Guía de scripts y automatización Python de IBM SPSS Modeler 18.4

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado "Avisos" en la página 481.

Información del producto

Esta edición se aplica a la versión 18, release 4, modificación 0 de IBM® SPSS Modeler y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

© Copyright International Business Machines Corporation .

Contenido

Capítulo 1. Scripts y lenguaje de scripts 1
- Visión general de scripts 1
- Tipos de scripts 1
- Scripts de ruta 1
  - Ejemplo de script de ruta: entrenamiento de una red neuronal 3
  - Límites de tamaño de código Jython 3
- Scripts autónomos 4
  - Ejemplo de script autónomo: guardar y cargar un modelo 4
  - Ejemplo de script autónomo: Generación de un modelo de selección de características 5
- Scripts de Supernodo 5
  - Script de supernodo de ejemplo 6
- Creación de bucles y ejecución condicional en rutas 6
  - Bucles en rutas 7
  - Ejecución condicional en rutas 10
- Ejecutar e interrumpir scripts 11
- Buscar y reemplazar 12

Capítulo 2. Language de scripts 15
- Visión general de lenguaje de script 15
- Python y Jython 15
- Scripts de Python 16
  - Operaciones 16
  - Listas 16
  - Cadenas 17
  - Observaciones 19
  - Sintaxis de las sentencias 19
  - Identificadores 19
  - Bloques de código 20
  - Pasar argumentos a un script 20
  - Ejemplos 21
  - Métodos matemáticos 21
  - Utilización de caracteres no ASCII 23
- Programación orientada a objetos 24
  - Definición de una clase 24
  - Creación de una instancia de clase 25
  - Añadir atributos a una instancia de clase 25
  - Definición de atributos de clase y métodos 25
  - Variables ocultas 26
  - Heredado 26

Capítulo 3. Scripts de IBM SPSS Modeler 27
- Tipos de scripts 27
- Rutas, rutas de supernodo y diagramas 27
  - Rutas 27
  - Rutas de Supernodo 27
  - Diagramas 27
- Ejecución de una ruta 27
- El contexto de los scripts 28
- Referencia a nodos existentes 29
  - Buscar nodos 29

Establecimiento de propiedades..... 30
Creación de nodos y modificación de rutas..... 31
Creación de nodos..... 31
Enlazar y desenlazar nodos..... 32
Importar, sustituir y eliminar nodos..... 33
Atravesar los nodos de una ruta..... 34
Borrado o eliminación de elementos..... 35
Obtener información sobre los nodos..... 35

Capítulo 4. API de scripts..... 39
Introducción a la API de scripts..... 39
Ejemplo 1: buscar nodos utilizando un filtro personalizado..... 39
Ejemplo 2: permitir a los usuarios obtener información de directorio o archivo basándose en sus privilegios..... 39
Metadatos: información sobre datos..... 40
Acceso a objetos generados..... 43
Manejo de errores..... 44
Parámetros de ruta, sesión y Supernodo..... 45
Valores globales..... 48
Trabajar con varias rutas: Scripts autónomos..... 49

Capítulo 5. Sugerencias para scripts..... 51
Modificación de ejecución de rutas..... 51
Nodos de recorrido en bucle..... 51
Acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository..... 51
Generación de una contraseña codificada..... 54
Comprobación de script..... 54
Scripts desde la línea de comandos..... 55
Compatibilidad con releases anteriores..... 55
resultados de ejecución de la secuencia, acceder..... 55
Modelo de contenido de tabla..... 56
Modelo de contenido XML..... 58
Modelo de contenido JSON..... 60
Modelo de contenido de estadísticas de columna y modelo de contenido de estadísticas por pares..... 61

Capítulo 6. Argumentos de la línea de comandos..... 65
Invocación del software..... 65
Utilización de argumentos de la línea de comandos..... 65
Argumentos del sistema..... 66
Argumentos de parámetros..... 67
Argumentos de conexión del servidor..... 68
Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository..... 69
IBM SPSS Analytic Server Argumentos de conexión..... 70
Combinación de varios argumentos..... 70

Capítulo 7. Referencia de propiedades..... 73
Visión general de referencia de propiedades..... 73
Sintaxis para propiedades..... 73
Ejemplos de las propiedades node y stream..... 75
Visión general de propiedades de nodo..... 75
Propiedades de nodos comunes..... 76

Capítulo 8. Propiedades de ruta..... 77

Capítulo 9. Propiedades de nodos de origen..... 81

Propiedades comunes de nodos de origen....81
Propiedades de asimport....88
Propiedades del nodo cognosimport....89
propiedades de databasenode....93
Propiedades de datacollectionimportnode....94
Propiedades de excelimportnode....98
Propiedades de extensionimportnode....99
Propiedades de fixedfilenode....101
Propiedades del nodo gsdata_import....106
Propiedades jsonimportnode....106
Propiedades de sasimportnode....107
propiedades de simgennode....107
Propiedades de statisticsimportnode....109
Propiedades del nodo tm1odataimport....109
Propiedades del nodo tm1import (en desuso)....110
Propiedades del nodo twcimport....111
propiedades de userinputnode....112
Propiedades de variablefilenode....113
Propiedades de xmlimportnode....119

Capítulo 10. Propiedades de nodos de operaciones con registros....121
propiedades de appendnode....121
propiedades de aggregatenode....121
propiedades de balancenode....123
Propiedades cplexoptnode....123
Propiedades derive_stbnode....127
propiedades de distinctnode....129
Propiedades de extensionprocessnode....131
propiedades de mergenode....132
propiedades rfmaggregatenode....134
propiedades de samplenode....137
propiedades de selectnode....139
propiedades de sortnode....139
Propiedades de spacetimeboxes....140
Propiedades streamingtimeseries....142

Capítulo 11. Propiedades de nodos de operaciones con campos....153
propiedades de anonymizenode....153
properties autodataprepnode....154
propiedades de astimeintervalsnode....158
propiedades de binningnode....159
propiedades de derivenode....162
propiedades de ensemblenode....166
propiedades de fillernode....167
propiedades de filternode....168
propiedades de historynode....169
propiedades de partitionnode....170
propiedades de reclassifynode....171
propiedades de reordernode....172
propiedades de reprojectnode....173
propiedades de restructurenode....173
propiedades de rfmanalysisnode....174
propiedades de settoflagnode....176
propiedades de statisticstransformnode....176
Propiedades timeintervalsnode (en desuso)....177
propiedades de transposenode....183
propiedades de typenode....185

Capítulo 12. Propiedades de nodos Gráfico.....193
Propiedades comunes del nodo Gráfico.....193
Propiedades de collectionnode.....194
Propiedades de distributionnode.....195
Propiedades de evaluationnode.....196
Propiedades de graphboardnode.....198
Propiedades de histogramnode.....203
Propiedades de mapvisualization.....204
Propiedades de multiplotnode.....209
Propiedades de plotnode.....210
Propiedades de timeplotnode.....213
Propiedades eplotnode.....214
Propiedades tsnenode.....215
Propiedades de webnode.....217

Capítulo 13. Propiedades de nodos de modelado.....219
Propiedades comunes de nodos de modelado.....219
propiedades de anomalydetectionnode.....220
propiedades de apriorinode.....221
propiedades associationrulesnode.....223
propiedades de autoclassifiernode.....226
Propiedades de ajustes de algoritmo.....228
propiedades de nodo de agrupación en clústeres automática.....229
propiedades de autonumericnode.....231
Propiedades de bayesnetnode.....233
propiedades de c50node.....235
propiedades de carmanode.....237
propiedades de cartnode.....238
propiedades de chaidnode.....241
propiedades de coxregnode.....244
Propiedades de decisionlistnode.....246
propiedades de discriminantnode.....248
Propiedades de extensionmodelnode.....250
propiedades de factornode.....253
propiedades de featureselectionnode.....255
propiedades de genlinnode.....257
Propiedades de glmmnode.....263
Propiedades de gle.....268
propiedades de kmeansnode.....275
Propiedades kmeansasnode.....276
propiedades de knnnode.....277
propiedades de kohonennode.....279
Propiedades de linearnode.....280
Propiedades de linearasnode.....282
Propiedades de logregnode.....284
propiedades de lsvmnode.....290
propiedades de neuralnetnode.....291
Propiedades de neuralnetwork.....294
propiedades de questnode.....296
propiedades randomtrees.....299
Propiedades de regressionnode.....301
propiedades de sequencenode.....304
propiedades de slrmnode.....305
propiedades de statisticsmodelnode.....306
propiedades de stpnode.....307
propiedades de svmnode.....313

Propiedades de tcmnode..... 315
propiedades ts..... 320
propiedades de treeas..... 331
Propiedades de twostepnode..... 334
Propiedades de twostepAS..... 335

Capítulo 14. Propiedades del nodo de nugget de modelo..... 339
Propiedades de applyanomalydetectionnode..... 339
Propiedades de applyapriorinode..... 339
Propiedades de applyassociationrulesnode..... 340
Propiedades de applyautoclassifiernode..... 341
Propiedades de applyautoclusternode..... 342
Propiedades de applyautonumericnode..... 342
Propiedades de applybayesnetnode..... 343
Propiedades de applyc50node..... 343
Propiedades de applycarmanode..... 344
Propiedades de applycartnode..... 344
Propiedades de applychaidnode..... 344
Propiedades de applycoxregnode..... 345
Propiedades de applydecisionlistnode..... 345
Propiedades de applydiscriminantnode..... 346
Propiedades de applyextension..... 346
Propiedades de applyfactornode..... 348
Propiedades de applyfeatureselectionnode..... 348
Propiedades de applygeneralizedlinearnode..... 348
Propiedades de applyglmmnode..... 349
Propiedades de applygle..... 350
Propiedades applygmm..... 350
Propiedades de applykmeansnode..... 351
Propiedades de applyknnnode..... 351
Propiedades de applykohonennode..... 351
Propiedades de applylinearnode..... 351
Propiedades de applylinearasnode..... 352
Propiedades de applylogregnode..... 352
Propiedades de applylsvmnode..... 352
Propiedades de applyneuralnetnode..... 352
propiedades de applyneuralnetworknode..... 353
Propiedades de applyocsvmnode..... 354
Propiedades de applyquestnode..... 354
Propiedades applyrandomtrees..... 355
Propiedades de applyregressionnode..... 356
propiedades de applyselflearningnode..... 356
Propiedades de applysequencenode..... 356
Propiedades de applysvmnode..... 356
Propiedades de applystpnode..... 356
Propiedades de applytcmnode..... 357
Propiedades applyts..... 357
Propiedades applytimeseriesnode (en desuso)..... 357
Propiedades de applytreeas..... 358
Propiedades de applytwostepnode..... 358
Propiedades de applytwostepAS..... 358
Propiedades de applyxgboosttreenode..... 359
propiedades de applyxgboostlinearnode..... 359
Propiedades de hdbscannugget..... 359
propiedades de kdeapply..... 359

Capítulo 15. Propiedades del nodo de modelado de base de datos..... 361

Propiedades de nodos de modelado de Microsoft 361
Propiedades de nodos de modelado de Microsoft 361
Propiedades de nugget de modelo de Microsoft 363
Propiedades de nodos de modelado de Oracle 366
Propiedades de nodos de modelado de Oracle 366
Propiedades de nugget de modelo de Oracle 372
Propiedades de nodos de modelado de IBM Netezza Analytics 373
Propiedades de nodos de modelado de Netezza 373
Propiedades de nugget de modelo de Netezza 390

Capítulo 16. Propiedades del nodo de resultados 393
propiedades de analysisnode 393
propiedades de dataauditnode 394
Propiedades de extensionoutputnode 396
Propiedades kdeexport 398
propiedades de matrixnode 399
propiedades de meansnode 401
propiedades de reportnode 403
propiedades de setglobalsnode 405
propiedades de simevalnode 406
propiedades de simfitnode 407
propiedades de statisticsnode 408
Propiedades de statisticsoutputnode 409
propiedades de tablenode 409
propiedades de transformnode 412

Capítulo 17. Propiedades de nodos Exportar 415
Propiedades de nodos Exportar comunes 415
Propiedades de asexport 415
Propiedades del nodo de exportación Cognos 416
propiedades de databaseexportnode 418
Propiedades de datacollectionexportnode 423
Propiedades de excelexportnode 424
Propiedades de extensionexportnode 425
Propiedades jsonexportnode 426
Propiedades de outputfilenode 427
Propiedades de sasexportnode 428
Propiedades de statisticsexportnode 428
Propiedades del nodo tm1odataexport 429
Propiedades del nodo tm1export (en desuso) 431
Propiedades de xmlexportnode 433

Capítulo 18. Propiedades de nodos de IBM SPSS Statistics 435
Propiedades de statisticsimportnode 435
propiedades de statisticstransformnode 435
propiedades de statisticsmodelnode 436
Propiedades de statisticsoutputnode 437
Propiedades de statisticsexportnode 437

Capítulo 19. Propiedades de nodo Python 439
Propiedades gmm 439
Propiedades hdbscannode 440
Propiedades kdemodel 442
Propiedades kdeexport 443
Propiedades gmm 444
ocsvmnode, propiedades 445
Propiedades rfnode 448

Propiedades de smotenode..... 450
Propiedades tsnenode.....451
Propiedades xgboostlinearnode..... 453
Propiedades de xgbosttreenode..... 454

Capítulo 20. Propiedades de nodo Spark..... 457
Propiedades isotonicasnode..... 457
Propiedades kmeansasnode..... 457
Propiedades multilayerperceptronnode..... 459
Propiedades xgboostasnode..... 459

Capítulo 21. Propiedades de supernodo..... 463

Apéndice A. Referencia de nombres de nodo..... 465
Nombres de nugget de modelo..... 465
Evitar nombres duplicados del modelo..... 467
Nombres de tipo de resultado..... 467

Apéndice B. Migración desde scripts de herencia a scripts Python..... 469
Visión general de la migración de scripts de herencia..... 469
Diferencias generales..... 469
El contexto de los scripts..... 469
Comparativa de comandos y funciones..... 469
Literales y comentarios..... 470
Operadores..... 470
Comandos condicionales y de bucle..... 471
Variables..... 472
Tipos modelo, resultado y nodo..... 472
Nombres de propiedades..... 473
Referencias de nodos..... 473
Obtener y establecer propiedades..... 473
Edición de rutas..... 474
Operaciones de nodo..... 475
Bucle..... 475
Ejecución de rutas..... 476
Acceso a objetos mediante el sistema de archivos y el repositorio..... 477
Operaciones de ruta..... 477
Operaciones de modelo..... 478
Operaciones de resultado de documento..... 478
Otras diferencias entre scripts heredados y scripts Python..... 478

Avisos..... 481
Marcas comerciales..... 482
Términos y condiciones de la documentación de producto..... 482

Índice..... 485

Capítulo 1. Scripts y lenguaje de scripts

Visión general de scripts

Los scripts en IBM SPSS Modeler son una herramienta potente para automatizar procesos en la interfaz de usuario. Los scripts pueden realizar los mismos tipos de acciones que se realizan con el ratón o el teclado y se utilizan para automatizar tareas que resultarían extremadamente repetitivas o llevarían mucho tiempo si se realizaran manualmente.

Puede utilizar los scripts para:

- Imponer un orden concreto para la ejecución de nodos en una ruta.
- Establecer propiedades de los nodos y realizar derivaciones usando un subconjunto de CLEM (Control Language for Expression Manipulation).
- Especificar una secuencia automática de acciones que normalmente implican la interacción del usuario (por ejemplo, puede generar un modelo y comprobarlo a continuación).
- Configurar procesos complejos que requieren una interacción sustancial del usuario, como los procedimientos de validación cruzada que requieren una repetitiva generación y comprobación de modelo.
- Configurar procesos que manipulen rutas; por ejemplo, puede tomar una ruta de entrenamiento de modelo, ejecutarla y producir la ruta de comprobación del modelo automáticamente.

Este capítulo proporciona descripciones de alto nivel y ejemplos de scripts de nivel de ruta, scripts autónomos y scripts en Supernodos en la interfaz de IBM SPSS Modeler. Para obtener más información sobre el lenguaje, la sintaxis y los comandos consulte los capítulos siguientes.

Nota:

No puede importar y ejecutar scripts creados en IBM SPSS Statistics dentro de IBM SPSS Modeler.

Tipos de scripts

IBM SPSS Modeler utiliza tres tipos de scripts:

- Los **scripts de la ruta** se guardan como una propiedad de ruta y se guardan y se cargan con una ruta específica. Por ejemplo, puede escribir un script de ruta que automatice el proceso de entrenamiento y aplicación de un nugget de modelo. También puede especificar que cuando se ejecute una ruta particular, se ejecute el script, en lugar del contenido del lienzo de la ruta.
- Los **scripts autónomos** no están asociados a ninguna ruta en particular y se guardan en archivos de texto externos. Puede utilizar un script autónomo, por ejemplo, para manipular varias rutas a la vez.
- Los **scripts Supernodos** se guardan como una propiedad de ruta de supernodo. Los scripts Supernodos sólo están disponibles en supernodos terminales. Puede utilizar un script de supernodo para controlar la secuencia de ejecución del contenido del supernodo. En supernodos no terminales (origen o proceso), puede definir propiedades del supernodo o los nodos que contiene en su script de ruta directamente.

Scripts de ruta

Los scripts se pueden utilizar para personalizar operaciones dentro de una ruta particular y se guardan con esa ruta. Los scripts de la ruta se pueden utilizar para especificar un orden de ejecución particular para los nodos terminales de una ruta. El cuadro de diálogo del script de ruta se utiliza para editar el script que está guardado con la ruta actual.

Para acceder a la pestaña de scripts de ruta en el cuadro de diálogo propiedades de ruta:

1. Desde el menú **Herramientas**, elija:

Propiedades de ruta > **Ejecución**

2. Pulse en la pestaña **Ejecución** para trabajar con scripts en la ruta actual.

Utilice los iconos de barra de herramientas en la parte superior del recuadro de diálogo del script de ruta para las operaciones siguientes:

- Importar el contenido de un script autónomo preexistente en la ventana.
- Guardar un script como archivo de texto.
- Imprimir un script.
- Añadir script predeterminado.
- Editar un script (deshacer, cortar, copiar, pegar y otras funciones de edición comunes).
- Ejecutar el script completo actual.
- Ejecutar líneas concretas de un script.
- Detener un script durante la ejecución. (Este icono sólo está habilitado cuando un script se está ejecutando).
- Comprobar la sintaxis del script y, si se encuentra algún error, mostrarlos para la revisión en el panel inferior del recuadro de diálogo.

Nota: A partir de la versión 16.0, SPSS Modeler utiliza el lenguaje de scripts Python. Todas las versiones anteriores a la 16.0 utilizaban un lenguaje de script exclusivo para SPSS Modeler, al que ahora se denomina como script de herencia. Según el tipo de script con el que trabaje, en la pestaña **Ejecución** seleccione la modalidad de ejecución **Predeterminada (script opcional)** y, a continuación, seleccione **Python** o **Legacy**.

Puede especificar si un script se va a ejecutar o no cuando se ejecuta la ruta. Para ejecutar el script cada vez que se ejecuta la ruta, respectando el orden de ejecución del script, seleccione **Ejecutar este script**. De este modo se proporciona una automatización a nivel de ruta para acelerar la generación del modelo. Sin embargo, la configuración predeterminada es omitir el script durante la ejecución de la ruta. Incluso si selecciona la opción **Omitir este script**, siempre puede ejecutar la ruta directamente desde este cuadro de diálogo.

El editor de scripts incluye las siguientes características que ayudan a crear scripts:

- Resaltado de sintaxis. Se resaltan las palabras claves, los valores literales (tales como cadenas y números) y los comentarios.
- Numeración de líneas.
- Coincidencia de bloques. Cuando se coloca el cursor al inicio de un bloque de programa, también se resalta el bloque final correspondiente.
- Finalización automática sugerida.

Los colores y los estilos de texto que utiliza la función de resaltado de la sintaxis se pueden personalizar utilizando las preferencias de visualización de IBM SPSS Modeler. Para acceder a las preferencia de visualización, elija **Herramientas** > **Opciones** > **Opciones de usuario** y seleccione la pestaña **Sintaxis**.

Se puede acceder a una lista de finalizaciones de sintaxis sugeridas seleccionando la **Sugerencia automática** en el menú de contexto o pulsando Ctrl más espacio. Utilice las teclas de cursor para desplazarse hacia arriba y hacia abajo por la lista y, a continuación, pulse Intro para insertar el texto seleccionado. Para salir de la modalidad de sugerencia automática sin modificar el texto existente, pulse Esc.

La pestaña **Depurar** muestra mensajes de depuración y se puede utilizar para evaluar el estado del script una vez que se haya ejecutado. La pestaña **Depurar** está formada por un área de texto de solo lectura y un campo de texto de entrada de una sola línea. El área de texto muestra el texto que se envía a la salida estándar o un error estándar mediante los scripts, por ejemplo, a través del texto del mensaje de error. El campo de texto de entrada toma la entrada del usuario. Esta entrada se evalúa dentro del contexto del script que se ha ejecutado más recientemente en el diálogo (conocido como el *contexto de los scripts*). El área de texto contiene el comando y la salida resultante, de modo que el usuario puede ver un rastreo de

los comandos. El campo de entrada de texto siempre contiene el indicador de mandatos (- -> for Legacy scripting).

Un contexto de script nuevo se crea en las circunstancias siguientes:

- Se ejecuta un script utilizando **Ejecutar este script** o **Ejecutar líneas seleccionadas**.
- Se modifica el lenguaje de script.

Si se crea un nuevo contexto de script, el área de texto se borra.

Nota: La ejecución de una ruta fuera del panel del script no modifica el contexto del script del panel del script. Los valores de las variables que se han creado como parte de la ejecución no son visibles dentro del recuadro de diálogo de script.

Ejemplo de script de ruta: entrenamiento de una red neuronal

Una ruta se puede usar para entrenar un modelo de red neuronal cuando se ejecute. Normalmente, para comprobar el modelo, se inserta el nodo de modelado para agregar el modelo a la ruta, realizar las conexiones adecuadas y ejecutar el nodo Análisis.

Mediante un script de IBM SPSS Modeler se puede automatizar el proceso de comprobar el nugget de modelo tras crearlo. Por ejemplo, el siguiente script de ruta para probar la secuencia de demostración `druglearn.str` (disponible en la carpeta `/Demos/streams/` bajo la instalación de IBM SPSS Modeler) se puede ejecutar desde el diálogo Propiedades de ruta (**Herramientas > Propiedades de corriente > Script**):

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

Los puntos siguientes describen cada línea de este ejemplo de script.

- La primera línea define una variable que apunta a la ruta actual.
- En la línea 2, el script busca el nodo generador Red neuronal.
- En la línea 3, el script crea una lista donde los resultados de la ejecución se pueden almacenar.
- En la línea 4, se crea el nugget de modelo Red neuronal. Se almacena en la lista definida en línea 3.
- En la línea 5, se crea un nodo de aplicación de modelo para el nugget de modelo y se coloca en el lienzo de rutas.
- En la línea 6, se crea un nodo de análisis denominado `Drug`.
- En la línea 7, el script busca el nodo Type.
- En la línea 8, el script conecta el nodo de aplicación de modelo creado en la línea 5 entre el nodo Type y el nodo Analysis.
- Finalmente, el nodo Análisis se ejecuta para producir el informe Análisis.

Es posible utilizar un script para crear y ejecutar una ruta desde cero, comenzando con un lienzo vacío.

Límites de tamaño de código Jython

Jython compila cada script para el código de bytes Java que, a continuación, es ejecutado por la máquina virtual Java (JVM). Sin embargo, Java impone un límite en el tamaño de un único archivo de código de bytes. Así, cuando Jython intenta cargar el código de bytes, puede hacer que la JVM se bloquee. IBM SPSS Modeler no puede evitar que esto suceda.

Asegúrese de escribir los scripts Jython utilizando buenas prácticas de codificación (por ejemplo minimizando el código duplicado utilizando variables o funciones para calcular valores intermedios comunes). Si es necesario, puede que sea necesario dividir el código en varios archivos de origen o definirlo utilizando módulos porque éstos se compilan en archivos de código de bytes independientes.

Scripts autónomos

El cuadro de diálogo script autónomo se usa para crear o editar un script que se ha guardado como archivo de texto. En él se muestra el nombre del archivo y se proporcionan recursos para la carga, almacenamiento, importación y ejecución de scripts.

Para acceder al cuadro de diálogo del script autónomo:

En el menú principal, elija:

Herramientas > **Script autónomo**

Los scripts autónomos y los de ruta comparten las mismas opciones de comprobación de sintaxis de scripts y barra de herramientas. Para obtener más información, consulte el tema "Scripts de ruta" en la página 1.

Ejemplo de script autónomo: guardar y cargar un modelo

Los scripts autónomos son útiles para la manipulación de rutas. Presuponga que tiene dos rutas, una que crea un modelo y otra que usa diagramas para examinar el conjunto de reglas generado a partir de la primera ruta con campos de datos existentes. Un script autónomo para este escenario tendría un aspecto similar a éste:

```
taskrunner = modeler.script.session().getTaskRunner()

# Modify this to the correct Modeler installation Demos folder.
# Note use of forward slash and trailing slash.
installation = "C:/Program Files/IBM/SPSS/Modeler/19/Demos/"

# First load the model builder stream from file and build a model
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Save the model to file
taskrunner.saveModelToFile(results[0], "rule.gm")

# Now load the plot stream, read the model from file and insert it into the stream
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivenode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
modelapplier.setPositionBetween(derivenode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivenode, plotnode)
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

Nota: Para obtener más información sobre el lenguaje de scripts en general, consulte "Visión general de lenguaje de script" en la página 15.

Ejemplo de script autónomo: Generación de un modelo de selección de características

Comenzando con un lienzo vacío, este ejemplo crea una ruta que genera un modelo de selección de características, aplica el modelo y crea una tabla que indica los 15 campos más importantes respecto al objetivo especificado.

```
stream = modeler.script.session().createProcessorStream("featureselection",
True)

statisticsimportnode = stream.createAt("statisticsimport", "Statistics
File", 150, 97)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/
customer_dbase.sav")

typenode = stream.createAt("type", "Type", 258, 97)
typenode.setKeyedPropertyValue("direction", "response_01", "Target")

featureselectionnode = stream.createAt("featureselection", "Feature
Selection", 366, 97)
featureselectionnode.setPropertyValue("top_n", 15)
featureselectionnode.setPropertyValue("max_missing_values", 80.0)
featureselectionnode.setPropertyValue("selection_mode", "TopN")
featureselectionnode.setPropertyValue("important_label", "Check Me Out!")
featureselectionnode.setPropertyValue("criteria", "Likelihood")

stream.link(statisticsimportnode, typenode)
stream.link(typenode, featureselectionnode)
models = []
featureselectionnode.run(models)

# Assumes the stream automatically places model apply nodes in the stream
applynode = stream.findByType("applyfeatureselection", None)
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96,
applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

El script crea un nodo de origen para leer en los datos, utiliza un nodo Tipo para establecer el rol (dirección) para el campo `response_01` en `Target` y, a continuación, crea y ejecuta un nodo Selección de característica. Este script también conecta cada nodo y posiciones en el lienzo de la ruta para producir un diseño legible. El nugget de modelo resultante se conecta al nodo Tabla, que indica los 15 campos más importantes, tal y como determinan las propiedades `selection_mode` y `top_n`. Para obtener más información, consulte el tema “propiedades de featureselectionnode” en la página 255.

Scripts de Supernodo

Puede crear y guardar scripts en cualquier supernodo de terminal utilizando el lenguaje de script de IBM SPSS Modeler. Estos scripts sólo están disponibles para supernodos terminales y se suelen utilizar cuando crea rutas de plantilla o para imponer un orden de ejecución especial del contenido del supernodo. Los scripts de supernodo también permiten ejecutar más de un script en una ruta.

Por ejemplo, supongamos que necesita especificar el orden de ejecución de una ruta compleja y su supernodo contiene varios nodos, incluyendo un nodo Val. globales, que se debe ejecutar antes de derivar un nuevo campo utilizado en un nodo Gráfico. En este caso, puede crear un script de supernodo que ejecute el nodo Val. globales en primer lugar. Los valores calculados por este nodo, como la media o la desviación estándar, se pueden usar posteriormente cuando se ejecute el nodo Gráfico.

En un script de Supernodo, puede especificar las propiedades del nodo de la misma manera que otros scripts. También puede cambiar y definir las propiedades de cualquier supernodo o sus nodos encapsulados directamente desde un script de ruta. Para obtener más información, consulte el tema

Capítulo 21, “Propiedades de supernodo”, en la página 463. Este método funciona para supernodos de origen y proceso y supernodos terminales.

Nota: Puesto que solo los supernodos de terminal pueden ejecutar sus propios scripts, la pestaña Scripts del recuadro de diálogo Supernodo solo está disponible para los supernodos de terminal.

Para abrir el cuadro de diálogo de script de supernodo desde el lienzo principal:

Seleccione un supernodo terminal en el lienzo de rutas y, en el menú de supernodo, seleccione:

Script de Supernodo...

Para abrir el cuadro de diálogo de script de supernodo desde el lienzo de supernodo aumentado:

Pulse con el botón derecho del ratón en el lienzo del supernodo y en el menú contextual elija:

Script de Supernodo...

Script de supernodo de ejemplo

El siguiente script de supernodo establece el orden en que se ejecutarán los nodos terminales del supernodo. Este orden garantiza que el nodo Val. globales se ejecuta primero para que los valores que calcula este nodo se puedan utilizar cuando se ejecute otro nodo.

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

Bloquear y desbloquear supernodos

El ejemplo siguiente ilustra cómo puede bloquear y desbloquear un supernodo:

```
stream = modeler.script.stream()
superNode=stream.findByID('id854RNTSD5MB')
# unlock one super node
print 'unlock the super node with password abcd'
if superNode.unlock('abcd'):
    print 'unlocked.'
else:
    print 'invalid password.'
# lock one super node
print 'lock the super node with password abcd'
superNode.lock('abcd')
```

Creación de bucles y ejecución condicional en rutas

A partir de la versión 16.0, SPSS Modeler permite crear scripts básicos desde dentro de una ruta seleccionando valores en varios cuadros de diálogo en lugar de tener que escribir instrucciones directamente en lenguaje de script. Los dos principales tipos de script que pueden crearse de este modo son los bucles sencillos y un modo de ejecutar nodos si se cumple una condición.

En una misma ruta pueden combinarse bucles y reglas de ejecución condicional. Por ejemplo, suponga que tiene datos relativos a ventas de vehículos de fabricantes de todo el mundo. Podría crearse un bucle para procesar los datos en una ruta, identificando los detalles por país del fabricante, y sacar los datos a distintas gráficas para mostrar detalles tales como volumen de ventas por modelo, niveles de emisión por fabricante y cilindrada, etc. Si solo le interesara analizar la información procedente de Europa, también podría añadir condiciones al bucle que impidieran la creación de gráficas de fabricantes procedentes de América y Asia.

Nota: Puesto que tanto un bucle como una ejecución condicional están basados en scripts de segundo plano, solo se aplican a una ruta entera cuando se ejecuta.

- **Bucles** Los bucles pueden utilizarse para automatizar tareas repetitivas. Por ejemplo, esto podría suponer añadir un determinado número de nodos a una ruta y modificar un parámetro del nodo cada vez. De forma opcional, podría controlarse la ejecución de una ruta o rama varias veces, como en los ejemplos siguientes:
  - Ejecutar la ruta un determinado número de veces y cambiar el origen cada vez.
  - Ejecutar la ruta un determinado número de veces cambiando el valor de una variable cada vez.
  - Ejecutar la ruta un determinado número de veces especificando un campo adicional en cada ejecución.
  - Construir un modelo un determinado número de veces y cambiar la configuración del modelo cada vez.
- **Ejecución condicional** Puede utilizarse para controlar cómo ejecutan los nodos en función de condiciones definidas previamente como, por ejemplo:
  - Dependiendo de si un determinado valor es verdadero o falso, se controla la ejecución de un nodo.
  - Definir si la iteración de nodos se ejecutará en paralelo o de forma secuencial.

Tanto bucles ejecuciones condicionales se configuran en la pestaña Ejecución dentro del cuadro de diálogo Propiedades de ruta. Los nodos que se utilicen en bucles o de forma condicional aparecerán con un símbolo adicional en el lienzo de rutas para indicar que forman parte de una ejecución por bucles o condicional.

Puede accederse a la pestaña Ejecución de tres maneras:

- Mediante los menús de la parte superior del cuadro de diálogo principal:
  1. En el menú Herramientas, seleccione:

     Propiedades de ruta > **Ejecución**
  2. Pulse en la pestaña Ejecución para trabajar con los scripts de la ruta actual.
- Dentro de una ruta:
  1. Pulse con el botón derecho en un nodo y seleccione **Bucles/Ejecución condicional**.
  2. Seleccione la opción de submenú que corresponda.
- En la barra de herramientas gráfica de la parte superior del cuadro de diálogo principal, pulse en el icono de propiedades de ruta.

Si es la primera vez que configura los detalles de un bucle o de una ejecución condicional, en la pestaña Ejecución seleccione el modo de ejecución **Ejecución de bucles/condicional** y después seleccione la subpestaña **Condicional** o **Bucles**.

Bucles en rutas

Con la creación de bucles puede automatizar las tareas repetitivas en las rutas, por ejemplo:

- Ejecutar la ruta un determinado número de veces y cambiar el origen cada vez.
- Ejecutar la ruta un determinado número de veces cambiando el valor de una variable cada vez.
- Ejecutar la ruta un determinado número de veces especificando un campo adicional en cada ejecución.
- Construir un modelo un determinado número de veces y cambiar la configuración del modelo cada vez.

Configurar las condiciones que deben cumplirse en la subpestaña **Bucle** de la pestaña Ejecución de la ruta. Para visualizar la subpestaña, seleccione el modo de ejecución **Ejecución en bucle/condicional**.

Los requisitos de bucle que defina entrarán en vigor cuando se ejecute la ruta, si se ha establecido la modalidad de ejecución **Ejecución en bucle/condicional**. Opcionalmente, puede generar el código de script para los requisitos de bucle y pegarlo en el editor de scripts pulsando **Pegar ...** en la esquina inferior derecha de la subficha Looping; la pestaña Ejecución principal cambia para mostrar el modo de

ejecución de **Valor predeterminado (script opcional)** con el script en la parte superior de la pestaña. Esto significa que puede definir bucles utilizando las diferentes opciones del cuadro de diálogo de bucle antes de generar un script que puede personalizar adicionalmente en el editor de scripts. Tenga en cuenta que al pulsar **Pegar ...** cualquier requisito de ejecución condicional que haya definido también se visualizará en el script generado.

Importante: Las variables de bucle que establezca en una ruta de SPSS Modeler pueden sustituirse si se ejecuta la ruta en un trabajo IBM SPSS Collaboration and Deployment Services. Esto se debe a que la entrada del editor del trabajo IBM SPSS Collaboration and Deployment Services sobrescribe la entrada de SPSS Modeler. Por ejemplo, si se establece una variable de bucle en la ruta para crear un nombre de archivo de salida diferente para cada bucle, los archivos se especifican correctamente en SPSS Modeler, pero son sustituidos por la entrada fija especificada en la pestaña Resultado del Gestor de despliegue de IBM SPSS Collaboration and Deployment Services.

Para configurar un bucle

1. Cree una clave de iteración para definir la estructura principal del bucle principal que se creará en una ruta. Consulte el tema Crear una clave de iteración para obtener más información.
2. Si es necesario, defina una o varias variables de iteración. Consulte el tema Crear una variable de iteración para obtener más información.
3. Las iteraciones y las variables que cree se muestran en el cuerpo principal de la subpestaña. De forma predeterminada, las iteraciones se ejecutan en el orden en que aparecen. Para subir o bajar una iteración en la lista, pulse la iteración para seleccionarla y, a continuación, utilice la flecha arriba o la flecha abajo de la columna de la derecha de la subpestaña para cambiar el orden.

Creación de una clave de iteración para bucles de rutas

Utilice una clave de iteración para definir la estructura principal del bucle principal que se creará en una ruta. Por ejemplo, si está analizando las ventas de automóviles, puede crear un parámetro de ruta *País de fabricación* y utilizarlo como la clave de la iteración. Cuando se ejecute la ruta, esta clave se establece en cada valor de país diferente de sus datos durante cada iteración. Utilice el cuadro de diálogo Definir clave de iteración para configurar la clave.

Para abrir el cuadro de diálogo, seleccione la opción **Clave de iteración ...** en la esquina inferior izquierda de la subficha Looping, o pulse con el botón derecho del ratón en cualquier nodo de la ruta y seleccione **Ejecución/Ejecución condicional** > **Definir clave de iteración (Campos)** o **Ejecución/Ejecución condicional** > **Definir clave de iteración (Valores)**. Si abre el cuadro de diálogo desde la ruta, algunos campos se completan automáticamente, tales como el nombre del nodo.

Para configurar una clave de iteración, complete los campos siguientes:

Iterar en. Puede seleccionar entre una de las opciones siguientes:

- **Parámetro de ruta - Campos**. Utilice esta opción para crear un bucle que establezca el valor de un parámetro de ruta existente en cada campo especificado de forma ordenada.
- **Parámetro de ruta - Valores**. Utilice esta opción para crear un bucle que establezca el valor de un parámetro de ruta existente en cada valor especificado de forma ordenada.
- **Propiedad del nodo - Campos**. Utilice esta opción para crear un bucle que establezca el valor de una propiedad de nodo en cada campo especificado de forma ordenada.
- **Propiedad del nodo - Valores**. Utilice esta opción para crear un bucle que establezca el valor de una propiedad de nodo en cada valor especificado de forma ordenada.

Qué se ha de establecer. Elija el elemento cuyo valor se establecerá cada vez que se ejecute el bucle. Puede seleccionar entre una de las opciones siguientes:

- **Parámetro**. Solo está disponible si se selecciona **Parámetro de ruta - Campos** o **Parámetro de ruta - Valores**. Seleccione el parámetro necesario en la lista disponible.
- **Nodo**. Solo está disponible si se selecciona **Propiedad del nodo - Campos** o **Propiedad del nodo - Valores**. Seleccione el nodo para el que desee configurar un bucle. Pulse el botón Examinar para abrir

el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.

- **Propiedad**. Solo está disponible si se selecciona **Propiedad del nodo - Campos** o **Propiedad del nodo - Valores**. Seleccione la propiedad del nodo en la lista disponible.

Campos de uso. Solo está disponible si se selecciona **Parámetro de ruta - Campos** o **Propiedad del nodo - Campos**. Seleccione el campo o los campos de un nodo que se utilizarán para proporcionar los valores de iteración. Puede seleccionar entre una de las opciones siguientes:

- **Nodo**. Solo está disponible si se selecciona **Parámetro de ruta - Campos**. Seleccione el nodo que contiene los detalles para los que desea configurar un bucle. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.
- **Lista de campos**. Pulse el botón de lista de la columna derecha para visualizar el cuadro de diálogo Seleccionar campos, donde puede seleccionar los campos del nodo para proporcionar los datos de iteración. Si desea más información, consulte "Selección de campos en iteraciones" en la página 10.

Valores de uso. Solo está disponible si se selecciona **Parámetro de ruta - Valores** o **Propiedad del nodo - Valores**. En el campo seleccionado, seleccione el o los valores que se utilizarán como valores de iteración. Puede seleccionar entre una de las opciones siguientes:

- **Nodo**. Solo está disponible si se selecciona **Parámetro de ruta - Valores**. Seleccione el nodo que contiene los detalles para los que desea configurar un bucle. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Origen, Proceso, Gráfico, Modelado, Resultados o Aplicar modelo.
- **Lista de campos**. Seleccione el campo del nodo para proporcionar los datos de iteración.
- **Lista de valores**. Pulse el botón de lista de la columna derecha para visualizar el cuadro de diálogo Seleccionar valores, donde puede seleccionar los valores del campo para proporcionar los datos de iteración.

Creación de una variable de iteración para bucles de rutas

Puede utilizar variables de iteración para cambiar los valores de los parámetros o las propiedades de ruta de los nodos seleccionados en una ruta, cada vez que se ejecute un bucle. Por ejemplo, si el bucle de ruta está analizando los datos de ventas de automóviles y utiliza *País de fabricación* como clave de iteración, puede tener un gráfico de resultados que muestre las ventas por modelo y otro gráfico de resultados que muestre información sobre emisiones contaminantes. En estos casos puede crear variables de iteración que creen nuevos títulos para los gráficos resultantes, tales como *Emisiones de vehículos suecos* y *Ventas de automóviles japoneses por modelo*. Utilice el cuadro de diálogo Definir variable de iteración para configurar las variables que necesite.

Para abrir el cuadro de diálogo, seleccione la opción **Añadir variable ...** en la esquina inferior izquierda de la subficha Looping, o pulse con el botón derecho del ratón en cualquier nodo de la ruta y seleccione:**Ejecución/Ejecución condicional** > **Definir variable de iteración**.

Para configurar una variable de iteración, complete los campos siguientes:

Cambio. Seleccione el tipo de atributo que desea enmendar. Puede elegir **Parámetro de ruta** o **Propiedad del nodo**.

- Si selecciona **Parámetro de ruta**, elija el parámetro necesario y, a continuación, utilizando una de las opciones siguientes, si están disponibles en su ruta, defina cómo se debe establecer el valor de dicho parámetro con cada iteración del bucle:
  - **Variable global**. Seleccione la variable global en la que se debe establecer el parámetro de ruta.
  - **Casilla de resultados de tabla**. Para que un parámetro de ruta sea el valor de una casilla de resultados de tabla, seleccione la tabla en la lista y especifique la **Fila** y la **Columna** que se han de utilizar.

  - **Especificar manualmente**. Seleccione esta opción si desea especificar manualmente el valor que tomará este parámetro en cada iteración. Cuando regrese a la subpestaña Bucle se habrá creado una columna nueva en la que puede especificar el texto necesario.
- Si selecciona **Propiedad del nodo**, elija el nodo necesario y una de sus propiedades, a continuación, establezca el valor que desea que se utilice para dicha propiedad. Establezca el nuevo valor de la propiedad utilizando una de las opciones siguientes:
  - **Solo**. El valor de la propiedad utilizará el valor de la clave de iteración. Si desea más información, consulte "Creación de una clave de iteración para bucles de rutas" en la página 8.
  - **Como prefijo de tallo**. Utiliza el valor de la clave de iteración como un prefijo para lo que especifique en el campo **Tallo**.
  - **Como sufijo de tallo**. Utiliza el valor de la clave de iteración como un sufijo para lo que especifique en el campo **Tallo**.

  Si selecciona la opción de prefijo o de sufijo se le solicitará que añada el texto adicional en el campo **Tallo**. Por ejemplo, si el valor de la clave de iteración es *País de fabricación* y selecciona **Como prefijo de tallo**, puede entrar - *ventas por modelo* en este campo.

Selección de campos en iteraciones

Cuando se crean iteraciones pueden seleccionarse uno o más campos mediante el cuadro de diálogo Seleccionar campos.

Ordenar por: puede ordenar campos disponibles para su visualización eligiendo una de las siguientes opciones:

- **Natural**: el orden de los campos es aquél en que pasaron desde la parte anterior de la ruta de datos al nodo actual.
- **Nombre**: ordena los campos siguiendo un orden alfabético para su visualización.
- **Tipo**: ordena los campos en función de su nivel de medición. Esta opción es útil cuando se seleccionan campos con un nivel de medición en particular.

Seleccione los campos de la lista de uno en uno o mantenga pulsada la tecla Mayús o Ctrl mientras selecciona otros campos para seleccionar varios campos. También puede utilizar los botones que se muestran bajo la lista para seleccionar grupos de campos en función de su nivel de medición o seleccionar y anular la selección de todos los campos de la tabla.

Tenga en cuenta que los campos disponibles para su selección se filtran para mostrar solo los campos que son adecuados para el parámetro de ruta o la propiedad de nodo que se están utilizando. Por ejemplo, si se está utilizando un parámetro de ruta que tiene un tipo de almacenamiento de cadena, solo se mostrarán los campos que tengan un tipo de almacenamiento de cadena.

Ejecución condicional en rutas

Con la ejecución condicional puede controlar cómo se ejecutan los nodos terminales, en función de las condiciones de coincidencia de contenido de ruta que defina. Ejemplos de ello pueden ser los siguientes:

- Dependiendo de si un determinado valor es verdadero o falso, se controla la ejecución de un nodo.
- Definir si la iteración de nodos se ejecutará en paralelo o de forma secuencial.

Configurar las condiciones que deben cumplirse en la subpestaña **Condicional** de la pestaña Ejecución de la ruta. Para visualizar la subpestaña, seleccione el modo de ejecución **Ejecución en bucle/condicional**.

Los requisitos de ejecución condicional que defina entrarán en vigor cuando se ejecute la ruta, si se ha establecido la modalidad de ejecución **Ejecución en bucle/condicional**. Opcionalmente, puede generar el código de script para los requisitos de ejecución condicional y pegarlo en el editor de scripts pulsando **Pegar ...** en la esquina inferior derecha de la subficha Condicional; la pestaña Ejecución principal cambia para mostrar el modo de ejecución de **Valor predeterminado (script opcional)** con el script en la parte superior de la pestaña. Esto significa que puede definir condiciones utilizando las diferentes opciones del cuadro de diálogo de bucle antes de generar un script que puede personalizar adicionalmente en el editor

de scripts. Tenga en cuenta que al pulsar **Pegar ...** Los requisitos de bucle que haya definido también se mostrarán en el script generado.

Para configurar una condición:

1. En la columna de la derecha de la subpestaña Condicional, pulse el botón Añadir condición nueva para abrir el cuadro de diálogo **Añadir sentencia de ejecución condicional**. En este diálogo especifica la condición que se debe cumplir para que se ejecute el nodo.
2. En el cuadro de diálogo **Sentencia de ejecución condicional**, especifique lo siguiente:
   a. **Nodo**. Seleccione el nodo para el que desee configurar una ejecución condicional. Pulse el botón Examinar para abrir el diálogo Seleccionar nodo y elija el nodo que desee. Si hay demasiados nodos en la lista, puede filtrar la visualización para que únicamente se muestren los nodos de una de las siguientes categorías: Exportar, Gráfico, Modelado o Resultados.
   b. **Condición basada en**. Especifique la condición que se debe cumplir para que se ejecute el nodo. Puede elegir una de estas cuatro opciones: **Parámetro de ruta**, **Variable global**, **Casilla de resultados de tabla** o **Siempre verdadero**. Los detalles que especifique en la mitad inferior del cuadro de diálogo están controlados por la condición que elija.
      - **Parámetro de ruta**. Seleccione el parámetro de la lista disponible y, a continuación, seleccione el **Operador** para ese parámetro; por ejemplo, el operador puede ser Más, Igual, Menor que, Entre, etc. A continuación especifique el **Valor**, o los valores mínimos o máximos, dependiendo del operador seleccionado.
      - **Variable global**. Seleccione la variable de la lista disponible; por ejemplo, esto podría incluir: Media, Suma, Valor mínimo, Valor máximo o Desviación estándar. A continuación, seleccione **Operador** y los valores necesarios.
      - **Casilla de resultados de tabla**. Seleccione el nodo de tabla de la lista disponible y, a continuación, seleccione la **Fila** y la **Columna** en la tabla. A continuación, seleccione **Operador** y los valores necesarios.
      - **Siempre verdadero**. Seleccione esta opción si siempre se ha de ejecutar el nodo. Si selecciona esta opción, no hay parámetros adicionales que seleccionar.
3. Repita los pasos 1 y 2 tantas veces como sea necesario hasta que haya configurado todas las condiciones que requiere. El nodo que ha seleccionado y la condición que se debe cumplir antes de que se ejecute el nodo se muestran en el cuerpo principal de la subpestaña de las columnas **Ejecutar nodo** y **Si esta condición es verdadera** respectivamente.
4. De forma predeterminada, los nodos y las condiciones se ejecutan en el orden en que aparecen. Para subir o bajar un nodo y condición en la lista, pulse el nodo para seleccionarlo y, a continuación, utilice la flecha arriba o la flecha abajo en la columna de la derecha de la subpestaña para cambiar el orden.

Además, puede establecer las siguientes opciones en la parte inferior de la subpestaña Condicional:

- **Evaluar todo en orden**. Seleccione esta opción para evaluar cada condición en el orden en que se muestra en la subpestaña. Los nodos para los que se han encontrado condiciones que son "True" se ejecutarán una vez evaluadas todas las condiciones.
- **Ejecutar uno por uno**. Sólo está disponible si se selecciona **Evaluar todo en orden**. Si se selecciona significa que si la condición se evalúa como "True", el nodo asociado con esa condición se ejecutará antes de que se evalúe la siguiente condición.
- **Evaluar hasta primer acierto**. Si se selecciona, significa que solo se ejecutará el primer nodo cuya evaluación de las condiciones devuelva el valor "True".

Ejecutar e interrumpir scripts

Existen diversas formas de ejecutar scripts. Por ejemplo, en el script de ruta o en el cuadro de diálogo del script, el botón "Ejecutar este script" ejecuta el script completo:

Figura 1. Botón Ejecutar este script

El botón "Ejecutar líneas seleccionadas" ejecuta una única línea, o un bloque de líneas adyacentes, que ha seleccionado en el script:

Figura 2. Botón Ejecutar líneas seleccionadas

Un script se puede ejecutar mediante cualquiera de los siguientes métodos:

- Pulse en el botón "Ejecutar este script" o "Ejecutar líneas seleccionadas" dentro de un script de ruta o un cuadro de diálogo de script.
- Ejecutando una ruta donde **Ejecutar este script** esté establecido como el método de ejecución predeterminado.
- Utilizando la marca `-execute` al inicio en modo interactivo. Para obtener más información, consulte el tema “Utilización de argumentos de la línea de comandos” en la página 65.

Nota: Un script de supernodo se ejecuta cuando el supernodo se ejecuta, siempre que haya seleccionado **Ejecutar este script** en el recuadro de diálogo Script de supernodo.

Interrumpir ejecución de script

En el cuadro de diálogo de scripts de ruta, se activará el botón rojo de detención durante la ejecución de scripts. Pulsando este botón, puede abandonar la ejecución del script y de cualquier ruta actual.

Buscar y reemplazar

El cuadro de diálogo Buscar/reemplazar está disponible en lugares donde edita texto de script o de expresión, incluido el editor de scripts, el generador de expresiones CLEM o cuando define una plantilla en el nodo Informe. Al editar texto en cualquiera de estas áreas, pulse `Ctrl+F` para acceder al cuadro de diálogo, asegurándose de que el cursor está centrado en un área de texto. Por ejemplo, si trabaja en un nodo Rellenar, puede acceder al cuadro de diálogo desde cualquiera de las áreas de texto de la pestaña Configuración o desde el campo de texto del generador de expresiones.

1. Con el cursor en un área de texto, pulse Ctrl+F para acceder al cuadro de diálogo Buscar/reemplazar.
2. Introduzca el texto que desee buscar o selecciónelo de la lista desplegable de elementos buscados recientemente.
3. Introduzca el texto de reemplazo, si lo tiene.
4. Pulse **Buscar siguiente** para iniciar la búsqueda.
5. Pulse **Reemplazar** para reemplazar la sección actual o en **Reemplazar todos** para actualizar todas las instancias o sólo las seleccionadas.
6. El cuadro de diálogo se cierra después de cada operación. Pulse F3 desde cualquier área de texto para repetir la operación de búsqueda más reciente o pulse Ctrl+F para volver a acceder al cuadro de diálogo.

Opciones de búsqueda

Coincidir mayúsculas y minúsculas. Especifica si la operación de búsqueda hace distinción entre mayúsculas y minúsculas; por ejemplo, si *mivar* es igual que *miVar*. El texto de reemplazo siempre se introduce exactamente como se ha introducido, independientemente de este ajuste.

Sólo palabras completas. Especifica si la operación de búsqueda tiene en cuenta el texto incluido dentro de las palabras. Por ejemplo, si se selecciona, la búsqueda de *fuego* no será igual que *cortafuegos* ni que *corta-fuegos*.

Expresiones regulares. Especifica si se utiliza la sintaxis de expresiones regulares (consulte la sección siguiente). Si está seleccionado, la opción **Sólo palabras completas** está desactivada y su valor se ignora.

Sólo texto seleccionado. Controla el ámbito de la búsqueda al utilizar la opción **Reemplazar todos**.

Sintaxis de expresiones regulares

Las expresiones regulares le permiten buscar caracteres especiales como caracteres de tabulador o de nueva línea, clases o rangos de caracteres como de la *a* a la *d*, cualquier dígito o no dígito y límites como el principio o el final de una línea. Se admiten los siguientes tipos de expresiones.

Tabla 1. Coincidencias de caracteres

Caracteres	**Coincidencias**
x	El carácter x
\\	El carácter de barra inclinada invertida
\0n	El carácter con valor octal 0n (0 <= n <= 7)
\0nn	El carácter con valor octal 0nn (0 <= n <= 7)
\0mnn	El carácter con valor octal 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	El carácter con valor hexadecimal 0xhh
\uhhhh	El carácter con valor hexadecimal 0xhhhh
\t	El carácter de tabulador ('\u0009')
\n	El carácter de nueva línea (avance de línea) ('\u000A')
\r	El carácter de retorno de carro ('\u000D')
\f	El carácter de avance de página ('\u000C')
\a	El carácter de alerta (campana) ('\u0007')
\e	El carácter de escape ('\u001B')
\cx	El carácter de control correspondiente a x

Tabla 2. Coincidencia de clases de caracteres

Clases de caracteres	**Coincidencias**
[abc]	a, b o c (clase simple)
[^abc]	Cualquier carácter excepto a, b o c (resta)
[a-zA-Z]	De la a a la z o de la A a la Z, ambas inclusive (rango)
[a-d[m-p]]	De la a a la d o de la m a la p (unión). Esto también puede especificarse como [a-dm-p].
[a-z&&[def]]	De la a a la z y d, e o f (intersección)
[a-z&&[^bc]]	De la a a la z, excepto b y c (resta). Esto también puede especificarse como [ad-z].
[a-z&&[^m-p]]	De la a a la z y no de la m a la p (resta). Esto también puede especificarse como [a-lq-z].

Tabla 3. Clases de caracteres predefinidas	
Clases de caracteres predefinidas	**Coincidencias**
.	Cualquier carácter (puede o no coincidir con los terminadores de línea)
\d	Cualquier dígito: [0-9]
\D	Un carácter que no es un dígito: [^0-9]
\s	Un carácter de espacio en blanco: [\t\n\x0B\f\r]
\S	Un carácter de espacio en blanco: [^\s]
\w	Un carácter de palabra: [a-zA-Z_0-9]
\W	Un carácter que no sea de palabra : [^\w]

Tabla 4. Coincidencias de límite	
Reconocedores de límite	**Coincidencias**
^	El comienzo de una línea
$	El final de una línea
\b	Un límite alfabético
\B	Un límite no alfabético
\A	El comienzo de la entrada
\Z	El final de la entrada pero para el terminador final, si lo hay
\z	El final de la entrada

Capítulo 2. Language de scripts

Visión general de lenguaje de script

El recurso de scripts para IBM SPSS Modeler le permite crear scripts que funcionen en la interfaz de usuario de SPSS Modeler, manipular los objetos de salida y ejecutar la sintaxis de comandos. Puede ejecutar scripts directamente desde SPSS Modeler.

Los scripts de IBM SPSS Modeler están escritos en el lenguaje de script Python. La implementación Java de Python que IBM SPSS Modeler se denomina Jython. El lenguaje de script consta de las siguientes características:

- Un formato para hacer referencia a nodos, rutas, proyectos, resultados y otros objetos de IBM SPSS Modeler.
- Un conjunto de instrucciones o comandos de scripts que se puede utilizar para manipular tales objetos.
- Un lenguaje de expresión de script para establecer los valores de las variables, los parámetros y otros objetos.
- Compatibilidad con comentarios, continuaciones y bloques de texto literal.

Las secciones siguientes describen el lenguaje de scripts Python, la implementación de Jython por parte de Python y la sintaxis básica para empezar a crear scripts en IBM SPSS Modeler. Las secciones siguientes recogen información sobre comandos y propiedades específicas.

Python y Jython

Jython es una implementación del lenguaje de scripts Python, escrito en el lenguaje Java e integrado con la plataforma Java. Python es un potente lenguaje de script orientado a objetos. Jython es útil porque proporciona las características de productividad de un lenguaje de script maduro y, a diferencia de Python, se ejecuta en cualquier entorno que soporte una máquina virtual Java (JVM). Esto significa que las bibliotecas Java de la máquina virtual Java están disponibles para utilizarlas cuando se escriben programas. Con Jython, puede beneficiarse de esta diferencia y utilizar la sintaxis y la mayoría de las características del lenguaje Python.

Como lenguaje de script, Python (y su implementación Jython) es fácil de aprender y ofrece una codificación eficaz con la estructura mínima necesaria para crear un programa de ejecución. El código se puede entrar de forma interactiva, es decir, una línea cada vez. Python es un lenguaje de script interpretado; no hay ningún paso de precompilación, tal como existe en Java. Los programas de Python simplemente son archivos de texto que se interpretan a medida que se entran (después de analizar los errores de sintaxis). Las expresiones simples, tales como los valores definidos, y también las acciones más complejas, tales como las definiciones de función, se ejecutan y están disponibles para su uso de forma inmediata. Los cambios realizados en el código se pueden probar fácilmente. Sin embargo, la interpretación del script tiene algunas desventajas. Por ejemplo, utilizar una variable no definida no es un error del compilador, por lo tanto solo se detecta si (y cuando) se ejecuta la sentencia en la que se utiliza la variable. En este caso, se puede editar y ejecutar el programa para depurar el error.

Python lo ve todo como un objeto, incluidos todos los datos y el código. Por lo tanto, puede manipular estos objetos con líneas de código. Algunos tipos de selección, tales como los números y cadenas, se consideran valores y no objetos, lo cual resulta más práctico, y Python da soporte a todo ello. Se da soporte a un valor nulo. Este valor nulo tiene el nombre reservado de `None`.

Para obtener una introducción más detallada del lenguaje de script Python y Jython y algunos scripts de ejemplo, consulte el tema http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html y el tema http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html.

Scripts de Python

Esta guía del lenguaje de script de Python es una introducción a los componentes que tienen más probabilidad de ser utilizados cuando se ejecutan scripts en IBM SPSS Modeler, incluidos conceptos y principios básicos de programación. Le proporcionará los conocimientos suficientes para comenzar a desarrollar sus propios scripts Python y utilizarlos en IBM SPSS Modeler.

Operaciones

La asignación se realiza utilizando un signo igual (=). Por ejemplo, para asignar el valor "3" a una variable denominada "x", utilizaría la siguiente sentencia:

```
x = 3
```

El signo igual también se utiliza para asignar datos de tipo de cadena a una variable. Por ejemplo, para asignar el valor "a string value" a la variable "y" utilice la sentencia siguiente:

```
y = "a string value"
```

La tabla siguiente enumera algunas de las operaciones numéricas y de comparación utilizadas con más frecuencia y sus descripciones.

Tabla 5. Operaciones numéricas y de comparación comunes

Operación	**Descripción**
`x < y`	¿Es x menor que y?
`x > y`	¿Es x mayor que y?
`x <= y`	¿Es x menor que o igual a y?
`x >= y`	¿Es x mayor que o igual a y?
`x == y`	¿Es x igual a y?
`x != y`	¿Es x no igual a y?
`x <> y`	¿Es x no igual a y?
`x + y`	Sumar y a x
`x - y`	Restar y de x
`x * y`	Multiplicar x por y
`x / y`	Dividir x por y
`x ** y`	Elevar x a la potencia de y

Listas

Las listas son secuencias de elementos. Una lista puede contener cualquier número de elementos, y los elementos de la lista pueden ser cualquier tipo de objeto. Las listas también se pueden considerar como matrices. El número de elementos de una lista puede aumentar o disminuir a medida que se añaden, eliminan o sustituyen elementos.

Ejemplos

`[]`	Cualquier lista vacía.
`[1]`	Una lista con un solo elemento, un entero.
`["Mike", 10, "Don", 20]`	Una lista con cuatro elementos, dos elementos de cadena y dos elementos de entero.

`[[],[7],[8,9]]`	Una lista de listas. Cada sublista es una lista vacía o una lista de elementos de enteros.
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	Una lista de enteros. Este ejemplo muestra el uso de variables y expresiones.

Puede asignar una lista a una variable, por ejemplo:

```
mylist1 = ["one", "two", "three"]
```

A continuación, puede acceder a los elementos específicos de la lista, por ejemplo:

```
mylist[0]
```

Esto genera el resultado siguiente:

```
one
```

El número entre corchetes (`[]`) se considera un *index* y hace referencia a un elemento concreto de la lista. Los elementos de una lista se indexan a partir de 0.

También puede seleccionar un rango de elementos de una lista; esto se denomina *porciones*. Por ejemplo, `x[1:3]` selecciona el segundo y tercer elemento de x. El índice final es uno pasado la selección.

Cadenas

Una *cadena* es una secuencia inmutable de caracteres que se trata como un valor. Las cadenas dan soporte a todas las funciones de secuencias inmutables y operadores que generan como resultado una nueva serie. Por ejemplo, `"abcdef"[1:4]` da como resultado la salida `"bcd"`.

En Python, los caracteres se representan mediante cadenas de caracteres de longitud uno.

Los literales de cadenas se definen mediante comillas simples o triples. Las cadenas definidas mediante comillas simples no pueden abarcar líneas, mientras que las series definidas mediante comillas triples sí que pueden. Una serie se puede incluir entre comillas simples (') o comillas dobles ("). Un carácter de cita puede contener el otro carácter de comillas sin escape o el carácter de comillas que se ha escapado, que va precedido por el carácter de barra inclinada invertida (\).

Ejemplos

```
"This is a string"
'This is also a string'
"It's a string"
'This book is called "Python Scripting and Automation Guide".'
"This is an escape quote (\") in a quoted string"
```

El analizador de Python automáticamente concatena varias cadenas separadas por un espacio en blanco. Esto facilita la entrada se cadenas largas y la combinación de tipos de comillas en una sola cadena, por ejemplo:

```
"This string uses ' and " 'that string uses "."
```

Esto resulta en la siguiente salida:

```
This string uses ' and that string uses ".
```

Las cadenas dan soporte a varios métodos útiles. Algunos de estos métodos se proporcionan en la tabla siguiente.

Tabla 6. Métodos de serie

Método	**Uso**
`s.capitalize()`	Mayúscula inicial s
`s.count(ss {,start {,end}})`	Recuento de apariciones de `ss` en `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Probar si s comienza por `str` Probar si s acaba en `str`
`s.expandtabs({size})`	Sustituir tabulaciones por espacios, el valor predeterminado de `size` es 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Busca el primer índice de `str` en s; si no se encuentra, el resultado es -1. `rfind` busca de derecha a izquierda.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Busca el primer índice de `str` en s; si no se encuentra: se genera `ValueError`. `rindex` busca de derecha a izquierda.
`s.isalnum`	Probar si la cadena es alfanumérica
`s.isalpha`	Probar si la cadena es alfabética
`s.isnum`	Probar si la cadena es numérica
`s.isupper`	Probar si la cadena está toda en mayúsculas
`s.islower`	Probar si la cadena está toda en minúsculas
`s.isspace`	Probar si la cadena está toda en espacios en blanco
`s.istitle`	Probar si la cadena es una secuencia de cadenas alfanuméricas con mayúscula inicial

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Convertir todo a minúsculas
	Convertir todo a mayúsculas
	Convertir de mayúsculas a minúsculas o viceversa
	Convertir todo a mayúsculas o minúsculas del título

`s.join(seq)`	Unir las cadenas de `seq` con s como separador
`s.splitlines({keep})`	Dividir s en líneas, si keep es `true`, mantener las nuevas líneas
`s.split({sep {, max}})`	Dividir s en "palabras" utilizando `sep` (el valor predeterminado de `sep` es un espacio en blanco) para un máximo de `max` veces

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Justificar la cadena a la izquierda con un ancho de campo de `width`
	Justificar la cadena a la derecha con un ancho de campo de `width`
	Justificar la cadena al centro con un ancho de campo de `width`
	Rellenar con 0.

Tabla 6. Métodos de serie (continuación)	
Método	**Uso**

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Eliminar espacios en blanco iniciales
	Eliminar espacios en blanco de cola
	Eliminar espacios en blanco iniciales y de cola

`s.translate(str {,delc})`	Traducir `s` utilizando la tabla, después de eliminar cualquier carácter en `delc`. `str` debe ser una serie con longitud == 256.
`s.replace(old, new {, max})`	Sustituye todas las apariciones de `max` de la cadena `old` por la cadena `new`

Observaciones

Las observaciones son comentarios que se introducen por el signo de almohadilla (o hash) (#). Todo el texto que sigue el signo de la libra en la misma línea se considera parte de la observación y se ignora. Un comentario puede comenzar en cualquier columna. El ejemplo siguiente muestra el uso de los comentarios:

```
#The HelloWorld application is one of the most simple
print 'Hello World' # print the Hello World line
```

Sintaxis de las sentencias

La sintaxis de las sentencias para Python es muy sencilla. En general, cada línea de origen es una sola sentencia. A excepción de las sentencias `expression` y `assignment`, cada sentencia se introduce mediante un nombre de palabra clave, tal como `if` o `for`. Las líneas en blanco o las líneas de comentarios se pueden insertar en cualquier lugar entre cualquier sentencia del código. Si existe más de una sentencia en una línea, cada sentencia debe estar separada por un signo de punto y coma (`;`).

Las sentencias muy largas puede continuar en más de una línea. En este caso, la sentencia que ha de continuar en la línea siguiente debe acabar con una barra invertida (\), por ejemplo:

```
x = "A looooooooooooooooooong string" + \
     "another looooooooooooooooooong string"
```

Cuando una estructura está encerrada entre paréntesis (`()`), corchetes (`[]`) o llaves (`{}`), la sentencia puede continuar en una línea nueva después de cualquier coma, sin tener que insertar una barra invertida, por ejemplo:

```
x = (1, 2, 3, "hello",
       "goodbye", 4, 5, 6)
```

Identificadores

Los identificadores se utilizan para el nombre de las variables, funciones, clases y palabras clave. Los identificadores pueden tener cualquier longitud, pero deben empezar por un carácter alfabético en mayúsculas o minúsculas o por el carácter de subrayado (_). Los nombres que empiezan por un subrayado generalmente están reservados para nombres internos o privados. Después del primer carácter, el identificador puede contener cualquier número y combinación de caracteres alfabéticos, los números del 0-9, y el carácter de subrayado.

Existen algunas palabras reservadas en Jython que no se pueden utilizar para el nombre de variables, funciones o clases. Estas palabras entran en las siguientes categorías:

- **Introducciones de sentencias:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` y `while`

- **Introducciones de parámetros:** as, import y in
- **Operadores:** and, in, is, lambda, not y or

El uso incorrecto de palabras claves suele generar SyntaxError.

Bloques de código

Bloques de código son grupos de sentencias que se utilizan donde se esperan sentencias individuales. Los bloques de código pueden seguir a cualquiera de las sentencias siguientes: if, elif, else, for, while, try, except, def y class. Estas sentencias introducen el bloque de código con el carácter de dos puntos (:), por ejemplo:

```
if x == 1:
    y = 2
    z = 3
elif:
    y = 4
    z = 5
```

Se utiliza la indentación para delimitar los bloques de código (en lugar de las llaves que se utilizan en Java). Todas las líneas de un bloque han de indentarse en la misma posición. Esto es debido a que un cambio en la indentación indica el final de un bloque de código. Normalmente la indentación es de cuatro espacios por nivel. Se recomienda utilizar espacios para la indentación, en lugar de tabulaciones. No se deben combinar espacios y tabulaciones. Las líneas del bloque de un módulo situado más al extremo deben comenzar en la columna uno, de lo contrario, se genera el error SyntaxError.

Las sentencias que componen un bloque de código (y siguen el signo de dos puntos) también deben estar en una sola línea, separadas por signos de punto y coma, por ejemplo:

```
if x == 1: y = 2; z = 3;
```

Pasar argumentos a un script

Pasar argumentos a un script puede resultar útil para poder utilizar un script reiteradamente sin modificarlo. Los argumentos se pasan en la línea de comandos como valores de la lista sys.argv. El número de valores que se pasan se puede obtener mediante el comando len(sys.argv). Por ejemplo:

```
import sys
print "test1"
print sys.argv[0]
print sys.argv[1]
print len(sys.argv)
```

En este ejemplo, el comando import importa toda la clase sys, por lo que se pueden utilizar los métodos existentes para esta clase, tales como argv.

El script de este ejemplo se puede invocar utilizando la línea siguiente:

```
/u/mjloos/test1 mike don
```

Esto genera el resultado siguiente:

```
/u/mjloos/test1 mike don
test1
mike
don
3
```

Ejemplos

La palabra clave `print` imprime los argumentos situados inmediatamente después de la misma. Si la sentencia va seguida de una coma, no se incluye una línea nueva en los resultados. Por ejemplo:

```
print "This demonstrates the use of a",
print " comma at the end of a print statement."
```

Esto genera el resultado siguiente:

```
This demonstrates the use of a comma at the end of a print statement.
```

La sentencia `for` se utiliza para la iteración por un bloque de código. Por ejemplo:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    print lv
    continue
```

En este ejemplo, se asignan tres cadenas a la lista `mylist1`. Los elementos de la lista se imprimen a continuación, con un elemento de cada línea. Esto genera el resultado siguiente:

```
one
two
three
```

En este ejemplo, el iterador `lv` toma el valor de cada elemento de la lista `mylist1` por orden, mientras el bucle for implementa el bloque de código de cada elemento. Un iterador puede ser cualquier identificador válido de cualquier longitud.

La sentencia `if` es una sentencia condicional. Evalúa la condición y devuelve true o false, en función del resultado de la evaluación. Por ejemplo:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    if lv == "two"
        print "The value of lv is ", lv
    else
        print "The value of lv is not two, but ", lv
    continue
```

En este ejemplo, se evalúa el valor del iterador `lv`. Si el valor de `lv` es `two` se devuelve una cadena diferente a la cadena que se devuelve si el valor de `lv` no es `two`. Esto resulta en la siguiente salida:

```
The value of lv is not two, but one
The value of lv is two
The value of lv is not two, but three
```

Métodos matemáticos

Desde el módulo `math` puede acceder a métodos matemáticos útiles. Algunos de estos métodos se proporcionan en la tabla siguiente. A menos que se especifique lo contrario, todos los valores se devuelven como valores flotantes.

Tabla 7. Métodos matemáticos

Método	**Uso**
`math.ceil(x)`	Devuelve el punto más alto de x como un valor flotante, que es el entero más pequeño mayor o igual a x
`math.copysign(x, y)`	Devolver x con el signo de y. `copysign(1, -0.0)` devuelve `-1`

Tabla 7. Métodos matemáticos (continuación)	
Método	**Uso**
`math.fabs(x)`	Devuelve el valor absoluto de x
`math.factorial(x)`	Devuelve el factor de x. Si x es negativo o no es un entero, se genera `ValueError`.
`math.floor(x)`	Devuelve el punto más bajo de x como un valor flotante, que es el entero más alto menor o igual a x
`math.frexp(x)`	Devolver la mantisa (`m`) y el exponente (`e`) de x como el par `(m, e)`. `m` es un valor flotante y e es un entero, tal como `x == m * 2**e` exactamente. Si x es cero, devuelve `(0.0, 0)`, de lo contrario `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Devuelve una suma de coma flotante precisa de los valores de `iterable`
`math.isinf(x)`	Comprueba si el valor flotante x es positivo o negativo infinito
`math.isnan(x)`	Comprueba si el valor flotante x es `NaN` (no es un número)
`math.ldexp(x, i)`	Devuelve `x * (2**i)`. Esencialmente es la función inversa de `frexp`.
`math.modf(x)`	Devuelve las partes de fracción y entero de x. Los dos resultados llevan el signo de x y son flotantes.
`math.trunc(x)`	Devuelve el valor `Real` de x, que se ha truncado en un `Integral`.
`math.exp(x)`	Devuelve `e**x`
`math.log(x[, base])`	Devuelve el logaritmo de x para el valor dado de `base`. Si no se especifica `base`, se devuelve el logaritmo natural de x.
`math.log1p(x)`	Devolver el logaritmo natural de `1+x (base e)`
`math.log10(x)`	Devuelve el logaritmo de base-10 de x
`math.pow(x, y)`	Devolver x elevado a la potencia y. `pow(1.0, x)` y `pow(x, 0.0)` siempre devuelven `1`, incluso cuando x es cero o `NaN`.
`math.sqrt(x)`	Devuelve la raíz cuadrada de x

Además de las funciones matemáticas, hay algunos métodos trigonométricos útiles. Estos métodos se muestran en la siguiente tabla.

Tabla 8. Métodos trigonométricos	
Método	**Uso**
`math.acos(x)`	Devuelve el arco coseno de x en radianes
`math.asin(x)`	Devuelve el arcoseno de x en radianes
`math.atan(x)`	Devuelve el arco tangente de x en radianes

Tabla 8. Métodos trigonométricos (continuación)	
Método	**Uso**
`math.atan2(y, x)`	Devuelve `atan(y / x)` en radianes.
`math.cos(x)`	Devuelve el coseno de x en radianes.
`math.hypot(x, y)`	Devolver la norma euclidiana `sqrt(x*x + y*y)`. Esta es la longitud del vector desde el origen al punto `(x, y)`.
`math.sin(x)`	Devuelve el seno de x en radianes
`math.tan(x)`	Devuelve la tangente de x en radianes
`math.degrees(x)`	Convierte el ángulo x de radianes a grados
`math.radians(x)`	Convierte el ángulo x de grados a radianes
`math.acosh(x)`	Devuelve el coseno hiperbólico inverso de x
`math.asinh(x)`	Devuelve el seno hiperbólico inverso de x
`math.atanh(x)`	Devuelve la tangente hiperbólica inversa de x
`math.cosh(x)`	Devuelve el coseno hiperbólico de x
`math.sinh(x)`	Devuelve el coseno hiperbólico de x
`math.tanh(x)`	Devuelve la tangente hiperbólica de x

También hay constantes matemáticas. El valor de `math.pi` es la constante matemática pi. El valor de `math.e` es la constante matemática e.

Utilización de caracteres no ASCII

Para utilizar caracteres no ASCII, Python requiere la codificación y descodificación explícitas de las cadenas en Unicode. En IBM SPSS Modeler, se presupone que los scripts Python están codificados UTF-8, la cual es una codificación Unicode estándar que da soporte a caracteres no ASCII. El script siguiente se compilará porque SPSS Modeler ha establecido el compilador Python en UTF-8.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "テストノード", 96, 64)
```

Sin embargo, el nodo resultante tendrá una etiqueta incorrecta.

Figura 3. Etiqueta del nodo que contiene caracteres no ASCII, visualiza incorrectamente

La etiqueta es incorrecta porque Python ha convertido el propio literal de serie en una cadena ASCII.

Python permite que los literales de cadenas Unicode se especifiquen añadiendo un prefijo con el carácter u antes del literal de cadena:

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", u"テストノード", 96, 64)
```

Esto crear una cadena Unicode y la etiqueta aparecerá correctamente.

Figura 4. Etiqueta de nodo que contiene caracteres no ASCII, visualizados correctamente

La utilización de Python y Unicode es un tema de gran volumen que no entra dentro del ámbito de este documento. Existen muchas publicaciones y recursos en línea disponibles que describen detalladamente este tema.

Programación orientada a objetos

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientada a objetos disminuye los errores y promociona la reutilización del código. Python es un lenguaje orientado a objetos. Los objetos definidos en Python tienen las características siguientes:

- **Identidad.** Cada objeto debe ser distinguido y ello debe poder demostrarse mediante pruebas. Las pruebas `is` e `is not` existen para este fin.
- **Estado.** Cada objeto debe ser capaz de almacenar el estado. Para este fin, existen atributos, tales como variables de instancias y campos.
- **Comportamiento.** Cada objeto debe ser capaz de manipular su estado. Para este fin existen métodos.

Python incluye las características siguientes para dar soporte a la programación orientada a objetos:

- **Creación de objetos basada en clases.** Las clases son plantillas para la creación de objetos. Los objetos son estructuras de datos con el comportamiento asociado.
- **Herencia con polimorfismo.** Python da soporte a la herencia individual y múltiple. Todos los métodos de instancias de Python son polimórficos y se pueden alterar temporalmente mediante subclases.
- **Encapsulación con ocultación de datos.** Python permite ocultar los atributos. Cuando se ocultan los atributos, se puede acceder a los mismos desde fuera de la clase únicamente mediante los métodos de la clase. Las clases implementan métodos para modificar los datos.

Definición de una clase

En una clase Python, se pueden definir tanto variables como métodos. A diferencia de Java, en Python puede definir cualquier número de clases públicas por archivo de origen (o *module*). Por lo tanto, un módulo en Python puede considerarse similar a un paquete en Java.

En Python, las clases se definen utilizando la sentencia `class`. La sentencia `class` tiene el formato siguiente:

```
class name (superclasses): statement
```

o

```
class name (superclasses):
    assignment
    .
    .
    function
    .
    .
```

Cuando define una clase tiene la opción de proporcionar cero o más sentencias *assignment*. Estos crean atributos de clase que comparten todas las instancias de la clase. Puede proporcionar cero o más definiciones de *function*. Estas definiciones de función crean métodos. La lista de superclases es opcional.

El nombre de clase debe ser exclusivo en el mismo ámbito, esto es, dentro de un módulo, función o clase. Puede definir varias variables para que hagan referencia a la misma clase.

Creación de una instancia de clase

Las clases se utilizan para contener (o compartir) los atributos de clase o para crear instancias de clase. Para crear una instancia de una clase, debe llamar a la clase como si fuera una función. Por ejemplo, considere las clases siguientes:

```
class MyClass:
    pass
```

Aquí, se utiliza la sentencia `pass` por que se requiere una sentencia para completar la clase, pero no se requiere ninguna acción de programación.

La sentencia siguiente crea una instancia de la clase `MyClass`:

```
x = MyClass()
```

Añadir atributos a una instancia de clase

A diferencia de Java, en Python los clientes pueden añadir atributos a una instancia de una clase. Solo se cambia la instancia. Por ejemplo, para añadir atributos a una instancia de `x`, establezca valores nuevos en dicha instancia:

```
x.attr1 = 1
x.attr2 = 2
    .
    .
x.attrN = n
```

Definición de atributos de clase y métodos

Cualquier variable enlazada a una clase es un *atributo de clase*. Cualquier función definida en una clase es un *método*. Los métodos reciben como primer argumento una instancia de la clase, que convencionalmente se denomina `self`. Por ejemplo, para definir algunos atributos de clase y métodos, puede entrar el siguiente código:

```
class MyClass
    attr1 = 10        #class attributes
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #reference the class attribute

    def method2(self):
        print MyClass.attr2   #reference the class attribute

    def method3(self, text):
        self.text = text        #instance attribute
        print text, self.text   #print my argument and my attribute

    method4 = method3   #make an alias for method3
```

Dentro de una clase, debe cualificar todas las referencias a los atributos de clase con el nombre de clase; por ejemplo, `MyClass.attr1`. Todas las referencias a los atributos de la instancia deben cualificarse con la variable `self`, por ejemplo, `self.text`. Fuera de la clase, debe cualificar todas las referencias a los atributos de clase con el nombre de clase (por ejemplo, `MyClass.attr1`) o con una instancia de la clase (por ejemplo, `x.attr1`, donde `x` es una instancia de la clase). Fuera de la clase, todas las referencias a las variables de la instancia deben cualificarse con una instancia de la clase, por ejemplo, `x.text`.

Variables ocultas

Los datos se pueden ocultar creando variables *privadas*. Solo la propia clase puede acceder a las variables privadas. Si declara nombres con el formato `__xxx` o `__xxx_yyy`, estos es, con dos signos de subrayado antes de los nombres, el analizador Python automáticamente añadirá el nombre de clase al nombre declarado y creará las variables ocultas, por ejemplo:

```
class MyClass:
    __attr = 10   #private class attribute

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #private attribute
```

A diferencia de Java, en Python todas las referencias a variables de instancia deben estar calificadas con `self`; no existe un uso implícito de `this`.

Heredado

La posibilidad de herencia de las clases es fundamental en la programación orientada a objetos. Python da soporte a la herencia individual y múltiple. *Herencia individual* significa que solo puede haber una superclase. *Herencia múltiple* significa que puede haber más de una superclase.

La herencia se implementa generando subclases de otras clases. Cualquier número de clases Python pueden ser superclases. En la implementación de Jython en Python, solo se puede heredar directa o indirectamente de una clase Java. No es necesario suministrar una superclase.

Cualquier atributo o método de una superclase también está en cualquier subclase y lo puede utilizar la propia clase o cualquier cliente, siempre que el atributo o método no esté oculto. Se puede utilizar cualquier instancia de una subclase; esto se denomina *polimorfismo*. Estas características permiten la reutilización y facilitan la extensión.

Ejemplo

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

class Class4(Class3, Class2): pass     #multiple inheritance
```

Capítulo 3. Scripts de IBM SPSS Modeler

Tipos de scripts

En IBM SPSS Modeler existen tres tipos de scripts:

- Los *scripts de ruta* se utilizan para controlar la ejecución de una sola ruta y se almacenan dentro de la ruta.
- Los *scripts Supernodo* se utilizan para controlar el comportamiento de los supernodos.
- Los *scripts autónomos o de sesión* se pueden utilizar para coordinar la ejecución entre un número de rutas diferentes.

Existen diferentes métodos disponibles que puede utilizar en scripts en IBM SPSS Modeler lo que le permite acceder a una amplia gama de funciones de SPSS Modeler. Estos métodos se utilizan también en Capítulo 4, “API de scripts”, en la página 39 para crear funciones más avanzadas.

Rutas, rutas de supernodo y diagramas

La mayoría de las veces, el término *ruta* significa lo mismo independientemente de que se trate de una ruta cargada de un archivo o utilizada dentro de un supernodo. En general significa una colección de nodos conectados entre sí que puede ejecutarse. Sin embargo, en la creación de scripts no todas las operaciones se soportan en todos los sitios, lo que significa que el autor de un script deberá tener en cuenta qué variante de ruta está utilizando.

Rutas

Una ruta es el principal tipo de documento de IBM SPSS Modeler. Se puede guardar, cargar, editar y ejecutar. Las rutas también pueden tener parámetros, valores globales, un script y otra información asociada a ellos.

Rutas de Supernodo

Una *ruta de Supernodo* es el tipo de ruta que se utiliza en un Supernodo. Al igual que una ruta normal, contiene nodos enlazados entre sí. Las rutas de Supernodo tienen una serie de diferencias respecto de una ruta normal.

- Los parámetros y scripts están asociados al Supernodo propietario de la ruta de Supernodo en lugar de a la propia ruta de Supernodo.
- Las rutas de Supernodo tienen nodos de conector de entrada y salida adicionales dependiendo del tipo de Supernodo. Estos nodos de conector se utilizan en los flujos de información entrantes y salientes de la ruta de Supernodo y se crean automáticamente cuando se crea el Supernodo.

Diagramas

El término *diagrama* abarca las funciones soportadas en rutas normales y en rutas de supernodo como, por ejemplo, la adición y eliminación de nodos y la modificación de conexiones entre nodos.

Ejecución de una ruta

El ejemplo siguiente ejecuta todos los nodos ejecutables en la ruta y es el tipo de script de ruta más sencillo:

```
modeler.script.stream().runAll(None)
```

El ejemplo siguiente también se ejecuta todos los nodos ejecutables de la ruta:

```
stream = modeler.script.stream()
stream.runAll(None)
```

En este ejemplo, la ruta se almacena en una variable denominada `stream`. Almacenar la ruta en una variable resulta útil ya que un script se utiliza generalmente para modificar la ruta o los nodos contenidos en una ruta. Si se crea una variable que almacena los resultados de la ruta, el script resultará más conciso.

El contexto de los scripts

El módulo `modeler.script` proporciona el contexto en el que se ejecuta un script. El módulo se importa automáticamente a un script de SPSS Modeler durante la ejecución. El módulo define cuatro funciones que proporcionan un script con acceso a su entorno de ejecución:

- La función `session()` devuelve la sesión para el script. La sesión define información, tal como el entorno local y el proceso de fondo de SPSS Modeler (ya sea un proceso local o un proceso de SPSS Modeler Server conectado a la red) que se está utilizando para ejecutar rutas.
- La función `stream()` se puede utilizar con la ruta y los scripts Supernodo. Esta función devuelve la ruta que es propietaria del script de ruta o el script Supernodo que se está ejecutando.
- La función `diagram()` se puede utilizar con los scripts Supernodo. Esta función devuelve el diagrama dentro del Supernodo. Para otros tipos de script, esta función devuelve el mismo que la función `stream()`.
- La función `supernode()` se puede utilizar con los scripts Supernodo. Esta función devuelve el Supernodo propietario del script que se está ejecutando.

En la tabla siguiente se resumen las cuatro funciones y sus resultados.

Tabla 9. Resumen de las funciones de `modeler.script`

Tipo de script	**`session()`**	**`stream()`**	**`diagram()`**	**`supernode()`**
Autónomo	Devuelve una sesión	Devuelve la ruta gestionada actual en el momento en que se invoca el script (por ejemplo, la ruta se pasa con la opción `-stream` de modalidad de proceso por lotes) o None.	Igual que para `stream()`	No aplicable
Flujo	Devuelve una sesión	Devuelve una ruta	Igual que para `stream()`	No aplicable
Supernodo	Devuelve una sesión	Devuelve una ruta	Devuelve una ruta Supernodo	Devuelve un Supernodo

El módulo `modeler.script` también define un modo de finalizar el script con un código de salida. La función `exit(`*`exit-code`*`)` detiene la ejecución del script y devuelve el código de salida de entero suministrado.

Uno de los métodos que se define para una ruta es `runAll(List)`. Este método ejecuta todos los nodos ejecutables. Los modelos o resultados que se generan mediante la ejecución de los nodos se añaden a la lista suministrada.

Es común que la ejecución de ruta genere resultados, tales como modelos, gráficos y otros. Para capturar este resultado, un script puede proporcionar una variable que se inicializa en una lista, por ejemplo:

```
stream = modeler.script.stream()
results = []
stream.runAll(results)
```

Cuando se completa la ejecución, se puede acceder a todos los objetos generados por la ejecución en la lista `results`.

Referencia a nodos existentes

Una ruta suele estar construida previamente con algunos parámetros que se deben modificar antes de ejecutar la ruta. Para modificar estos parámetros se han de realizar las tareas siguientes:

1. Localizar los nodos en la ruta relevante.
2. Cambiar los valores de los nodos o de la ruta (o de ambas cosas).

Buscar nodos

Las rutas proporcionan varios modos de localizar un nodo existente. Estos métodos se resumen en la siguiente tabla.

Tabla 10. Métodos para localizar un nodo existente

Método	Tipo devuelto	Descripción
`s.findAll(type, label)`	Colección	Devuelve una lista de todos los nodos con el tipo y la etiqueta. El tipo o la etiqueta pueden ser None, en cuyo caso se utiliza el otro parámetro.
`s.findAll(filter, recursive)`	Colección	Devuelve una colección de todos los nodos que están aceptados por el filtro especificado. Si el distintivo recursivo es `True`, también se buscan los supernodos contenidos en la ruta especificada.
`s.findByID(id)`	Nodo	Devuelve el nodo con el ID proporcionado o None si no existe dicho nodo. La búsqueda se limita a la ruta actual.
`s.findByType(type, label)`	Nodo	Devuelve el nodo con el tipo, etiqueta o ambas cosas. El tipo o el nombre pueden ser None, en cuyo caso se utiliza el otro parámetro. Si varios nodos dan como resultado una coincidencia, se elige uno arbitrario y se devuelve. Si ningún nodo da como resultado una coincidencia, se devuelve el valor None.

Tabla 10. Métodos para localizar un nodo existente (continuación)		
Método	**Tipo devuelto**	**Descripción**
`s.findDownstream(fromNodes)`	Colección	Busca en la lista de nodos suministrada y devuelve el conjunto de nodos en sentido descendente de los nodos suministrados. La lista devuelta incluye los nodos proporcionados originalmente.
`s.findUpstream(fromNodes)`	Colección	Busca en la lista de nodos suministrada y devuelve el conjunto de nodos en sentido ascendente de los nodos suministrados. La lista devuelta incluye los nodos proporcionados originalmente.
`s.findProcessorForID(String id, boolean recursive)`	Nodo	Devuelve el nodo con el ID proporcionado o `None` si no existe dicho nodo. Si el distintivo recursivo es `true`, también se buscan los nodos compuestos contenidos en la ruta especificada.

Por ejemplo, si una ruta contiene un nodo Filtro único que el script necesita para acceso, el nodo Filtro se puede encontrar utilizando el siguiente script:

```
stream = modeler.script.stream()
node = stream.findByType("filter", None)
...
```

Como alternativa, si se conoce el ID del nodo (tal como se muestra en la pestaña Anotaciones del cuadro de diálogo del nodo) se puede utilizar el ID para buscar el nodo, por ejemplo:

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # the filter node ID
...
```

Establecimiento de propiedades

Los nodos, rutas, modelos y resultados tienen propiedades a las que se puede acceder y que, en la mayor parte de los casos, se pueden establecer. Las propiedades suelen utilizarse para modificar el aspecto o el comportamiento del objeto. En la tabla siguiente se resumen los métodos disponibles para establecer y acceder a las propiedades de los objetos.

Tabla 11. Métodos para establecer y acceder a las propiedades de los objetos		
Método	**Tipo devuelto**	**Descripción**
`p.getPropertyValue(propertyName)`	Objeto	Devuelve el valor de la propiedad con nombre, o `None` si no existe tal propiedad.
`p.setPropertyValue(propertyName, value)`	No aplicable	Establece el valor de la propiedad con nombre.

Tabla 11. Métodos para establecer y acceder a las propiedades de los objetos (continuación)		
Método	**Tipo devuelto**	**Descripción**
`p.setPropertyValues(properties)`	No aplicable	Establece los valores de la propiedad con nombre. Cada entrada de la correlación de propiedades consta de una clave que representa el nombre de la propiedad y del valor que debe asignarse a la propiedad.
`p.getKeyedPropertyValue(propertyName, keyName)`	Objeto	Devuelve el valor de la propiedad con nombre, o None si no existe dicha propiedad o clave.
`p.setKeyedPropertyValue(propertyName, keyName, value)`	No aplicable	Establece el valor de la propiedad con nombre y de la clave.

Por ejemplo, si desea establecer el valor de un nodo Archivo variable al comienzo de una ruta, puede utilizar el siguiente script:

```
stream = modeler.script.stream()
node = stream.findByType("variablefile", None)
node.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
...
```

Como alternativa, puede que desee filtrar un campo desde un nodo Filtrar. En este caso, el valor es con clave en el nombre de campo, por ejemplo:

```
stream = modeler.script.stream()
# Locate the filter node ...
node = stream.findByType("filter", None)
# ... and filter out the "Na" field
node.setKeyedPropertyValue("include", "Na", False)
```

Creación de nodos y modificación de rutas

En algunas situaciones, es posible que desee añadir nuevos nodos a rutas existentes. Para añadir nodos a rutas existentes suele ser necesario realizar las tareas siguientes:

1. Crear los nodos.
2. Enlazar los nodos con el flujo de ruta existente.

Creación de nodos

Las rutas proporcionan varios modos de crear nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 12. Métodos para crear nodos		
Método	**Tipo devuelto**	**Descripción**
`s.create(nodeType, name)`	Nodo	Crea un nodo del tipo especificado y lo añade a la ruta especificada.
`s.createAt(nodeType, name, x, y)`	Nodo	Crea un nodo del tipo especificado y lo añade a la ruta especificada en la ubicación especificada. Si x< 0 o y < 0, no se establece la ubicación.

Tabla 12. Métodos para crear nodos (continuación)		
Método	**Tipo devuelto**	**Descripción**
`s.createModelApplier(modelOutput, name)`	Nodo	Crea un nodo aplicador de modelos que se deriva del objeto de resultados del modelo proporcionado.

Por ejemplo, para crear un tipo de nodo nuevo en una ruta puede utilizar el siguiente script:

```
stream = modeler.script.stream()
# Create a new type node
node = stream.create("type", "My Type")
```

Enlazar y desenlazar nodos

Cuando un nodo nuevo se crea dentro de una ruta, debe estar conectado a una ruta de nodos para poder utilizarlo. Las rutas proporciona varios métodos para enlazar y desenlazar nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 13. Métodos para enlazar y desenlazar nodos		
Método	**Tipo devuelto**	**Descripción**
`s.link(source, target)`	No aplicable	Crea un nuevo enlace entre los nodos de origen y destino.
`s.link(source, targets)`	No aplicable	Crea nuevos enlaces entre el nodo de origen y cada nodo de destino de la lista suministrada.
`s.linkBetween(inserted, source, target)`	No aplicable	Conecta un nodo entre dos instancias de otro nodo (los nodos de origen y de destino) y establece la posición del nodo insertado de modo que quede entre ellos. Cualquier enlace directo entre los nodos de origen y de destino se elimina en primer lugar.
`s.linkPath(path)`	No aplicable	Crea una nueva ruta entre instancias de nodo. El primer nodo se enlaza con el segundo, el segundo nodo se enlaza con el tercero y así sucesivamente.
`s.unlink(source, target)`	No aplicable	Elimina cualquier enlace directo entre los nodos de origen y de destino.
`s.unlink(source, targets)`	No aplicable	Elimina los enlaces directos entre el nodo de origen y cada objeto de la lista de destinos.
`s.unlinkPath(path)`	No aplicable	Elimina cualquier ruta que existe entre las instancias del nodo.
`s.disconnect(node)`	No aplicable	Elimina los enlaces entre el nodo suministrado y todos los demás nodos de la ruta especificada.

Tabla 13. Métodos para enlazar y desenlazar nodos (continuación)		
Método	**Tipo devuelto**	**Descripción**
`s.isValidLink(source, target)`	*Boolean*	Devuelve `True` si es válido crear un enlace entre los nodos de origen y de destino especificados. Este método comprueba que ambos objetos pertenezcan a la ruta especificada, que el nodo de origen puede proporcionar un enlace y que el nodo de destino puede recibir un enlace, y que la creación de un enlace de este tipo no creará un circularidad en la ruta.

El script de ejemplo siguiente realiza estas cinco tareas:

1. Crea un nodo de entrada Archivo de variables, un nodo Filtro y un nodo de salida Tabla.
2. Conecta los nodos entre sí.
3. Establece el nombre de archivo del nodo de entrada Archivo de variables.
4. Filtra el campo "Drug" de la salida resultante.
5. Ejecute el nodo Tabla.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "My File Input ", 96, 64)
filternode = stream.createAt("filter", "Filter", 192, 64)
tablenode = stream.createAt("table", "Table", 288, 64)
stream.link(filenode, filternode)
stream.link(filternode, tablenode)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
filternode.setKeyedPropertyValue("include", "Drug", False)
results = []
tablenode.run(results)
```

Importar, sustituir y eliminar nodos

Además de crear y conectar nodos, a menudo es necesario sustituir y suprimir nodos de la ruta. Los métodos que están disponibles para importar, sustituir y suprimir nodos se resumen en la tabla siguiente.

Tabla 14. Métodos para importar, sustituir y suprimir nodos		
Método	**Tipo devuelto**	**Descripción**
`s.replace(originalNode, replacementNode, discardOriginal)`	No aplicable	Sustituye el nodo especificado de la ruta actual. Tanto el nodo original como el nodo de sustitución deben ser propiedad de la ruta especificada.

Tabla 14. Métodos para importar, sustituir y suprimir nodos (continuación)		
Método	**Tipo devuelto**	**Descripción**
`s.insert(source, nodes, newIDs)`	Lista	Inserta copias de los nodos en la lista suministrada. Se presupone que todos los nodos de la lista suministrada se encuentran dentro de la ruta especificada. El distintivo `newIDs` indica si se deben generar nuevos ID para cada nodo o si se debe copiar y utilizar el ID existente. Se presupone que todos los nodos de una ruta tienen ID exclusivos, por lo tanto este distintivo se debe establecer en `True` si la ruta de origen es la misma que la ruta especificada. El método devuelve la lista de nodos recién insertados, en la que el orden de los nodos está sin definir (es decir, el orden no es necesariamente el mismo que el orden de los nodos de la lista de entrada).
`s.delete(node)`	No aplicable	Elimina el nodo especificado de la ruta especificada. El nodo debe ser propiedad de la ruta especificada.
`s.deleteAll(nodes)`	No aplicable	Elimina todos los nodos especificados de la ruta especificada. Todos los nodos de la colección debe pertenecer a la ruta especificada.
`s.clear()`	No aplicable	Elimina todos los nodos de la ruta especificada.

Atravesar los nodos de una ruta

Un requisito común es identificar los nodos que están en un punto de la ruta anterior o posterior a un determinado nodo. La ruta proporciona una serie de métodos que pueden utilizarse para identificar estos nodos. Estos métodos se resumen en la siguiente tabla.

Tabla 15. Métodos para identificar los nodos en sentido ascendente y descendente		
Método	**Tipo devuelto**	**Descripción**
`s.iterator()`	Iterador	Devuelve un iterador de los objetos de nodo que están contenidos en la ruta especificada. Si la ruta se modifica entre llamadas a la función `next()`, el comportamiento del iterador no está definido.

Tabla 15. Métodos para identificar los nodos en sentido ascendente y descendente (continuación)		
Método	**Tipo devuelto**	**Descripción**
`s.predecessorAt(node, index)`	Nodo	Devuelve el predecesor inmediato especificado del nodo suministrado o None si el índice está fuera de los límites.
`s.predecessorCount(node)`	*int*	Devuelve el número de predecesores inmediatos del nodo suministrado.
`s.predecessors(node)`	Lista	Devuelve los predecesores inmediatos del nodo suministrado.
`s.successorAt(node, index)`	Nodo	Devuelve el sucesor inmediato especificado del nodo suministrado o None si el índice está fuera de los límites.
`s.successorCount(node)`	*int*	Devuelve el número de sucesores inmediatos del nodo suministrado.
`s.successors(node)`	Lista	Devuelve los sucesores inmediatos del nodo suministrado.

Borrado o eliminación de elementos

El script heredado da soporte a varios usos del mandato `clear` , por ejemplo:

- `clear outputs` Para suprimir todos los elementos de salida de la paleta del gestor.
- `clear generated palette` Para borrar todos los nugget de modelos de la paleta de modelos.
- `clear stream` Para eliminar el contenido de una ruta.

Los scripts Python admiten un conjunto de funciones parecido; el comando `removeAll()` se utiliza para borrar las rutas, las salidas y los gestores de modelos. Por ejemplo:

- Para borrar el gestor de rutas:

```
session = modeler.script.session()
session.getStreamManager.removeAll()
```

- Para borrar el gestor de salidas:

```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```

- Para borrar el gestor de modelos:

```
session = modeler.script.session()
session.getModelOutputManager().removeAll()
```

Obtener información sobre los nodos

Los nodos entran en diferentes categorías, tales como nodos de importación y exportación de datos, nodos de construcción de modelos y otros tipos de nodos. Cada nodo proporciona una serie de métodos que pueden utilizarse para obtener información sobre el nodo.

Los métodos que se pueden utilizar para obtener el ID, el nombre y la etiqueta de un nodo se resumen en la tabla siguiente.

Tabla 16. Métodos para obtener el ID, el nombre y la etiqueta de un nodo

Método	Tipo devuelto	Descripción
`n.getLabel()`	*serie*	Devuelve la etiqueta de visualización del nodo especificado. La etiqueta es el valor de la propiedad `custom_name` sólo si la propiedad es una serie no vacía y la propiedad `use_custom_name` no está establecida; de lo contrario, la etiqueta es el valor de `getName()`.
`n.setLabel(label)`	No aplicable	Establece la etiqueta de visualización del nodo especificado. Si la nueva etiqueta es una cadena no vacía se asigna a la propiedad `custom_name`, y `False` se asigna a la propiedad `use_custom_name`, de tal modo que la etiqueta especificada tiene prioridad; de lo contrario, se asigna una serie vacía a la propiedad `custom_name` y se asigna `True` a la propiedad `use_custom_name`.
`n.getName()`	*serie*	Devuelve el nombre del nodo especificado.
`n.getID()`	*serie*	Devuelve el ID del nodo especificado. Se crea un ID nuevo cada vez que se crea un nuevo nodo. El ID se conserva con el nodo cuando se guarda como parte de una ruta de modo que, cuando se abre la ruta, los ID de nodo se conservan. Sin embargo, si un nodo guardado se inserta en una ruta, el nodo insertado se considera un nuevo objeto y se le asigna un nuevo ID.

Los métodos que se pueden utilizar para obtener información acerca de un nodo se resumen en la tabla siguiente.

Tabla 17. Métodos para obtener información acerca de un nodo

Método	Tipo devuelto	Descripción
`n.getTypeName()`	*serie*	Devuelve el nombre de script de este nodo. Este es el mismo nombre que puede utilizarse para crear una nueva instancia de este nodo.

Tabla 17. Métodos para obtener información acerca de un nodo (continuación)

Método	**Tipo devuelto**	**Descripción**
`n.isInitial()`	*boolean*	Devuelve `True` si es un nodo *initial*, esto es, un nodo que aparece al inicio de una ruta.
`n.isInline()`	*boolean*	Devuelve `True` si es un nodo *in-line*, esto es, un nodo que aparece a mitad de una ruta.
`n.isTerminal()`	*boolean*	Devuelve `True` si es un nodo *terminal*, esto es, un nodo que aparece al final de una ruta.
`n.getXPosition()`	*int*	Devuelve el desplazamiento de la posición x del nodo en la ruta.
`n.getYPosition()`	*int*	Devuelve el desplazamiento de la posición y del nodo en la ruta.
`n.setXYPosition(x, y)`	No aplicable	Devuelve la posición del nodo en la ruta.
`n.setPositionBetween(source, target)`	No aplicable	Establece la posición del nodo en la ruta, de modo que esté posicionado entre los nodos suministrados.
`n.isCacheEnabled()`	*boolean*	Devuelve `True` si la memoria caché está habilitada; devuelve `False` de lo contrario.
`n.setCacheEnabled(val)`	No aplicable	Habilita o inhabilita la memoria caché para este objeto. Si la memoria caché está llena y la memoria caché pasa a estar inhabilitada, la memoria caché se vacía.
`n.isCacheFull()`	*boolean*	Devuelve `True` si la memoria caché está llena; devuelve `False` de lo contrario.
`n.flushCache()`	No aplicable	Vacía la memoria caché de este nodo. No tiene efecto si la memoria caché no está activada o no está llena.

Capítulo 4. API de scripts

Introducción a la API de scripts

La API de scripts proporciona acceso a una amplia gama de funciones de SPSS Modeler. Todos los métodos descritos hasta ahora forman parte de la API y se puede acceder a los mismos de forma implícita en el script sin importaciones adicionales. Sin embargo, si desea hacer referencia a las clases de la API, debe importar la API explícitamente con la sentencia siguiente:

```
import modeler.api
```

Esta sentencia import es necesaria para muchos de los ejemplos de la API de scripts.

Se puede encontrar una guía completa de las clases, los métodos y parámetros que están disponibles a través de la API de scripts en el documento *Guía de referencia de la API de scripts Python de IBM SPSS Modeler*.

Ejemplo 1: buscar nodos utilizando un filtro personalizado

La sección "Buscar nodos" en la página 29 se incluye un ejemplo de cómo buscar un nodo en una ruta utilizando el nombre de tipo del nodo como criterio de búsqueda. En algunas situaciones, se requiere una búsqueda más genérica y ésta se puede implementar utilizando la clase `NodeFilter` y el método `findAll()` de la ruta. Este tipo de búsqueda requiere los dos pasos siguientes:

1. Crear una clase nueva que amplíe `NodeFilter` e implemente una versión personalizada del método `accept()`.
2. Llamar al método `findAll()` de la ruta con una instancia de esta clase nueva. Esto devuelve todos los nodos que cumplen el criterio definido en el método `accept()`.

El ejemplo siguiente muestra cómo buscar nodos de una ruta que tienen habilitada la memoria caché de nodo. La lista de nodos devuelta se puede utilizar para vaciar o inhabilitar las memorias caché de estos nodos.

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
    """A node filter for nodes with caching enabled"""
    def accept(this, node):
        return node.isCacheEnabled()

cachingnodes = modeler.script.stream().findAll(CacheFilter(), False)
```

Ejemplo 2: permitir a los usuarios obtener información de directorio o archivo basándose en sus privilegios

Para evitar que la PSAPI se abra a los usuarios, se puede utilizar un método denominado `session.getServerFileSystem()` mediante la llamada de la función PSAPI para crear un objeto de sistema de archivos.

El ejemplo siguiente muestra cómo permitir que un usuario obtenga información de directorio o archivo basándose en los privilegios del usuario que se conecta a IBM SPSS Modeler Server.

```
import modeler.api
stream = modeler.script.stream()
sourceNode = stream.findByID('')
session = modeler.script.session()
fileSystem = session.getServerFileSystem()
parameter = stream.getParameterValue('VPATH')
serverDirectory = fileSystem.getServerFile(parameter)
```

```
files = fileSystem.getFiles(serverDirectory)
for f in files:
    if f.isDirectory():
        print 'Directory:'
    else:
        print 'File:'
        sourceNode.setPropertyValue('full_filename',f.getPath())
        break
    print f.getName(),f.getPath()
stream.execute()
```

Metadatos: información sobre datos

Puesto que en una ruta los nodos se conectan entre sí, está disponible la información relativa a las columnas o los campos disponibles en cada nodo. Por ejemplo, en la interfaz de usuario de Modeler esto permite seleccionar por qué campos hay que ordenar o agregar. Esta información se llama modelo de datos.

Los scripts también pueden acceder al modelo de datos inspeccionando los campos que entran en un nodo o que salen de él. En algunos nodos coinciden los modelos de datos de entrada y salida; por ejemplo, un nodo Ordenar se limita a cambiar el orden de los registros, sin alterar el modelo de datos. Otros, como el nodo Derivar, pueden añadir nuevos campos. Otros, como el nodo Filtrar, pueden renombrar o eliminar campos.

En el ejemplo siguiente, el script toma la corriente IBM SPSS Modeler `druglearn.str` estándar, y para cada campo, crea un modelo con uno de los campos de entrada descartados. Lo hace de la siguiente manera:

1. Accede al modelo de datos de salida del nodo Tipo.
2. Itera cada campo del modelo de datos de salida.
3. Modifica el nodo Filtro de cada campo de entrada.
4. Cambia el nombre del modelo que se construye.
5. Ejecuta el nodo de construcción de modelos.

Nota: Antes de ejecutar el script en la corriente `druglean.str` , recuerde establecer el lenguaje de scripts en Python (la corriente se creó en una versión anterior de IBM SPSS Modeler para que el lenguaje de scripts de corriente se establezca en Legacy).

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Always use a custom model name
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # If this is the target field then ignore it
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Re-enable the field that was most recently removed
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Remove the field
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Set the name of the new model then run the build
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

El objeto DataModel (modelo de datos) proporciona una serie de métodos de acceso a la información relativa a los campos y columnas del modelo de datos. Estos métodos se resumen en la siguiente tabla.

Tabla 18. Métodos del objeto DataModel de acceso a la información relativa a campos o columnas		
Método	**Tipo devuelto**	**Descripción**
`d.getColumnCount()`	*int*	Devuelve el número de columnas del modelo de datos.
`d.columnIterator()`	Iterador	Devuelve un iterador que devuelve cada columna en el orden "natural" de inserción. El iterador devuelve instancias de Column.
`d.nameIterator()`	Iterador	Devuelve un iterador que devuelve el nombre de cada columna en el orden "natural" de inserción.
`d.contains(name)`	*boolean*	Devuelve `True` si en este DataModel existe una columna con el nombre proporcionado y `False` en caso contrario.
`d.getColumn(name)`	Columna	Devuelve la columna cuyo nombre es el especificado.
`d.getColumnGroup(name)`	ColumnGroup	Devuelve el grupo de columnas nombrado o `None` si no existe dicho grupo de columnas.
`d.getColumnGroupCount()`	*int*	Devuelve el número de grupos de columnas de este modelo de datos.
`d.columnGroupIterator()`	Iterador	Devuelve un iterador que devuelve a su vez cada grupo de columnas.
`d.toArray()`	Columna []	Devuelve el modelo de datos como un vector de columnas. Las columnas van ordenadas según su orden "natural" de inserción.

Cada campo (objeto Column) incluye una serie de métodos de acceso a la información de la columna. La tabla que se muestra a continuación muestra una selección de los mismos.

Tabla 19. Métodos del objeto Column de acceso a la información de una columna		
Método	**Tipo devuelto**	**Descripción**
`c.getColumnName()`	*serie*	Devuelve el nombre de la columna.
`c.getColumnLabel()`	*serie*	Devuelve la etiqueta de la columna o una cadena vacía si no hay ninguna etiqueta asociada a la columna.
`c.getMeasureType()`	MeasureType	Devuelve el tipo de medición de la columna.
`c.getStorageType()`	StorageType	Devuelve el tipo de almacenamiento de la columna.

Tabla 19. Métodos del objeto Column de acceso a la información de una columna (continuación)		
Método	**Tipo devuelto**	**Descripción**
`c.isMeasureDiscrete()`	*boolean*	Devuelve `True` si la columna es discreta. Se consideran discretas las columnas que son un conjunto o un distintivo.
`c.isModelOutputColumn()`	*boolean*	Devuelve `True` si la columna es una columna de resultado del modelo.
`c.isStorageDatetime()`	*boolean*	Devuelve `True` si el almacenamiento de la columna es un valor de hora, fecha o indicación de fecha y hora.
`c.isStorageNumeric()`	*boolean*	Devuelve `True` si el almacenamiento de la columna es un entero o un número real.
`c.isValidValue(value)`	*boolean*	Devuelve `True` si el valor especificado es válido para este almacenamiento y `valid` cuando se conocen los valores de columna válidos.
`c.getModelingRole()`	ModelingRole	Devuelve el rol de modelado de la columna.
`c.getSetValues()`	Objeto []	Devuelve un vector de valores válidos para la columna o `None` si se desconocen los valores o si la columna no es un conjunto.
`c.getValueLabel(value)`	*serie*	Devuelve la etiqueta del valor de la columna o una cadena vacía si no hay ninguna etiqueta asociada al valor.
`c.getFalseFlag()`	Objeto	Devuelve el valor indicador de "falso" de la columna o `None` si se desconoce el valor o la columna no es un indicador.
`c.getTrueFlag()`	Objeto	Devuelve el valor indicador de "verdadero" de la columna o `None` si se desconoce el valor o la columna no es un indicador.
`c.getLowerBound()`	Objeto	Devuelve el valor del límite inferior de los valores de la columna o `None` si se desconoce el valor o si la columna no es continua.
`c.getUpperBound()`	Objeto	Devuelve el valor del límite superior de los valores de la columna o `None` si se desconoce el valor o si la columna no es continua.

Observe que la mayoría de los métodos de acceso a la información de una columna tienen métodos equivalentes definidos en el propio objeto DataModel. Por ejemplo, las dos sentencias siguientes son equivalentes:

```
dataModel.getColumn("someName").getModelingRole()
dataModel.getModelingRole("someName")
```

Acceso a objetos generados

Ejecución de una ruta normalmente implica producir objetos de salida adicionales. Estos objetos adicionales pueden ser un nuevo modelo o un fragmento de la salida que proporciona información para utilizarla en las ejecuciones posteriores.

En el ejemplo siguiente, la ruta `druglearn.str` se utiliza de nuevo como punto de partida para la ruta. En este ejemplo, se ejecutan todos los nodos de la ruta y los resultados se almacenan en una lista. The script then loops through the results, and any model outputs that result from the execution are saved as an IBM SPSS Modeler model (`.gm`) file, and the model is PMML exported.

```
import modeler.api

stream = modeler.script.stream()

# Set this to an existing folder on your system.
# Include a trailing directory separator
modelFolder = "C:/temp/models/"

# Execute the stream
models = []
stream.runAll(models)

# Save any models that were created
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # If the stream execution built other outputs then ignore them
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

    label = model.getLabel()
    algorithm = model.getModelDetail().getAlgorithmName()

    # save each model...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...and export each model PMML...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

La clase taskrunner proporciona un modo práctico de ejecutar diferentes tareas comunes. Los métodos que están disponibles en esta clase se resumen en la tabla siguiente.

Tabla 20. Métodos de la clase taskrunner para realizar tareas comunes

Método	Tipo devuelto	Descripción
`t.createStream(name, autoConnect, autoManage)`	Flujo	Crea y devuelve una nueva ruta. Tenga en cuenta que el código que debe crear las rutas de forma privada sin que las vea el usuario debe establecer el distintivo `autoManage` en `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	No aplicable	Exporta la descripción de la ruta a un archivo utilizando el formato de archivo especificado.

Tabla 20. Métodos de la clase taskrunner para realizar tareas comunes (continuación)		
Método	**Tipo devuelto**	**Descripción**
`t.exportModelToFile(modelOutput, filename, fileFormat)`	No aplicable	Exporta el modelo a un archivo utilizando el formato de archivo especificado.
`t.exportStreamToFile(stream, filename, fileFormat)`	No aplicable	Exporta la ruta a un archivo utilizando el formato de archivo especificado.
`t.insertNodeFromFile(filename, diagram)`	Nodo	Lee un nodo en el archivo especificado y lo devuelve insertándolo en el diagrama suministrado. Tenga en cuenta que lo pueden utilizar los objetos Nodo y Supernodo.
`t.openDocumentFromFile(filename, autoManage)`	DocumentOutput	Lee un nodo en el archivo especificado y lo devuelve.
`t.openModelFromFile(filename, autoManage)`	ModelOutput	Lee un modelo en el archivo especificado y lo devuelve.
`t.openStreamFromFile(filename, autoManage)`	Flujo	Lee una ruta en el archivo especificado y la devuelve.
`t.saveDocumentToFile( documentOutput, filename)`	No aplicable	Guarda el documento en la ubicación de archivo especificada.
`t.saveModelToFile(modelOutput, filename)`	No aplicable	Guarda el modelo en la ubicación de archivo especificada.
`t.saveStreamToFile(stream, filename)`	No aplicable	Guarda la ruta en la ubicación de archivo especificada.

Manejo de errores

El lenguaje Python proporciona manejo de errores mediante el bloque de código `try...except`. Se puede utilizar en los scripts para capturar excepciones y manejar los problemas que podrían ocasionar la finalización del script.

En el script de ejemplo siguiente, se realiza un intento para recuperar un modelo desde el IBM SPSS Collaboration and Deployment Services Repository. Esta operación puede hacer que se genere una excepción, por ejemplo, es posible que las credenciales de inicio de sesión en el repositorio no se hayan configurado correctamente o que la ruta del repositorio sea errónea. En el script, esto puede hacer que se genere un `ModelerException` (todas las excepciones generadas por IBM SPSS Modeler se derivan de `modeler.api.ModelerException`).

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
```

Nota: Algunas operaciones de scripts pueden generar excepciones Java estándar, estas excepciones no se derivan de `ModelerException`. Para capturar estas excepciones, se puede utilizar un bloque except adicional que capture todas las excepciones Java, por ejemplo:

```
import modeler.api
import java.lang.Exception

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

Parámetros de ruta, sesión y Supernodo

Los parámetros proporcionan una forma útil de pasar valores en el momento de la ejecución, en lugar de codificarlos directamente en un script. Los parámetros y sus valores se definen de la misma forma que las rutas, es decir, como entradas de la tabla de los parámetros de una ruta o Supernodo o como parámetros de la línea de comandos. Las clases de Ruta y Supernodo implementan un conjunto de funciones definidas por el objeto ParameterProvider, como se muestra en la tabla siguiente. La sesión proporciona una llamada `getParameters()` que devuelve un objeto que define dichas funciones.

Tabla 21. Funciones definidas por el objeto ParameterProvider

Método	Tipo devuelto	Descripción
`p.parameterIterator()`	Iterador	Devuelve un iterador de nombres de parámetro para este objeto.
`p.getParameterDefinition(parameterName)`	ParameterDefinition	Devuelve la definición de parámetro para el parámetro con el nombre especificado, o None si no existe tal parámetro en este proveedor. El resultado puede ser una instantánea de la definición en el momento en que el método se ha llamado y que no necesariamente refleja las modificaciones posteriores realizadas en el parámetro a través de este proveedor.
`p.getParameterLabel(parameterName)`	*serie*	Devuelve la etiqueta del parámetro con nombre, o None si no existe tal parámetro.
`p.setParameterLabel(parameterName, label)`	No aplicable	Establece la etiqueta del parámetro con nombre.
`p.getParameterStorage(parameterName)`	ParameterStorage	Devuelve el almacenamiento del parámetro con nombre, o None si no existe tal parámetro.
`p.setParameterStorage(parameterName, storage)`	No aplicable	Establece el almacenamiento del parámetro con nombre.

Tabla 21. Funciones definidas por el objeto ParameterProvider (continuación)		
Método	**Tipo devuelto**	**Descripción**
p.getParameterType(parameterName)	ParameterType	Devuelve el tipo del parámetro con nombre, o None si no existe tal parámetro.
p.setParameterType(parameterName, type)	No aplicable	Establece el tipo del parámetro con nombre.
p.getParameterValue(parameterName)	Objeto	Devuelve el valor del parámetro con nombre, o None si no existe tal parámetro.
p.setParameterValue(parameterName, value)	No aplicable	Establece el valor del parámetro con nombre.

En el ejemplo siguiente, el script agrega algunos datos Telco para averiguar qué región tiene los datos de promedio de ingresos más bajos. A continuación, se establece un parámetro de ruta con esta región. Este parámetro de ruta se utiliza en un nodo Seleccionar para excluir dicha región de los datos, antes de que se cree un modelo de abandono en el resto.

El ejemplo es artificial porque el script genera el propio nodo Seleccionar y, por lo tanto, podría haber generado el valor correcto directamente en la expresión del nodo Seleccionar. Sin embargo, las rutas se suelen construir previamente, de modo que establecer los parámetros de este modo proporciona un ejemplo útil.

La primera parte del script de ejemplo crea el parámetro de ruta que contendrá la región con el promedio de ingresos más bajo. El script también crea los nodos de la rama de agregación y la rama de creación de modelos y los conecta.

```
import modeler.api

stream = modeler.script.stream()

# Initialize a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)
```

El script de ejemplo crea la ruta siguiente.

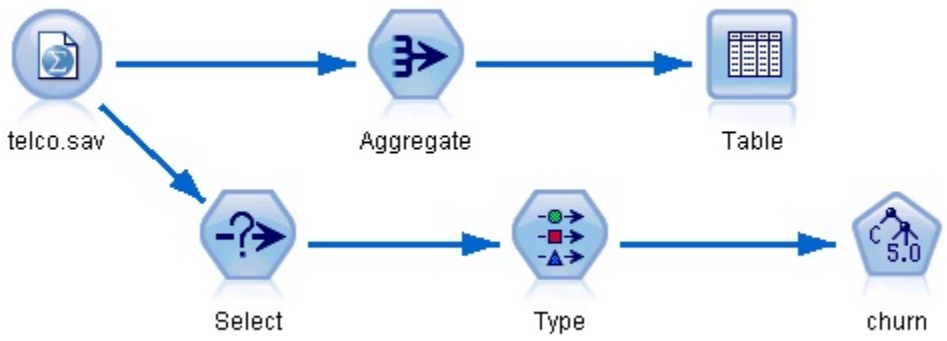

Figura 5. Ruta resultante del script de ejemplo

La parte siguiente del script de ejemplo ejecuta el nodo Tabla al final de la rama de agregación.

```
# First execute the table node
results = []
tablenode.run(results)
```

La parte siguiente del script de ejemplo accede a la salida de la tabla que ha generado la ejecución del nodo Tabla. A continuación, el script itera por las filas de la tabla, buscando la región con el promedio de ingresos más bajo.

```
# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1
```

La parte siguiente del script utiliza la región con el promedio de ingresos más bajo para establecer el parámetro de ruta "LowestRegion" creado anteriormente. El script ejecuta el constructor de modelos excluyendo la región especificada de los datos de formación.

```
# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

A continuación, se muestra el script de ejemplo.

```
import modeler.api

stream = modeler.script.stream()

# Create a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
```

```
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)


# First execute the table node
results = []
tablenode.run(results)

# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

Valores globales

Los valores globales se utilizan para calcular diferentes estadísticas de resumen para los campos especificados. Se puede acceder a estos valores de resumen desde cualquier lugar de la ruta. Los valores globales son similares a los parámetros de ruta, ya que se puede acceder a los mismos por nombre a través de la ruta. Se diferencian de los parámetros de ruta en que los valores asociados se actualizan automáticamente cuando se ejecuta uno nodo Establecer globales, en lugar de asignarlos mediante script o desde la línea de comandos. Se accede a los valores globales de una ruta invocando el método `getGlobalValues()` de la ruta.

El objeto GlobalValues define las funciones que se muestran en la tabla siguiente.

Tabla 22. Funciones definidas por el objeto GlobalValues		
Método	**Tipo devuelto**	**Descripción**
`g.fieldNameIterator()`	Iterador	Devuelve un iterador para cada nombre de campo con al menos un valor global.
`g.getValue(type, fieldName)`	Objeto	Devuelve el valor global para el tipo especificado y nombre de campo, o None si no se puede localizar ningún valor. Generalmente se espera que el valor devuelto sea un número, aunque en las funciones futuras se pueden devolver tipos de valores diferentes.
`g.getValues(fieldName)`	Mapa	Devuelve un mapa que contiene las entradas conocidas para el nombre de campo especificado o None si no hay entradas para el campo.

`GlobalValues.Type` define el tipo de estadísticas de resumen disponibles. Están disponibles las siguientes estadísticas de resumen:

- `MAX`: el valor máximo del campo.
- `MEAN`: el valor medio del campo.
- `MIN`: el valor mínimo del campo.
- `STDDEV`: la desviación estándar del campo.
- `SUM`: la suma de los valores del campo.

Por ejemplo, el script siguiente accede el valor medio del campo "income" que calcula un nodo Val. globales:

```
import modeler.api

globals = modeler.script.stream().getGlobalValues()
mean_income = globals.getValue(modeler.api.GlobalValues.Type.MEAN, "income")
```

Trabajar con varias rutas: Scripts autónomos

Para trabajar con varias rutas se debe utilizar un script autónomo. El script autónomo se puede editar y ejecutar dentro de la interfaz de usuario de IBM SPSS Modeler o se puede pasar como un parámetro de línea de comandos en modalidad de proceso por lotes.

El siguiente script autónomo abre dos rutas. Una de estas rutas crea un modelo y la segunda ruta traza la distribución de los valores predichos.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/18.4.0/DEMOS/streams/"

session = modeler.script.session()
tasks = session.getTaskRunner()

# Open the model build stream, locate the C5.0 node and run it
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
```

```
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Create a model applier node, insert it between the derive and histogram nodes
# then run the histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

# Finally, tidy up the streams
buildstream.close()
plotstream.close()
```

El ejemplos siguiente muestra cómo también puede iterar en las rutas abiertas (todas las rutas abiertas en la pestaña Rutas). Tenga en cuenta que esto solo está soportado en scripts autónomos.

```
for stream in modeler.script.streams():
    print stream.getName()
```

Capítulo 5. Sugerencias para scripts

Esta sección proporciona una visión general de las técnicas y sugerencias para utilizar scripts, incluida la modificación de la ejecución de la ruta, la utilización de una contraseña codificada en un script y el acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository.

Modificación de ejecución de rutas

Cuando se ejecuta una ruta, sus nodos terminales se ejecutan en un orden optimizado para la situación predeterminada. En algunos casos, es posible que prefiera un orden de ejecución diferente. Para modificar el orden de ejecución de una ruta, lleve a cabo los siguientes pasos desde la pestaña Ejecución del cuadro de diálogo Propiedades de ruta:

1. Comience con un script vacío.
2. Pulse en el botón **Añadir script predeterminado** de la barra de herramientas y añada el script de ruta predeterminado.
3. Cambie el orden de las instrucciones del script de ruta predeterminado por el orden en que desee que se ejecuten las instrucciones.

Nodos de recorrido en bucle

Puede utilizar un bucle `for` para recorrer en bucle todos los nodos de una ruta. Por ejemplo, los siguientes dos ejemplos de script recorren en bucle todos los nodos y cambia los nombres de campos de cualquier nodo Filtrar a mayúsculas.

Estos scripts pueden emplearse en cualquier ruta que tenga un nodo Filtrar, incluso si no hay campos filtrados. Simplemente, añada un nodo Filtrar que recorra todos los campos para cambiar todos los nombres de campo a mayúsculas.

```
# Alternative 1: using the data model nameIterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() returns the field names
        for field in node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Alternative 2: using the data model iterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() returns the field objects so we need
        # to call getColumnName() to get the name
        for field in node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

El script recorre en bucle los nodos de la ruta actual y comprueba cada nodo para ver si es un Filtro. Si es así, el script recorre en bucle cada campo del nodo y utiliza la función `field.upper()` o `field.getColumnName().upper()` para cambiar el nombre a mayúsculas.

Acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository

Si tiene una licencia para IBM SPSS Collaboration and Deployment Services Repository, puede almacenar y recuperar objetos del repositorio utilizando comandos de script. Utilice el repositorio para gestionar el

ciclo de vida de los modelos de minería de datos y los objetos predictivos relacionados en el contexto de aplicaciones empresariales, herramientas y soluciones.

Conexión con IBM SPSS Collaboration and Deployment Services Repository

Para acceder al repositorio, en primer lugar, debe configurar una conexión válida con el mismo, a través del menú **Herramientas** de la interfaz de usuario de SPSS Modeler o a través de la línea de comandos. Para obtener más información, consulte "Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository" en la página 69.

Obtener acceso al repositorio

Se puede acceder al repositorio desde la sesión, por ejemplo:

```
repo = modeler.script.session().getRepository()
```

Recuperación de objetos del repositorio

En un script, utilice las funciones `retrieve*` para acceder a distintos objetos, incluyendo rutas, modelos, salidas y nodos. Se muestra un resumen de las funciones de recuperación en la tabla siguiente.

Tabla 23. Recuperar funciones de script

Tipo de objeto	**Función de repositorio**
Flujo	repo.retrieveStream(String path, String version, String label, Boolean autoManage)
Modelo	repo.retrieveModel(String path, String version, String label, Boolean autoManage)
Salida	repo.retrieveDocument(String path, String version, String label, Boolean autoManage)
Nodo	repo.retrieveProcessor(String path, String version, String label, ProcessorDiagram diagram)

Por ejemplo, puede recuperar una ruta del repositorio con la función siguiente:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", None, "production", True)
```

Este ejemplo recupera la ruta `risk_score.str` de la carpeta especificada. La etiqueta `production` identifica qué versión de la corriente se va a recuperar y el último parámetro especifica que SPSS Modeler debe gestionar la corriente (por ejemplo, de modo que la corriente aparece en la pestaña **Flujos** si la interfaz de usuario de SPSS Modeler es visible). Como alternativa, para utilizar una versión específica sin etiquetar:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", "0:2015-10-12 14:15:41.281",
None, True)
```

Nota: Si ambos parámetros, el de versión y el de etiqueta, son None,se devuelve la última versión.

Almacenar objetos en el repositorio

Para utilizar scripts para almacenar objetos en el repositorio, utilice las funciones `store*`. Se muestra un resumen de las funciones de almacenamiento en la tabla siguiente.

Tabla 24. Funciones de script de almacenamiento	
Tipo de objeto	**Función de repositorio**
Flujo	repo.storeStream(ProcessorStream stream, String path, String label)
Modelo	repo.storeModel(ModelOutput modelOutput, String path, String label)
Salida	repo.storeDocument(DocumentOutput documentOutput, String path, String label)
Nodo	repo.storeProcessor(Processor node, String path, String label)

Por ejemplo, puede almacenar una nueva versión de la ruta `risk_score.str` con la función siguiente:

```
versionId = repo.storeStream(stream, "/projects/retention/risk_score.str", "test")
```

Este ejemplo almacena una nueva versión de la ruta, asocia la etiqueta `"test"` a la misma y devuelve el marcador de la versión para la versión recién creada.

Nota: Si no desea asociar una etiqueta a la nueva versión, pase None para la etiqueta.

Gestión de carpetas de repositorio

Al utilizar las carpetas dentro del repositorio, puede organizar objetos en grupos lógicos y hacer que sea más fácil ver qué objetos están relacionados. Cree carpetas utilizando la función `createFolder()`, como en el ejemplo siguiente:

```
newpath = repo.createFolder("/projects", "cross-sell")
```

Este ejemplo crea una carpeta nueva que se llama `"cross-sell"` en la carpeta `"/projects"`. La función devuelve la vía de acceso completa de la carpeta nueva.

Para cambiar el nombre de una carpeta, utilice la función `renameFolder()`:

```
repo.renameFolder("/projects/cross-sell", "cross-sell-Q1")
```

El primer parámetro es la vía de acceso completa que se va a renombrar, y el segundo es el nombre nuevo que se va a proporcionar a dicha carpeta.

Para suprimir una carpeta vacía, utilice la función `deleteFolder()`:

```
repo.deleteFolder("/projects/cross-sell")
```

Bloquear y desbloquear objetos

Puede bloquear un objeto desde un script para evitar que otros usuarios actualicen cualquiera de las versiones existentes o creen nuevas versiones. También puede desbloquear un objeto que haya bloqueado.

La sintaxis para bloquear y desbloquear un objeto es:

```
repo.lockFile(REPOSITORY_PATH)
repo.lockFile(URI)

repo.unlockFile(REPOSITORY_PATH)
repo.unlockFile(URI)
```

Al igual que con el almacenamiento y la recuperación de objetos, REPOSITORY_PATH proporciona la ubicación del objeto en el repositorio. La ruta debe estar entre comillas y utilizar barras inclinadas como delimitadores. No distingue entre mayúsculas y minúsculas.

```
repo.lockFile("/myfolder/Stream1.str")
repo.unlockFile("/myfolder/Stream1.str")
```

Si lo prefiere, puede utilizar un identificador de recursos uniforme (URI) en lugar de una ruta de repositorio para proporcionar la ubicación del proyecto. El URI debe incluir el prefijo `spsscr:` y debe estar entre comillas. Sólo se puede utilizar barras inclinadas como delimitadores, y los espacios deben estar codificados. Es decir, utilizar `%20` en lugar de un espacio en la ruta. El URI no distingue entre mayúsculas y minúsculas. A continuación aparecen algunos ejemplos:

```
repo.lockFile("spsscr:///myfolder/Stream1.str")
repo.unlockFile("spsscr:///myfolder/Stream1.str")
```

Tenga en cuenta que el bloqueo de objetos se aplica a todas las versiones de un objeto: no puede bloquear o desbloquear versiones por separado.

Generación de una contraseña codificada

En algunos casos, puede que necesite incluir una contraseña en un script. Por ejemplo, es posible que desee acceder a un origen de datos protegido con contraseña. Las contraseñas codificadas pueden utilizarse en:

- Propiedades de los nodos para un origen de base de datos y nodos de resultado
- Argumentos de línea de comando para conectarse al servidor
- Propiedades de conexión con la base de datos almacenadas en un archivo *.par* (archivo de parámetro generado desde la pestaña Publicar de un nodo de exportación)

A través de la interfaz de usuario, está disponible una herramienta para generar contraseñas codificadas basándose en el algoritmo Blowfish (consulte http://www.schneier.com/blowfish.html si desea más información). Una vez codificada, puede copiar y almacenar la contraseña en archivos de script y argumentos de líneas de comando. La propiedad de nodo `epassword` utilizada para `databasenode` y `databaseexportnode` almacena la contraseña codificada.

1. Para generar una contraseña codificada, en el menú Herramientas seleccione:

   Codificar contraseña...
2. Especifique una contraseña en el cuadro de texto Contraseña.
3. Pulse **Codificar** para generar una codificación aleatoria de la contraseña.
4. Pulse en el botón Copiar para copiar la contraseña codificada al Portapapeles.
5. Pegue la contraseña en el script o parámetro deseado.

Comprobación de script

Puede comprobar rápidamente la sintaxis de todos los tipos de scripts pulsando en el botón de comprobación de la barra de herramientas del cuadro de diálogo Script de ruta.

Figura 6. Iconos de barra de herramientas del script de ruta

En la comprobación por script se avisa de cualquier error que se detecte en el código y se sugieren recomendaciones de mejora. Para ver la línea con errores, pulse en los comentarios, en la mitad inferior del cuadro de diálogo. Los errores se señalan en rojo.

Scripts desde la línea de comandos

El uso scripts permite ejecutar operaciones típicamente desarrolladas en la interfaz de usuario. Simplemente especifique y ejecute un script independiente en la línea de comandos cuando ejecute IBM SPSS Modeler. Por ejemplo:

```
client -script scores.txt -execute
```

La marca `-script` carga el script especificado, mientras que la marca `-execute` ejecuta todos los comandos del archivo de script.

Compatibilidad con releases anteriores

Los scripts creados en versiones anteriores de IBM SPSS Modeler deberían funcionar normalmente sin cambios en la versión actual. Sin embargo, los nuggets de modelos podrán ahora insertarse en la ruta automáticamente (es el comportamiento predeterminado) y podrán sustituir o complementar un nugget existente del tipo en la ruta. Si esto sucede realmente depende de los valores de las opciones **Añadir modelo a corriente** y **Sustituir modelo anterior** (**Herramientas > Opciones > Opciones de usuario > Notificaciones**). Por ejemplo, es posible que tenga que modificar un script de una versión anterior en el que la sustitución del nugget se trate borrando el nugget existente e insertando uno nuevo.

Es posible que los scripts creados en esta versión no funcionen en versiones anteriores.

Si un script creó una liberación antigua utiliza un comando que se ha sustituido desde entonces (o desaprobado), la forma antigua se seguirá admitiendo, pero aparecerá un mensaje de advertencia. Por ejemplo, la palabra clave antigua `generated` se ha sustituido por `model`, y `clear generated` se ha sustituido por `clear generated palette`. Los scripts que utilizan las formas antiguas se seguirán ejecutando, pero se mostrará una advertencia.

resultados de ejecución de la secuencia, acceder

Muchos nodos de IBM SPSS Modeler producen datos de salida tales como modelos, diagramas y datos tabulares. Muchas de estas salidas contienen valores útiles que los scripts pueden utilizar para guiar las ejecuciones posteriores. Estos valores se agrupan en contenedores de contenido (denominados simplemente contenedores) a los que se puede acceder utilizando etiquetas o identificadores que identifican cada contenedor. La forma en que se accede a estos valores depende del formato o "modelo de contenido" utilizado por el contenedor.

Por ejemplo, muchos resultados de modelo predictivo utilizan una variante de XML llamada PMML para representar información sobre el modelo, tal como qué campos utiliza un árbol de decisiones en cada bifurcación o cómo están conectadas las neuronas de una red neuronal y con qué intensidades. Los resultados del modelo que utilizan PMML proporcionan un modelo de contenido XML que se puede utilizar para acceder a esa información. Por ejemplo:

```
stream = modeler.script.stream()
# Assume the stream contains a single C5.0 model builder node
# and that the datasource, predictors and targets have already been
# set up
modelbuilder = stream.findByType("c50", None)
results = []
modelbuilder.run(results)
modeloutput = results[0]

# check how many contents are there and what are their names
tags = modeloutput.getContentModelTags()
print "Content Model Tags :" , tags

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")
if (cm != None) :
```

```
    # The PMML content model is a generic XML-based content model that
    # uses XPath syntax. Use that to find the names of the data fields.
    # The call returns a list of strings match the XPath values
    dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField",
"name")
    print "Data Field Names:" , dataFieldNames
```

IBM SPSS Modeler es compatible con los modelos de contenido siguientes en los scripts:

- El **Modelo de contenido de tabla** proporciona acceso a los datos tabulares sencillos representados como filas y columnas.
- El **Modelo de contenido XML** proporciona acceso a contenido almacenado en formato XML.
- El **modelo de contenido JSON** proporciona acceso a contenido almacenado en formato JSON.
- El **Modelo de estadísticas de columna** proporciona acceso a estadísticas de resumen sobre un campo específico.
- El **Modelo de contenido de estadísticas de columna por pares** proporciona acceso a estadísticas de resumen entre dos campos o valores entre dos campos separados.

Tenga en cuenta que los nodos siguientes no contienen estos modelos de contenido:

- Serie temporal
- Discriminante
- SLRM
- TCM
- Todos los nodos Python
- Todos los nodos Spark
- Todos los nodos de modelado de bases de datos
- Modelo de extensión
- STP

Modelo de contenido de tabla

El modelo de contenido de tabla proporciona un modelo sencillo para acceder a los datos simples de fila y columna. Los valores en una columna determinada deben tener todos el mismo tipo de almacenamiento (por ejemplo, series o enteros).

API

Tabla 25. API

Volver	Método	Descripción
`int`	`getRowCount()`	Devuelve el número de filas en esta tabla.
`int`	`getColumnCount()`	Devuelve el número de columnas en esta tabla.
`String`	`getColumnName(int columnIndex)`	Devuelve el nombre de la columna en el índice de columna especificado. El índice de columna comienza en el 0.

Tabla 25. API (continuación)		
Volver	**Método**	**Descripción**
StorageType	getStorageType(int columnIndex)	Devuelve el tipo de almacenamiento de la columna en el índice especificado. El índice de columna comienza en el 0.
Object	getValueAt(int rowIndex, int columnIndex)	Devuelve el valor en los índices de fila y columna especificados. Los índices de fila y columna comienzan en el 0.
void	reset()	Desecha cualquier almacenamiento interno asociado con el modelo de contenido.

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 26. Nodos y datos de salida		
Nombre de nodo	**Nombre del resultado**	**ID de contenedor**
table	table	"table"

Script de ejemplo

```
stream = modeler.script.stream()
from modeler.api import StorageType

# Set up the variable file import node
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Next create the aggregate node and connect it to the variable file node
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configure the aggregate node
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Then create the table output node and connect it to the aggregate node
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Execute the table node and capture the resulting table output object
results = []
tablenode.run(results)
tableoutput = results[0]

# Access the table output's content model
tablecontent = tableoutput.getContentModel("table")

# For each column, print column name, type and the first row
# of values from the table content
col = 0
```

```
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

La salida en la pestaña Depuración de scripts tendrá un aspecto similar a este:

```
Age_Min Integer 15
Age_Max Integer 74
Na_Mean Real 0.730851098901
Na_SDev Real 0.116669731242
Drug String drugY
Record_Count Integer 91
```

Modelo de contenido XML

El modelo de contenido XML proporciona acceso a contenido basado en XML.

El modelo de contenido XML permite acceder a componentes utilizando expresiones XPath. Las expresiones XPath son series que definen qué elementos o atributos son necesarios para el solicitante. El modelo de contenido XML hace que sea transparente la creación de diversos objetos y expresiones de compilación que normalmente son necesarios para el soporte de XPath. Esto hace que sea más sencillo hacer llamadas desde scripts Python.

El modelo de contenido XML incluye una función que devuelve el documento XML como serie de caracteres. Esto permite que los usuarios del script Python utilicen su biblioteca preferida de Python para analizar XML.

API

Tabla 27. API

Devolución	Método	Descripción
`String`	`getXMLAsString()`	Devuelve el XML en forma de serie de caracteres.
`number`	`getNumericValue(String xpath)`	Devuelve un resultado de tipo numérico al analizar la vía de acceso (por ejemplo, contar el número de elementos que coinciden con la expresión de vía de acceso).
`boolean`	`getBooleanValue(String xpath)`	Devuelve un resultado de tipo booleano al evaluar la expresión de vía de acceso especificada.
`String`	`getStringValue(String xpath, String attribute)`	Devuelve el valor de atributo o valor de nodo XML que coincide con la vía de acceso especificada.
`List of strings`	`getStringValues(String xpath, String attribute)`	Devuelve una lista de todos los valores de atributo o valores de nodo XML que coinciden con la vía de acceso especificada.

Tabla 27. API (continuación)		
Devolución	**Método**	**Descripción**
`List of lists of strings`	`getValuesList(String xpath, <List of strings> attributes, boolean includeValue)`	Devuelve una lista de todos los valores de atributo que coinciden con la vía de acceso especificada junto con el valor de nodo XML si es necesario.
`Hash table (key:string, value:list of string)`	`getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)`	Devuelve una tabla hash que utiliza el atributo de clave o valor de nodo XML como clave y la lista de valores de atributo especificados como valores de la tabla.
`boolean`	`isNamespaceAware()`	Indica si los analizadores XML deben tener en cuenta los espacios de nombres. El valor predeterminado es `False`.
`void`	`setNamespaceAware(boolean value)`	Establece si los analizadores XML deben tener en cuenta los espacios de nombres. Esto también invoca a `reset()` para asegurarse de que los cambios sean captados por llamadas posteriores.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido (por ejemplo, un objeto DOM almacenado en memoria caché).

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 28. Nodos y datos de salida		
Nombre de nodo	**Nombre del resultado**	**ID de contenedor**
`Most model builders`	`Most generated models`	`"PMML"`
`"autodataprep"`	`n/a`	`"PMML"`

Script de ejemplo

El código del script Python para acceder al contenido puede tener este aspecto:

```
results = []
modelbuilder.run(results)
modeloutput = results[0]
cm = modeloutput.getContentModel("PMML")

dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
predictedNames = cm.getStringValues("//MiningSchema/
MiningField[@usageType='predicted']", "name")
```

Modelo de contenido JSON

El modelo de contenido JSON se utiliza para proporcionar soporte para contenido con formato JSON. Esto proporciona una API básica para permitir que los solicitantes extraigan valores bajo la asunción de que saben qué valores se deben acceder.

API

Tabla 29. API

Devolución	Método	Descripción
`String`	`getJSONAsString()`	Devuelve el contenido JSON como serie de caracteres.
`Object`	`getObjectAt(<List of cbjecta> path, JSONArtifact artifact) throws Exception`	Devuelve el objeto situado en la vía de acceso especificada. El artefacto raíz proporcionado puede ser nulo, en cuyo caso se utiliza la raíz del contenido. El valor devuelto puede ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).
`Hash table (key:object, value:object>`	`getChildValuesAt(<List of object> path, JSONArtifact artifact) throws Exception`	Devuelve los valores hijo de la vía de acceso especificada si la vía conduce a un objeto JSON, o devuelve nulo en caso contrario. Las claves de la tabla son series, mientras que el valor asociado puede ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).
`List of objects`	`getChildrenAt(<List of object> path path, JSONArtifact artifact) throws Exception`	Devuelve la lista de objetos situados en la vía de acceso especificada si la vía conduce a una matriz JSON, o devuelve nulo en caso contrario. Los valores devueltos pueden ser una serie literal, un entero, un número real, un valor booleano o un artefacto JSON (ya sea un objeto JSON o una matriz JSON).
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido (por ejemplo, un objeto DOM almacenado en memoria caché).

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 30. Nodos y datos de salida		
Nombre de nodo	**Nombre del resultado**	**ID de contenedor**
"applykmeansas"	n/a	"JSON_MV"
"applyxgboosttree"	n/a	"JSON_MV"

Scripts de ejemplo

Los scripts siguientes recuperan archivos JSON:

```
applykmeansas = stream.findByType("applykmeansas",None)
mvjson = applykmeansas.getContentModel("JSON_MV")
print(mvjson.getJSONAsString())
```

```
applyxgboosttree = stream.findByType("applyxgboosttree",None)
mvjson = applyxgboosttree.getContentModel("JSON_MV")
print(mvjson.getJSONAsString())
```

Modelo de contenido de estadísticas de columna y modelo de contenido de estadísticas por pares

El modelo de contenido de estadísticas de columna proporciona acceso a estadísticas que se pueden calcular para cada campo (estadísticas univariables). El modelo de contenido de estadísticas por pares proporciona acceso a estadísticas que se pueden calcular para pares de campos o pares de valores de un campo.

Las medidas estadísticas posibles son:

- `Count`
- `UniqueCount`
- `ValidCount`
- `Mean`
- `Sum`
- `Min`
- `Max`
- `Range`
- `Variance`
- `StandardDeviation`
- `StandardErrorOfMean`
- `Skewness`
- `SkewnessStandardError`
- `Kurtosis`
- `KurtosisStandardError`
- `Median`
- `Mode`
- `Pearson`
- `Covariance`
- `TTest`

- FTest

Algunos valores sólo son adecuados para estadísticas de una sola columna, mientras que otros sólo son adecuados para estadísticas por pares.

Los nodos que generan estadísticas son los siguientes:

- El **nodo Estadísticas** produce estadísticas de columna y puede producir estadísticas por pares cuando se especifican campos de correlación
- El **nodo Auditoría de datos** produce estadísticas de columna y puede producir estadísticas por pares cuando se especifica un campo de preformato.
- El **nodo Medias** produce estadísticas por pares cuando compara pares de campos o cuando compara valores de un campo con otros resúmenes de campo.

Qué modelos de contenido y estadísticas se pueden utilizar depende de las prestaciones del nodo en cuestión y de los valores contenidos en ese nodo.

La API ColumnStatsContentModel

Tabla 31. La API ColumnStatsContentModel

Devolución	Método	Descripción
`List<StatisticType>`	`getAvailableStatistics()`	Devuelve las estadísticas disponibles en este modelo. No todos los campos tendrán necesariamente valores para todas las estadísticas.
`List<String>`	`getAvailableColumns()`	Devuelve los nombres de columna para los que se han calculado estadísticas.
`Number`	`getStatistic(String column, StatisticType statistic)`	Devuelve los valores estadísticos asociados a la columna.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido.

La API PairwiseStatsContentModel

Tabla 32. La API PairwiseStatsContentModel

Devolución	Método	Descripción
`List<StatisticType>`	`getAvailableStatistics()`	Devuelve las estadísticas disponibles en este modelo. No todos los campos tendrán necesariamente valores para todas las estadísticas.
`List<String>`	`getAvailablePrimaryColumns()`	Devuelve los nombres de columna primaria para los que se han calculado estadísticas.
`List<Object>`	`getAvailablePrimaryValues()`	Devuelve los valores de la columna primaria para la que se han calculado estadísticas.

Tabla 32. La API PairwiseStatsContentModel (continuación)		
Devolución	**Método**	**Descripción**
`List<String>`	`getAvailableSecondaryColumns()`	Devuelve los nombres de columna secundaria para los que se han calculado estadísticas.
`Number`	`getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)`	Devuelve los valores estadísticos asociados a las columnas.
`Number`	`getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)`	Devuelve los valores estadísticos asociados al valor de la columna primaria y la columna secundaria.
`void`	`reset()`	Desecha cualquier almacenamiento interno asociado con el modelo de contenido.

Nodos y datos de salida

Esta tabla lista los nodos que generan datos de salida y que incluyen este tipo de modelo de contenido.

Tabla 33. Nodos y datos de salida			
Nombre de nodo	**Nombre del resultado**	**ID de contenedor**	**Notas**
`"means"` (nodo Medias)	`"means"`	`"columnStatistics"`	
`"means"` (nodo Medias)	`"means"`	`"pairwiseStatistics"`	
`"dataaudit"` (nodo Auditoría de datos)	`"means"`	`"columnStatistics"`	
`"statistics"` (nodo Estadísticas)	`"statistics"`	`"columnStatistics"`	Sólo se genera cuando se examinan campos determinados.
`"statistics"` (nodo Estadísticas)	`"statistics"`	`"pairwiseStatistics"`	Sólo se genera cuando se correlacionan campos.

Script de ejemplo

```
from modeler.api import StatisticType
stream = modeler.script.stream()

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
```

```
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

# Now create the statistics node. This can produce both
# column statistics and pairwise statistics
statisticsnode = stream.createAt("statistics", "Stats", 192, 96)
statisticsnode.setPropertyValue("examine", ["Age", "Na", "K"])
statisticsnode.setPropertyValue("correlate", ["Age", "Na", "K"])
stream.link(varfile, statisticsnode)

results = []
statisticsnode.run(results)
statsoutput = results[0]
statscm = statsoutput.getContentModel("columnStatistics")
if (statscm != None):
    cols = statscm.getAvailableColumns()
    stats = statscm.getAvailableStatistics()
    print "Column stats:", cols[0], str(stats[0]), " = ",
statscm.getStatistic(cols[0], stats[0])

statscm = statsoutput.getContentModel("pairwiseStatistics")
if (statscm != None):
    pcols = statscm.getAvailablePrimaryColumns()
    scols = statscm.getAvailableSecondaryColumns()
    stats = statscm.getAvailableStatistics()
    corr = statscm.getStatistic(pcols[0], scols[0], StatisticType.Pearson)
    print "Pairwise stats:", pcols[0], scols[0], " Pearson = ", corr
```

Capítulo 6. Argumentos de la línea de comandos

Invocación del software

Puede utilizar la línea de comandos del sistema operativo para iniciar IBM SPSS Modeler de la siguiente manera:

1. En un ordenador en el que se haya instalado IBM SPSS Modeler, abra una ventana de DOS o del indicador de comandos.
2. Para iniciar la interfaz IBM SPSS Modeler en modalidad interactiva, escriba el mandato `modelerclient` seguido de los argumentos necesarios; por ejemplo:

```
modelerclient -stream report.str -execute
```

Los argumentos disponibles (modificadores) permiten conectar con un servidor, cargar rutas, ejecutar scripts o especificar otros parámetros, según sea necesario.

Utilización de argumentos de la línea de comandos

Puede añadir argumentos de línea de mandatos (también denominados *Banderas*) al mandato `modelerclient` inicial para alterar la invocación de IBM SPSS Modeler.

Hay varios tipos de argumentos de línea de comandos disponibles que se describen más adelante en esta sección.

Tabla 34. Tipos de argumentos de línea de comandos

Tipo de argumento	**Dónde se describe**
Argumentos del sistema	Para obtener más información, consulte el tema “Argumentos del sistema” en la página 66.
Argumentos de parámetros	Para obtener más información, consulte el tema “Argumentos de parámetros” en la página 67.
Argumentos de conexión del servidor	Para obtener más información, consulte el tema “Argumentos de conexión del servidor” en la página 68.
IBM SPSS Collaboration and Deployment Services Repository Argumentos de conexión	Para obtener más información, consulte el tema “Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository” en la página 69.
IBM SPSS Analytic Server Argumentos de conexión	Para obtener más información, consulte el tema “IBM SPSS Analytic Server Argumentos de conexión” en la página 70.

Por ejemplo, se pueden utilizar las marcas `-server`, `-stream` y `-execute` para conectar con un servidor y, a continuación, cargar y ejecutar una ruta, de la siguiente forma:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Tenga en cuenta que al ejecutarse en una instalación cliente local, no se necesitan argumentos de conexión con el servidor.

Los valores de parámetros que contienen espacios se pueden poner entre comillas dobles, por ejemplo:

```
modelerclient -stream mystream.str -Pusername="Joe User" -execute
```

También puede ejecutar estados y scripts de IBM SPSS Modeler de esta manera, utilizando los distintivos `-state` y `-script` , respectivamente.

Nota: Si utiliza un parámetro estructurado en un comando, delante de las comillas debe poner barras invertidas. Así evitará que se quiten las comillas durante la interpretación de la serie.

Depuración de argumentos de línea de comandos

Para depurar una línea de mandatos, utilice el mandato `modelerclient` para iniciar IBM SPSS Modeler con los argumentos deseados. Esto permite comprobar que los comandos se ejecutarán como se espera. También puede confirmar los valores de cualquier parámetro pasado desde la línea de comandos en el cuadro de diálogo Parámetros de sesión (menú Herramientas, Definir parámetros de sesión).

Argumentos del sistema

En la siguiente tabla se describen los argumentos del sistema disponibles para la invocación de la línea de comandos de la interfaz de usuario.

Tabla 35. Argumentos del sistema

Argumento	**Comportamiento/Descripción**
`@` < commandFile>	El carácter @ seguido de un nombre de archivo especifica una lista de comandos. Cuando `modelerclient` o encuentra un argumento que comienza por @, opera en los comandos de este archivo como si hubieran estado en la línea de comandos. Para obtener más información, consulte el tema “Combinación de varios argumentos” en la página 70.
`-directory` < dir>	Define el directorio de trabajo predeterminado. En el modo local, este directorio se utiliza tanto para datos como para resultados. Ejemplo: `-directory c:/` o `-directory c:\\`
`-server_directory` < dir>	Define el directorio de servidor predeterminado para datos. El directorio de trabajo, especificado con la marca `-directory`, se utiliza para resultados.
`-execute`	Después del inicio, ejecuta cualquier ruta, estado o script que se haya cargado en el inicio. Si se carga un script además de una ruta o un estado, el script se ejecutará solo.
`-stream` < stream>	Carga en el inicio la ruta especificada. Se pueden especificar varias rutas, pero la última se definirá como la actual.
`-script` <script>	Carga en el inicio el script autónomo especificado. Se puede especificar además de una ruta o un estado, tal y como se describe a continuación, pero sólo se puede cargar un único script en el inicio.
`-model` < modelo>	En el inicio, carga el modelo generado (archivo de formato `.gm`) especificado.
`-state` < estado>	Carga en el inicio el estado especificado guardado.
`-project` < project>	Carga el proyecto especificado. Sólo se puede cargar un único proyecto en el inicio.
`-output` < output>	Carga en el inicio el objeto de resultados guardado (archivo de formato `.cou`).
`-help`	Muestra una lista de argumentos de la línea de comandos. Cuando se especifica esta opción, todos los demás argumentos se ignoran y se muestra la pantalla Ayuda.

Tabla 35. Argumentos del sistema (continuación)	
Argumento	**Comportamiento/Descripción**
-P < nombre> = < valor>	Se utiliza para definir un parámetro de inicio. También se puede utilizar para definir propiedades de nodos (parámetros de intervalo).

Nota: Los directorios predeterminados también se pueden establecer en la interfaz de usuario. Para acceder a las opciones en el menú Archivo, seleccione **Definir directorio** o **Definir directorio de servidor**.

Carga de varios archivos

Desde la línea de comandos puede cargar varias rutas, estados y resultados en el inicio repitiendo el argumento relevante para cada objeto cargado. Por ejemplo, para cargar y ejecutar dos rutas llamadas `report.str` y `train.str`, utilizaría el siguiente comando:

```
modelerclient -stream report.str -stream train.str -execute
```

Carga de objetos desde el IBM SPSS Collaboration and Deployment Services Repository

Puesto que puede cargar determinados objetos de un archivo o de IBM SPSS Collaboration and Deployment Services Repository (si tiene licencia), el prefijo de nombre de archivo `spsscr:` y, opcionalmente, `file:` (para objetos en disco) indica a IBM SPSS Modeler dónde buscar el objeto. El prefijo funciona con las siguientes marcas:

- `-stream`
- `-script`
- `-output`
- `-model`
- `-project`

Puede utilizar el prefijo para crear un URI que especifique la ubicación del objeto, por ejemplo, `-stream "spsscr:///folder_1/scoring_stream.str"`. La presencia del prefijo `spsscr:` requiere que se haya especificado una conexión válida con IBM SPSS Collaboration and Deployment Services Repository en el mismo mandato. Así, por ejemplo, el comando completo sería:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

Recuerde que en la línea de comandos *debe* utilizar un URI. El `REPOSITORY_PATH` más simple no se admite. (Sólo funciona en scripts.) Para obtener más detalles sobre los URI para objetos en el IBM SPSS Collaboration and Deployment Services Repository, consulte el tema “Acceso a objetos en el IBM SPSS Collaboration and Deployment Services Repository” en la página 51.

Argumentos de parámetros

Los parámetros se pueden utilizar como marcas durante la ejecución de la línea de comandos de IBM SPSS Modeler. En los argumentos de línea de mandatos, el distintivo `-P` se utiliza para indicar un parámetro del formulario `-P` <*Nombre*> = <*valor*>.

Los parámetros pueden ser:

- **Parámetros simples** (o parámetros utilizados directamente en expresiones CLEM).
- **Parámetros de intervalo**, también denominados propiedades de nodos. Estos parámetros se utilizan para modificar la configuración de los nodos en la ruta. Para obtener más información, consulte el tema “Visión general de propiedades de nodo” en la página 75.

- **Parámetros de la línea de comandos**, que son parámetros utilizados para alterar la invocación de IBM SPSS Modeler.

Por ejemplo, puede proporcionar contraseñas y nombres de usuario de orígenes de datos como una marca de la línea de comandos, por ejemplo:

```
modelerclient -stream response.str -P:databasenode.datasource="{\"ORA
10gR2\",user1,mypsw,false}"
```

El formato es el mismo que el del parámetro `datasource` de la propiedad de nodo `databasenode`. Para obtener más información, consulte: “propiedades de databasenode” en la página 93.

El último parámetro se debe establecer en `true` si está pasando una contraseña codificada. También tenga en cuenta que no deben utilizarse espacios iniciales delante del nombre de usuario y de la contraseña de la base de datos (a menos, por supuesto, que el nombre de usuario o la contraseña realmente contenga un espacio inicial).

Nota: Si el nodo tiene nombre, debe encerrar el nombre del nodo entre comillas dobles y utilizar una barra inclinada invertida como carácter de escape antes de las comillas. Por ejemplo, si el nodo de origen de datos del ejemplo anterior tiene el nombre `Source_ABC`, la entrada sería la siguiente:

```
modelerclient -stream response.str
-P:databasenode.\"Source_ABC\".datasource="{\"ORA 10gR2\",
 user1,mypsw,true}"
```

También se requiere una barra inclinada invertida delante de las comillas que identifican un parámetro estructurado, como en el siguiente ejemplo de origen de datos de TM1:

```
clemb -server -hostname 9.115.21.169 -port 28053 -username administrator
    -execute -stream C:\Share\TM1_Script.str -P:tm1import.pm_host="http://9.115.21.163:9510/
pmhub/pm"
    -P:tm1import.tm1_connection={\"SData\",\"\",\"admin\",\"apple\"}
    -P:tm1import.selected_view={\"SalesPriorCube\",\"salesmargin%\"}
```

Nota: Si el nombre de la base de datos (en la propiedad `datasource`) contiene uno o más espacios, puntos (llamados también "punto y aparte") o subrayados, puede utilizar el formato de "barra inclinada invertida y comillas dobles" para tratarlo como una cadena. Por ejemplo: `"{\"db2v9.7.6_linux\"}"` o: `"{\"TDATA 131\"}"`. Además, escriba siempre los valores de serie de `datasource` entre comillas dobles y llaves, como en el ejemplo siguiente: `"{\"SQL Server\",spssuser,abcd1234,false}"`.

Argumentos de conexión del servidor

El distintivo `-server` indica a IBM SPSS Modeler que debe conectarse a un servidor público, y los distintivos `-hostname`, `-use_ssl`, `-port`, `-username`, `-password`y `-domain` se utilizan para indicar a IBM SPSS Modeler cómo conectarse al servidor público. Si no se especifica ningún argumento -server, se utilizará el servidor predeterminado o local.

Ejemplos

Para conectarse con un servidor público:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Para conectarse con un clúster de servidores:

```
modelerclient -server -cluster "QA Machines" \
-spsscr_hostname pes_host -spsscr_port 8080 \
-spsscr_username asmith -spsscr_epassword xyz
```

Tenga en cuenta que la conexión a un clúster de servidores requiere el coordinador de procesos a través de IBM SPSS Collaboration and Deployment Services, por lo que el argumento `-cluster` debe utilizarse

en combinación con las opciones de conexión de repositorio (spsscr_*). Consulte el tema "Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository" en la página 69 para obtener más información.

Tabla 36. Argumentos de conexión del servidor

Argumento	Comportamiento/Descripción
-server	Ejecuta IBM SPSS Modeler en el modo servidor, conectando con un servidor público utilizando las marcas -hostname, -port, -username, -password y -domain.
-hostname < nombre>	Nombre de host del equipo servidor. Disponible en el modo servidor solamente.
-use_ssl	Especifica que la conexión debería utilizar SSL (secure socket layer). La marca es opcional, el parámetro predeterminado *no* utiliza SSL.
-port < número>	Número de puerto del servidor especificado. Disponible en el modo servidor solamente.
-cluster < nombre>	Especifica una conexión a un clúster de servidores en lugar de un servidor especificado; este argumento es una alternativa a los argumentos hostname, port y use_ssl. El nombre es el del clúster o un URI exclusivo que identifica el clúster en el IBM SPSS Collaboration and Deployment Services Repository. Coordinator of Processes gestiona el clúster de servidores a través de IBM SPSS Collaboration and Deployment Services. Para obtener más información, consulte el tema "Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository" en la página 69.
-username < nombre>	Nombre de usuario con el que iniciar sesión en el servidor. Disponible en el modo servidor solamente.
-password <password>	Contraseña con la que iniciar sesión en el servidor. Disponible en el modo servidor solamente. **Nota:** Si el argumento -password no se utiliza, se le pediría una contraseña.
-epassword < encodedpasswordstring>	Contraseña codificada con la que iniciar sesión en el servidor. Disponible en el modo servidor solamente. **Nota:** Se puede generar una contraseña codificada desde el menú Herramientas de la aplicación IBM SPSS Modeler.
-domain < nombre>	Dominio utilizado para iniciar sesión en el servidor. Disponible en el modo servidor solamente.
-P < nombre> = < valor>	Se utiliza para definir un parámetro de inicio. También se puede utilizar para definir propiedades de nodos (parámetros de intervalo).

Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository

Si desea almacenar o recuperar objetos de IBM SPSS Collaboration and Deployment Services a través de la línea de comandos, debe especificar una conexión válida con IBM SPSS Collaboration and Deployment Services Repository. Por ejemplo:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

La siguiente tabla enumera los argumentos que pueden utilizarse para configurar la conexión.

Tabla 37. Argumentos de conexión de IBM SPSS Collaboration and Deployment Services Repository

Argumento	**Comportamiento/Descripción**
`-spsscr_hostname` < nombre de host o dirección IP >	El nombre del host o la dirección IP del servidor en que se ha instalado IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_port` < número>	Número de puerto en el que el IBM SPSS Collaboration and Deployment Services Repository acepta las conexiones (normalmente 8080 de forma predeterminada).
`-spsscr_use_ssl`	Especifica que la conexión debería utilizar SSL (secure socket layer). La marca es opcional, el parámetro predeterminado *no* utiliza SSL.
`-spsscr_username` < nombre>	Nombre de usuario con el que iniciar la sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <password>	Contraseña con la que iniciar la sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` < contraseña codificada >	Contraseña cifrada con la que iniciar la sesión en el IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_providername` < nombre>	El proveedor de autenticación utilizado para iniciar la sesión en el IBM SPSS Collaboration and Deployment Services Repository (Active Directory o LDAP). Esto no resulta necesario si se utiliza el proveedor nativo (repositorio local).

IBM SPSS Analytic Server Argumentos de conexión

Si desea almacenar o recuperar objetos de IBM SPSS Analytic Server mediante la línea de comandos, debe indicar una conexión válida con IBM SPSS Analytic Server.

Nota: La ubicación predeterminada de Analytic Server se obtiene de SPSS Modeler Server. Los usuarios también pueden definir sus propias conexiones de Analytic Server a través de **Herramientas** > **Conexiones de Analytic Server**.

La siguiente tabla enumera los argumentos que pueden utilizarse para configurar la conexión.

Tabla 38. IBM SPSS Analytic Server Argumentos de conexión

Argumento	**Comportamiento/Descripción**
`-analytic_server_username`	Nombre de usuario con el que iniciar sesión en IBM SPSS Analytic Server.
`-analytic_server_password`	La contraseña con la que se inicia sesión en IBM SPSS Analytic Server.
`-analytic_server_epassword`	La contraseña codificada con la que se inicia sesión en IBM SPSS Analytic Server.
`-analytic_server_credential`	Las credenciales utilizadas para iniciar sesión en IBM SPSS Analytic Server.

Combinación de varios argumentos

Es posible combinar varios argumentos en un único archivo de comandos especificado en la invocación utilizando el símbolo @ seguido del nombre de archivo. De este modo podrá acortar la invocación de la línea de comandos y superar cualquier limitación del sistema operativo en la longitud del comando. Por

ejemplo, el siguiente mandato de inicio utiliza los argumentos especificados en el archivo al que hace referencia <commandFileName>.

```
modelerclient @<commandFileName>
```

Ponga el nombre del archivo y la ruta del archivo de comandos entre comillas si hay que incluir espacios, de la siguiente forma:

```
modelerclient @ "C:\Program
Files\IBM\SPSS\Modeler\nn\scripts\my_command_file.txt"
```

El archivo de comandos puede contener todos los argumentos especificados previamente a nivel individual en el inicio. Por ejemplo:

```
-stream report.str
-Porder.full_filename=APR_orders.dat
-Preport.filename=APR_report.txt
-execute
```

Cuando escriba y referencie archivos de comandos, asegúrese de cumplir estas restricciones:

- Utilice sólo un comando por línea.
- No incluya un argumento `@CommandFile` en un archivo de mandatos.

Capítulo 7. Referencia de propiedades

Visión general de referencia de propiedades

Puede especificar una serie de propiedades distintas para nodos, rutas, proyectos y supernodos. Algunas propiedades son comunes a todos los nodos, como el nombre, la anotación y la información sobre herramientas, mientras que otras son específicas para determinados tipos de nodos. Otras propiedades hacen referencia a operaciones de rutas de alto nivel, como el comportamiento del Supernodo o el almacenamiento en caché. Se puede acceder a las propiedades a través de la interfaz de usuario estándar (por ejemplo, al abrir un cuadro de diálogo para editar opciones para un nodo) y se pueden utilizar también de varias otras formas.

- Las propiedades se pueden modificar a través de los scripts, como se describe en esta sección. Para obtener más información, consulte “Sintaxis para propiedades” en la página 73.
- Las propiedades de los nodos se pueden utilizar en los parámetros de Supernodo.
- Las propiedades de nodo también se pueden utilizar como parte de una opción de línea de mandatos (utilizando el distintivo `-P`) al iniciar IBM SPSS Modeler.

En el contexto de los scripts de IBM SPSS Modeler, las propiedades de nodos y rutas se suelen llamar **parámetros de intervalo**. En esta guía, se denominan propiedades de nodos y rutas.

Sintaxis para propiedades

Las propiedades se pueden establecer con la sintaxis siguiente

```
OBJECT.setPropertyValue(PROPERTY, VALUE)
```

o bien:

```
OBJECT.setKeyedPropertyValue(PROPERTY, KEY, VALUE)
```

El valor de propiedades se puede recuperar usando la sintaxis siguiente:

```
VARIABLE = OBJECT.getPropertyValue(PROPERTY)
```

o bien:

```
VARIABLE = OBJECT.getKeyedPropertyValue(PROPERTY, KEY)
```

donde `OBJECT` es un nodo o salida, `PROPERTY` es el nombre de la propiedad de nodo al que la expresión se refiere, y `KEY` es el valor de la clave para las propiedades clave. Por ejemplo, la siguiente sintaxis se utiliza para buscar el nodo de filtro y, a continuación, establecer el valor predeterminado para incluir todos los campos y filtrar el campo `Age` en los datos en sentido descendente:

```
filternode = modeler.script.stream().findByType("filter", None)
filternode.setPropertyValue("default_include", True)
filternode.setKeyedPropertyValue("include", "Age", False)
```

Todos los nodos utilizados en IBM SPSS Modeler se pueden localizar utilizando la función de corriente `findByType(TYPE, LABEL)` . Al menos debe especificarse `TYPE` o `LABEL`.

Propiedades estructuradas

Hay dos formas en las que los scripts utilizan propiedades estructuradas para mejorar la claridad durante el análisis:

- Otorgando estructura a los nombres de las propiedades para los nodos complejos, como Tipo, Filtro o Equilibrar.
- Proporcionando un formato para especificar varias propiedades a la vez.

Estructuración para las interfaces complejas

Los scripts para los nodos con tablas y otras interfaces complejas, como, por ejemplo, los nodos Tipo, Filtro o Equilibrar, deben seguir una estructura determinada para realizar el análisis correctamente. Estas propiedades necesitan un nombre más complejo que el de un solo identificador. Este nombre se denomina clave. Por ejemplo, en un nodo Filtrar, cada campo disponible (en la parte superior) se activa o desactiva. Para poder consultar esta información, el nodo Filtrar almacena un elemento de información por campo (independientemente de que el campo sea verdadero o falso). Esta propiedad debe tener (o se le ha dado) el valor `True` o `False`. Suponga que un nodo Filtrar denominado `mynode` tiene (en sentido ascendente) un campo llamado `Age`. Para desactivar esto, establezca la propiedad `include`, con la clave `Age`, en el valor `False`, del modo siguiente:

```
mynode.setKeyedPropertyValue("include", "Age", False)
```

Estructuración para definir varias propiedades

Si hay muchos nodos, puede asignar más de una propiedad de nodo o ruta al mismo tiempo. Esto se denomina **comando de conjunto múltiple** o **bloque de conjuntos**.

En algunos casos, una propiedad estructurada puede ser bastante compleja. A continuación se muestra un ejemplo:

```
sortnode.setPropertyValue("keys", [["K", "Descending"], ["Age",
"Ascending"], ["Na", "Descending"]])
```

Otra ventaja de las propiedades estructuradas es la capacidad de definir varias propiedades en un nodo antes de que éste sea estable. De forma predeterminada, un conjunto múltiple define todas las propiedades del bloque antes de realizar una acción basada en una configuración de propiedades individuales. Por ejemplo, al definir un nodo Archivo fijo, el uso de dos pasos para definir las propiedades del campo daría lugar a errores porque el nodo no será constante hasta que las dos configuraciones sean válidas. La definición de las propiedades como un conjunto múltiple salva este problema al definir ambas propiedades antes de actualizar el modelo de datos.

Abreviaturas

Las abreviaturas estándar se utilizan en la sintaxis para las propiedades de nodos. El aprendizaje de las abreviaturas le ayudará en la creación de scripts.

Tabla 39. Abreviaturas estándar utilizadas en toda la sintaxis

Abreviación	**Descripción**
abs	Valor absoluto
lon	Longitud
min	Minimum
max	Máximo
correl	Correlación
covar	Covarianza
núm	Número o numérico
porcentaje	Porcentaje

Tabla 39. Abreviaturas estándar utilizadas en toda la sintaxis (continuación)	
Abreviación	**Descripción**
transp	Transparencia
xval	Validación cruzada
var	Varianza o variable (en nodos de origen)

Ejemplos de las propiedades node y stream

Las propiedades de nodos y rutas se pueden utilizar de varias formas con IBM SPSS Modeler. Normalmente se utilizan como parte de un script, bien un **script autónomo**, utilizado para automatizar rutas u operaciones o un **script de ruta**, utilizado para automatizar procesos en una sola ruta. Los parámetros de nodo se pueden especificar también utilizando las propiedades para los nodos del Supernodo. En el nivel más básico, las propiedades se pueden utilizar también como una opción de línea de comandos para iniciar IBM SPSS Modeler. Si utiliza el argumento `-p` como parte de la invocación de la línea de comandos, podrá utilizar una propiedad de ruta para cambiar una configuración de la ruta.

Tabla 40. Ejemplos de las propiedades node y stream	
Propiedad	**Descripción**
`s.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s`.
`s:samplenode.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s` que debe ser un nodo Muestrear.
`:samplenode.max_size`	Hace referencia a la propiedad `max_size` del nodo Muestrear de la ruta actual (debe haber sólo un nodo Muestrear).
`s:sample.max_size`	Hace referencia a la propiedad `max_size` del nodo denominado `s` que debe ser un nodo Muestrear.
`t.direction.Age`	Hace referencia al rol del campo *Edad* del nodo Tipo `t`.
`:.max_size`	*** NO ES LEGAL *** Debe especificar el nombre o el tipo de nodo.

El ejemplo `s:sample.max_size` muestra que no es necesario deletrear los tipos de nodos al completo.

El ejemplo `t.direction.Age` muestra que algunos nombres de intervalo se pueden estructurar por sí mismos, en aquellos casos en que los atributos de un nodo sean más complejos que los intervalos individuales con valores individuales. Dichos intervalos se denominan **estructurados** o **complejos**.

Visión general de propiedades de nodo

Cada tipo de nodo tiene su propio conjunto de propiedades legales y cada propiedad tiene un tipo. Este tipo puede ser un tipo general, número, marca o cadena, en cuyo caso, las configuraciones de la propiedad se forzarán en el tipo correcto. Surgirá un error en caso de que no se puedan forzar. También se puede dar el caso de que la referencia de la propiedad pueda especificar el rango de valores legales como `Discard`, `PairAndDiscard` e `IncludeAsText`, en cuyo caso se producirá un error si se utiliza otro valor. Las propiedades de distintivo deben leerse o establecerse utilizando los valores de `true` y `false`. (Las variaciones que contengan `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` o `0` también se reconocen al configurar los valores, pero pueden provocar errores al leer los valores de propiedad en algunos casos. El resto de valores se consideran verdaderos. Utilizar `true` y `false` de forma coherente evitará cualquier confusión.) En las tablas de referencia de esta guía, las propiedades estructuradas se indican como tales en la columna **Descripción de la propiedad** y se proporcionan los formatos de uso.

Propiedades de nodos comunes

Existen varias propiedades que son comunes a todos los nodos (incluidos los Supernodos) en IBM SPSS Modeler.

Tabla 41. Propiedades comunes de nodos

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`use_custom_name`	*Flag*	
`name`	*serie*	Propiedad de sólo lectura que lee el nombre (automático o personalizado) para un nodo del lienzo.
`custom_name`	*serie*	Especifica un nombre personalizado para el nodo.
`tooltip`	*serie*	
`annotation`	*serie*	
`keywords`	*serie*	Ranura estructurada que especifica una lista de palabras clave asociadas al objeto (por ejemplo, `["Keyword1" "Keyword2"]`).
`cache_enabled`	*Flag*	
`node_type`	`source_supernode` `process_supernode` `terminal_supernode` todos los nombres de nodos tal como se especifican para scripts	Propiedad de sólo lectura utilizada para hacer referencia a un nodo por tipo. Por ejemplo, en lugar de hacer referencia a un nodo sólo por nombre, como por ejemplo `real_income`, también puede especificar el tipo, como por ejemplo `userinputnode` o `filternode`.

Las propiedades específicas del Supernodo se tratan aparte como con los demás nodos. Para obtener más información, consulte el tema Capítulo 21, “Propiedades de supernodo”, en la página 463.

Capítulo 8. Propiedades de ruta

Los scripts pueden controlar una serie de propiedades de la ruta. Para hacer referencia a propiedades de ruta, debe establecer el método de ejecución para que utilice scripts:

```
stream = modeler.script.stream()
stream.setPropertyValue("execute_method", "Script")
```

Ejemplo

La propiedad de nodo se utiliza para hacer referencia a los nodos en la ruta actual. El siguiente script de ruta muestra un ejemplo:

```
stream = modeler.script.stream()
annotation = stream.getPropertyValue("annotation")

annotation = annotation + "\n\nThis stream is called \"" + stream.getLabel() + "\" and
 contains the following nodes:\n"

for node in stream.iterator():
    annotation = annotation + "\n" + node.getTypeName() + " node called \"" + node.getLabel()
 + "\""

stream.setPropertyValue("annotation", annotation)
```

El ejemplo anterior utiliza la propiedad node para crear una lista con todos los nodos de la ruta y escribir dicha lista en las anotaciones. La anotación generada tendrá el siguiente aspecto:

```
This stream is called "druglearn" and contains the following nodes:

type node called "Define Types"
derive node called "Na_to_K"
variablefile node called "DRUG1n"
neuralnetwork node called "Drug"
c50 node called "Drug"
filter node called "Discard Fields"
```

Las propiedades de la ruta se describen en la tabla siguiente.

Tabla 42. Propiedades de ruta

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
execute_method	Normal Script	

Tabla 42. Propiedades de ruta (continuación)		
Nombre de la propiedad	**Tipo de datos**	**Descripción de la propiedad**
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	*number*	
date_2digit_baseline	*number*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	*Flag*	
import_datetime_as_string	*Flag*	
decimal_places	*number*	
decimal_symbol	Default Period Comma	
angles_in_radians	*Flag*	
use_max_set_size	*Flag*	
max_set_size	*number*	

Tabla 42. Propiedades de ruta (continuación)		
Nombre de la propiedad	**Tipo de datos**	**Descripción de la propiedad**
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	*Flag*	Se utiliza para actualizar los nodos de origen de forma automática al realizar la ejecución de la ruta.
script	*serie*	
annotation	*serie*	
name	*serie*	**Nota:** Esta propiedad es de sólo lectura. Si desea cambiar el nombre de una ruta, debe guardarla con un nombre diferente.
parameters		Utilice esta propiedad para actualizar los parámetros de ruta desde dentro de un script autónomo.
nodes		Consulte la información detallada que se muestra a continuación.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*Boolean*	
stream_rewriting_maximise_sql	*Boolean*	
stream_rewriting_optimise_clem_execution	*Boolean*	
stream_rewriting_optimise_syntax_execution	*Boolean*	
enable_parallelism	*Boolean*	
sql_generation	*Boolean*	
database_caching	*Boolean*	
sql_logging	*Boolean*	
sql_generation_logging	*Boolean*	
sql_log_native	*Boolean*	
sql_log_prettyprint	*Boolean*	
record_count_suppress_input	*Boolean*	

Tabla 42. Propiedades de ruta (continuación)		
Nombre de la propiedad	**Tipo de datos**	**Descripción de la propiedad**
record_count_feedback_interval	*integer*	
use_stream_auto_create_node_settings	*Boolean*	Si es true, se utilizan los valores específicos de la ruta, de lo contrario, se utilizan las preferencias de usuario.
create_model_applier_for_new_models	*Boolean*	Si es true, cuando un constructor de modelos crea un modelo nuevo y no tiene enlaces de actualización activos, se añade un nuevo aplicador de modelos. **Nota:** Si utiliza IBM SPSS Modeler Batch versión 15, debe añadir de forma explícita el aplicador de modelos dentro de su script.
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Define el tipo de enlace creado cuando se añade automáticamente un nodo aplicador de modelos.
create_source_node_from_builders	*Boolean*	Si es true, cuando un constructor de modelos crea un resultado de origen nuevo y no tiene enlaces de actualización activos, se añade un nuevo nodo de origen.
create_source_node_update_links	createEnabled createDisabled doNotCreate	Define el tipo de enlace creado cuando se añade automáticamente un nodo de origen.
has_coordinate_system	*Boolean*	Si es verdadero, aplica un sistema de coordenadas a la ruta completa.
coordinate_system	*serie*	El nombre del sistema de coordenadas proyectadas seleccionado.
deployment_area	ModelRefresh Scoring None	Selecciona cómo desea desplegar la ruta. Si este valor se establece en None, no se utilizan otras entradas de despliegue.
scoring_terminal_node_id	*serie*	Elija la rama de puntuación en la ruta. Puede ser cualquier terminal de la ruta.
scoring_node_id	*serie*	Elija el nugget de la rama de puntuación.
model_build_node_id	*serie*	Elija el nodo de modelado de la ruta.

Capítulo 9. Propiedades de nodos de origen

Propiedades comunes de nodos de origen

Las propiedades comunes a todos los nodos de origen se enumeran a continuación, con información sobre nodos específicos en los temas siguientes.

Ejemplo 1

```
varfilenode = modeler.script.stream().create("variablefile", "Var. File")
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
varfilenode.setKeyedPropertyValue("check", "Age", "None")
varfilenode.setKeyedPropertyValue("values", "Age", [1, 100])
varfilenode.setKeyedPropertyValue("type", "Age", "Range")
varfilenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Ejemplo 2

Este script presupone que el archivo de datos especificado contiene un campo Region que representa una cadena de varias líneas.

```
from modeler.api import StorageType
from modeler.api import MeasureType

# Create a Variable File node that reads the data set containing
# the "Region" field
varfilenode = modeler.script.stream().create("variablefile", "My Geo Data")
varfilenode.setPropertyValue("full_filename", "C:/mydata/mygeodata.csv")
varfilenode.setPropertyValue("treat_square_brackets_as_lists", True)

# Override the storage type to be a list...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region",
StorageType.LIST)
# ...and specify the type if values in the list and the list depth
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region",
StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Now change the measurement to indentify the field as a geospatial value...
varfilenode.setKeyedPropertyValue("measure_type", "Region",
MeasureType.GEOSPATIAL)
# ...and finally specify the necessary information about the specific
# type of geospatial object
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
varfilenode.setKeyedPropertyValue("coordinate_system", "Region",
 "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabla 43. Propiedades comunes de nodos de origen

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
direction	Input Target Both None Partition Split Frequency RecordID	Propiedad con clave para los roles de los campos. Formato de uso: NODE.direction.FIELDNAME **Nota:** Los valores In y Out han quedado en desuso. Es posible que ya no puedan utilizarse en futuras versiones.
type	Range Flag Set Typeless Discrete Ordered Set Default	Tipo de campo. Si se establece esta propiedad como *Default*, se borrará cualquier configuración del parámetro values y si value_mode tiene el valor *Specify*, se restablecerá a *Read*. Si value_mode se establece en *Pass* o *Read*, la configuración de type no le afectará. Formato de uso: NODE.type.FIELDNAME
storage	Unknown String Integer Real Time Date Timestamp	Propiedad con clave de solamente lectura para el tipo de almacenamiento de campos. Formato de uso: NODE.storage.FIELDNAME

Tabla 43. Propiedades comunes de nodos de origen (continuación)

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
check	None Nullify Coerce Discard Warn Abort	Propiedad con clave para la comprobación del rango y el tipo de campo. Formato de uso: NODE.check.FIELDNAME
values	[*value value*]	Para un campo continuo (rango), el primer valor es el mínimo y el último valor es el máximo. Para campos nominales (conjunto), especifique todos los valores. Para los campos marca, el primer valor representa *falso* y el último, *verdadero*. La configuración de esta propiedad establece de forma automática la propiedad value_mode en *Specify*. El almacenamiento se determina en función del primer valor de la lista, por ejemplo, si el primer valor es una *cadena*, el almacenamiento se establece en Cadena. Formato de uso: NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	Determina la forma en que se han establecido los valores para un campo en la siguiente lectura de datos. Formato de uso: NODE.value_mode.FIELDNAME Tenga en cuenta que no puede establecer esta propiedad directamente en *Specify*. Para utilizar valores específicos, establezca la propiedad values.
default_value_mode	Read Pass	Especifica el método predeterminado para configurar los valores de todos los campos. Formato de uso: NODE.default_value_mode Esta configuración puede anularse para determinados campos mediante la propiedad value_mode.

Tabla 43. Propiedades comunes de nodos de origen (continuación)

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
`extend_values`	*Flag*	Se aplica cuando `value_mode` se establece en *Read*. Establézcala en *T* para añadir nuevos valores de lectura a los valores existentes del campo. Establézcala en *F* para descartar los valores existentes y favorecer a los nuevos valores de lectura. Formato de uso: `NODE.extend_values.FIELDNAME`
`value_labels`	*serie*	Se utiliza para especificar una etiqueta de valor. Tenga en cuenta que estos valores se deben especificar primero.
`enable_missing`	*Flag*	Cuando está definida como *T*, activa el seguimiento de los valores perdidos para el campo. Formato de uso: `NODE.enable_missing.FIELDNAME`
`missing_values`	[*value value ...*]	Especifica los valores de datos que denotan los datos perdidos. Formato de uso: `NODE.missing_values.FIELDNAME`
`range_missing`	*Flag*	Cuando esta propiedad se establece como *T*, especifica si se define un rango de valores perdidos (en blanco) para un campo. Formato de uso: `NODE.range_missing.FIELDNAME`
`missing_lower`	*serie*	Si `range_missing` es verdadero, especifica el límite inferior del rango de valores perdidos. Formato de uso: `NODE.missing_lower.FIELDNAME`
`missing_upper`	*serie*	Si `range_missing` es verdadero, especifica el límite superior del rango de valores perdidos. Formato de uso: `NODE.missing_upper.FIELDNAME`

Tabla 43. Propiedades comunes de nodos de origen (continuación)

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
null_missing	*Flag*	Cuando esta propiedad se establece en *T*, los valores nulos (valores no definidos que se muestran como $null$ en el software) se consideran valores perdidos. Formato de uso: NODE.null_missing.FIELDNAME
whitespace_missing	*Flag*	Cuando esta propiedad está definida como *T*, los valores que solamente contienen un espacio en blanco (espacios, tabulaciones y líneas nuevas) se consideran valores perdidos. Formato de uso: NODE.whitespace_missing.FIELDNAME
description	*serie*	Se utiliza para especificar la descripción o etiqueta de un campo.
default_include	*Flag*	Propiedad con clave para especificar si el comportamiento predeterminado es para pasar o filtrar los campos: NODE.default_include Ejemplo: set mynode:filternode.default_include = false
include	*Flag*	Propiedad con clave que se utiliza para determinar si los campos individuales se han incluido o se han filtrado: NODE.include.FIELDNAME.
new_name	*serie*	

Tabla 43. Propiedades comunes de nodos de origen (continuación)

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Esta propiedad con clave es similar a type en cuanto a que puede utilizarse para definir la medición asociada al campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores MeasureType, mientras que la función de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad con clave define el tipo de medición asociado con los valores subyacentes.

Tabla 43. Propiedades comunes de nodos de origen (continuación)

Nombre de la propiedad	Tipo de datos	Descripción de la propiedad
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	En campos geoespaciales, esta propiedad con clave define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores.
has_coordinate_system	*Boolean*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas
coordinate_system	*serie*	En campos geoespaciales, esta propiedad con clave define el sistema de coordenadas para este campo.
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Esta propiedad con clave es similar a custom_storage en cuanto a que puede utilizarse para definir el almacenamiento de alteración temporal para el campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores StorageType, mientras que la función de obtención siempre devolverá los valores StorageType.

Tabla 43. Propiedades comunes de nodos de origen (continuación)		
Nombre de la propiedad	**Tipo de datos**	**Descripción de la propiedad**
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Para campos de lista, esta propiedad con clave especifica el tipo de almacenamiento de los valores subyacentes.
custom_list_depth	*integer*	Para campos de lista, esta propiedad con clave especifica la profundidad del campo.
max_list_length	*integer*	Solo está disponible para datos con un nivel de medición de *Geoespacial* o *Colección*. Establezca la longitud máxima de la lista especificando el número de elementos que puede contener la lista.
max_string_length	*integer*	Solo está disponible para datos *sin tipos* y se utiliza al generar SQL para crear una tabla. Entre el valor de la serie más larga en los datos; esto genera una columna en la tabla que es lo suficientemente grande para contener la serie.

Propiedades de asimport

El origen de Analytic Server permite ejecutar una ruta en un sistema de archivos distribuido de Hadoop (HDFS en sus siglas inglesas).

Ejemplo

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false
","","","","",""])
```

Tabla 44. Propiedades de asimport		
Propiedades de asimport	**Tipo de datos**	**Descripción de la propiedad**
data_source	*serie*	Nombre del origen de datos.

Tabla 44. Propiedades de asimport (continuación)		
Propiedades de asimport	**Tipo de datos**	**Descripción de la propiedad**
use_default_as	*Boolean*	Si se establece en True, utiliza la conexión Analytic Server predeterminada configurada en el archivo options.cfg del servidor. Si se establece en False, utiliza la conexión de este nodo.
connection	["string","string","string", "string","string","string","string", "string" ,"string","string", "string" ,"string"]	Una propiedad de la lista que contiene los detalles de conexión de Analytic Server. El formato es: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] Donde: is_secure_connect: indica si se utiliza una conexión segura, y es true o false. use-kerberos-auth: indica si se utiliza autenticación Kerberos y es true o false. enable-kerberos-debug: indica si se utiliza el modo de depuración de la autenticación Kerberos y es true o false.

Propiedades del nodo cognosimport

El nodo de origen de IBM Cognos importa datos de bases de datos Cognos Analytics .

Ejemplo

```
node = stream.create("cognosimport", "My node")
node.setPropertyValue("cognos_connection", ["http://mycogsrv1:9300/p2pd/
servlet/dispatch",
 True, "", "", ""])
node.setPropertyValue("cognos_package_name", "/Public Folders/GOSALES")
node.setPropertyValue("cognos_items", ["[GreatOutdoors].[BRANCH].
[BRANCH_CODE]", "[GreatOutdoors]
.[BRANCH].[COUNTRY_CODE]"])
```

Tabla 45. propiedades del nodo cognosimport

Propiedades del nodo `cognosimport`	**Tipo de datos**	**Descripción de la propiedad**

`mode`	`Data`
	
	`Report`
		Especifica si se importarán los datos de Cognos (valor predeterminado) o informes.

Tabla 45. propiedades del nodo cognosimport (continuación)

Propiedades del nodo cognosimport	**Tipo de datos**	**Descripción de la propiedad**
cognos_connection	*["string",flag,"string", "string", "string"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de Cognos. El formato es: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` donde: `Cognos_server_URL` es la URL del servidor de Cognos que contiene el origen. `login_mode` indica si se utiliza el inicio de sesión anónimo, y es `true` o `false`; si se establece en `true`, los campos siguientes deben establecerse en `""`. `namespace` especifica el proveedor de autenticación de seguridad utilizado para registrarse en el servidor. `username` y `password` son los utilizados para registrarse en el servidor de Cognos. En lugar de `login_mode`, también hay disponibles las modalidades siguientes: • `anonymousMode`. For example: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. For example: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]` • `storedCredentialMode`. For example: `['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]` Donde `stored_credential_name` es el nombre de una credencial de Cognos del repositorio.

Tabla 45. propiedades del nodo cognosimport (continuación)		
Propiedades del nodo cognosimport	**Tipo de datos**	**Descripción de la propiedad**
cognos_package_name	*serie*	La ruta y el nombre del paquete de Cognos desde el que está importando objetos de datos, por ejemplo: /Public Folders/GOSALES **Nota:** solo es válida la barra diagonal.
cognos_items	*["campo", "campo", ..., "campo"]*	El nombre de uno o más objetos de datos que van a importarse. El formato de *campo* es [*espacio de nombre*].[*asunto de consulta*].[*elemento de consulta*]
cognos_filters	*campo*	El nombre de uno o más filtros que van a aplicarse antes de importar datos.
cognos_data_parameters	*lista*	Valores de parámetros de solicitud de datos. Los pares nombre-valor van encerrados entre corchetes, los pares múltiples están separados por comas y toda la cadena está encerrada entre corchetes. Formato: [["*param1*", "*valor*"],...,["*paramN*", "*valor*"]]
cognos_report_directory	*campo*	La ruta de Cognos de una carpeta o paquete de la que importar informes, por ejemplo: /Public Folders/GOSALES **Nota:** solo es válida la barra diagonal.
cognos_report_name	*campo*	La ruta y nombre dentro de la ubicación del informe de un informe que se ha de importar.
cognos_report_parameters	*lista*	Valores de parámetros de informe. Los pares nombre-valor van encerrados entre corchetes, los pares múltiples están separados por comas y toda la cadena está encerrada entre corchetes. Formato: [["*param1*", "*valor*"],...,["*paramN*", "*valor*"]]

propiedades de databasenode

Puede utilizar el nodo Base de datos para importar datos de una variedad de otros paquetes utilizando ODBC (Open Database Connectivity), incluidos Microsoft SQL Server, Db2, Oracle y otros.

Ejemplo

```
import modeler.api
stream = modeler.script.stream()
node = stream.create("database", "My node")
node.setPropertyValue("mode", "Table")
node.setPropertyValue("query", "SELECT * FROM drug1n")
node.setPropertyValue("datasource", "Drug1n_db")
node.setPropertyValue("username", "spss")
node.setPropertyValue("password", "spss")
node.setPropertyValue("tablename", ".Drug1n")
```

Tabla 46. propiedades de databasenode

Propiedades de databasenode	Tipo de datos	Descripción de la propiedad
mode	Table Query	Especifique *Table* para conectarse a una tabla de base de datos mediante los controles del cuadro de diálogo, o *Query* para realizar una consulta a la base de datos seleccionada mediante SQL.
datasource	*serie*	Nombre de la base de datos (consulte la siguiente nota).
username	*serie*	Detalles de conexión de la base de datos (consulte la siguiente nota).
password	*serie*	
credential	*serie*	Nombre de la credencial almacenada en IBM SPSS Collaboration and Deployment Services. Se puede utilizar en lugar de las propiedades username y password. El nombre de usuario y la contraseña de la credencial deben coincidir con el nombre de usuario y la contraseña necesarios para acceder a la base de datos
use_credential		Se establece en True o False.
epassword	*serie*	Especifica una contraseña codificada como una alternativa a codificar una contraseña en un script. Para obtener más información, consulte el tema "Generación de una contraseña codificada" en la página 54. Esta propiedad es de sólo lectura durante la ejecución.
tablename	*serie*	Nombre de la tabla a la que se desea tener acceso.

Tabla 46. propiedades de databasenode (continuación)		
Propiedades de databasenode	**Tipo de datos**	**Descripción de la propiedad**
strip_spaces	None Left Right Both	Opciones para descartar los espacios iniciales y finales en las cadenas.
use_quotes	AsNeeded Always Never	Especifica si los nombres de columna y tabla aparecen entre comillas cuando las consultas se envían a la base de datos (en el caso, por ejemplo, de que contengan espacios o signos de puntuación).
query	*serie*	Especifica el código SQL para la consulta que desea enviar.

Nota: Si el nombre de la base de datos (en la propiedad datasource) contiene espacios, entonces en vez de las propiedades individuales para datasource, username y password, utilice un único origen de datos en el siguiente formato:

Tabla 47. Propiedades de databasenode - específicas de datasource		
Propiedades de databasenode	**Tipo de datos**	**Descripción de la propiedad**
datasource	*serie*	Formato: [database_name,username,password[,true | false]] El último parámetro se usa con contraseñas cifradas. Si se define como true, la contraseña se cifrará antes de usarse.

Utilice este formato también si está cambiando el origen de datos, sin embargo, si tan sólo desea cambiar el nombre de usuario o contraseña, puede usar las propiedades username o password.

Propiedades de datacollectionimportnode

El nodo Importación de datos de Data Collection importa datos de encuesta basándose en el modelo de datos de Data Collection utilizado por productos de investigación de mercado. Se debe instalar la biblioteca de datos de Data Collection para utilizar este nodo.

Ejemplo

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Program Files/IBM/SPSS/
DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("casedata_name", "mrQvDsc")
node.setPropertyValue("casedata_source_type", "File")
```

```
node.setPropertyValue("casedata_file", "C:/Program Files/IBM/SPSS/
DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

Tabla 48. propiedades de datacollectionimportnode

Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
metadata_name	*serie*	El nombre del MDSC. El valor especial DimensionsMDD indica que se debería utilizar el documento de metadatos de Data Collection estándar. Otro posibles valores podrían ser: mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC El valor especial none indica que no existe ningún MDSC.
metadata_file	*serie*	Nombre del archivo en el que se almacenan los metadatos.

Tabla 48. propiedades de datacollectionimportnode (continuación)		
Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
casedata_name	*serie*	El nombre del CDSC. Entre los posibles valores se encuentran: mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc El valor especial none indica que no existe ningún CDSC.
casedata_source_type	Unknown File Folder UDL DSN	Indica el tipo de origen del CDSC.
casedata_file	*serie*	Cuando casedata_source_type es *File*, especifica el archivo que contiene los datos de casos.
casedata_folder	*serie*	Cuando casedata_source_type es *Folder*, especifica la carpeta que contiene los datos de casos.
casedata_udl_string	*serie*	Cuando casedata_source_type es *UDL*, especifica la cadena de conexión OLD-DB del origen de datos que contiene los datos de casos.

Tabla 48. propiedades de datacollectionimportnode (continuación)

Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
casedata_dsn_string	*serie*	Cuando casedata_source_type es *DSN*, especifica la cadena de conexión ODBC del origen de datos.
casedata_project	*serie*	Al leer datos de casos de una base de datos de Data Collection, puede escribir el nombre del proyecto. Para el resto de tipos de datos de casos, esta configuración se deberá dejar en blanco.
version_import_mode	All Latest Specify	Define el modo en que deben tratarse las versiones.
specific_version	*serie*	Cuando version_import_mode es *Specify*, define la versión de los datos de casos que se van a importar.
use_language	*serie*	Determina si deben usarse las etiquetas de un idioma concreto.
language	*serie*	Si use_language es verdadero, define el código de idioma que se va a usar en la importación. Este código de idioma debe incluirse entre aquellos disponibles en los datos de casos.
use_context	*serie*	Determina si se debe importar un contexto específico. Los contextos se utilizan para modificar la descripción asociada con las respuestas.
context	*serie*	Cuando use_context es verdadero, define el contexto de la importación. Este contexto debe encontrarse entre aquellos disponibles en los datos de casos.
use_label_type	*serie*	Determina si se debe importar un tipo de etiqueta específico.
label_type	*serie*	Cuando use_label_type es verdadero, define el tipo de etiqueta de la importación. Este tipo de etiqueta debe encontrarse entre aquellos disponibles en los datos de casos.
user_id	*serie*	En el caso de las bases de datos que requieren un inicio de sesión explícito, puede proporcionar un ID de usuario y una contraseña para acceder al origen de datos.
password	*serie*	

Tabla 48. propiedades de datacollectionimportnode (continuación)		
Propiedades de datacollectionimportnode	**Tipo de datos**	**Descripción de la propiedad**
import_system_variables	Common None All	Especifica las variables del sistema que se importan.
import_codes_variables	*Flag*	
import_sourcefile_variables	*Flag*	
import_multi_response	MultipleFlags Single	

Propiedades de excelimportnode

El nodo de importación Excel importa datos de Microsoft Excel en el formato de archivo .xlsx. No es necesario un origen de datos ODBC.

Ejemplos

```
#To use a named range:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("use_named_range", True)
node.setPropertyValue("named_range", "DRUG")
node.setPropertyValue("read_field_names", True)

#To use an explicit range:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("worksheet_mode", "Name")
node.setPropertyValue("worksheet_name", "Drug")
node.setPropertyValue("explicit_range_start", "A1")
node.setPropertyValue("explicit_range_end", "F300")
```

Tabla 49. propiedades de excelimportnode		
Propiedad de excelimportnode	**Tipo de datos**	**Descripción de la propiedad**
excel_file_type	Excel2007	
full_filename	*serie*	El nombre completo del archivo, incluyendo la vía de acceso.
use_named_range	*boolean*	Si usar un rango con nombre. Si es verdadero, la propiedad named_range se utiliza para especificar el rango de lectura y se ignoran el resto de configuraciones de rango de datos y hojas de trabajo.

Tabla 49. propiedades de excelimportnode (continuación)		
Propiedad de excelimportnode	**Tipo de datos**	**Descripción de la propiedad**
named_range	*serie*	
worksheet_mode	Index Name	Determina si la hoja de trabajo se define por el índice o por el nombre.
worksheet_index	*integer*	Índice de la hoja de trabajo que se va a leer, siendo 0 la primera hoja de trabajo, 1 la segunda, etc.
worksheet_name	*serie*	Nombre de la hoja de trabajo que se va a leer.
data_range_mode	FirstNonBlank ExplicitRange	Especifica cómo debe establecerse el rango.
blank_rows	StopReading ReturnBlankRows	Cuando data_range_mode es *FirstNonBlank*, especifica cómo deben tratarse las filas en blanco.
explicit_range_start	*serie*	Cuando data_range_mode es *ExplicitRange*, especifica el punto de partida del rango de lectura.
explicit_range_end	*serie*	
read_field_names	*boolean*	Determina si la primera fila del rango concreto debería usarse como nombres de campo (columna).
scanLineCount	*integer*	Especifica el número de filas que se deben explorar para la columna y el tipo de almacenamiento. El valor predeterminado es 200.

Propiedades de extensionimportnode

Con el nodo de Importación de extensión, puede ejecutar scripts R o Python para Spark para importar datos.

Ejemplo de Python para Spark

```
##### Script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_importer", "extension_importer")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import spss.pyspark
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()
```

```
_schema = StructType([StructField('id', LongType(), nullable=False), \
StructField('age', LongType(), nullable=True), \
StructField('Sex', StringType(), nullable=True), \
StructField('BP', StringType(), nullable=True), \
StructField('Cholesterol', StringType(), nullable=True), \
StructField('K', DoubleType(), nullable=True), \
StructField('Na', DoubleType(), nullable=True), \
StructField('Drug', StringType(), nullable=True)])

if cxt.isComputeDataModelOnly():
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    if df is None:
        drugList=[(1,23,'F','HIGH','HIGH',0.792535,0.031258,'drugY'), \
(2,47,'M','LOW','HIGH',0.739309,0.056468,'drugC'),\
              (3,47,'M','LOW','HIGH',0.697269,0.068944,'drugC'),\
              (4,28,'F','NORMAL','HIGH',0.563682,0.072289,'drugX'),\
              (5,61,'F','LOW','HIGH',0.559294,0.030998,'drugY'),\
              (6,22,'F','NORMAL','HIGH',0.676901,0.078647,'drugX'),\
              (7,49,'F','NORMAL','HIGH',0.789637,0.048518,'drugY'),\
              (8,41,'M','LOW','HIGH',0.766635,0.069461,'drugC'),\
              (9,60,'M','NORMAL','HIGH',0.777205,0.05123,'drugY'),\
              (10,43,'M','LOW','NORMAL',0.526102,0.027164,'drugY')]
        sqlcxt = cxt.getSparkSQLContext()
        rdd = cxt.getSparkContext().parallelize(drugList)
        print 'pyspark read data count = '+str(rdd.count())
        df = sqlcxt.createDataFrame(rdd, _schema)

    cxt.setSparkOutputData(df)
"""

node.setPropertyValue("python_syntax", python_script)
```

Ejemplo de R

```
#### Script example for R
node.setPropertyValue("syntax_type", "R")

R_script = """# 'JSON Import' Node v1.0 for IBM SPSS Modeler
# 'RJSONIO' package created by Duncan Temple Lang - http://cran.r-project.org/web/packages/
RJSONIO
# 'plyr' package created by Hadley Wickham http://cran.r-project.org/web/packages/plyr
# Node developer: Danil Savine - IBM Extreme Blue 2014
# Description: This node allows you to import into SPSS a table data from  a JSON.
# Install function for packages
packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x,character.only=TRUE)){
    install.packages(pkgs=x,repos="http://cran.r-project.org")
    require(x,character.only=TRUE)
  }
}
#  packages
packages(RJSONIO)
packages(plyr)
### This function is used to generate automatically the dataModel
getMetaData <- function (data) {
  if (dim(data)[1]<=0) {

    print("Warning : modelerData has no line, all fieldStorage fields set to strings")
    getStorage <- function(x){return("string")}

  } else {

    getStorage <- function(x) {
      res <- NULL
      #if x is a factor, typeof will return an integer so we treat the case on the side
      if(is.factor(x)) {
        res <- "string"
      } else {
        res <- switch(typeof(unlist(x)),
                      integer = "integer",
                      double = "real",
                      character = "string",
                      "string")
      }
      return (res)
    }
```

```
  }

  col = vector("list", dim(data)[2])
  for (i in 1:dim(data)[2]) {
    col[[i]] <- c(fieldName=names(data[i]),
                  fieldLabel="",
                  fieldStorage=getStorage(data[i]),
                  fieldMeasure="",
                  fieldFormat="",
                  fieldRole="")
  }
  mdm<-do.call(cbind,col)
  mdm<-data.frame(mdm)
  return(mdm)
}
# From JSON to a list
txt <- readLines('C:/test.json')
formatedtxt <- paste(txt, collapse = '')
json.list <- fromJSON(formatedtxt)
  # Apply path to json.list
  if(strsplit(x='true', split='
  ' ,fixed=TRUE)[[1]][1]) {
    path.list <- unlist(strsplit(x='id_array', split=','))
    i = 1
    while(i<length(path.list)+1){
      if(is.null(getElement(json.list, path.list[i]))){
         json.list <- json.list[[1]]
      }else{
      json.list <- getElement(json.list, path.list[i])
      i <- i+1
      }
    }
  }
# From list to dataframe via unlisted json
i <-1
filled <- data.frame()
while(i < length(json.list)+ 1){
  unlisted.json <- unlist(json.list[[i]])
  to.fill <- data.frame(t(as.data.frame(unlisted.json, row.names = names(unlisted.json))),
stringsAsFactors=FALSE)
  filled <- rbind.fill(filled,to.fill)
  i <-  1 + i
}
# Export to SPSS Modeler Data
modelerData <- filled
print(modelerData)
modelerDataModel <- getMetaData(modelerData)
print(modelerDataModel)

"""

node.setPropertyValue("r_syntax", R_script)
```

Tabla 50. Propiedades de extensionimportnode

Propiedades de extensionimportnode	**Tipo de datos**	**Descripción de la propiedad**
syntax_type	*E* *Python*	Especifica qué script se ejecuta: R o Python (R es el valor predeterminado).
r_syntax	*serie*	Sintaxis de scripts R a ejecutar.
python_syntax	*serie*	Sintaxis de scripts Python a ejecutar.

Propiedades de fixedfilenode

El nodo Archivo fijo importa datos desde archivos de texto de campo fijo; esto es, archivos cuyos campos no están delimitados pero empiezan en la misma posición y tienen una longitud fija. Los datos heredados o generados por la máquina se suelen almacenar en formato de campo fijo.

Ejemplo

```
node = stream.create("fixedfile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("record_len", 32)
node.setPropertyValue("skip_header", 1)
node.setPropertyValue("fields", [["Age", 1, 3], ["Sex", 5, 7], ["BP", 9,
10], ["Cholesterol",
 12, 22], ["Na", 24, 25], ["K", 27, 27], ["Drug", 29, 32]])
node.setPropertyValue("decimal_symbol", "Period")
node.setPropertyValue("lines_to_scan", 30)
```

Tabla 51. Propiedades de fixedfilenode

Propiedad de `fixedfilenode`	**Tipo de datos**	**Descripción de la propiedad**
`record_len`	*number*	Especifica el número de caracteres de cada registro.
`line_oriented`	*Flag*	Omite el carácter de nueva línea al final de cada registro.

`decimal_symbol`	`Default`
	
	`Comma`
	
	`Period`
		Tipo de separador decimal utilizado en el origen de datos.

`skip_header`	*number*	Especifica el número de líneas que se ignorarán al principio del primer registro. Esto resulta útil para ignorar las cabeceras de columna.
`auto_recognize_datetime`	*Flag*	Especifica si las fechas o las horas se identifican automáticamente en los datos de origen.
`lines_to_scan`	*number*	
`fields`	*lista*	Propiedad estructurada.
`full_filename`	*serie*	Nombre completo del archivo que se va a leer, incluido el directorio.

`strip_spaces`	`None`
	
	`Left`
	
	`Right`
	
	`Both`
		Descarta los espacios iniciales y finales en las cadenas de importación.

`invalid_char_mode`	`Discard`
	
	`Replace`
		Elimina los caracteres no válidos (nulo, 0 o cualquier carácter que no exista en la codificación actual) de la entrada de datos o sustituye los caracteres no válidos con el símbolo especificado de un carácter.

`invalid_char_replacement`	*serie*	
`use_custom_values`	*Flag*	

Tabla 51. Propiedades de fixedfilenode (continuación)

Propiedad de `fixedfilenode`	Tipo de datos	Descripción de la propiedad

`custom_storage`	`Unknown`
	
	`String`
	
	`Integer`
	
	`Real`
	
	`Time`
	
	`Date`
	
	`Timestamp`	

Tabla 51. Propiedades de fixedfilenode (continuación)		
Propiedad de `fixedfilenode`	**Tipo de datos**	**Descripción de la propiedad**

`custom_date_format`	`"DDMMYY"`
	`"MMDDYY"`
	`"YYMMDD"`
	`"YYYYMMDD"`
	`"YYYYDDD"`
	`DAY`
	`MONTH`
	`"DD-MM-YY"`
	`"DD-MM-YYYY"`
	`"MM-DD-YY"`
	`"MM-DD-YYYY"`
	`"DD-MON-YY"`
	`"DD-MON-YYYY"`
	`"YYYY-MM-DD"`
	`"DD.MM.YY"`
	`"DD.MM.YYYY"`
	`"MM.DD.YY"`
	`"MM.DD.YYYY"`
	`"DD.MON.YY"`
	`"DD.MON.YYYY"`
		Aplicable solamente si ha especificado un almacenamiento personalizado.

Tabla 51. Propiedades de fixedfilenode (continuación)		
Propiedad de fixedfilenode	**Tipo de datos**	**Descripción de la propiedad**
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Aplicable solamente si ha especificado un almacenamiento personalizado.

Tabla 51. Propiedades de fixedfilenode (continuación)		
Propiedad de fixedfilenode	**Tipo de datos**	**Descripción de la propiedad**
custom_decimal_symbol	*campo*	Aplicable solamente si ha especificado un almacenamiento personalizado.
encoding	StreamDefault SystemDefault "UTF-8"	Especifica el método de codificación de textos.

Propiedades del nodo gsdata_import

Utilice el nodo de origen Geoespacial para introducir datos de mapas o espaciales en la sesión de minería de datos.

Tabla 52. Propiedades del nodo gsdata_import		
Propiedades del nodo gsdata_import	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	Entre la vía de acceso del archivo .shp que desea cargar.
map_service_URL	*serie*	Escriba el URL de servicio de mapas al que conectarse.
map_name	*serie*	Sólo si se utiliza map_service_URL; esto contiene la estructura de carpeta de nivel superior del servicio de mapa.

Propiedades jsonimportnode

El nodo de origen JSON importa datos de un archivo JSON.

Tabla 53. propiedades jsonimportnode		
Propiedades jsonimportnode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	El nombre completo del archivo, incluyendo la vía de acceso.
string_format	*Registros* *Valores*	Especifique el formato de la cadena JSON. El valor predeterminado es records.
auto_label		Se ha añadido en la versión 18.2.1.1.

Propiedades de sasimportnode

El nodo importar SAS importa datos SAS a IBM SPSS Modeler.

Ejemplo

```
node = stream.create("sasimport", "My node")
node.setPropertyValue("format", "Windows")
node.setPropertyValue("full_filename", "C:/data/retail.sas7bdat")
node.setPropertyValue("member_name", "Test")
node.setPropertyValue("read_formats", False)
node.setPropertyValue("full_format_filename", "Test")
node.setPropertyValue("import_names", True)
```

Tabla 54. propiedades de sasimportnode

Propiedad de sasimportnode	Tipo de datos	Descripción de la propiedad
format	Windows UNIX Transport SAS7 SAS8 SAS9	Formato del archivo que se va a importar.
full_filename	*serie*	El nombre completo del archivo introducido, incluyendo su ruta.
member_name	*serie*	Especifique el miembro para importar desde el archivo de transporte SAS especificado.
read_formats	*Flag*	Lee formatos de datos (como etiquetas de variables) del archivo de formatos especificado.
full_format_filename	*serie*	
import_names	NamesAndLabels LabelsasNames	Especifica el método para la correlación de nombres y etiquetas de variables en la importación.

propiedades de simgennode

El nodo Generar simulación permite, de forma sencilla, generar datos simulados —ya sea a partir de cero utilizando distribuciones estadísticas especificadas por el usuario o de forma automática utilizando las distribuciones obtenidas de la ejecución de un nodo Ajustar simulación sobre datos históricos existentes. Esto es útil cuando se desea evaluar el resultado de un modelo predictivo en presencia de incertidumbre en las entradas del modelo.

Tabla 55. propiedades de simgennode		
propiedades de simgennode	**Tipo de datos**	**Descripción de la propiedad**
fields	Propiedad estructurada	Ver ejemplo
correlations	Propiedad estructurada	Ver ejemplo
keep_min_max_setting	*Boolean*	
refit_correlations	*Boolean*	
max_cases	*integer*	El valor mínimo es 1000, el valor máximo es 2,147,483,647.
create_iteration_field	*Boolean*	
iteration_field_name	*serie*	
replicate_results	*Boolean*	
random_seed	*integer*	
parameter_xml	*serie*	Devuelve el XML del parámetro como una cadena.

Ejemplo de fields

Este es un parámetro de ranura estructurado con la sintaxis siguiente:

```
simgennode.setPropertyValue("fields", [
    [field1, storage, locked, [distribution1], min, max],
    [field2, storage, locked, [distribution2], min, max],
    [field3, storage, locked, [distribution3], min, max]
])
```

distribution es una declaración del nombre de distribución seguido de una lista que contiene pares de nombres y valores de atributo. Cada distribución se define de la siguiente forma:

```
[distributionname, [[par1], [par2], [par3]]]

simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [["Age", "integer", False, ["Uniform",[["min","1"],
["max","2"]]], "", ""]])
```

Por ejemplo, para crear un nodo que genere un solo campo con una distribución binomial puede utilizar el siguiente script:

```
simgen_node1 = modeler.script.stream().createAt("simgen", u"Sim Gen", 200, 200)
simgen_node1.setPropertyValue("fields", [["Education", "Real", False, ["Binomial", [["n", 32],
 ["prob", 0.7]]], "", ""]])
```

La distribución binomial tiene 2 parámetros: n y prob. Puesto que binomial no admite los valores mínimo y máximo, éstos se suministran como una serie vacía.

Nota: No se puede establecer distribution directamente; utilícelo en combinación con la propiedad fields.

Los ejemplos siguientes muestran todos los tipos de distribución posibles. Tenga en cuenta que el umbral se especifica como thresh tanto en NegativeBinomialFailures como en NegativeBinomialTrial.

```
stream = modeler.script.stream()

simgennode = stream.createAt("simgen", u"Sim Gen", 200, 200)

beta_dist = ["Field1", "Real", False, ["Beta",[["shape1","1"],["shape2","2"]]], "", ""]
binomial_dist = ["Field2", "Real", False, ["Binomial",[["n" ,"1"],["prob","1"]]], "", ""]
categorical_dist = ["Field3", "String", False, ["Categorical", [["A",0.3],["B",0.5],["C",0.2]]], "", ""]
dice_dist = ["Field4", "Real", False, ["Dice", [["1" ,"0.5"],["2","0.5"]]], "", ""]
exponential_dist = ["Field5", "Real", False, ["Exponential", [["scale","1"]]], "", ""]
```

```
fixed_dist = ["Field6", "Real", False, ["Fixed", [["value","1" ]]], "", ""]
gamma_dist = ["Field7", "Real", False, ["Gamma", [["scale","1"],["shape"," 1"]]], "", ""]
lognormal_dist = ["Field8", "Real", False, ["Lognormal", [["a","1"],["b","1" ]]], "", ""]
negbinomialfailures_dist = ["Field9", "Real", False, ["NegativeBinomialFailures",[["prob","0.5"],["thresh","1"]]], "", ""]
negbinomialtrial_dist = ["Field10", "Real", False, ["NegativeBinomialTrials",[["prob","0.2"],["thresh","1"]]], "", ""]
normal_dist = ["Field11", "Real", False, ["Normal", [["mean","1"] ,["stddev","2"]]], "", ""]
poisson_dist = ["Field12", "Real", False, ["Poisson", [["mean","1"]]], "", ""]
range_dist = ["Field13", "Real", False, ["Range", [["BEGIN","[1,3]"] ,["END","[2,4]"],["PROB","[[0.5],[0.5]]"]]], "", ""]
triangular_dist = ["Field14", "Real", False, ["Triangular", [["min","0"],["max","1"],["mode","1"]]], "", ""]
uniform_dist = ["Field15", "Real", False, ["Uniform", [["min","1"],["max","2"]]], "", ""]
weibull_dist = ["Field16", "Real", False, ["Weibull", [["a","0"],["b","1 "],["c","1"]]], "", ""]
```

```
simgennode.setPropertyValue("fields", [\
beta_dist, \
binomial_dist, \
categorical_dist, \
dice_dist, \
exponential_dist, \
fixed_dist, \
gamma_dist, \
lognormal_dist, \
negbinomialfailures_dist, \
negbinomialtrial_dist, \
normal_dist, \
poisson_dist, \
range_dist, \
triangular_dist, \
uniform_dist, \
weibull_dist
])
```

Ejemplo de correlations

Este es un parámetro de ranura estructurado con la sintaxis siguiente:

```
simgennode.setPropertyValue("correlations", [
    [field1, field2, correlation],
    [field1, field3, correlation],
    [field2, field3, correlation]
])
```

La correlación puede ser cualquier número entre +1 y -1. Puede especificar tantas correlaciones como desee. Las correlaciones no especificadas se establecen en cero. Si alguno de los campos se desconocen, el valor de la correlación debe establecerse en la matriz de correlación (o tabla) y se muestra en texto rojo. Cuando hay campos desconocidos, no es posible ejecutar el nodo.

Propiedades de statisticsimportnode

El nodo Archivo de IBM SPSS Statistics lee los datos desde un formato de archivo *.sav* que utiliza IBM SPSS Statistics y archivos caché guardados en IBM SPSS Modeler, que también puede utilizar el mismo formato.

Las propiedades de este nodo están descritas en “Propiedades de statisticsimportnode” en la página 435.

Propiedades del nodo tm1odataimport

El nodo fuente de IBM Cognos TM1 importa datos de las bases de datos de Cognos TM1.

Tabla 56. Propiedades del nodo tm1odataimport

Propiedades del nodo `tm1odataimport`	**Tipo de datos**	**Descripción de la propiedad**
`credential_type`	*inputCredential* o *storedCredential*	Se utiliza para indicar el tipo de credencial.

Tabla 56. Propiedades del nodo tm1odataimport (continuación)		
Propiedades del nodo tm1odataimport	**Tipo de datos**	**Descripción de la propiedad**
input_credential	*lista*	Cuando credential_type es inputCredential; especifique el dominio, el nombre de usuario y la contraseña.
stored_credential_name	*serie*	Cuando credential_type es storedCredential; especifique el nombre de la credencial en el servidor C & DS.
selected_view	*["campo" "campo"]*	Una propiedad de la lista que contiene los detalles del cubo TM1 seleccionado y el nombre de la vista de cubo donde los datos se importarán en SPSS. Por ejemplo: TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])
is_private_view	*Flag*	Especifica si selected_view es una vista privada. El valor predeterminado es false.
selected_columns	*["campo"]*	Especifique la columna seleccionada; solo se puede especificar un elemento. Por ejemplo: setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["campo" "campo"]*	Especifique las filas seleccionadas. Por ejemplo: setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])
connection_type	AdminServer TM1Server	Indica el tipo de conexión. El valor predeterminado es AdminServer.
admin_host	*serie*	El URL del nombre de host de la API REST. Obligatorio si connection_type es AdminServer.
server_name	*serie*	El nombre del servidor TM1 seleccionado de admin_host. Obligatorio si connection_type es AdminServer.
server_url	*serie*	El URL de la API REST del Servidor TM1. Obligatorio si connection_type es TM1Server.

Propiedades del nodo tm1import (en desuso)

El nodo fuente de IBM Cognos TM1 importa datos de las bases de datos de Cognos TM1.

Nota: Este nodo estaba en desuso en Modeler 18.0. El nombre del script del nodo de sustitución es *tm1odataimport*.

Tabla 57. Propiedades del nodo tm1import		
Propiedades del nodo `tm1import`	**Tipo de datos**	**Descripción de la propiedad**
`pm_host`	*serie*	**Nota:** sólo para la versión 16.0 y 17.0 Nombre del host. Por ejemplo: `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
`tm1_connection`	*["campo", "campo", ..., "campo"]*	**Nota:** sólo para la versión 16.0 y 17.0 Una propiedad de la lista que contiene los detalles de conexión para el servidor de TM1. El formato es: `[ "TM1_Server_Name","tm1_username","tm1_ password"]` Por ejemplo: `TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])`
`selected_view`	*["campo" "campo"]*	Una propiedad de la lista que contiene los detalles del cubo TM1 seleccionado y el nombre de la vista de cubo donde los datos se importarán en SPSS. Por ejemplo: `TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])`
`selected_column`	*["campo"]*	Especifique la columna seleccionada; solo se puede especificar un elemento. Por ejemplo: `setPropertyValue("selected_columns", ["Measures"])`
`selected_rows`	*["campo" "campo"]*	Especifique las filas seleccionadas: Por ejemplo: `setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])`

Propiedades del nodo twcimport

El nodo de origen de TWC importa datos meteorológicos de The Weather Company, una empresa de IBM. Puede utilizarlo para obtener datos meteorológicos históricos o predicción para una ubicación. Esto puede ayudarle a desarrollar soluciones empresariales dirigidas a la meteorología para una mejor toma de decisiones utilizando los datos meteorológicos más exactos y precisos disponibles.

Tabla 58. Propiedades del nodo twcimport		
Propiedades del nodo `twcimport`	**Tipo de datos**	**Descripción de la propiedad**
`TWCDataImport.latitude`	*real*	Especifica un valor de latitud con el formato `[-90.0~90.0]`
`TWCDataImport.longitude`	*real*	Especifica un valor de longitud con el formato `[-180.0~180.0]`.
`TWCDataImport.licenseKey`	*serie*	Especifica la clave de licencia obtenida de The Weather Company.

`TWCDataImport.measurmentUnit`	`English`
	
	`Metric`
	
	`Hybrid`
		Especifica la unidad de medida. Los valores posibles son `English`, `Metric` o `Hybrid`. `Metric` es el valor predeterminado.

`TWCDataImport.dataType`	`Historical`
	
	`Forecast`
		Especifica el tipo de datos meteorológicos para la entrada. Los valores posibles son `Historical` o `Prediction`. `Histórico` es el valor predeterminado.

`TWCDataImport.startDate`	*integer*	Si se especifica `Histórico` para `TWCDataImport.dataType`, especifique una fecha de inicio con el formato `aaaaMMdd`.
`TWCDataImport.endDate`	*integer*	Si se especifica `Histórico` para `TWCDataImport.dataType`, especifique una fecha de finalización con el formato `aaaaMMdd`.

`TWCDataImport.forecastHour`	`6`
	
	`12`
	
	`24`
	
	`48`
		Si se especifica `Predicción` para `TWCDataImport.dataType`, especifique 6, 12, 24 o 48 para la hora.

propiedades de userinputnode

El nodo Datos de usuario proporciona una manera fácil de crear datos sintéticos, ya sea partiendo de cero o modificando los datos existentes. Esto resulta útil, por ejemplo, cuando desee crear un conjunto de datos de comprobación para el modelado.

Ejemplo

```
node = stream.create("userinput", "My node")
node.setPropertyValue("names", ["test1", "test2"])
node.setKeyedPropertyValue("data", "test1", "2, 4, 8")
node.setKeyedPropertyValue("custom_storage", "test1", "Integer")
node.setPropertyValue("data_mode", "Ordered")
```

Tabla 59. propiedades de userinputnode		
Propiedad de userinputnode	**Tipo de datos**	**Descripción de la propiedad**
data		
names		Intervalo estructurado que establece o devuelve una lista de nombres de campos generados por el nodo.
custom_storage	Unknown String Integer Real Time Date Timestamp	Intervalo con clave que establece o devuelve el almacenamiento para un campo.
data_mode	Combined Ordered	Si se especifica Combined, los registros se generarán para cada combinación de valores del conjunto y valores mínimos y máximos. El número de registros generados será igual al producto del número de valores de cada campo. Si se especifica Ordered, se tomará un valor de cada columna para cada registro con el fin de generar una fila de datos. El número de registros generados será igual al número más grande de valores asociados a un campo. Los campos que tengan menos valores de datos se rellenarán con valores nulos.
values		**Nota:** Esta propiedad ya no se utiliza, y no debe usarse; en su lugar, se usa userinputnode.data.

Propiedades de variablefilenode

El nodo Archivo variable lee datos desde los archivos de texto de campo libre, es decir, campos cuyos registros contienen un número constante de campos, pero un número variado de caracteres. Este nodo resulta también útil para los archivos con texto de cabecera de longitud fija y determinados tipos de anotaciones.

Ejemplo

```
node = stream.create("variablefile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("read_field_names", True)
node.setPropertyValue("delimit_other", True)
node.setPropertyValue("other", ",")
```

```
node.setPropertyValue("quotes_1", "Discard")
node.setPropertyValue("decimal_symbol", "Comma")
node.setPropertyValue("invalid_char_mode", "Replace")
node.setPropertyValue("invalid_char_replacement", "|")
node.setKeyedPropertyValue("use_custom_values", "Age", True)
node.setKeyedPropertyValue("direction", "Age", "Input")
node.setKeyedPropertyValue("type", "Age", "Range")
node.setKeyedPropertyValue("values", "Age", [1, 100])
```

Tabla 60. Propiedades de variablefilenode

Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
skip_header	*number*	Especifica el número de caracteres que se ignorarán al principio del primer registro.
num_fields_auto	*Flag*	Determina el número de campos de cada registro de forma automática. Los registros deben terminar con un carácter de nueva línea.
num_fields	*number*	Especifica manualmente el número de campos de cada registro.
delimit_space	*Flag*	Especifica el carácter utilizado para delimitar los límites de los campos del archivo.
delimit_tab	*Flag*	
delimit_new_line	*Flag*	
delimit_non_printing	*Flag*	
delimit_comma	*Flag*	En aquellos casos en los que la coma sea el delimitador del campo y el separador decimal para rutas, establezca delimit_other en *True* y especifique una coma como delimitador mediante la propiedad other.
delimit_other	*Flag*	Permite especificar un delimitador personalizado mediante la propiedad other.
other	*serie*	Especifica el delimitador utilizado cuando delimit_other es *True*.
decimal_symbol	Default Comma Period	Especifica el separador decimal utilizado en el origen de datos.
multi_blank	*Flag*	Trata varios caracteres delimitadores vacíos adyacentes como un único delimitador.
read_field_names	*Flag*	Trata la primera fila del archivo de datos como etiquetas para la columna.

Tabla 60. Propiedades de variablefilenode (continuación)

Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
strip_spaces	None Left Right Both	Descarta los espacios iniciales y finales en las cadenas de importación.
invalid_char_mode	Discard Replace	Elimina los caracteres no válidos (nulo, 0 o cualquier carácter que no exista en la codificación actual) de la entrada de datos o sustituye los caracteres no válidos con el símbolo especificado de un carácter.
invalid_char_replacement	*serie*	
break_case_by_newline	*Flag*	Especifica que el delimitador de línea es el carácter de nueva línea.
lines_to_scan	*number*	Especifica cuántas líneas se van a explorar para los tipos de datos especificados.
auto_recognize_datetime	*Flag*	Especifica si las fechas o las horas se identifican automáticamente en los datos de origen.
quotes_1	Discard PairAndDiscard IncludeAsText	Especifica cómo se tratarán las comillas simples en la importación.
quotes_2	Discard PairAndDiscard IncludeAsText	Especifica cómo se tratarán las comillas dobles en la importación.
full_filename	*serie*	Nombre completo del archivo que se va a leer, incluido el directorio.
use_custom_values	*Flag*	

Tabla 60. Propiedades de variablefilenode (continuación)

Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
custom_storage	Unknown String Integer Real Time Date Timestamp	

Tabla 60. Propiedades de variablefilenode (continuación)		
Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	Aplicable solamente si ha especificado un almacenamiento personalizado.

Tabla 60. Propiedades de variablefilenode (continuación)		
Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	

Tabla 60. Propiedades de variablefilenode (continuación)		
Propiedad de variablefilenode	**Tipo de datos**	**Descripción de la propiedad**
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Aplicable solamente si ha especificado un almacenamiento personalizado.
custom_decimal_symbol	*campo*	Aplicable solamente si ha especificado un almacenamiento personalizado.
encoding	StreamDefault SystemDefault "UTF-8"	Especifica el método de codificación de textos.

Propiedades de xmlimportnode

El nodo de origen XML importa datos en formato XML en la ruta. Puede importar un único archivo o todos los archivos en un directorio. Puede especificar opcionalmente un archivo de esquema para leer la estructura XML.

Ejemplo

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

Tabla 61. propiedades de xmlimportnode

propiedades de xmlimportnode	**Tipo de datos**	**Descripción de la propiedad**
read	single directory	Lee un único archivo de datos (valor predeterminado) o todos los archivos XML de un directorio.
recurse	*Flag*	Especifica si leer además archivos XML de todos los subdirectorios del directorio especificado.
full_filename	*serie*	(necesario) Vía de acceso completa y nombre de archivo del archivo XML a importar (si `read = single`).
directory_name	*serie*	(obligatorio) Ruta completa y nombre del directorio desde el que importar los archivos XML (si `read = directory`).
full_schema_filename	*serie*	Ruta completa y nombre de archivo del archivo XSD o DTD desde el que leer la estructura XML. Si omite este parámetro, se leerá la estructura desde el archivo de origen XML.
records	*serie*	Expresión XPath (p.ej. /author/name) para definir el límite del registro. Cada vez que este elemento se encuentra en el archivo de origen se crea un nuevo registro.
mode	read specify	Lee todos los datos (valor predeterminado) o especifica qué elementos leer.
fields		Lista de elementos (elementos y atributos) para importar. Cada elemento de la lista es una expresión XPath.

Capítulo 10. Propiedades de nodos de operaciones con registros

propiedades de appendnode

El nodo Añadir concatena conjuntos de registros. Es útil para combinar conjuntos de datos con estructuras parecidas, pero con datos diferentes.

Ejemplo

```
node = stream.create("append", "My node")
node.setPropertyValue("match_by", "Name")
node.setPropertyValue("match_case", True)
node.setPropertyValue("include_fields_from", "All")
node.setPropertyValue("create_tag_field", True)
node.setPropertyValue("tag_field_name", "Append_Flag")
```

Tabla 62. propiedades de appendnode

Propiedad de appendnode	Tipo de datos	Descripción de la propiedad
match_by	Position Name	Se pueden añadir conjuntos de datos basándose en la posición que tienen los campos en el origen de datos principal o el nombre de los campos en los conjuntos de datos de entrada.
match_case	*Flag*	Activa la coincidencia de mayúsculas y minúsculas al hacer coincidir nombres de campos.
include_fields_from	Main All	
create_tag_field	*Flag*	
tag_field_name	*serie*	

propiedades de aggregatenode

El nodo Agregar reemplaza una secuencia de registros de entrada con registros de salida agregados y resumidos.

Ejemplo

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
```

```
node.setPropertyValue("inc_record_count", True)
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

Tabla 63. propiedades de aggregatenode

Propiedad de aggregatenode	**Tipo de datos**	**Descripción de la propiedad**
keys	*lista*	Enumera los campos que se pueden usar como claves en la agregación. Por ejemplo, si Sex y Region son los campos de clave, cada combinación exclusiva de M y F con las regiones N y S (cuatro combinaciones exclusivas) tendrá un registro agregado.
contiguous	*Flag*	Seleccione esta opción si sabe que todos los registros con los mismos valores clave se agrupan en la entrada (por ejemplo, si la entrada se clasifica en los campos clave). Con ello puede mejorar el rendimiento.
aggregates		Enumera los campos numéricos cuyos valores se añadirán, así como los modos de agregación elegidos.
aggregate_exprs		Propiedad con clave que aplica claves al nombre de campo derivado con la expresión agregada utilizada para calcularla. Por ejemplo: `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*serie*	Especifica un prefijo o sufijo para campos agregados duplicados (consulte un ejemplo a continuación).
add_as	Suffix Prefix	
inc_record_count	*Flag*	Crea un campo adicional que indica la cantidad de registros de entrada agregados para conformar cada registro agregado.
count_field	*serie*	Especifica el nombre del campo de recuento de registros.
allow_approximation	*boolean*	Permite la aproximación de estadísticas de ordenación cuando se realiza la agregación en Analytic Server
bin_count	*integer*	Especifica el número de intervalos a utilizar en la aproximación

propiedades de balancenode

El nodo Equilibrar corrige los desequilibrios de un conjunto de datos para que cumpla una condición determinada. La directiva de equilibrado ajusta la proporción de registros si una condición es verdadera por el factor determinado.

Ejemplo

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

Tabla 64. propiedades de balancenode

Propiedad de balancenode	Tipo de datos	Descripción de la propiedad
directives		Propiedad estructurada para equilibrar la proporción de los valores de campos basados en un número determinado (consulte el ejemplo a continuación).
training_data_only	*Flag*	Especifica que sólo se deben equilibrar los datos de entrenamiento. Si no se incluye ningún campo de partición en la ruta, se ignorará esta opción.

Esta propiedad de nodo utiliza el siguiente formato:

[[*number, string*] \ [*number, string*] \ ... [*number, string*]].

Nota: Si las cadenas (que utilizan comillas dobles) están incrustadas en la expresión, han de estar precedidas del carácter de escape " \ ". El carácter " \ " es también el carácter de continuación de línea, que puede utilizar para alinear los argumentos para mayor claridad.

Propiedades cplexoptnode

El nodo Optimización de CPLEX proporciona la capacidad de utilizar la optimización basada en CPLEX (matemática compleja) a través de un archivo de modelo OPL (Optimization Programming Language - Lenguaje de programación de optimización). Esta funcionalidad estaba disponible en el producto IBM Analytical Decision Management, que ya no recibe soporte. Pero también puede utilizar el nodo CPLEX en SPSS Modeler sin necesidad de IBM Analytical Decision Management.

Tabla 65. Propiedades cplexoptnode

Propiedades cplexoptnode	Tipo de datos	Descripción de la propiedad
opl_model_text	*serie*	El programa de scripts OPL (Lenguaje de programación de optimización) que ejecutará el nodo Optimización de CPLEX y, después, generará el resultado de la optimización.

Tabla 65. Propiedades cplexoptnode (continuación)

Propiedades cplexoptnode	Tipo de datos	Descripción de la propiedad
opl_tuple_set_name	*serie*	Nombre de conjunto de tuplas en el modelo OPL que corresponde a los datos de entrada. Esto no es necesario y, normalmente, no se establece a través del script. Solo se debe utilizar para editar correlaciones de campos de una fuente de datos seleccionada.
data_input_map	*Lista de propiedades estructuradas*	Las correlaciones de campos de entrada para una fuente de datos. Esto no es necesario y, normalmente, no se establece a través del script. Solo se debe utilizar para editar correlaciones de campos de una fuente de datos seleccionada.

Tabla 65. Propiedades cplexoptnode (continuación)

Propiedades `cplexoptnode`	Tipo de datos	Descripción de la propiedad
`md_data_input_map`	*Lista de propiedades estructuradas*	Las correlaciones de campos entre cada tupla definida en el OPL, con cada fuente de datos de campo correspondiente (datos de entrada). Los usuarios pueden editarlas de forma individual por fuente de datos. Con este script, puede establecer la propiedad directamente para definir todas las correlaciones a la vez. Este valor no se muestra en la interfaz de usuario. Cada entidad de la lista corresponde a datos estructurados. **Etiqueta de fuente de datos.** La etiqueta de la fuente de datos, que se puede encontrar en el desplegable de fuentes de datos. Por ejemplo, para `0_Products_Type` la etiqueta es `0`. **Índice de fuentes de datos.** La secuencia física (índice) de la fuente de datos. Esto se determina mediante el orden de conexión. **Nodo de origen.** El nodo de origen (anotación) de la fuente de datos. Este se puede encontrar en el desplegable de fuentes de datos. Por ejemplo, para `0_Products_Type` el nodo de origen es `Products`. **Nodo conectado.** El nodo anterior (anotación) que se conecta al nodo de optimización de CPLEX actual. Este se puede encontrar en el desplegable de fuentes de datos. Por ejemplo, para `0_Products_Type`, el nodo conectado es `Type`. **Nombre de conjunto de tuplas.** El nombre de conjunto de tuplas de la fuente de datos. Debe coincidir con lo que está definido en el OPL. **Nombre de campo de tupla.** El nombre del campo de conjunto de tuplas de la fuente de datos. Debe coincidir con lo que está definido en la definición del conjunto de tuplas de OPL. **Tipo de almacenamiento.** El tipo de almacenamiento de campo. Los valores posibles son `int`, `float` o `string`.

Tabla 65. Propiedades cplexoptnode (continuación)

Propiedades cplexoptnode	Tipo de datos	Descripción de la propiedad
		Nombre de campo de datos. El nombre de campo de la fuente de datos. **Ejemplo:** `[[0,0,'Product','Type','Products','prod_id_tup','int','prod_id'], [0,0,'Product','Type','Products','prod_name_tup','string', 'prod_name'], [1,1,'Components','Type','Components', 'comp_id_tup','int','comp_id'], [1,1,'Components','Type', 'Components','comp_name_tup','string','comp_name']]`
opl_data_text	*serie*	La definición de algunas variables o datos utilizados para el OPL.
output_value_mode	*serie*	Los valores posibles son raw o dvar. Si se proporciona dvar, en la pestaña Resultado el usuario debe especificar el nombre de variable de función de objeto en OPL para el resultado. Si se especifica raw, la función de objetivo se producirá directamente, independientemente del nombre.
decision_variable_name	*serie*	El nombre de variable de la función de objetivo definido en el OPL. Este valor solo está habilitado cuando la propiedad output_value_mode está establecida en dvar.
objective_function_value_fieldname	*serie*	El nombre del campo para el valor de la función de objetivo para utilizar en la salida. El valor predeterminado es _OBJECTIVE.
output_tuple_set_names	*serie*	El nombre de las tuplas predefinidas a partir de los datos entrantes. Este actúa como los índices de la variable de decisión y se espera que sea la salida con las salidas de variable. La tupla de salida debe ser coherente con la definición de la variable de decisión en el OPL. Si hay varios índices, los nombres de tupla se deben unir con una coma (,). Un ejemplo para una sola tupla es Products, siendo la definición de OPL correspondiente `dvar float+ Production[Products];` Un ejemplo para varias tuplas es Products,Components, siendo la definición de OPL correspondiente `dvar float+ Production[Products][Components];`

Tabla 65. Propiedades cplexoptnode (continuación)		
Propiedades cplexoptnode	**Tipo de datos**	**Descripción de la propiedad**
decision_output_map	*Lista de propiedades estructuradas*	La correlación de campos entre variables definidas en el OPL que será la salida y los campos de salidas. Cada entidad de la lista corresponde a datos estructurados. **Nombre de variable.** El nombre de variable en el OPL para el resultado. **Tipo de almacenamiento.** Los valores posibles son `int`, `float` o `string`. **Nombre de campo de salida.** El nombre de campo previsto en los resultados (salida o exportación). **Ejemplo:** `[['Production','int','res'], ['Remark','string','res_1']['Cost', 'float','res_2']]`

Propiedades derive_stbnode

El nodo Cajas-espacio-tiempo deriva Cajas-espacio-tiempo de los campos latitud, longitud e indicación de fecha y hora. Las Cajas-espacio-tiempo también pueden identificarse como lugares comunes.

Ejemplo

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Individual Records mode
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")

# Hangouts mode
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

Tabla 66. Propiedades del nodo Cajas-Espacio-Tiempo		
Propiedades de derive_stbnode	**Tipo de datos**	**Descripción de la propiedad**
mode	`IndividualRecords` `Hangouts`	
latitude_field	*campo*	

Tabla 66. Propiedades del nodo Cajas-Espacio-Tiempo (continuación)

Propiedades de derive_stbnode	**Tipo de datos**	**Descripción de la propiedad**
longitude_field	*campo*	
timestamp_field	*campo*	
hangout_density	*densidad*	Una sola densidad. Consulte en densities los valores de densidad válidos.
densities	[*densidad*,*densidad*,..., *densidad*]	Cada densidad es una cadena, por ejemplo, STB_GH8_1DAY. **Nota:** Existen límites para que las densidades sean válidas. En geohash, se pueden utilizar los valores de GH1 a GH15. Para la parte temporal, se pueden utilizar los valores siguientes: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	*campo*	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Debe ser una cadena.
min_events	*integer*	El valor de entero válido mínimo es de 2.
qualifying_pct	*integer*	Debe estar en el rango de 1 a 100.

Tabla 66. Propiedades del nodo Cajas-Espacio-Tiempo (continuación)		
Propiedades de derive_stbnode	**Tipo de datos**	**Descripción de la propiedad**
add_extension_as	Prefix Suffix	
name_extension	*serie*	

propiedades de distinctnode

El nodo Distinguir se puede usar para eliminar registros duplicados pasando el primero de los registros distintos a la ruta de datos o descartando el primer registro y pasando cualquier duplicado a la ruta de datos en su lugar.

Ejemplo

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

Tabla 67. propiedades de distinctnode		
Propiedad de distinctnode	**Tipo de datos**	**Descripción de la propiedad**
mode	Include Discard	Se puede incluir el primer nodo distinto en la ruta o descartar el primer nodo distinto y pasar en su lugar todos los registros duplicados a la ruta de datos.
grouping_fields	*lista*	Enumera los campos utilizados para determinar si los registros son idénticos. **Nota:** Esta propiedad está en desuso desde IBM SPSS Modeler 16 y posterior.
composite_value	Intervalo estructurado	Vea el ejemplo que se muestra a continuación.
composite_values	Intervalo estructurado	Vea el ejemplo que se muestra a continuación.
inc_record_count	*Flag*	Crea un campo adicional que indica la cantidad de registros de entrada agregados para conformar cada registro agregado.
count_field	*serie*	Especifica el nombre del campo de recuento de registros.
sort_keys	Intervalo estructurado	**Nota:** Esta propiedad está en desuso desde IBM SPSS Modeler 16 y posterior.
default_ascending	*Flag*	
low_distinct_key_count	*Flag*	Especifica que sólo tiene un pequeño número de registros y/o un pequeño número de valores exclusivos del campo(s) clave.

Tabla 67. propiedades de distinctnode (continuación)		
Propiedad de distinctnode	**Tipo de datos**	**Descripción de la propiedad**
keys_pre_sorted	*Flag*	Especifica que todos los registros con los mismos valores clave se agrupan en la entrada.
disable_sql_generation	*Flag*	

Ejemplo para la propiedad composite_value

La propiedad composite_value tiene el formato general siguiente:

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION tiene el formato [FillType, Option1, Option2, ...].

Ejemplos:

```
node.setKeyedPropertyValue("composite_value", "Age", ["First"])
node.setKeyedPropertyValue("composite_value", "Age", ["last"])
node.setKeyedPropertyValue("composite_value", "Age", ["Total"])
node.setKeyedPropertyValue("composite_value", "Age", ["Average"])
node.setKeyedPropertyValue("composite_value", "Age", ["Min"])
node.setKeyedPropertyValue("composite_value", "Age", ["Max"])
node.setKeyedPropertyValue("composite_value", "Date", ["Earliest"])
node.setKeyedPropertyValue("composite_value", "Date", ["Latest"])
node.setKeyedPropertyValue("composite_value", "Code", ["FirstAlpha"])
node.setKeyedPropertyValue("composite_value", "Code", ["LastAlpha"])
```

Las opciones personalizadas requieren más de un argumento, añadidos como una lista, por ejemplo:

```
node.setKeyedPropertyValue("composite_value", "Name", ["MostFrequent", "FirstRecord"])
node.setKeyedPropertyValue("composite_value", "Date", ["LeastFrequent", "LastRecord"])
node.setKeyedPropertyValue("composite_value", "Pending", ["IncludesValue", "T", "F"])
node.setKeyedPropertyValue("composite_value", "Marital", ["FirstMatch", "Married", "Divorced",
"Separated"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Space"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Comma"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "UnderScore"])
```

Ejemplo para la propiedad composite_values

La propiedad composite_values tiene el formato general siguiente:

```
node.setPropertyValue("composite_values", [
                      [FIELD1, [FILLOPTION1]],
                      [FIELD2, [FILLOPTION2]],
.
.
])
```

Ejemplo:

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

Propiedades de extensionprocessnode

Con el nodo de Transformación de extensión, puede tomar datos de una ruta y aplicar transformaciones a los datos utilizando scripts R o scripts Python para Spark.

Ejemplo de Python para Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_process", "extension_process")
node.setPropertyValue("syntax_type", "Python")

process_script = """
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = StructType([StructField("Age", LongType(), nullable=True), \
                      StructField("Sex", StringType(), nullable=True), \
                      StructField("BP", StringType(), nullable=True), \
                      StructField("Na", DoubleType(), nullable=True), \
                      StructField("K", DoubleType(), nullable=True), \
                      StructField("Drug", StringType(), nullable=True)])
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    print df.dtypes[:]
    _newDF = df.select("Age","Sex","BP","Na","K","Drug")
    print _newDF.dtypes[:]
    cxt.setSparkOutputData(_newDF)
"""

node.setPropertyValue("python_syntax", process_script)
```

Ejemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabla 68. Propiedades de extensionprocessnode

Propiedades de extensionprocessnode	**Tipo de datos**	**Descripción de la propiedad**
syntax_type	*E* *Python*	Especifica qué script se ejecuta: R o Python (R es el valor predeterminado).
r_syntax	*serie*	Sintaxis de scripts R a ejecutar.
python_syntax	*serie*	Sintaxis de scripts Python a ejecutar.
use_batch_size	*Flag*	Habilitar uso de proceso por lotes.

Tabla 68. Propiedades de extensionprocessnode (continuación)		
Propiedades de extensionprocessnode	**Tipo de datos**	**Descripción de la propiedad**
batch_size	*integer*	Especifique el número de registros de datos a incluir en cada lote.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opción para convertir campos de distintivos.
convert_missing	*Flag*	Opción para convertir valores perdidos en el valor R NA.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.

propiedades de mergenode

El nodo Fundir toma varios registros de entrada y crea un registro de salida único que contiene todos o algunos de los campos de entrada. Es útil para fusionar datos desde diferentes orígenes, como datos de clientes internos y datos demográficos adquiridos.

Ejemplo

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

Tabla 69. propiedades de mergenode

Propiedad de mergenode	**Tipo de datos**	**Descripción de la propiedad**
method	Order Keys Condition Rankedcondition	Especifique si los registros se fusionan en el orden en que aparecen en los archivos de datos, si se utilizarán uno o más campos clave para fusionar los registros que tengan el mismo valor en sus campos clave, si los registros se fusionarán si se satisface una condición especificada, o si se debe fusionar cada emparejamiento de fila de los conjuntos primario y todos los secundarios; utilizando la expresión de clasificación para ordenar coincidencias múltiples de menor a mayor.
condition	*serie*	Si method se establece en Condition, especifica la condición para incluir o descartar registros.
key_fields	*lista*	
common_keys	*Flag*	
join	Inner FullOuter PartialOuter Anti	
outer_join_tag.n	*Flag*	En esta propiedad, *n* es el nombre de etiqueta tal y como recoge el cuadro de diálogo Seleccionar conjunto de datos. Tenga en cuenta que es posible que existan varios nombres de etiquetas especificados, ya que pueden ser varios los conjuntos de datos que aporten registros incompletos.
single_large_input	*Flag*	Determina si se va a usar la optimización para tener una entrada relativamente grande en comparación con el resto de entradas.
single_large_input_tag	*serie*	Especifica el nombre de etiqueta tal y como se muestra en el cuadro de diálogo Seleccionar conjunto de datos grande. Tenga en cuenta que el uso de esta propiedad es ligeramente distinto que el de la propiedad outer_join_tag (marca frente a cadena), ya que solamente se puede especificar un único conjunto de datos de entrada.
use_existing_sort_keys	*Flag*	Determina si las entradas ya se han ordenado en función de uno o varios campos clave.

Tabla 69. propiedades de mergenode (continuación)		
Propiedad de mergenode	**Tipo de datos**	**Descripción de la propiedad**
existing_sort_keys	[['string', 'Ascending'] \ ['string', 'Descending']]	Especifica los campos que ya están ordenados y la dirección en que dicho orden se ha establecido.
primary_dataset	*serie*	Si method es Rankedcondition, seleccione el conjunto de datos primario de la fusión. Esto se puede considerar como el lado izquierdo de una fusión de unión externa.
rename_duplicate_fields	*boolean*	Si method es Rankedcondition, y este valor está establecido en Y, si el conjunto de datos fusionado resultante contiene varios campos con el mismo nombre procedentes de orígenes de datos distintos, se añaden las etiquetas respectivas de los orígenes de datos al comienzo de las cabeceras de columna de campo.
merge_condition	*serie*	
ranking_expression	*serie*	
Num_matches	*integer*	Número de coincidencias a devolver, en función de merge_condition y ranking_expression. Mínimo 1, máximo 100.

propiedades rfmaggregatenode

El nodo Aggregate de Recency, Frequency, Monetary (RFM) le permite tomar los datos de transacciones históricas de los clientes, eliminar cualquier dato no utilizado y combinar todos los datos de transacción restantes en una sola fila que lista cuando se trataron por última vez con usted, cuántas transacciones han realizado y el valor monetario total de esas transacciones.

Ejemplo

```
node = stream.create("rfmaggregate", "My node")
node.setPropertyValue("relative_to", "Fixed")
node.setPropertyValue("reference_date", "2007-10-12")
node.setPropertyValue("id_field", "CardID")
node.setPropertyValue("date_field", "Date")
node.setPropertyValue("value_field", "Amount")
node.setPropertyValue("only_recent_transactions", True)
node.setPropertyValue("transaction_date_after", "2000-10-01")
```

Tabla 70. propiedades rfmaggregatenode		
propiedades rfmaggregatenode	**Tipo de datos**	**Descripción de la propiedad**
relative_to	Fixed Today	Especifica la fecha a partir de la que se calculará la actualidad de las transacciones.

Tabla 70. propiedades rfmaggregatenode (continuación)		
propiedades rfmaggregatenode	**Tipo de datos**	**Descripción de la propiedad**
reference_date	*Date*	Sólo está disponible si se selecciona Fixed en relative_to.
contiguous	*Flag*	Si los datos se han clasificado previamente de forma que todos los registros con el mismo ID aparecen en la ruta de datos, al seleccionar esta opción acelerará el procesamiento.
id_field	*campo*	Especifica el campo que desea utilizar para identificar el cliente y sus transacciones.
date_field	*campo*	Especifica el campo de fecha que se utilizará para calcular la actualidad.
value_field	*campo*	Especifica el campo que se utilizará para calcular el valor monetario.
extension	*serie*	Especifica un prefijo o sufijo para campos agregados duplicados.
add_as	Suffix Prefix	Especifique si el extension debe añadirse como sufijo o como prefijo.
discard_low_value_records	*Flag*	Permite utilizar la configuración discard_records_below.
discard_records_below	*number*	Especifique un valor mínimo por debajo del cual no se utilice ningún detalle de transacción al calcular los totales de RFM. Las unidades de valor se relacionan con el campo value seleccionado.
only_recent_transactions	*Flag*	Permite utilizar de la configuración specify_transaction_date o transaction_within_last.
specify_transaction_date	*Flag*	
transaction_date_after	*Date*	Sólo está disponible si selecciona specify_transaction_date. Especifica la fecha de transacción tras la que se incluirán los registros en su análisis.
transaction_within_last	*number*	Sólo está disponible si selecciona transaction_within_last. Especifica el número y tipo de períodos (días, semanas, meses o años) desde la fecha Calcular actualidad relativa a tras la cual se incluirán los registros en su análisis.

Tabla 70. propiedades rfmaggregatenode (continuación)		
propiedades rfmaggregatenode	**Tipo de datos**	**Descripción de la propiedad**
transaction_scale	Days Weeks Months Years	Sólo está disponible si selecciona transaction_within_last. Especifica el número y tipo de períodos (días, semanas, meses o años) desde la fecha Calcular actualidad relativa a tras la cual se incluirán los registros en su análisis.
save_r2	*Flag*	Muestra la fecha de la segunda transacción más reciente para cada cliente.
save_r3	*Flag*	Sólo está disponible si selecciona save_r2. Muestra la fecha de la tercera transacción más reciente para cada cliente.

Propiedades de Rprocessnode

El nodo Transformación R le permite tomar datos de una ruta de IBM(r) SPSS(r) Modeler y modificar los datos utilizando su propio script R personalizado. Después de modificar los datos, se devuelven a la ruta.

Ejemplo

```
node = stream.create("rprocess", "My node")
node.setPropertyValue("custom_name", "my_node")
node.setPropertyValue("syntax", """day<-as.Date(modelerData$dob, format="%Y-
%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next
day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
node.setPropertyValue("convert_datetime", "POSIXct")
```

Tabla 71. Propiedades de Rprocessnode		
Propiedades de Rprocessnode	**Tipo de datos**	**Descripción de la propiedad**
syntax	*serie*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*Flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*Flag*	

Tabla 71. Propiedades de Rprocessnode (continuación)		
Propiedades de Rprocessnode	**Tipo de datos**	**Descripción de la propiedad**
use_batch_size	*Flag*	Habilite el uso del proceso por lotes
batch_size	*integer*	Especifique el número de registros de datos que se incluirán en cada lote

propiedades de samplenode

El nodo Muestrear selecciona un subconjunto de registros. Se admite una variedad de tipos de muestras, entre las que se incluyen las muestras estratificadas, agrupadas en clústeres y no aleatorias (estructuradas). El muestreo puede ser de gran utilidad para mejorar el rendimiento y para seleccionar grupos de registros o transacciones relacionadas para un análisis.

Ejemplo

```
/* Create two Sample nodes to extract
   different samples from the same data */

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"], ["M",
"Normal", "Default"],
 ["F", "High", 0.3], ["F", "Normal", 0.3]])
```

Tabla 72. propiedades de samplenode		
Propiedad de samplenode	**Tipo de datos**	**Descripción de la propiedad**
method	Simple Complejo	
mode	Include Discard	Incluye o descarta los registros que reúnan la condición especificada.
sample_type	First OneInN RandomPct	Especifica el método de muestreo.
first_n	*integer*	Se incluirán o descartarán los registros hasta el punto de corte especificado.
one_in_n	*number*	Incluye o descarta cada *n* registros.

Tabla 72. propiedades de samplenode (continuación)

Propiedad de samplenode	Tipo de datos	Descripción de la propiedad
rand_pct	*number*	Especifica el porcentaje de registros que incluir o descartar.
use_max_size	*Flag*	Activa el uso del parámetro maximum_size.
maximum_size	*integer*	Especifica la muestra más grande que se va a incluir o descartar de la ruta de datos. Esta opción es redundante, por lo que se desactiva cuando se especifican las opciones First e Include.
set_random_seed	*Flag*	Activa el uso del parámetro de semillas aleatorias.
random_seed	*integer*	Especifica el valor utilizado como semilla aleatoria.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*integer*	
variable_proportions	*campo*	
variable_counts	*campo*	
use_min_stratum_size	*Flag*	
minimum_stratum_size	*integer*	Esta opción sólo se aplica cuando se toma una muestra compleja con Sample units=Proportions.
use_max_stratum_size	*Flag*	
maximum_stratum_size	*integer*	Esta opción sólo se aplica cuando se toma una muestra compleja con Sample units=Proportions.
clusters	*campo*	

Tabla 72. propiedades de samplenode (continuación)		
Propiedad de samplenode	**Tipo de datos**	**Descripción de la propiedad**
stratify_by	*[field1 ... fieldn]*	
specify_input_weight	*Flag*	
input_weight	*campo*	
new_output_weight	*serie*	
sizes_proportions	[[string *valor cadena*] [string *valor cadena*]...]	Si sample_units=proportions y sample_size_proportions=Custom, especifica un valor para cada combinación posible de valores de campos de especificación.
default_proportion	*number*	
sizes_counts	[[string *valor cadena*] [string *valor cadena*]...]	Especifica un valor para cada combinación de valores posible de campos de estratificación. Se utiliza de forma similar a sizes_proportions pero especificando un entero en lugar de una proporción.
default_count	*number*	

propiedades de selectnode

El nodo Seleccionar selecciona o descarta un subconjunto de registros de la ruta de datos en función de una condición específica. Por ejemplo, podría seleccionar los registros que pertenezcan a una región de ventas determinada.

Ejemplo

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

Tabla 73. propiedades de selectnode		
Propiedad de selectnode	**Tipo de datos**	**Descripción de la propiedad**
mode	Include Discard	Especifica si incluir o descartar los registros seleccionados.
condition	*serie*	Condición para incluir o descartar registros.

propiedades de sortnode

Los nodos Ordenar organizan registros en orden ascendente o descendente atendiendo a los valores de uno o varios campos.

Ejemplo

```
node = stream.create("sort", "My node")
node.setPropertyValue("keys", [["Age", "Ascending"], ["Sex", "Descending"]])
node.setPropertyValue("default_ascending", False)
node.setPropertyValue("use_existing_keys", True)
node.setPropertyValue("existing_keys", [["Age", "Ascending"]])
```

Tabla 74. propiedades de sortnode

Propiedad de sortnode	**Tipo de datos**	**Descripción de la propiedad**
keys	*lista*	Especifica los campos que desea ordenar. Si no se especifica ninguna dirección, se utilizará la predeterminada.
default_ascending	*Flag*	Especifica el orden de clasificación predeterminado.
use_existing_keys	*Flag*	Determina si la clasificación se optimiza usando el orden de clasificación anterior de los campos que ya están ordenados.
existing_keys		Especifica los campos que ya están ordenados y la dirección en que dicho orden se ha establecido. Utiliza el mismo formato que las propiedades keys.

Propiedades de spacetimeboxes

Las cajas-espacio-tiempo (STB) son una extensión de las ubicaciones espaciales demarcadas mediante Geohash. En concreto, una STB es una cadena alfanumérica que representa una región de forma regular de espacio y tiempo.

Tabla 75. Propiedades de spacetimeboxes

Propiedades de spacetimeboxes	**Tipo de datos**	**Descripción de la propiedad**
mode	*IndividualRecords* *Hangouts*	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	

Tabla 75. Propiedades de spacetimeboxes (continuación)		
Propiedades de spacetimeboxes	**Tipo de datos**	**Descripción de la propiedad**

densities	*[density, density, density...]*	Cada densidad es una serie. Por ejemplo: STB_GH8_1DAY
		Tenga en cuenta que existen límites con respecto a qué densidades son válidas.
		Para geohash, se pueden utilizar los valores de GH1-GH15.
		Para la parte temporal, se pueden utilizar los valores siguientes:
		`EVER`
		`1YEAR`
		`1MONTH`
		`1DAY`
		`12HOURS`
		`8HOURS`
		`6HOURS`
		`4HOURS`
		`3HOURS`
		`2HOURS`
		`1HOUR`
		`30MINS`
		`15MINS`
		`10MINS`
		`5MINS`
		`2 MINS`
		`1 MIN`
		`30SECS`
		`15SECS`
		`10SECS`
		`5 SECS`
		`2 SECS`
		`1SEC`

field_name_extension	*serie*	
add_extension_as	*Prefix* *Suffix*	
hangout_density	*densidad*	Densidad única (ver más arriba)
id_field	*campo*	

qualifying_duration	`1DAY`
	`12HOURS`
	`8HOURS`
	`6HOURS`
	`4HOURS`
	`2HOURS`
	`1HOUR`
	`30MIN`
	`15MIN`
	`10MIN`
	`5MIN`
	`2MIN`
	`1MIN`
	`30SECS`
	`15SECS`
	`10SECS`
	`5SECS`
	`2SECS`
	`1SECS`
		Esto debe ser una serie.

min_events	*integer*	El valor mínimo es 2

Tabla 75. Propiedades de spacetimeboxes (continuación)		
Propiedades de spacetimeboxes	**Tipo de datos**	**Descripción de la propiedad**
qualifying_pct	*integer*	Debe estar dentro del rango de 1-100

Propiedades streamingtimeseries

El nodo Serie temporal de modalidad continua genera y puntúa los modelos de serie temporal en un solo paso.

Nota: Este nodo Serie temporal de modalidad continua sustituye el nodo Resolución de TS original que estaba en desuso en la versión 18 de SPSS Modeler.

Tabla 76. propiedades streamingtimeseries		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
targets	*campo*	El nodo Serie temporal de modalidad continua prevé uno o más objetivos, de forma opcional, utilizando uno o más campos de entrada como predictores. Los campos de frecuencia y ponderación no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
candidate_inputs	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
use_period	*Flag*	
date_time_field	*campo*	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*campo*	
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
timestamp_increments	*integer*	
cyclic_increments	*integer*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*Flag*	
cross_hour	*Flag*	
aggregate_and_distribute	*lista*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	

Tabla 76. propiedades streamingtimeseries (continuación)

Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*integer*	
use_estimation_period	*Flag*	
estimation_period	Observations Times	
date_estimation	*lista*	Solo está disponible si se utiliza date_time_field
period_estimation	*lista*	Solo está disponible si se utiliza use_period
observations_type	Latest Earliest	
observations_num	*integer*	
observations_exclude	*integer*	
method	ExpertModeler Exsmooth Arima	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*Flag*	
detect_outliers	*Flag*	
expert_outlier_additive	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_innovational	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_transient	*Flag*	
expert_outlier_seasonal_additive	*Flag*	
expert_outlier_local_trend	*Flag*	
expert_outlier_additive_patch	*Flag*	
consider_newesmodels	*Flag*	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	
futureValue_type_method	Compute specify	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*integer*	
arima.d	*integer*	
arima.q	*integer*	
arima.sp	*integer*	
arima.sd	*integer*	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
arima.sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*Flag*	
tf_arima.p. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.d. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.q. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sp. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sd. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sq. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.delay. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.transformation_type. *fieldname*	None SquareRoot NaturalLog	Para funciones de transferencia.
arima_detect_outliers	*Flag*	
arima_outlier_additive	*Flag*	
arima_outlier_level_shift	*Flag*	
arima_outlier_innovational	*Flag*	
arima_outlier_transient	*Flag*	
arima_outlier_seasonal_additive	*Flag*	
arima_outlier_local_trend	*Flag*	
arima_outlier_additive_patch	*Flag*	
conf_limit_pct	*REAL*	
events	*Campos*	
forecastperiods	*integer*	
extend_records_into_future	*Flag*	

Tabla 76. propiedades streamingtimeseries (continuación)		
Propiedades streamingtimeseries	**Valores**	**Descripción de la propiedad**
conf_limits	*Flag*	
noise_res	*Flag*	

Propiedades streamingts (en desuso)

Nota: Este nodo Serie temporal de modalidad continua estaba en desuso en la versión 18 de SPSS Modeler y se sustituye por el nodo Serie temporal de modalidad continua que se ha designado para aplicar la potencia de IBM SPSS Analytic Server y procesas grandes cantidades de datos (Big Data).

El nodo Generación de análisis TS construye y puntúa modelos de series temporales en un paso, sin la necesidad de un nodo Intervalos de tiempo.

Ejemplo

```
node = stream.create("streamingts", "My node")
node.setPropertyValue("deployment_force_rebuild", True)
node.setPropertyValue("deployment_rebuild_mode", "Count")
node.setPropertyValue("deployment_rebuild_count", 3)
node.setPropertyValue("deployment_rebuild_pct", 11)
node.setPropertyValue("deployment_rebuild_field", "Year")
```

Tabla 77. propiedades streamingts		
Propiedades de streamingts	**Tipo de datos**	**Descripción de la propiedad**
custom_fields	*Flag*	Si custom_fields=false, se utilizarán los valores de un nodo Tipo situado en un punto anterior de la ruta. Si custom_fields=true, deberán especificarse targets e inputs.
targets	[*field1...fieldN*]	
inputs	[*field1...fieldN*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*Flag*	
conf_limit_pct	*REAL*	
use_time_intervals_node	*Flag*	Si use_time_intervals_node=true, se utilizarán los valores de un nodo Intervalos de tiempo situado en un punto anterior de la ruta. Si use_time_intervals_node=false, deberán especificarse interval_offset_position, interval_offset e interval_type.
interval_offset_position	LastObservation LastRecord	LastObservation se refiere a la **Última observación válida**. LastRecord se refiere a la **Cuenta hacia atrás a partir del último registro**.

Tabla 77. propiedades streamingts (continuación)		
Propiedades de streamingts	**Tipo de datos**	**Descripción de la propiedad**
interval_offset	*number*	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*Campos*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*Flag*	
detect_outliers	*Flag*	
expert_outlier_additive	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_innovational	*Flag*	
expert_outlier_transient	*Flag*	
expert_outlier_seasonal_additive	*Flag*	
expert_outlier_local_trend	*Flag*	
expert_outlier_additive_patch	*Flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_d	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_q	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo

Tabla 77. propiedades streamingts (continuación)

Propiedades de streamingts	**Tipo de datos**	**Descripción de la propiedad**
arima_sp	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_sd	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_sq	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo
arima_transformation_type	None SquareRoot NaturalLog	Misma propiedad que el nodo de modelado Serie de tiempo
arima_include_constant	*Flag*	Misma propiedad que el nodo de modelado Serie de tiempo
tf_arima_p.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_d.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_q.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sp.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sd.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_sq.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_delay.*fieldname*	*integer*	Misma propiedad que el nodo de modelado Serie de tiempo. Para funciones de transferencia.
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*Flag*	
arima_outlier_level_shift	*Flag*	
arima_outlier_innovational	*Flag*	

Tabla 77. propiedades streamingts (continuación)		
Propiedades de `streamingts`	**Tipo de datos**	**Descripción de la propiedad**
arima_outlier_transient	*Flag*	
arima_outlier_seasonal_additive	*Flag*	
arima_outlier_local_trend	*Flag*	
arima_outlier_additive_patch	*Flag*	
deployment_force_rebuild	*Flag*	
deployment_rebuild_mode	`Count` `Percent`	
deployment_rebuild_count	*number*	
deployment_rebuild_pct	*number*	
deployment_rebuild_field	*<field>*	

Capítulo 11. Propiedades de nodos de operaciones con campos

propiedades de anonymizenode

El nodo Anonimizar transforma la manera en que se representan los nombres y los valores de los campos a partir de ese punto de la ruta, lo que permite disfrazar los datos originales. Puede resultar útil si desea permitir que otros usuarios generen modelos utilizando datos confidenciales, como los nombres de los clientes u otros detalles.

Ejemplo

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

Tabla 78. propiedades de anonymizenode

Propiedades de anonymizenode	**Tipo de datos**	**Descripción de la propiedad**
enable_anonymize	*Flag*	Cuando se establece en True, activa la anonimización de los valores de los campos (equivale a seleccionar **Sí** para dicho campo en la columna Anonimizar valores).
use_prefix	*Flag*	Cuando se establece en True, se utilizará un prefijo personalizado, si es que se ha establecido uno. Se aplica a los campos que se anonimizarán mediante el método Hash y es equivalente a elegir el botón de radio **Personalizado** en el cuadro de diálogo Reemplazar valores correspondiente a dicho campo.
prefix	*serie*	Equivale a escribir un prefijo en el cuadro de texto del cuadro de diálogo Reemplazar valores. El prefijo predeterminado es el valor predeterminado, si no se ha especificado otra cosa.
transformation	Random Fixed	Determina si los parámetros de transformación que se aplican a un campo anonimizado mediante el método Transformar serán aleatorios o fijos.
set_random_seed	*Flag*	Cuando se establece en True, se utilizará el valor inicial especificado (si transformation también se ha establecido en Random).

Tabla 78. propiedades de anonymizenode (continuación)		
Propiedades de anonymizenode	**Tipo de datos**	**Descripción de la propiedad**
random_seed	*integer*	Cuando set_random_seed se establece en True, esta es la semilla para el número aleatorio.
scale	*number*	Cuando transformation se establece en Fixed, se utiliza este valor para "scale by" (escalar por). El valor máximo de la escala suele ser 10, pero puede reducirse para evitar desbordamientos.
translate	*number*	Cuando transformation se establece en Fixed, se utiliza este valor para "translate" (convertir). El valor máximo de traslación suele ser 1000, pero puede reducirse para evitar desbordamientos.

properties autodataprepnode

El nodo de preparación automática de datos (ADP) puede analizar sus datos e identificar los valores fijos, cribar los campos problemáticos o que no serán útiles y derivar nuevos atributos cuando sea necesario y mejorar el rendimiento mediante técnicas de cribado y muestreo inteligente. Puede utilizar el nodo de forma totalmente automática, permitiendo que el nodo seleccione y aplique valores fijos, o bien puede tener una vista previa de los cambios antes de que se apliquen y aceptarlos o rechazarlos.

Ejemplo

```
node = stream.create("autodataprep", "My node")
node.setPropertyValue("objective", "Balanced")
node.setPropertyValue("excluded_fields", "Filter")
node.setPropertyValue("prepare_dates_and_times", True)
node.setPropertyValue("compute_time_until_date", True)
node.setPropertyValue("reference_date", "Today")
node.setPropertyValue("units_for_date_durations", "Automatic")
```

Tabla 79. properties autodataprepnode		
properties autodataprepnode	**Tipo de datos**	**Descripción de la propiedad**
objective	Balanced Speed Accuracy Custom	
custom_fields	*Flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
target	*campo*	Especifica un campo de objetivo único.

Tabla 79. properties autodataprepnode (continuación)		
properties autodataprepnode	**Tipo de datos**	**Descripción de la propiedad**
inputs	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
use_frequency	*Flag*	
frequency_field	*campo*	
use_weight	*Flag*	
weight_field	*campo*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*Flag*	Controla el acceso a todos los campos de fecha y hora
compute_time_until_date	*Flag*	
reference_date	Today Fixed	
fixed_date	*Date*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*Flag*	
reference_time	CurrentTime Fixed	
fixed_time	*Tiempo*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	

Tabla 79. properties autodataprepnode (continuación)		
properties autodataprepnode	**Tipo de datos**	**Descripción de la propiedad**
extract_year_from_date	*Flag*	
extract_month_from_date	*Flag*	
extract_day_from_date	*Flag*	
extract_hour_from_time	*Flag*	
extract_minute_from_time	*Flag*	
extract_second_from_time	*Flag*	
exclude_low_quality_inputs	*Flag*	
exclude_too_many_missing	*Flag*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*Flag*	
maximum_number_categories	*number*	
exclude_if_large_category	*Flag*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*Flag*	
adjust_type_inputs	*Flag*	
adjust_type_target	*Flag*	
reorder_nominal_inputs	*Flag*	
reorder_nominal_target	*Flag*	
replace_outliers_inputs	*Flag*	
replace_outliers_target	*Flag*	
replace_missing_continuous_inputs	*Flag*	
replace_missing_continuous_target	*Flag*	
replace_missing_nominal_inputs	*Flag*	
replace_missing_nominal_target	*Flag*	
replace_missing_ordinal_inputs	*Flag*	
replace_missing_ordinal_target	*Flag*	
maximum_values_for_ordinal	*number*	

Tabla 79. properties autodataprepnode (continuación)

properties autodataprepnode	Tipo de datos	Descripción de la propiedad
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*Flag*	
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*Flag*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*Flag*	
maximize_association_with_target	*Flag*	
p_value_for_merging	*number*	
merge_ordinal_features	*Flag*	
merge_nominal_features	*Flag*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*Flag*	
p_value_for_binning	*number*	
perform_feature_selection	*Flag*	
p_value_for_selection	*number*	
perform_feature_construction	*Flag*	
transformed_target_name_extension	*serie*	
transformed_inputs_name_extension	*serie*	
constructed_features_root_name	*serie*	

Tabla 79. properties autodataprepnode (continuación)		
properties autodataprepnode	**Tipo de datos**	**Descripción de la propiedad**
years_duration_ name_extension	*serie*	
months_duration_ name_extension	*serie*	
days_duration_ name_extension	*serie*	
hours_duration_ name_extension	*serie*	
minutes_duration_ name_extension	*serie*	
seconds_duration_ name_extension	*serie*	
year_cyclical_name_extens ion	*serie*	
month_cyclical_name_exten sion	*serie*	
day_cyclical_name_extensi on	*serie*	
hour_cyclical_name_extens ion	*serie*	
minute_cyclical_name_exte nsion	*serie*	
second_cyclical_name_exte nsion	*serie*	

propiedades de astimeintervalsnode

Utilice el nodo Intervalos de tiempo para especificar intervalos y derivar un nuevo campo de tiempo para la estimación o predicción. Se admite una gama completa de intervalos de tiempo que abarca desde segundos a años.

Tabla 80. propiedades de astimeintervalsnode		
Propiedades de astimeintervalsnode	**Tipo de datos**	**Descripción de la propiedad**
time_field	*campo*	Solo puede aceptar un único campo continuo. Ese campo lo utiliza el nodo como la clave de agregación para convertir el intervalo. Si se utiliza aquí un campo de entero, se considera como un índice de tiempo.
dimensions	*[campo1 campo2 ... campon]*	Estos campos se utilizan para crear series temporales individuales basándose en los valores de campo.

Tabla 80. propiedades de astimeintervalsnode (continuación)		
Propiedades de astimeintervalsnode	**Tipo de datos**	**Descripción de la propiedad**
fields_to_aggregate	*[campo1 campo2 ... campon]*	Estos campos se agregan como parte del cambio del período del campo de tiempo. Los campos no incluidos en este selector se filtran y excluyen de los datos que salen del nodo.

propiedades de binningnode

El nodo Intervalos crea automáticamente nuevos campos nominales (conjunto) en función de los valores de uno o más campos continuos (rango numérico) existentes. Por ejemplo, puede transformar un campo de ingresos continuo en un campo categórico nuevo que contenga grupos de ingresos como desviaciones desde la media. Una vez creados los intervalos para el campo nuevo, puede generar un nodo Derivar en función de los puntos de corte.

Ejemplo

```
node = stream.create("binning", "My node")
node.setPropertyValue("fields", ["Na", "K"])
node.setPropertyValue("method", "Rank")
node.setPropertyValue("fixed_width_name_extension", "_binned")
node.setPropertyValue("fixed_width_add_as", "Suffix")
node.setPropertyValue("fixed_bin_method", "Count")
node.setPropertyValue("fixed_bin_count", 10)
node.setPropertyValue("fixed_bin_width", 3.5)
node.setPropertyValue("tile10", True)
```

Tabla 81. propiedades de binningnode		
Propiedad de binningnode	**Tipo de datos**	**Descripción de la propiedad**
fields	*[field1 field2 ... fieldn]*	Los campos continuos (rango numérico) pendientes de transformación. Se pueden crear intervalos de varios campos de forma simultánea.
method	FixedWidth EqualCount Rank SDev Optimal	Método utilizado para determinar los puntos de corte de los intervalos de campo nuevos (categorías).
rcalculate_bins	Always IfNecessary	Especifica si se vuelven a calcular los intervalos y los datos se colocan en el intervalo adecuado cada vez que se ejecuta el nodo o si los datos sólo se añaden a los intervalos existentes y cualquier nuevo intervalo que se haya añadido.

Tabla 81. propiedades de binningnode (continuación)

Propiedad de binningnode	Tipo de datos	Descripción de la propiedad
fixed_width_name_extension	*serie*	La extensión predeterminada es *_BIN*.
fixed_width_add_as	Suffix Prefix	Determina si la extensión se debe añadir al principio (prefijo) o al final (sufijo) del nombre de campo. La extensión predeterminada es *income_BIN*.
fixed_bin_method	Width Count	
fixed_bin_count	*integer*	Especifica un número entero para determinar el número de intervalos de anchura fija (categorías) para los nuevos campos.
fixed_bin_width	*REAL*	Valor (entero o real) para calcular el ancho del intervalo.
equal_count_name_ extension	*serie*	La extensión predeterminada es *_TILE*.
equal_count_add_as	Suffix Prefix	Especifica una extensión, sufijo o prefijo, utilizada para el nombre de los campos generados con p-tiles estándar. La extensión predeterminada es *_TILE* más *N*, donde *N* es el número de cuantil.
tile4	*Flag*	Genera cuatro intervalos de cuantiles, cada uno con el 25% de los casos.
tile5	*Flag*	Genera cinco intervalos de quintiles.
tile10	*Flag*	Genera 10 intervalos de deciles.
tile20	*Flag*	Genera 20 intervalos de veintiles.
tile100	*Flag*	Genera 100 intervalos de percentiles.
use_custom_tile	*Flag*	
custom_tile_name_extension	*serie*	La extensión predeterminada es *_TILEN*.
custom_tile_add_as	Suffix Prefix	
custom_tile	*integer*	
equal_count_method	RecordCount ValueSum	El método RecordCount trata de asignar el mismo número de registros a cada intervalo, mientras que ValueSum asigna registros de manera que la suma de los valores de cada intervalo sea la misma.

Tabla 81. propiedades de binningnode (continuación)

Propiedad de binningnode	Tipo de datos	Descripción de la propiedad
tied_values_method	Next Current Random	Especifica en qué intervalo se van a insertar los datos de valor empatado.
rank_order	Ascending Descending	Esta propiedad incluye Ascending (el valor más bajo se marca con 1) o Descending (el valor más alto se marca con 1).
rank_add_as	Suffix Prefix	Esta opción se aplica al rango, rango fraccional y rango como porcentaje.
rank	*Flag*	
rank_name_extension	*serie*	La extensión predeterminada es *_RANK*.
rank_fractional	*Flag*	Establece rangos de casos en los que el valor del campo nuevo es igual al rango dividido por la suma de las ponderaciones de los casos que no están perdidos. Los rangos fraccionales están dentro del rango 0–1.
rank_fractional_name_ extension	*serie*	La extensión predeterminada es *_F_RANK*.
rank_pct	*Flag*	Cada rango se divide por el número de registros con valores válidos y se multiplica por 100. Los rangos fraccionales de porcentaje están dentro del rango 1–100.
rank_pct_name_extension	*serie*	La extensión predeterminada es *_P_RANK*.
sdev_name_extension	*serie*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*serie*	La extensión predeterminada es *_OPTIMAL*.
optimal_add_as	Suffix Prefix	

Tabla 81. propiedades de binningnode (continuación)		
Propiedad de binningnode	**Tipo de datos**	**Descripción de la propiedad**
optimal_supervisor_field	*campo*	Campo elegido como campo supervisor, con el que se relacionan los campos seleccionados para los intervalos.
optimal_merge_bins	*Flag*	Especifica que todos los intervalos con un número pequeño de casos se añadirán a un intervalo vecino de mayor tamaño.
optimal_small_bin_threshold	*integer*	
optimal_pre_bin	*Flag*	Indica si debe agruparse previamente en intervalos el conjunto de datos.
optimal_max_bins	*integer*	Especifica un límite superior con el fin de evitar que se genere un número desmesurado de intervalos.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

propiedades de derivenode

El nodo Derivar modifica los valores de datos o crea campos nuevos desde uno o más campos existentes. Crea campos de fórmula de tipo, distintivo, nominal, estado, recuento y condicional.

Ejemplo 1

```
# Create and configure a Flag Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Create and configure a Conditional Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" ><
@INDEX")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

Ejemplo 2

Este script presupone que existen dos columnas numéricas denominadas XPos e YPos que representan las coordenadas X e Y de un punto (por ejemplo, donde se ha producido un evento). El script crea un nodo Derivar que calcula una columna geoespacial de coordenadas X e Y que representan dicho punto en un sistema de coordenadas específico:

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabla 82. propiedades de derivenode

Propiedad de derivenode	**Tipo de datos**	**Descripción de la propiedad**
new_name	*serie*	Nombre del campo nuevo.
mode	Single Multiple	Especifica si los campos son únicos o múltiples.
fields	*lista*	Se utiliza en modo Múltiple solamente para seleccionar varios campos.
name_extension	*serie*	Especifica la extensión de los nombres de los nuevos campos.
add_as	Suffix Prefix	Añade la extensión como un prefijo (al principio) o como un sufijo (al final) del nombre de los campos.
result_type	Formula Flag Set State Count Conditional	Los seis tipos de campos nuevos que se pueden crear.
formula_expr	*serie*	Expresión para calcular un nuevo valor de campo en el nodo Derivar.
flag_expr	*serie*	
flag_true	*serie*	
flag_false	*serie*	

Tabla 82. propiedades de derivenode (continuación)		
Propiedad de derivenode	**Tipo de datos**	**Descripción de la propiedad**
set_default	*serie*	
set_value_cond	*serie*	Estructurada para proporcionar la condición asociada a un valor dado.
state_on_val	*serie*	Especifica el valor del campo nuevo cuando se cumple la condición Activado.
state_off_val	*serie*	Especifica el valor del campo nuevo cuando se cumple la condición Desactivado.
state_on_expression	*serie*	
state_off_expression	*serie*	
state_initial	On Off	Asigna a cada registro del nuevo campo un valor inicial activado (On) o desactivado (Off). Este valor puede cambiar a medida que se cumplan las condiciones.
count_initial_val	*serie*	
count_inc_condition	*serie*	
count_inc_expression	*serie*	
count_reset_condition	*serie*	
cond_if_cond	*serie*	
cond_then_expr	*serie*	
cond_else_expr	*serie*	

Tabla 82. propiedades de derivenode (continuación)

Propiedad de derivenode	**Tipo de datos**	**Descripción de la propiedad**
formula_measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Esta propiedad se puede utilizar para definir la medición asociada con el campo derivado. La función de establecimiento se puede pasar como una cadena o uno de los valores MeasureType. El método de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad define el tipo de medición asociado con los valores subyacentes.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	En campos geoespaciales, esta propiedad define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores
has_coordinate_system	*Boolean*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas

Tabla 82. propiedades de derivenode (continuación)		
Propiedad de derivenode	**Tipo de datos**	**Descripción de la propiedad**
coordinate_system	*serie*	En campos geoespaciales, esta propiedad define el sistema de coordenadas para este campo

propiedades de ensemblenode

El nodo Conjunto combina dos o más nuggets de modelo para obtener predicciones más precisas de las que se pueden obtener de cualquier modelo individual.

Ejemplo

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

Tabla 83. propiedades de ensemblenode		
propiedades de ensemblenode	**Tipo de datos**	**Descripción de la propiedad**
ensemble_target_field	*campo*	Especifica el campo objetivo de todos los modelos utilizados en el conjunto.
filter_individual_model_output	*Flag*	Especifica si los resultados de puntuación de los modelos individuales se deben eliminar.
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.

Tabla 83. propiedades de ensemblenode (continuación)		
propiedades de ensemblenode	**Tipo de datos**	**Descripción de la propiedad**
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
set_voting_tie_selection	Random HighestConfidence	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.
calculate_standard_error	*Flag*	Si el campo objetivo es continuo, se ejecuta un error estándar de forma predeterminada para calcular la diferencia entre los valores medidos o estimados y los valores true; y para mostrar si las estimaciones coinciden.

propiedades de fillernode

El nodo Rellenar sustituye valores de campos y cambia el almacenamiento. Puede sustituir los valores en función de una condición CLEM, como `@BLANK(@FIELD)`. También puede sustituir todos los espacios vacíos o valores nulos por un valor específico. Un nodo Rellenar suelen utilizarse junto con un nodo Tipo para sustituir valores perdidos.

Ejemplo

```
node = stream.create("filler", "My node")
node.setPropertyValue("fields", ["Age"])
node.setPropertyValue("replace_mode", "Always")
node.setPropertyValue("condition", "(\"Age\" > 60) and (\"Sex\" = \"M\"")
node.setPropertyValue("replace_with", "\"old man\"")
```

Tabla 84. propiedades de fillernode		
Propiedad de fillernode	**Tipo de datos**	**Descripción de la propiedad**
fields	*lista*	Campos del conjunto de datos cuyos valores se van a examinar y sustituir.

Tabla 84. propiedades de fillernode (continuación)		
Propiedad de `fillernode`	**Tipo de datos**	**Descripción de la propiedad**

`replace_mode`	`Always`
	
	`Conditional`
	
	`Blank`
	
	`Null`
	
	`BlankAndNull`
		Se pueden sustituir todos los valores, valores vacíos o valores nulos, o bien reemplazar aquellos basados en una condición específica.

`condition`	*serie*	
`replace_with`	*serie*	

propiedades de filternode

El nodo Filtrar filtra (descarta) campos, vuelve a nombrar campos y correlaciona campos de nodo de origen a otro.

Ejemplo:

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

Usando la propiedad default_include. Tenga en cuenta que, si establece el valor de la propiedad `default_include`, no se incluirán o excluirán automáticamente todos los campos, sino que simplemente se determinará el valor predeterminado de los seleccionados actualmente. Esto equivale funcionalmente a pulsar en el botón **Incluir campos de forma predeterminada** del cuadro de diálogo del nodo Filtrar. Por ejemplo, imagine que ejecuta el siguiente script:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

Esto hará que el nodo pase los campos Age y Sex y descarte todos los demás. Después de ejecutar el script anterior, ahora suponga que añade las líneas siguientes al script para nombrar dos campos más:

```
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

Esto añadirá dos campos más al filtro para que se pasen un total de cuatro campos (Age, Sex, BP, Na). En otras palabras, restablecer el valor de `default_include` a `False` no restablece automáticamente todos los campos.

Si lo desea, si ahora cambia `default_include` a `True` (ya sea usando un script o en el cuadro de diálogo del nodo Filtrar), cambiará el comportamiento de forma que los cuatro campos enumerados

anteriormente no se incluirían, sino que quedarían descartados. Si no está seguro, se recomienda experimentar con los controles del cuadro de diálogo del nodo Filtrar para entender esta interacción.

Tabla 85. propiedades de filternode

Propiedad de filternode	Tipo de datos	Descripción de la propiedad
default_include	*Flag*	Propiedad con clave para especificar si el comportamiento predeterminado es para pasar o filtrar los campos: Tenga en cuenta que, si se establece esta propiedad, no se incluirán o excluirán automáticamente todos los campos, sino que simplemente se determinará si los campos seleccionados se incluyen o excluyen de forma predeterminada. Vea el ejemplo que se muestra a continuación para ver más comentarios.
include	*Flag*	Propiedad con clave para incluir y eliminar el campo.
new_name	*serie*	

propiedades de historynode

El nodo Historial se utiliza para crear campos nuevos que contienen datos de los campos de registros anteriores. Los nodos Historial se suelen utilizar para los datos secuenciales, como los datos de series temporales. Antes de utilizar un nodo Historial, es posible que desee clasificar los datos utilizando un nodo Ordenar.

Ejemplo

```
node = stream.create("history", "My node")
node.setPropertyValue("fields", ["Drug"])
node.setPropertyValue("offset", 1)
node.setPropertyValue("span", 3)
node.setPropertyValue("unavailable", "Discard")
node.setPropertyValue("fill_with", "undef")
```

Tabla 86. propiedades de historynode

Propiedad de historynode	Tipo de datos	Descripción de la propiedad
fields	*lista*	Campos para los que desea un historial.
offset	*number*	Especifica el último registro anterior al registro actual desde el que desea extraer valores del campo histórico.
span	*number*	Especifica el número de registros anteriores de los que desea extraer valores.
unavailable	Discard Leave Fill	Para tratar registros que no tienen valores de historial, suele hacer referencia a los primeros registros (en la parte superior del conjunto de datos), de los que no hay registros previos que utilizar como historial.

Tabla 86. propiedades de historynode (continuación)		
Propiedad de historynode	**Tipo de datos**	**Descripción de la propiedad**
fill_with	String Number	Especifica un valor o cadena que utilizar en el caso de los registros en los que no existen valores de historial disponibles.

propiedades de partitionnode

El nodo Partición genera un campo de partición, que divide los datos en subconjuntos diferentes para las fases de entrenamiento, comprobación y validación en la generación del modelo.

Ejemplo

```
node = stream.create("partition", "My node")
node.setPropertyValue("create_validation", True)
node.setPropertyValue("training_size", 33)
node.setPropertyValue("testing_size", 33)
node.setPropertyValue("validation_size", 33)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 123)
node.setPropertyValue("value_mode", "System")
```

Tabla 87. propiedades de partitionnode		
Propiedad de partitionnode	**Tipo de datos**	**Descripción de la propiedad**
new_name	*serie*	Nombre del campo de partición generado por el nodo.
create_validation	*Flag*	Especifica si se debe crear una partición de validación.
training_size	*integer*	Porcentaje de registros (0-100) que se van a asignar a la partición de entrenamiento.
testing_size	*integer*	Porcentaje de registros (0-100) que se van a asignar a la partición de comprobación.
validation_size	*integer*	Porcentaje de registros (0-100) que se van a asignar a la partición de entrenamiento. Se ignora si no se crea una partición de validación.
training_label	*serie*	Etiqueta para la partición de entrenamiento.
testing_label	*serie*	Etiqueta para la partición de comprobación.
validation_label	*serie*	Etiqueta para la partición de validación. Se ignora si no se crea una partición de validación.
value_mode	System SystemAndLabel Label	Especifica los valores utilizados para representar cada partición en los datos. Por ejemplo, la muestra de entrenamiento puede estar representado por el entero del sistema 1, la etiqueta Training o una combinación de ambos, 1_Training.

Tabla 87. propiedades de partitionnode (continuación)		
Propiedad de partitionnode	**Tipo de datos**	**Descripción de la propiedad**
set_random_seed	*boolean*	Especifica si se debe utilizar una semilla aleatoria especificada por el usuario.
random_seed	*integer*	Valor de semilla aleatoria especificada por el usuario. Para que se pueda utilizar este valor, set_random_seed se debe establecer en True.
enable_sql_generation	*boolean*	Especifica si se utiliza la retrotracción SQL para asignar registros a particiones.
unique_field		Especifica el campo de entrada que se utiliza para garantizar que los registros se asignan a particiones de una forma aleatoria pero reproducible. Para que se pueda utilizar este valor, enable_sql_generation se debe establecer en True.

propiedades de reclassifynode

El nodo Reclasificar transforma un conjunto de valores categóricos en otro. La reclasificación es útil para contraer categorías o reagrupar datos para su análisis.

Ejemplo

```
node = stream.create("reclassify", "My node")
node.setPropertyValue("mode", "Multiple")
node.setPropertyValue("replace_field", True)
node.setPropertyValue("field", "Drug")
node.setPropertyValue("new_name", "Chemical")
node.setPropertyValue("fields", ["Drug", "BP"])
node.setPropertyValue("name_extension", "reclassified")
node.setPropertyValue("add_as", "Prefix")
node.setKeyedPropertyValue("reclassify", "drugA", True)
node.setPropertyValue("use_default", True)
node.setPropertyValue("default", "BrandX")
node.setPropertyValue("pick_list", ["BrandX", "Placebo", "Generic"])
```

Tabla 88. propiedades de reclassifynode		
Propiedad de reclassifynode	**Tipo de datos**	**Descripción de la propiedad**
mode	Single Multiple	Single reclasifica las categorías de un campo. Multiple activa las opciones que permiten la transformación de varios campos al mismo tiempo.
replace_field	*Flag*	
field	*serie*	Sólo se utiliza en modo Single.
new_name	*serie*	Sólo se utiliza en modo Single.

Tabla 88. propiedades de reclassifynode (continuación)		
Propiedad de reclassifynode	**Tipo de datos**	**Descripción de la propiedad**
fields	*[field1 field2 ... fieldn]*	Sólo se utiliza en modo Multiple.
name_extension	*serie*	Sólo se utiliza en modo Multiple.
add_as	Suffix Prefix	Sólo se utiliza en modo Multiple.
reclassify	*serie*	Propiedad estructurada para valores de campos.
use_default	*Flag*	Utiliza el valor predeterminado.
default	*serie*	Especifica un valor predeterminado.
pick_list	*[string string ... string]*	Permite al usuario importar una lista de valores nuevos conocidos para rellenar la lista desplegable de la tabla.

propiedades de reordernode

El nodo Reorg. campos define el orden natural utilizado para mostrar los campos en la parte posterior de la ruta. Este orden afecta a la visualización de los campos en diversas ubicaciones, como las tablas, las listas y el selector de campos. Esta operación es útil cuando se trabaja con conjuntos de datos amplios para hacer más visibles los campos de interés.

Ejemplo

```
node = stream.create("reorder", "My node")
node.setPropertyValue("mode", "Custom")
node.setPropertyValue("sort_by", "Storage")
node.setPropertyValue("ascending", False)
node.setPropertyValue("start_fields", ["Age", "Cholesterol"])
node.setPropertyValue("end_fields", ["Drug"])
```

Tabla 89. propiedades de reordernode		
Propiedad de reordernode	**Tipo de datos**	**Descripción de la propiedad**
mode	Custom Auto	Se pueden ordenar los valores de forma automática o especificar un orden personalizado.
sort_by	Name Type Storage	
ascending	*Flag*	
start_fields	*[campo1 campo2 ... campon]*	Los campos nuevos se han insertado después de estos campos.

Tabla 89. propiedades de reordernode (continuación)		
Propiedad de reordernode	**Tipo de datos**	**Descripción de la propiedad**
end_fields	*[campo1 campo2 ... campon]*	Los campos nuevos se han insertado antes de estos campos.

propiedades de reprojectnode

En SPSS Modeler, elementos como las funciones espaciales Creador de expresiones, el nodo de Predicción espacio-temporal (STP) y el nodo Visualización de mapas utilizan el sistema de coordenadas proyectado. Utilice el nodo Reproyectar para cambiar el sistema de coordenadas de los datos que importa y que utilizan un sistema de coordenadas geográficas.

Tabla 90. propiedades de reprojectnode		
Propiedades de reprojectnode	**Tipo de datos**	**Descripción de la propiedad**
reproject_fields	*[campo1 campo2 ... campon]*	Lista todos los campos que se van a reproyectar.
reproject_type	Streamdefault Specify	Elija cómo reproyectar los campos.
coordinate_system	*serie*	El nombre del sistema de coordenadas que se aplicará a los campos. Ejemplo: set reprojectnode.coordinate_system = “WGS_1984_World_Mercator”

propiedades de restructurenode

El nodo Reestructurar convierte un campo nominal o marca en un grupo de campos que se puede rellenar con los valores todavía de otro campo. Por ejemplo, dado un campo llamado *tipo de pago*, con valores de *crédito*, *Efectivo*y *Débito*, se crearían tres nuevos campos (*crédito*, *Efectivo*, *Débito*), cada uno de los cuales podría contener el valor del pago real realizado.

Ejemplo

```
node = stream.create("restructure", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("include_field_name", True)
node.setPropertyValue("value_mode", "OtherFields")
node.setPropertyValue("value_fields", ["Age", "BP"])
```

Tabla 91. propiedades de restructurenode		
Propiedad de restructurenode	**Tipo de datos**	**Descripción de la propiedad**
fields_from	[*category category category*] all	

Tabla 91. propiedades de restructurenode (continuación)		
Propiedad de `restructurenode`	**Tipo de datos**	**Descripción de la propiedad**
`include_field_name`	*Flag*	Indica si se debe usar el nombre del campo en el nombre de campo reestructurado.

`value_mode`	`OtherFields`
	
	`Flags`
		Indica el modo de definir los valores para los campos reestructurados. Con `OtherFields`, debe especificar los campos que se van a usar (consulte a continuación). Con `Flags`, los valores son marcas numéricas.

`value_fields`	*lista*	Es necesario en caso de que `value_mode` sea `OtherFields`. Especifica los campos que se van a usar como campos de valores.

propiedades de rfmanalysisnode

El nodo de Análisis de Recencia, Frecuencia, Monetaria (RFM) le permite determinar cuantitativamente qué clientes son más propensos a ser los mejores examinando la última vez que compraron de usted (recency), con qué frecuencia compraron (frecuencia) y cuánto gastaron en todas las transacciones (monetaria).

Ejemplo

```
node = stream.create("rfmanalysis", "My node")
node.setPropertyValue("recency", "Recency")
node.setPropertyValue("frequency", "Frequency")
node.setPropertyValue("monetary", "Monetary")
node.setPropertyValue("tied_values_method", "Next")
node.setPropertyValue("recalculate_bins", "IfNecessary")
node.setPropertyValue("recency_thresholds", [1, 500, 800, 1500, 2000, 2500])
```

Tabla 92. propiedades de rfmanalysisnode		
propiedades de `rfmanalysisnode`	**Tipo de datos**	**Descripción de la propiedad**
`recency`	*campo*	Especifica el campo de actualidad. Puede ser una fecha, marca de tiempo o un número simple.
`frequency`	*campo*	Especifica el campo de frecuencia.
`monetary`	*campo*	Especifica el campo de monetario.
`recency_bins`	*integer*	Especifica el número de intervalos de actividades recientes que se van a generar.
`recency_weight`	*number*	Especifica la ponderación que se aplicará a los datos de actividades recientes. El valor predeterminado es 100.
`frequency_bins`	*integer*	Especifica el número de intervalos de frecuencia que se van a generar.

Tabla 92. propiedades de rfmanalysisnode (continuación)		
propiedades de `rfmanalysisnode`	**Tipo de datos**	**Descripción de la propiedad**
`frequency_weight`	*number*	Especifica la ponderación que se aplicará a los datos de frecuencia. El valor por omisión es 10.
`monetary_bins`	*integer*	Especifica el número de intervalos de monetario que se van a generar.
`monetary_weight`	*number*	Especifica la ponderación que se aplicará a los datos de monetario. El valor predeterminado es 1.

`tied_values_method`	`Next`
	
	`Current`
		Especifica en qué intervalo se van a insertar los datos de valor empatado.

`recalculate_bins`	`Always`
	
	`IfNecessary`	

`add_outliers`	*Flag*	Sólo está disponible si `recalculate_bins` se define como `IfNecessary`. Si se selecciona, los registros por debajo del intervalo más inferior se añaden al intervalo inferior y los registros por encima, se añaden al intervalo superior.

`binned_field`	`Recency`
	
	`Frequency`
	
	`Monetary`	

`recency_thresholds`	*valor valor*	Sólo está disponible si `recalculate_bins` se define como `Always`. Especifica los umbrales superior e inferior de los intervalos de actividades recientes. El umbral superior de un intervalo se utiliza como el umbral inferior del siguiente, por ejemplo, [10 30 60] definiría dos intervalos, el primer intervalo con los umbrales superior e inferior de 10 y 30, con los umbrales del segundo intervalo de 30 y 60.
`frequency_thresholds`	*valor valor*	Sólo está disponible si `recalculate_bins` se define como `Always`.
`monetary_thresholds`	*valor valor*	Sólo está disponible si `recalculate_bins` se define como `Always`.

propiedades de settoflagnode

El nodo Marcas deriva varios campos de marcas en función de los valores categóricos definidos para uno o más campos nominales.

Ejemplo

```
node = stream.create("settoflag", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("true_value", "1")
node.setPropertyValue("false_value", "0")
node.setPropertyValue("use_extension", True)
node.setPropertyValue("extension", "Drug_Flag")
node.setPropertyValue("add_as", "Suffix")
node.setPropertyValue("aggregate", True)
node.setPropertyValue("keys", ["Cholesterol"])
```

Tabla 93. propiedades de settoflagnode

Propiedad de settoflagnode	**Tipo de datos**	**Descripción de la propiedad**
fields_from	[*category category category*] all	
true_value	*serie*	Especifica el valor para verdadero utilizado por el nodo al configurar una marca. El valor predeterminado es T (del inglés 'True').
false_value	*serie*	Especifica el valor para falso utilizado por el nodo al configurar una marca. El valor predeterminado es F (del inglés 'True').
use_extension	*Flag*	Utiliza una extensión como sufijo o prefijo para el nuevo campo de marca.
extension	*serie*	
add_as	Suffix Prefix	Especifica si la extensión es un sufijo o un prefijo.
aggregate	*Flag*	Agrupa registros en función de campos clave. Si algún registro se establece como verdadero, se activarán todos los campos de marca de un grupo.
keys	*lista*	Campos clave.

propiedades de statisticstransformnode

El nodo Transformación Statistics ejecuta una selección de comandos de sintaxis de IBM SPSS Statistics en los orígenes de datos de IBM SPSS Modeler. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Las propiedades de este nodo están descritas en “propiedades de statisticstransformnode” en la página 435.

Propiedades timeintervalsnode (en desuso)

Nota: Este nodo estaba en desuso en la versión 18 de SPSS Modeler y se sustituye por el nuevo nodo Serie temporal.

El nodo Intervalos de tiempo especifica intervalos y genera etiquetas (si es necesario) para modelar los datos de series temporales. Si los valores no están espaciados de manera uniforme, el nodo puede rellenar o agregar valores según sea necesario para crear un intervalo uniforme entre registros.

Ejemplo

```
node = stream.create("timeintervals", "My node")
node.setPropertyValue("interval_type", "SecondsPerDay")
node.setPropertyValue("days_per_week", 4)
node.setPropertyValue("week_begins_on", "Tuesday")
node.setPropertyValue("hours_per_day", 10)
node.setPropertyValue("day_begins_hour", 7)
node.setPropertyValue("day_begins_minute", 5)
node.setPropertyValue("day_begins_second", 17)
node.setPropertyValue("mode", "Label")
node.setPropertyValue("year_start", 2005)
node.setPropertyValue("month_start", "January")
node.setPropertyValue("day_start", 4)
node.setKeyedPropertyValue("pad", "AGE", "MeanOfRecentPoints")
node.setPropertyValue("agg_mode", "Specify")
node.setPropertyValue("agg_set_default", "Last")
```

Tabla 94. propiedades de timeintervalsnode		
Propiedad de `timeintervalsnode`	**Tipo de datos**	**Descripción de la propiedad**

`interval_type`	`None`
	`Periods`
	`CyclicPeriods`
	`Years`
	`Quarters`
	`Months`
	`DaysPerWeek`
	`DaysNonPeriodic`
	`HoursPerDay`
	`HoursNonPeriodic`
	`MinutesPerDay`
	`MinutesNonPeriodic`
	`SecondsPerDay`
	`SecondsNonPeriodic`	

`mode`	`Label` `Create`	Determina si desea etiquetar los registros de manera consecutiva o crear la serie según una fecha, marca de tiempo o campo de tiempo concretos.
`field`	*campo*	Al crear la serie a partir de los datos, especifica el campo que informa de la fecha u hora de cada registro.
`period_start`	*integer*	Especifica el intervalo de inicio para períodos o períodos cíclicos.
`cycle_start`	*integer*	Ciclo de inicio de los períodos cíclicos.
`year_start`	*integer*	En el caso de los tipos de intervalo que procedan, el año en que el primer intervalo tiene lugar.
`quarter_start`	*integer*	En el caso de los tipos de intervalo que procedan, el trimestre en que el primer intervalo tiene lugar.

Tabla 94. propiedades de timeintervalsnode (continuación)		
Propiedad de `timeintervalsnode`	**Tipo de datos**	**Descripción de la propiedad**
month_start	January February March April May June July August September October November December	
day_start	*integer*	
hour_start	*integer*	
minute_start	*integer*	
second_start	*integer*	
periods_per_cycle	*integer*	En el caso de los períodos cíclicos, número en cada ciclo.
fiscal_year_begins	January February March April May June July August September October November December	En el caso de los intervalos trimestrales, especifica el mes en el que comienza el año fiscal.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	En el caso de los intervalos periódicos (días a la semana, horas al día, minutos al día y segundos al día), especifica el día en el que comienza la semana.

Tabla 94. propiedades de timeintervalsnode (continuación)		
Propiedad de `timeintervalsnode`	**Tipo de datos**	**Descripción de la propiedad**
`day_begins_hour`	*integer*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica la hora a la que comienza el día. Se puede usar junto con `day_begins_minute` y `day_begins_second` para determinar una hora exacta, como *8:05:01*. Vea el ejemplo de uso que se muestra a continuación.
`day_begins_minute`	*integer*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el minuto en el que comienza el día (por ejemplo, *5* en *8:05*).
`day_begins_second`	*integer*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el segundo en el que comienza el día (por ejemplo, *17* en *8:05:17*).
`days_per_week`	*integer*	En el caso de los intervalos periódicos (días a la semana, horas al día, minutos al día y segundos al día), especifica el número de días a la semana.
`hours_per_day`	*integer*	En el caso de los intervalos periódicos (horas al día, minutos al día y segundos al día), especifica el número de horas al día.

`interval_increment`	`1`
	`2`
	`3`
	`4`
	`5`
	`6`
	`10`
	`15`
	`20`
	`30`
		En el caso de los minutos al día y de los segundos al día, especifica el número de minutos o segundos que se va a aumentar en cada registro.

Tabla 94. propiedades de timeintervalsnode (continuación)		
Propiedad de timeintervalsnode	**Tipo de datos**	**Descripción de la propiedad**
field_name_extension	*serie*	
field_name_extension_as_prefix	*Flag*	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	

Tabla 94. propiedades de timeintervalsnode (continuación)		
Propiedad de `timeintervalsnode`	**Tipo de datos**	**Descripción de la propiedad**
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	Especifica el método de agregación para un campo.
pad	Blank MeanOfRecentPoints True False	Especifica el método de relleno para un campo.
agg_mode	All Specify	Determina si se agregan o rellenan todos los campos con las funciones predeterminadas según sea necesario o bien si se especifican los campos y las funciones que deben usarse.
agg_range_default	Mean Sum Mode Min Max	Especifica la función predeterminada que se va a usar al agregar campos continuos.
agg_set_default	Mode First Last	Especifica la función predeterminada que se va a usar al agregar campos nominales.

Tabla 94. propiedades de timeintervalsnode (continuación)		
Propiedad de `timeintervalsnode`	**Tipo de datos**	**Descripción de la propiedad**

`agg_flag_default`	`TrueIfAnyTrue`
	
	`Mode`
	
	`First`
	
	`Last`	

`pad_range_default`	`Blank`
	
	`MeanOfRecentPoints`
		Especifica la función predeterminada que se va a usar al agregar campos continuos.

`pad_set_default`	`Blank`
	
	`MostRecentValue`	

`pad_flag_default`	`Blank`
	
	`True`
	
	`False`	

`max_records_to_create`	*integer*	Especifica el número máximo de registros que se van a crear al rellenar la serie.
`estimation_from_beginning`	*Flag*	
`estimation_to_end`	*Flag*	
`estimation_start_offset`	*integer*	
`estimation_num_holdouts`	*integer*	
`create_future_records`	*Flag*	
`num_future_records`	*integer*	
`create_future_field`	*Flag*	
`future_field_name`	*serie*	

propiedades de transposenode

El nodo Transponer cambia los datos de filas y columnas para que los registros se conviertan en campos y los campos se conviertan en registros.

Ejemplo

```
node = stream.create("transpose", "My node")
node.setPropertyValue("transposed_names", "Read")
node.setPropertyValue("read_from_field", "TimeLabel")
```

```
node.setPropertyValue("max_num_fields", "1000")
node.setPropertyValue("id_field_name", "ID")
```

Tabla 95. propiedades de transposenode

Propiedad de transposenode	**Tipo de datos**	**Descripción de la propiedad**
transpose_method	*enum*	Especifica el método de transposición: Normal (normal), CASE a VAR (casetovar) o VAR a CASE (vartocase).
transposed_names	Prefix Read	Propiedad para el método de transposición normal. Se pueden generar nuevos nombres de campo automáticamente a partir de un prefijo concreto o bien se pueden leer desde un campo existente en los datos.
prefix	*serie*	Propiedad para el método de transposición normal.
num_new_fields	*integer*	Propiedad para el método de transposición normal. Al usar un prefijo, especifica el número máximo de campos nuevos que se van a crear.
read_from_field	*campo*	Propiedad para el método de transposición normal. Campo del que se leen los nombres. Debe tratarse de un campo instanciado o, de lo contrario, se producirá un error al ejecutar el nodo.
max_num_fields	*integer*	Propiedad para el método de transposición normal. Al leer nombres de un campo, especifica un límite superior con el fin de evitar que se genere un número desmesurado de campos.
transpose_type	Numeric String Custom	Propiedad para el método de transposición normal. De forma predeterminada, solamente los campos continuos (rango numérico) se transponen, si bien se puede elegir un subconjunto personalizado de campos numéricos o, en su lugar, transponer todos los campos de cadena.
transpose_fields	*lista*	Propiedad para el método de transposición normal. Especifica los campos que se van a transponer cuando se usa la opción Custom.
id_field_name	*campo*	Propiedad para el método de transposición normal.
transpose_casetovar_id fields	*campo*	Propiedad para el método de transposición CASE a VAR (casetovar). Acepta varios campos a utilizar como campos de índice. field1 ... fieldN

Tabla 95. propiedades de transposenode (continuación)		
Propiedad de transposenode	**Tipo de datos**	**Descripción de la propiedad**
transpose_casetovar_columnfields	*campo*	Propiedad para el método de transposición CASE a VAR (casetovar). Acepta varios campos que se utilizarán como campos de columna. field1 ... fieldN
transpose_casetovar_valuefields	*campo*	Propiedad para el método de transposición CASE a VAR (casetovar). Acepta varios campos que se utilizarán como campos de valor. field1 ... fieldN
transpose_vartocase_idfields	*campo*	Propiedad para el método de transposición VAR a CASE (vartocase). Acepta varios campos que se deberán utilizar como campos de variable de ID. field1 ... fieldN
transpose_vartocase_valfields	*campo*	Propiedad para el método de transposición VAR a CASE (vartocase). Acepta varios campos que se deberán utilizar como campos de variable de valor. field1 ... fieldN

propiedades de typenode

El nodo Tipo especifica propiedades y metadatos de campo. Por ejemplo, puede especificar un nivel de medición (continuo, nominal, ordinal o marca) para cada campo, establecer las opciones para gestionar valores perdidos y nulos del sistema, establecer el rol de un campo con fines de modelado, especificar las etiquetas de valor y campo y especificar los valores de un campo.

Ejemplo

```
node = stream.createAt("type", "My node", 50, 50)
node.setKeyedPropertyValue("check", "Cholesterol", "Coerce")
node.setKeyedPropertyValue("direction", "Drug", "Input")
node.setKeyedPropertyValue("type", "K", "Range")
node.setKeyedPropertyValue("values", "Drug", ["drugA", "drugB", "drugC",
"drugD", "drugX",
 "drugY", "drugZ"])
node.setKeyedPropertyValue("null_missing", "BP", False)
node.setKeyedPropertyValue("whitespace_missing", "BP", False)
node.setKeyedPropertyValue("description", "BP", "Blood Pressure")
node.setKeyedPropertyValue("value_labels", "BP", [["HIGH", "High Blood
Pressure"],
 ["NORMAL", "normal blood pressure"]])
```

Observe que en algunos casos puede que sea necesario instanciar totalmente el nodo Tipo para que otros nodos funcionen adecuadamente, como, por ejemplo, la propiedad `fields from` del nodo Marcas. Simplemente conecte un nodo Tabla y ejecútelo para instanciar los campos:

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

Tabla 96. propiedades de typenode

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
direction	Input Target Both None Partition Split Frequency RecordID	Propiedad con clave para los roles de los campos. **Nota:** Los valores In y Out han quedado en desuso. Es posible que ya no puedan utilizarse en futuras versiones.
type	Range Flag Set Typeless Discrete OrderedSet Default	Nivel de medición del campo (anteriormente llamado el "tipo" de campo). Si se establece type en Default, se borrará cualquier valor del parámetro values, y si value_mode tiene el valor Specify, se restablecerá en Read. Si value_mode se establece en Pass o Read, el establecimiento de type no afectará a value_mode. **Nota:** Los tipos de datos utilizados internamente difieren de los que son visibles en el nodo tipo. La correspondencia es la siguiente: Range -> Continuous Set - > Nominal OrderedSet -> Ordinal Discrete- > Categorical

Tabla 96. propiedades de typenode (continuación)

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
storage	Unknown String Integer Real Time Date Timestamp	Propiedad con clave de solamente lectura para el tipo de almacenamiento de campos.
check	None Nullify Coerce Discard Warn Abort	Propiedad con clave para la comprobación del rango y el tipo de campo.
values	[*value value*]	Para un campo continuos, el primer valor es el mínimo y el último es el máximo. Para campos nominales, especifique todos los valores. Para los campos marca, el primer valor representa *falso* y el último, *verdadero*. La configuración de esta propiedad establece de forma automática la propiedad value_mode en Specify.
value_mode	Read Pass Read+ Current Specify	Determina la forma en la que se establecen los valores. Tenga en cuenta que no puede establecer esta propiedad directamente en Specify. Para utilizar valores específicos, establezca la propiedad values.

Tabla 96. propiedades de typenode (continuación)

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
extend_values	*Flag*	Se aplica cuando value_mode se establece en Read. Establézcala en T para añadir nuevos valores de lectura a los valores existentes del campo. Establézcala en F para descartar los valores existentes y favorecer a los nuevos valores de lectura.
enable_missing	*Flag*	Cuando está definida como T, activa el seguimiento de los valores perdidos para el campo.
missing_values	[*value value ...*]	Especifica los valores de datos que denotan los datos perdidos.
range_missing	*Flag*	Especifica si se ha definido un rango de valores perdidos (vacíos) para un campo.
missing_lower	*serie*	Si range_missing es verdadero, especifica el límite inferior del rango de valores perdidos.
missing_upper	*serie*	Si range_missing es verdadero, especifica el límite superior del rango de valores perdidos.
null_missing	*Flag*	Cuando se establece en T, los valores *nulos* (valores no definidos que se muestran como $null$ en el software) se consideran valores perdidos.
whitespace_missing	*Flag*	Cuando se establece en T, los valores que sólo contienen un espacio en blanco (espacios, tabulaciones y líneas nuevas) se consideran valores perdidos.
description	*serie*	Especifica la descripción de un campo.
value_labels	*[[Value LabelString] [Value LabelString] ...]*	Se utiliza para especificar etiquetas para los pares de valores.
display_places	*integer*	Establece el número de cifras decimales para el campo cuando se muestra (sólo se aplica a campos con almacenamiento REAL). Un valor de –1 utilizará la corriente predeterminada.
export_places	*integer*	Establece el número de cifras decimales para el campo cuando se exporta (sólo se aplica a campos con almacenamiento REAL). Un valor de –1 utilizará la corriente predeterminada.

Tabla 96. propiedades de typenode (continuación)

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
decimal_separator	DEFAULT PERIOD COMMA	Establece el separador decimal para el campo (sólo se aplica a campos con almacenamiento REAL).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Establece el formato de fecha del campo (sólo se aplica a los campos con almacenamiento DATE o TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Establece el formato de hora para el campo (sólo se aplica a campos con almacenamiento TIME o TIMESTAMP).
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Establece el formato de presentación de los números para el campo.

Tabla 96. propiedades de typenode (continuación)		
Propiedad de typenode	**Tipo de datos**	**Descripción de la propiedad**
standard_places	*integer*	Establece el número de cifras decimales para el campo cuando se muestra en formato estándar. Un valor de –1 utilizará el valor predeterminado del flujo. Tenga en cuenta que el intervalo display_places existente también cambiará esto, pero ahora se ha desaprobado.
scientific_places	*integer*	Establece el número de cifras decimales para el campo cuando se muestra en formato científico. Un valor de –1 utilizará el valor predeterminado del flujo.
currency_places	*integer*	Establece el número de cifras decimales para el campo cuando se muestra en formato moneda. Un valor de –1 utilizará el valor predeterminado del flujo.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Establece el símbolo de agrupación para el campo.
column_width	*integer*	Establece el ancho de columna para el campo. Un valor de –1 establecerá el ancho de columna en Auto.
justify	AUTO CENTER LEFT RIGHT	Establece la justificación de columna para el campo.

Tabla 96. propiedades de typenode (continuación)

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Esta propiedad con clave es similar a type en cuanto a que puede utilizarse para definir la medición asociada al campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores MeasureType, mientras que la función de obtención siempre devolverá los valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de recopilación (listas con profundidad 0), esta propiedad con clave define el tipo de medición asociado con los valores subyacentes.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	En campos geoespaciales, esta propiedad con clave define el tipo del objeto geoespacial representado por este campo. Debería ser coherente con la profundidad de lista de los valores.
has_coordinate_ system	*Boolean*	En campos geoespaciales, esta propiedad define si este campo tiene un sistema de coordenadas
coordinate_ system	*serie*	En campos geoespaciales, esta propiedad con clave define el sistema de coordenadas para este campo.

Tabla 96. propiedades de typenode (continuación)

Propiedad de typenode	Tipo de datos	Descripción de la propiedad
custom_storage_ type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Esta propiedad con clave es similar a custom_storage en cuanto a que puede utilizarse para definir el almacenamiento de alteración temporal para el campo. La diferencia es que, en los scripts Python, la función de establecimiento puede pasar también uno de los valores StorageType, mientras que la función de obtención siempre devolverá los valores StorageType.
custom_list_ storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Para campos de lista, esta propiedad con clave especifica el tipo de almacenamiento de los valores subyacentes.
custom_list_depth	*integer*	Para campos de lista, esta propiedad con clave especifica la profundidad del campo.
max_list_length	*integer*	Solo está disponible para datos con un nivel de medición de *Geoespacial* o *Colección*. Establezca la longitud máxima de la lista especificando el número de elementos que puede contener la lista.
max_string_length	*integer*	Solo está disponible para datos *sin tipos* y se utiliza al generar SQL para crear una tabla. Entre el valor de la serie más larga en los datos; esto genera una columna en la tabla que es lo suficientemente grande para contener la serie.

Capítulo 12. Propiedades de nodos Gráfico

Propiedades comunes del nodo Gráfico

Esta sección describe las propiedades disponibles para los nodos de gráficos, incluidas las comunes y aquellas específicas de cada tipo de nodo.

Tabla 97. Propiedades comunes de nodos de gráficos

Propiedades comunes de nodos de gráficos	**Tipo de datos**	**Descripción de la propiedad**
`title`	*serie*	Especifica el título. Ejemplo: "Esto es un título".
`caption`	*serie*	Especifica el pie. Por ejemplo: "Esto es un pie".

`output_mode`	`Screen`
	
	`File`
		Determina si el resultado del nodo de gráficos se muestra o si se guarda en un archivo.

`output_format`	`BMP`
	
	`JPEG`
	
	`PNG`
	
	`HTML`
	
	`output` (*.cou*)
		Especifica el tipo de resultado. El tipo exacto de resultado permitido para cada nodo varía.

`full_filename`	*serie*	Especifica la ruta de destino y el nombre de archivo del resultado generado por el nodo de gráficos.
`use_graph_size`	*Flag*	Controla si el tamaño del gráfico se ha establecido de manera explícita mediante las propiedades de ancho y altura a continuación. Afecta solamente a los gráficos que tienen salida a pantalla. No disponible para el nodo Distribución.
`graph_width`	*number*	Cuando `use_graph_size` es `True`, establece el ancho del gráfico en píxeles.
`graph_height`	*number*	Cuando `use_graph_size` es `True`, establece la altura del gráfico en píxeles.

Desactivación de los campos opcionales

Los campos opcionales, como un campo de preformato para gráficos, se pueden desactivar estableciendo el valor de propiedad en `" "` (serie vacía), como se muestra en el ejemplo siguiente:

```
plotnode.setPropertyValue("color_field", "")
```

Especificación de colores

Los colores de los títulos, pies, fondos y etiquetas se pueden especificar mediante las cadenas hexadecimales que comiencen con el símbolo almohadilla (#). Por ejemplo, para establecer el fondo del gráfico en cielo azul, debe utilizar la siguiente instrucción:

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

Aquí, los dos primeros dígitos, 87, especifican el contenido rojo, los dos del medio, CE, especifican el contenido verde y los dos últimos, EB, el contenido azul. Cada dígito puede tomar un valor del rango 0-9 o A-F. Juntos, estos valores pueden especificar red-green-blue, o RGB o color.

Nota: Al especificar colores en RVA, puede utilizar el selector de campos en la interfaz de usuario para determinar el código de color correcto. Basta con colocarse sobre el color para ver un texto con la información deseada.

Propiedades de collectionnode

El nodo Colección muestra la distribución de valores de un campo numérico relativo a los valores de otro. (Crea gráficos parecidos a los histogramas.) Es útil para ilustrar una variable o un campo cuyos valores cambian con el tiempo. Con los gráficos 3D también puede incluir un eje simbólico que muestra las distribuciones por categoría.

Ejemplo

```
node = stream.create("collection", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

Tabla 98. propiedades de collectionnode

Propiedad de collectionnode	**Tipo de datos**	**Descripción de la propiedad**
over_field	*campo*	
over_label_auto	*Flag*	
over_label	*serie*	
collect_field	*campo*	
collect_label_auto	*Flag*	
collect_label	*serie*	
three_D	*Flag*	

Tabla 98. propiedades de collectionnode (continuación)		
Propiedad de collectionnode	**Tipo de datos**	**Descripción de la propiedad**
by_field	*campo*	
by_label_auto	*Flag*	
by_label	*serie*	
operation	Sum Mean Min Max SDev	
color_field	*serie*	
panel_field	*serie*	
animation_field	*serie*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*Flag*	
graph_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.

Propiedades de distributionnode

El nodo Distribución muestra las instancias de valores simbólicos (categóricos), como el tipo de hipoteca o el género. Normalmente, podría usar el nodo Distribución para mostrar los desequilibrios de los datos, que pueden rectificarse mediante el nodo Equilibrar antes de crear un modelo.

Ejemplo

```
node = stream.create("distribution", "My node")
# "Plot" tab
```

```
node.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

Tabla 99. propiedades de distributionnode

Propiedades de distributionnode	**Tipo de datos**	**Descripción de la propiedad**
plot	SelectedFields Flags	
x_field	*campo*	
color_field	*campo*	Campo de superposición.
normalize	*Flag*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*Flag*	

Propiedades de evaluationnode

El nodo Evaluación ayuda a evaluar y comparar modelos predictivos. El diagrama de evaluación muestra la calidad con que los modelos predicen resultados particulares. Ordena registros en función del valor predicho y la confianza de la predicción. Divide el registro en grupos de igual tamaño (**cuantiles**) y, a continuación, representa el valor del criterio de negocio de cada cuantil de mayor a menor. El gráfico muestra múltiples modelos como líneas independientes.

Ejemplo

```
node = stream.create("evaluation", "My node")
# "Plot" tab
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

Tabla 100. propiedades de evaluationnode		
Propiedad de evaluationnode	**Tipo de datos**	**Descripción de la propiedad**
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*Flag*	
field_detection_method	Metadata Name	
use_fixed_cost	*Flag*	
cost_value	*number*	
cost_field	*serie*	
use_fixed_revenue	*Flag*	
revenue_value	*number*	
revenue_field	*serie*	
use_fixed_weight	*Flag*	
weight_value	*number*	
weight_field	*campo*	
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*Flag*	
style	Line Point	

Tabla 100. propiedades de evaluationnode (continuación)		
Propiedad de evaluationnode	**Tipo de datos**	**Descripción de la propiedad**
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
export_data	*Flag*	
data_filename	*serie*	
delimiter	*serie*	
new_line	*Flag*	
inc_field_names	*Flag*	
inc_best_line	*Flag*	
inc_business_rule	*Flag*	
business_rule_condition	*serie*	
plot_score_fields	*Flag*	
score_fields	*[field1 ... fieldn]*	
target_field	*campo*	
use_hit_condition	*Flag*	
hit_condition	*serie*	
use_score_expression	*Flag*	
score_expression	*serie*	
caption_auto	*Flag*	

Propiedades de graphboardnode

El nodo Tablero ofrece muchos tipos diferentes de gráficos en un único nodo. Con este nodo puede seleccionar los campos de datos que desee explorar y seleccionar un gráfico de los disponibles para los datos seleccionados. El nodo filtra automáticamente cualquier tipo de gráfico que no sea compatible con las selecciones de campo.

Nota: Si establece una propiedad que no es válida para el tipo de gráfico (por ejemplo, si especifica y_field para un histograma), se ignora dicha propiedad.

Nota: En la interfaz de usuario, en la ficha Detallado de muchos tipos diferentes de gráficos, hay un campo **Resumen**; este campo no está soportado actualmente utilizando scripts.

Ejemplo

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

Tabla 101. Propiedades de graphboardnode

Propiedades de graphboard	Tipo de datos	Descripción de la propiedad
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues	Identifica el tipo de gráfico.

Tabla 101. Propiedades de graphboardnode (continuación)		
Propiedades de graphboard	**Tipo de datos**	**Descripción de la propiedad**
	ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap	

Tabla 101. Propiedades de graphboardnode (continuación)

Propiedades de graphboard	Tipo de datos	Descripción de la propiedad
	PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	
x_field	*campo*	Especifica una etiqueta personalizada para el eje *x*. Disponible solamente para etiquetas.
y_field	*campo*	Especifica una etiqueta personalizada para el eje *y*. Disponible solamente para etiquetas.
z_field	*campo*	Se utiliza en algunos gráficos 3D.
color_field	*campo*	Se utiliza en mapas de calor.
size_field	*campo*	Se utiliza en gráficos de burbujas.
categories_field	*campo*	
values_field	*campo*	
rows_field	*campo*	
columns_field	*campo*	
fields	*campo*	
start_longitude_field	*campo*	Se utiliza con flechas en un mapa de referencia.
end_longitude_field	*campo*	
start_latitude_field	*campo*	
end_latitude_field	*campo*	
data_key_field	*campo*	Se utiliza en diversos mapas.
panelrow_field	*serie*	
panelcol_field	*serie*	
animation_field	*serie*	

Tabla 101. Propiedades de graphboardnode (continuación)		
Propiedades de graphboard	**Tipo de datos**	**Descripción de la propiedad**
longitude_field	*campo*	Se utiliza en mapas de coordenadas.
latitude_field	*campo*	
map_color_field	*campo*	

Propiedades de histogramnode

El nodo Histograma muestra las instancias de valores de los campos numéricos. Se suele utilizar para explorar los datos antes de las manipulaciones y la generación de modelos. Al igual que con el nodo Distribución, con frecuencia el nodo Histograma detecta desequilibrios en los datos.

Ejemplo

```
node = stream.create("histogram", "My node")
# "Plot" tab
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

Tabla 102. propiedades de histogramnode		
Propiedad de histogramnode	**Tipo de datos**	**Descripción de la propiedad**
field	*campo*	
color_field	*campo*	
panel_field	*campo*	
animation_field	*campo*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*Flag*	

Tabla 102. propiedades de histogramnode (continuación)		
Propiedad de histogramnode	**Tipo de datos**	**Descripción de la propiedad**
separate_bands	*Flag*	
x_label_auto	*Flag*	
x_label	*serie*	
y_label_auto	*Flag*	
y_label	*serie*	
use_grid	*Flag*	
graph_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
normal_curve	*Flag*	Indica si se debe mostrar la curva de distribución normal en los resultados.

Propiedades de mapvisualization

El nodo Visualización de mapas puede aceptar varias conexiones de entrada y visualizar datos geoespaciales en un mapa como una serie de capas. Cada capa es un único campo geoespacial; por ejemplo, la capa base podría ser el mapa de un país, y por encima puede tener una capa de carreteras, una capa de ríos y una capa de ciudades.

Tabla 103. Propiedades de mapvisualization		
Propiedades de mapvisualization	**Tipo de datos**	**Descripción de la propiedad**
tag	*serie*	Establece el nombre de la etiqueta para la entrada. La etiqueta predeterminada es un número basado en el orden en el que las entradas estaban conectadas al nodo (la primera etiqueta de conexión es 1, la segunda etiqueta de conexión es 2, etc.

Tabla 103. Propiedades de mapvisualization (continuación)

Propiedades de `mapvisualization`	**Tipo de datos**	**Descripción de la propiedad**
`layer_field`	*campo*	Selecciona qué geocampo se muestra del conjunto de datos como una capa en el mapa. La selección predeterminada se basa en el orden de clasificación siguiente: • Primero - Punto • Cadena lineal • Polígono • Multipunto • Cadena multilínea • Último - Multipolígono Si existen dos campos con el mismo tipo de medida, de forma predeterminada, se seleccionará el primer campo alfabéticamente (por nombre).
`color_type`	*Boolean*	Especifica si se va a aplicar un color estándar a todas las características del geocampo, o un campo de superposición que colorea las características basándose en valores de otro campo del conjunto de datos. Los valores posibles son `standard` u `overlay`. El valor predeterminado es `standard`.
`color`	*serie*	Si `standard` está seleccionado para `color_type`, el desplegable contiene la misma paleta de colores que el orden de colores de la categoría de gráfico en la pestaña Visualización de opciones de usuario. El valor predeterminado es el color de categoría de gráfico 1.
`color_field`	*campo*	Si `overlay` está seleccionado para `color_type`, el desplegable contiene todos los campos del mismo conjunto de datos que el geocampo seleccionado como la capa.
`symbol_type`	*Boolean*	Especifica si se va a aplicar un símbolo estándar a todos los registro del geocampo, o un símbolo de superposición que cambie el icono de símbolo para los puntos basándose en los valores de otro campo del conjunto de datos. Los valores posibles son `standard` u `overlay`. El valor predeterminado es `standard`.

Tabla 103. Propiedades de mapvisualization (continuación)

Propiedades de `mapvisualization`	**Tipo de datos**	**Descripción de la propiedad**
`symbol`	*serie*	Si `standard` está seleccionado para `symbol_type`, el desplegable contiene una selección de símbolos que se puede utilizar para mostrar puntos en el mapa.
`symbol_field`	*campo*	Si `overlay` está seleccionado para `symbol_type`, el desplegable contiene todos los campos nominales, ordinales o categóricos del mismo conjunto de datos como el geocampo seleccionado como la capa.
`size_type`	*Boolean*	Especifica si se va a aplicar un tamaño estándar a todos los registros del geocampo, o un tamaño de superposición que cambia el tamaño del icono de símbolo o el grosor de la línea basándose en valores de otro campo del conjunto de datos. Los valores posibles son `standard` u `overlay`. El valor predeterminado es `standard`.
`size`	*serie*	Si `standard` está seleccionado para `size_type`, para `point` o `multipoint`, el desplegable contiene una selección de tamaños para el símbolo seleccionado. Para `linestring` o `multilinestring`, el desplegable contiene una selección del grosor de línea.
`size_field`	*campo*	Si `overlay` está seleccionado para `size_type`, el desplegable contiene todos los campos del mismo conjunto de datos que el geocampo seleccionado como la capa.
`transp_type`	*Boolean*	Especifica si se va a aplicar una transparencia estándar a todos los registros del geocampo, o una transferencia de superposición que cambia el nivel de transparencia para el símbolo, línea o polígono basándose en valores de otro campo del conjunto de datos. Los valores posibles son `standard` u `overlay`. El valor predeterminado es `standard`.

Tabla 103. Propiedades de mapvisualization (continuación)

Propiedades de `mapvisualization`	**Tipo de datos**	**Descripción de la propiedad**
`transp`	*integer*	Si `standard` está seleccionado para `transp_type`, el desplegable contiene una selección de niveles de transparencia que empieza en `0%` (opaco) y va aumentando hasta `100%` (transparente) en incrementos del 10 %. Establece la transparencia de puntos, líneas o polígonos en el mapa. Si `overlay` está seleccionado para `size_type`, el desplegable contiene todos los campos del mismo conjunto de datos que el geocampo seleccionado como la capa. Para `points`, `multipoints`, `linestrings` y `multilinestrings`, `polygons` y `multipolygons` (que están en la capa inferior), el valor predeterminado es `0%`. Para `polygons` y `multipolygons` que no son la capa inferior, el valor predeterminado es `50%` (para evitar oscurecer las capas debajo de estos polígonos).
`transp_field`	*campo*	Si `overlay` está seleccionado para `transp_type`, el desplegable contiene todos los campos del mismo conjunto de datos que el geocampo seleccionado como la capa.
`data_label_field`	*campo*	Especifica el campo para utilizar como etiquetas de datos en el mapa. Por ejemplo, si la capa donde se aplica este valor es un capa de polígono, la etiqueta de datos podría ser el campo `name` – que contiene el nombre de cada polígono. Así si aquí se selecciona el campo `name`, se mostrarán estos nombres en el mapa.
`use_hex_binning`	*Boolean*	Habilita la agrupación hexadecimal y habilita todos los desplegables de agregación. Este valor está desactivado de forma predeterminada.

Tabla 103. Propiedades de mapvisualization (continuación)

Propiedades de `mapvisualization`	**Tipo de datos**	**Descripción de la propiedad**
`color_aggregation` y `transp_aggregation`	*serie*	Si selecciona un campo de superposición para una capa de puntos que utiliza la agrupación hexadecimal, todos los valores para dicho campo se deben agregar para todos los puntos del hexágono. Por lo tanto, debe especificar una función de agregación para cualquier otro campo de superposición que desea aplicar al mapa. Las funciones de agregación disponibles son: Continuas (almacenamiento de Real o Entero) • Suma • Media • Mín • Máx • Mediana • Primer cuartil • Tercer cuartil Continuas (almacenamiento de Hora, Fecha o Indicación de fecha y hora): • Media • Mín • Máx Nominal/Categórico: • Modo • Mín • Máx Distintivo: • True si alguno es True • False si alguno es False
`custom_storage`	*serie*	Establece el tipo de almacenamiento general del campo. El valor predeterminado es `List`. Si se ha especificado `List`, los controles `custom_value_storage` y `list_depth` siguientes están inhabilitados.
`custom_value_storage`	*serie*	Establece los tipos de almacenamiento de los elementos de la lista, en lugar de los campos como un todo. EL valor predeterminado es `Real`.

Tabla 103. Propiedades de mapvisualization (continuación)		
Propiedades de `mapvisualization`	**Tipo de datos**	**Descripción de la propiedad**
`list_depth`	*integer*	Establece la profundidad del campo de lista. Su profundidad necesaria depende del tipo de geocampo, siguiendo estos criterios: • Punto - 0 • Serie lineal - 1 • Polígono - 2 • Multipunto - 1 • Multi serie lineal - 2 • Multipolígono - 3 Debe conocer el tipo de campo geoespacial que está volviendo a convertir a una lista y la profundidad necesaria para dicho tipo de campo. Si se ha establecido de forma incorrecta, el campo no se puede utilizar. El valor predeterminado es `0`, el valor mínimo es `0`, y el valor máximo es `10`.

Propiedades de multiplotnode

El nodo G. múltiple crea un gráfico que muestra varios campos *Y* sobre un campo *X* único. Los campos *Y* están representados como líneas coloreadas; cada uno equivale a un nodo Gráfico con el estilo establecido en **Línea** y el Modo para X establecido en **Ordenar**. Los gráficos múltiples son útiles cuando quiere explorar la fluctuación de varias variables a través del tiempo.

Ejemplo

```
node = stream.create("multiplot", "My node")
# "Plot" tab
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# "Overlay" section
node.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

Tabla 104. propiedades de multiplotnode		
Propiedad de `multiplotnode`	**Tipo de datos**	**Descripción de la propiedad**
`x_field`	*campo*	
`y_fields`	*lista*	
`panel_field`	*campo*	
`animation_field`	*campo*	

Tabla 104. propiedades de multiplotnode (continuación)		
Propiedad de multiplotnode	**Tipo de datos**	**Descripción de la propiedad**
normalize	*Flag*	
use_overlay_expr	*Flag*	
overlay_expression	*serie*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*Flag*	
x_label	*serie*	
y_label_auto	*Flag*	
y_label	*serie*	
use_grid	*Flag*	
graph_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.

Propiedades de plotnode

El nodo Gráfico muestra la relación entre los campos numéricos. Puede crear un gráfico mediante puntos (un diagrama de dispersión) o líneas.

Ejemplo

```
node = stream.create("plot", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# "Output" tab
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/
plot_output.jpeg")
```

Tabla 105. propiedades de plotnode

Propiedad de plotnode	Tipo de datos	Descripción de la propiedad
x_field	*campo*	Especifica una etiqueta personalizada para el eje *x*. Disponible solamente para etiquetas.
y_field	*campo*	Especifica una etiqueta personalizada para el eje *y*. Disponible solamente para etiquetas.
three_D	*Flag*	Especifica una etiqueta personalizada para el eje *y*. Disponible sólo para etiquetas en gráficos 3D.
z_field	*campo*	
color_field	*campo*	Campo de superposición.
size_field	*campo*	
shape_field	*campo*	
panel_field	*campo*	Especifica un campo de marcas o nominal para crear un gráfico independiente para cada categoría. Los gráficos aparecerán juntos en una ventana de resultados.
animation_field	*campo*	Especifica un campo de marcas o nominal para ilustrar las categorías de los valores de datos creando una serie de gráficos secuenciados mediante la animación.
transp_field	*campo*	Especifica un campo para ilustrar las categorías de los valores de datos utilizando un nivel de transparencia distinto para cada categoría. No disponible para gráficos de líneas.
overlay_type	None Smoother Function	Determina si se muestra una función superpuesta o suavizado LOESS.
overlay_expression	*serie*	Especifica la expresión utilizada cuando overlay_type se establece en Function.
style	Point Line	

Tabla 105. propiedades de plotnode (continuación)

Propiedad de plotnode	Tipo de datos	Descripción de la propiedad
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Automatic UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*Flag*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	

Tabla 105. propiedades de plotnode (continuación)		
Propiedad de plotnode	**Tipo de datos**	**Descripción de la propiedad**
x_label_auto	*Flag*	
x_label	*serie*	
y_label_auto	*Flag*	
y_label	*serie*	
z_label_auto	*Flag*	
z_label	*serie*	
use_grid	*Flag*	
graph_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
page_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
use_overlay_expr	*Flag*	Desaprobado en favor de overlay_type.

Propiedades de timeplotnode

El nodo Gráfico de tiempo muestra uno o más conjuntos de datos de series temporales. Normalmente, primero utilizaría un nodo Intervalos de tiempo para crear un campo *Etiqueta de tiempo* , que se utilizaría para etiquetar el eje *x* .

Ejemplo

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Appearance settings
node.setPropertyValue("symbol_size", 2.0)
```

Tabla 106. propiedades de timeplotnode		
Propiedad de timeplotnode	**Tipo de datos**	**Descripción de la propiedad**
plot_series	Series Models	
use_custom_x_field	*Flag*	
x_field	*campo*	
y_fields	*lista*	
panel	*Flag*	
normalize	*Flag*	
line	*Flag*	

Tabla 106. propiedades de timeplotnode (continuación)		
Propiedad de timeplotnode	**Tipo de datos**	**Descripción de la propiedad**
points	*Flag*	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	*Flag*	Puede añadir suavizados al gráfico únicamente si establece panel en True.
use_records_limit	*Flag*	
records_limit	*integer*	
symbol_size	*number*	Especifica el tamaño del símbolo.
panel_layout	Horizontal Vertical	

Propiedades eplotnode

El nodo Gráfico electrónico (Beta) muestra la relación entre campos numéricos. Es similar al nodo Gráfico, pero sus opciones difieren y su salida utiliza una nueva interfaz de gráficos específica a este nodo. Utilice el nodo de nivel beta para probar las nuevas características de gráficos.

Tabla 107. propiedades eplotnode		
Propiedades eplotnode	**Tipo de datos**	**Descripción de la propiedad**
x_field	*serie*	Especifique el campo para mostrar en el eje X horizontal.
y_field	*serie*	Especifique el campo para mostrar en el eje Y vertical.
color_field	*serie*	Especifique el campo para utilizar para el preformato de mapa de colores en la salida, si se desea.
size_field	*serie*	Especifique el campo para utilizar para el preformato de mapa de tamaño en la salida, si se desea.

Tabla 107. propiedades eplotnode (continuación)		
Propiedades eplotnode	**Tipo de datos**	**Descripción de la propiedad**
shape_field	*serie*	Especifique el campo para utilizar para el preformato de mapa de formas en la salida, si se desea.
interested_fields	*serie*	Especifique los campos que desea incluir en la salida.
records_limit	*integer*	Especifique un número para el número máximo de registros para trazar en la salida. 2000 es el valor predeterminado.
if_over_limit	*boolean*	Especifique si se debe utilizar la opción Sample o la opción Use all data si se supera records_limit . Sample es el valor predeterminado y muestra los datos de forma aleatoria hasta que llega a records_limit. Si especifica Use all data para ignorar el records_limit y trazar todos los puntos de datos, tenga en cuenta que esto puede disminuir drásticamente el rendimiento.

Propiedades tsnenode

t-SNE (vecino estocástico con t distribuida incorporado) es una herramienta para visualizar datos de alta dimensión. Convierte afinidades de puntos de datos a probabilidades. Este nodo t-SNE en SPSS Modeler se implementa en Python y requiere la biblioteca scikit-learn© Python.

Tabla 108. propiedades tsnenode		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
mode_type	*serie*	Especifique la modalidad simple o expert .
n_components	*serie*	Dimensión del espacio incorporado (2D o 3D). Especifique 2 o 3. El valor predeterminado es 2.
method	*serie*	Especifique barnes_hut o exact. El valor predeterminado es barnes_hut.
init	*serie*	Inicialización de la incorporación. Especifique random o pca. El valor predeterminado es random.
target_field Renombrado como target a partir de la versión 18.2.1.1	*serie*	El nombre del campo objetivo. Será un mapa de colores en el gráfico de salida. El gráfico utilizará un color si no se especifica ningún campo objetivo.

Tabla 108. propiedades tsnenode (continuación)		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
perplexity	*Float*	La perplejidad está relacionada con el número de los vecinos más cercanos utilizados en otros algoritmos de aprendizaje de colector. Normalmente, los conjuntos de datos más grandes requieren una mayor perplejidad. Considere seleccionar un valor entre 5 y 50. El valor predeterminado es 30.
early_exaggeration	*Float*	Controla cómo de ajustados están los clústeres naturales del espacio original en el espacio incorporado, y cuánto espacio habrá entre ellos. El valor predeterminado es 12.0.
learning_rate	*Float*	El valor predeterminado es 200.
n_iter	*integer*	Número máximo de iteraciones para la optimización. Se establece en 250, como mínimo. El valor predeterminado es 1000.
angle	*Float*	El tamaño angular del nodo distante como se ha medido a partir de un punto. Especifique un valor dentro del rango de 0-1. El valor predeterminado es 0.5.
enable_random_seed	*boolean*	Establezca el valor en true para habilitar el parámetro random_seed. El valor predeterminado es false.
random_seed	*integer*	La semilla de aleatorización para utilizar. El valor predeterminado es None.
n_iter_without_progress	*integer*	Máximo de iteraciones sin progreso. El valor predeterminado es 300.
min_grad_norm	*serie*	Si la norma gradiente está por debajo de este umbral, la optimización se detendrá. El valor predeterminado es 1.0E-7. Los valores posibles son: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*boolean*	Se establece en true para realizar t-SNE con varias perplejidades distintas. El valor predeterminado es false.

Tabla 108. propiedades tsnenode (continuación)		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
output_Rename	*boolean*	Especifique true si desea proporcionar un nombre personalizado, o false para denominar la salida automáticamente. El valor predeterminado es false.
output_to	*serie*	Especifique Screen o Output. El valor predeterminado es Screen.
full_filename	*serie*	Especifique el nombre de archivo de salida.
output_file_type	*serie*	Formato de archivo de salida. Especifique HTML o Output object. El valor predeterminado es HTML.

Propiedades de webnode

El nodo Malla muestra la fuerza de las relaciones entre los valores de dos o más campos simbólicos (categóricos). El gráfico utiliza líneas de varios anchos para indicar la fuerza de la conexión. Podría utilizar un nodo Malla, por ejemplo, para explorar la relación entre la compra de un conjunto de artículos en un sitio de comercio electrónico.

Ejemplo

```
node = stream.create("web", "My node")
# "Plot" tab
node.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# "Options" tab
node.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

Tabla 109. propiedades de webnode		
Propiedad de webnode	**Tipo de datos**	**Descripción de la propiedad**
use_directed_web	*Flag*	
fields	*lista*	
to_field	*campo*	
from_fields	*lista*	
true_flags_only	*Flag*	

Tabla 109. propiedades de webnode (continuación)

Propiedad de webnode	Tipo de datos	Descripción de la propiedad
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*Flag*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*number*	
links_above	*number*	
discard_links_min	*Flag*	
links_min_records	*number*	
discard_links_max	*Flag*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*Flag*	
web_display	Circular Network Directed Grid	
graph_background	*Color*	Al principio de esta sección se describen los colores de gráficos estándar.
symbol_size	*number*	Especifica el tamaño del símbolo.

Capítulo 13. Propiedades de nodos de modelado

Propiedades comunes de nodos de modelado

Las siguientes propiedades son comunes a algunos o todos los nodos de modelado. Las excepciones se indican en la documentación de los nodos de modelado individuales según sea adecuado.

Tabla 110. Propiedades comunes de nodos de modelado

Propiedad	Valores	Descripción de la propiedad
`custom_fields`	*Flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.

`target`
 	
o
 	
`targets`
	campo
	
	o
	
	[*field1 ... fieldN*]
		Especifica un único campo objetivo o varios campos objetivo dependiendo del tipo de modelo.

`inputs`	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
`partition`	*campo*	
`use_partitioned_data`	*Flag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
`use_split_data`	*Flag*	
`splits`	*[field1 ... fieldn]*	Especifica el campo o campos para utilizar en el modelado de divisiones. Sólo funciona si `use_split_data` está establecido como `True`.
`use_frequency`	*Flag*	Los modelos específicos utilizan campos de ponderación y frecuencia como se indica en cada tipo de modelo.
`frequency_field`	*campo*	
`use_weight`	*Flag*	
`weight_field`	*campo*	
`use_model_name`	*Flag*	
`model_name`	*serie*	Nombre personalizado para nuevo modelo.

Tabla 110. Propiedades comunes de nodos de modelado (continuación)		
Propiedad	**Valores**	**Descripción de la propiedad**
mode	Simple Expert	

propiedades de anomalydetectionnode

El nodo Detección de anomalías identifica casos extraños, o valores atípicos, que no se ajustan a patrones de datos "normales". Con este nodo, es posible identificar los valores atípicos incluso si no se ajustan a los patrones conocidos anteriormente y aunque no esté exactamente seguro de lo que está buscando.

Ejemplo

```
node = stream.create("anomalydetection", "My node")
node.setPropertyValue("anomaly_method", "PerRecords")
node.setPropertyValue("percent_records", 95)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("peer_group_num_auto", True)
node.setPropertyValue("min_num_peer_groups", 3)
node.setPropertyValue("max_num_peer_groups", 10)
```

Tabla 111. propiedades de anomalydetectionnode		
Propiedades de anomalydetectionnode	**Valores**	**Descripción de la propiedad**
inputs	*[field1 ... fieldn]*	Los modelos de detección de anomalías criban registros basándose en los campos de entrada especificados. No utilizan un campo objetivo. Los campos de ponderación y frecuencia tampoco se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Especifica el método utilizado para determinar el valor de corte para marcar los registros como anómalos.
index_level	*number*	Especifica el valor de corte mínimo con el que se van a marcar anomalías.
percent_records	*number*	Establece el umbral para marcar registros de acuerdo al porcentaje de registros en los datos de entrenamiento.

Tabla 111. propiedades de anomalydetectionnode (continuación)		
Propiedades de anomalydetectionnode	**Valores**	**Descripción de la propiedad**
num_records	*number*	Establece el umbral para marcar registros de acuerdo al número de registros en los datos de entrenamiento.
num_fields	*integer*	El número de campos de los que se va a informar por cada registro anómalo.
impute_missing_values	*Flag*	
adjustment_coeff	*number*	Valor que se usa para equilibrar la ponderación relativa asignada a los campos categóricos y continuos al calcular la distancia.
peer_group_num_auto	*Flag*	Calcula automáticamente el número de grupos de homólogos.
min_num_peer_groups	*integer*	Especifica el número mínimo de grupos de homólogos empleado cuando peer_group_num_auto se establece en True.
max_num_per_groups	*integer*	Especifica el número máximo de grupos de homólogos.
num_peer_groups	*integer*	Especifica el número de grupos de homólogos empleado cuando peer_group_num_auto se establece en False.
noise_level	*number*	Determina el modo en que se tratan los valores atípicos durante el clúster. Especifique un valor entre 0 y 0,5.
noise_ratio	*number*	Especifica la parte de memoria asignada al componente que debería usarse para el almacenamiento en búfer de ruido. Especifique un valor entre 0 y 0,5.

propiedades de apriorinode

El nodo Apriori extrae un conjunto de reglas de los datos y destaca aquellas reglas con un mayor contenido de información. Apriori ofrece cinco métodos diferentes para la selección de reglas y utiliza un sofisticado esquema de indización para procesar eficientemente grandes conjuntos de datos. En los problemas de mucho volumen, Apriori se entrena más rápidamente, no tiene un límite arbitrario para el número de reglas que puede retener y puede gestionar reglas que tengan hasta 32 precondiciones. Apriori requiere que todos los campos de entrada y salida sean categóricos, pero ofrece un mejor rendimiento ya que está optimizado para este tipo de datos.

Ejemplo

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Tabla 112. propiedades de apriorinode

Propiedades de apriorinode	**Valores**	**Descripción de la propiedad**
consequents	*campo*	Los modelos Apriori utilizan Consecuentes y Antecedentes en lugar de los campos estándar objetivo y de entrada. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
antecedents	[*field1 ... fieldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*Flag*	
optimize	Speed Memory	
use_transactional_data	*Flag*	Cuando el valor es true, la puntuación para cada ID de transacción es independiente de otros ID de transacción. Cuando los datos que se van a puntuar son demasiado grandes para obtener un rendimiento aceptable, se recomienda separar los datos.
contiguous	*Flag*	

Tabla 112. propiedades de apriorinode (continuación)		
Propiedades de apriorinode	**Valores**	**Descripción de la propiedad**
id_field	*serie*	
content_field	*serie*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.

propiedades associationrulesnode

El nodo Reglas de asociación es parecido al nodo Apriori; sin embargo, a diferencia de Apriori, el nodo Reglas de asociación puede procesar datos de lista. Además, el nodo Reglas de asociación se puede utilizar con IBM SPSS Analytic Server para procesar big data y aprovechar el procesamiento paralelo.

Tabla 113. propiedades associationrulesnode		
Propiedades associationrulesnode	**Tipo de datos**	**Descripción de la propiedad**
predictions	*campo*	Los campos en esta lista sólo pueden aparecer como un predictor de una regla
conditions	*[field1...fieldN]*	Los campos en esta lista sólo pueden aparecer como una condición de una regla
max_rule_conditions	*integer*	El número máximo de condiciones que pueden incluirse en una sola regla. Mínimo 1, máximo 9.
max_rule_predictions	*integer*	El número máximo de predicciones que pueden incluirse en una sola regla. Mínimo 1, máximo 5.
max_num_rules	*integer*	El número máximo de reglas que pueden considerarse como parte de la generación de regla. Mínimo 1, máximo 10.000.

Tabla 113. propiedades associationrulesnode (continuación)		
Propiedades associationrulesnode	**Tipo de datos**	**Descripción de la propiedad**
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Deployability	El criterio de reglas que determina el valor por el cual se eligen las reglas "N" superiores en el modelo.
true_flags	*boolean*	Establecer en *Y* determina que sólo se considerarán los valores verdaderos para campos de distintivo durante la generación de la regla.
rule_criterion	*boolean*	Establecer en *Y* determina que los valores del criterio de regla se utilizan para excluir reglas durante la generación de modelos.
min_confidence	*number*	0,1 a 100 - el valor de porcentaje para el nivel de confianza mínimo necesario para una regla producida por el modelo. Si el modelo genera una regla con un nivel de confianza inferior al valor especificado aquí, la regla se descarta.
min_rule_support	*number*	0,1 a 100 - el valor de porcentaje para el soporte de regla mínimo necesario para una regla producida por el modelo. Si el modelo genera una regla con un nivel de soporte de regla inferior al valor especificado, la regla se descarta.
min_condition_support	*number*	0,1 a 100 - el valor de porcentaje para el soporte de condición mínima necesaria para una regla producida por el modelo. Si el modelo genera una regla con un nivel de soporte de condición inferior al valor especificado, la regla se descarta.
min_lift	*integer*	1 a 10 – representa la elevación mínima necesaria para una regla producida por el modelo. Si el modelo genera una regla con un nivel de elevación inferior al valor especificado, la regla se descarta.
exclude_rules	*boolean*	Se utiliza para seleccionar una lista de campos relacionados a partir de los cuales no desea que el modelo cree reglas. Ejemplo: set :gsarsnode.exclude_rules = [[[field1,field2, field3]], [[field4, field5]]]- donde cada lista de campos separados por [] es una fila de la tabla.

Tabla 113. propiedades associationrulesnode (continuación)		
Propiedades associationrulesnode	**Tipo de datos**	**Descripción de la propiedad**
num_bins	*integer*	Establezca el número de intervalos automáticos en los que se agrupan los campos continuos. Mínimo 2, máximo 10.
max_list_length	*integer*	Se aplica a cualquier campo de lista del que no se conoce la longitud máxima. Los elementos de la lista hasta el número especificado aquí se incluyen en la generación de modelos; los elementos adicionales se descartan. Mínimo 1, máximo 100.
output_confidence	*boolean*	
output_rule_support	*boolean*	
output_lift	*boolean*	
output_condition_support	*boolean*	
output_deployability	*boolean*	
rules_to_display	upto all	El número máximo de reglas a visualizar en las tablas de salida.
display_upto	*integer*	Si upto se establece en rules_to_display, establezca el número de reglas a visualizar en las tablas de salida. El mínimo es 1.
field_transformations	*boolean*	
records_summary	*boolean*	
rule_statistics	*boolean*	
most_frequent_values	*boolean*	
most_frequent_fields	*boolean*	
word_cloud	*boolean*	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	*integer*	Mínimo 1, máximo 20.
max_predictions	*integer*	El número máximo de reglas que se pueden aplicar a cada entrada de la puntuación.

Tabla 113. propiedades associationrulesnode (continuación)		
Propiedades associationrulesnode	**Tipo de datos**	**Descripción de la propiedad**
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Seleccione la medida utilizada para determinar la fuerza de las reglas.
allow_repeats	*boolean*	Determine si las reglas con la misma predicción se incluyen en la puntuación.
check_input	NoPredictions Predictions NoCheck	

propiedades de autoclassifiernode

El nodo Clasificador automático crea y compara varios modelos diferentes para obtener resultados binarios (sí o no, abandono o no de clientes, etc.), lo que le permite seleccionar el mejor enfoque para un análisis determinado. Son compatibles varios algoritmos de modelado, por lo que es posible seleccionar los métodos que desee utilizar, las opciones específicas para cada uno y los criterios para comparar los resultados. El nodo genera un conjunto de modelos basados en las opciones especificadas y clasifica los mejores candidatos según los criterios que especifique.

Ejemplo

```
node = stream.create("autoclassifier", "My node")
node.setPropertyValue("ranking_measure", "Accuracy")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_accuracy_limit", True)
node.setPropertyValue("accuracy_limit", 0.9)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("use_costs", True)
node.setPropertyValue("svm", False)
```

Tabla 114. propiedades de autoclassifiernode		
Propiedades de autoclassifiernode	**Valores**	**Descripción de la propiedad**
target	*campo*	En objetivos de marca, el nodo Clasificador binario requiere un único campo objetivo y uno o más campos de entrada. También se puede especificar campos de ponderación y frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*integer*	Número de modelos que se incluirán en el nugget de modelo. Especifique un entero entre 1 y 100.
calculate_variable_importance	*Flag*	
enable_accuracy_limit	*Flag*	
accuracy_limit	*integer*	Entero entre 0 y 100.
enable_ area_under_curve _limit	*Flag*	
area_under_curve_limit	*number*	Número real entre 0,0 y 1,0.
enable_profit_limit	*Flag*	
profit_limit	*number*	Entero mayor que 0.
enable_lift_limit	*Flag*	
lift_limit	*number*	Número real mayor que 1,0.
enable_number_of_variables_lim it	*Flag*	
number_of_variables_limit	*number*	Entero mayor que 0.
use_fixed_cost	*Flag*	
fixed_cost	*number*	Número real mayor que 0.0.
variable_cost	*campo*	

Tabla 114. propiedades de autoclassifiernode (continuación)		
Propiedades de autoclassifiernode	**Valores**	**Descripción de la propiedad**
use_fixed_revenue	*Flag*	
fixed_revenue	*number*	Número real mayor que 0.0.
variable_revenue	*campo*	
use_fixed_weight	*Flag*	
fixed_weight	*number*	Número real mayor que 0,0
variable_weight	*campo*	
lift_percentile	*number*	Entero entre 0 y 100.
enable_model_build_time_limit	*Flag*	
model_build_time_limit	*number*	Entero que indica el número máximo de minutos que se puede tardar en generar cada uno de los modelos.
enable_stop_after_time_limit	*Flag*	
stop_after_time_limit	*number*	Número real que indica el número máximo de horas que puede tardar una ejecución del clasificador automático.
enable_stop_after_valid_model_produced	*Flag*	
use_costs	*Flag*	
<algorithm>	*Flag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*serie*	Define un valor de propiedad para un algoritmo específico. Para obtener más información, consulte el tema “Propiedades de ajustes de algoritmo” en la página 228.

Propiedades de ajustes de algoritmo

En el caso de los nodos Clasificador automático, Autonumérico y Agrupación en clústeres automática, las propiedades de determinados algoritmos utilizados por el nodo se pueden establecer utilizando el formato general:

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

Por ejemplo:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Los nombres de algoritmos del nodo Clasificador automático son `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` y `knn`.

Los nombres de algoritmos del nodo Autonumérico son `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` y `knn`.

Los nombres de algoritmos del nodo Autoclúster son `twostep`, `k-means` y `kohonen`.

Los nombres de las propiedades son los nombres estándar, según se han documentado para cada nodo de algoritmo.

Las propiedades de algoritmos que contienen puntos u otros signos de puntuación deben encerrarse entre comillas simples. Por ejemplo:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

También es posible asignar varios valores a una propiedad. Por ejemplo:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up",
"Down"])
```

Para activar o desactivar el uso de un determinado algoritmo:

```
node.setPropertyValue("chaid", True)
```

Nota: En los casos en los que determinadas opciones de algoritmos no están disponibles en el nodo Clasificador automático o cuando sólo se puede especificar un único valor, en lugar de un intervalo de valores, se aplican los mismos límites que tienen los scripts cuando se accede al nodo de la manera estándar.

propiedades de nodo de agrupación en clústeres automática

El nodo Agrupación en clústeres automática calcula y compara los modelos de agrupación en clústeres que identifican grupos de registros con características similares. El nodo funciona de la misma manera que otros nodos de modelado automático, permitiéndole experimentar con múltiples combinaciones de opciones en una única pasada de modelado. Los modelos se pueden comparar utilizando medidas básicas con las que se intenta filtrar y clasificar la utilidad de los modelos de clúster y proporcionar una medida según la importancia de campos concretos.

Ejemplo

```
node = stream.create("autocluster", "My node")
node.setPropertyValue("ranking_measure", "Silhouette")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_silhouette_limit", True)
node.setPropertyValue("silhouette_limit", 5)
```

Tabla 115. propiedades de nodo de agrupación en clústeres automática

Propiedades de autoclusternode	**Valores**	**Descripción de la propiedad**
`evaluation`	*campo*	**Nota:** Solo nodo Agrupación en clústeres automática. Identifica el campo cuyo valor de importancia se calculará. Como alternativa, se puede utilizar para identificar la calidad con la que el clúster diferencia el valor de este campo y, por lo tanto; la calidad con la que el modelo predecirá este campo.

Tabla 115. propiedades de nodo de agrupación en clústeres automática (continuación)

Propiedades de autoclusternode	**Valores**	**Descripción de la propiedad**
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Training Test	
summary_limit	*integer*	Número de modelos que se incluirán en el informe. Especifique un entero entre 1 y 100.
enable_silhouette_limit	*Flag*	
silhouette_limit	*integer*	Entero entre 0 y 100.
enable_number_less_limit	*Flag*	
number_less_limit	*number*	Número real entre 0,0 y 1,0.
enable_number_greater_limit	*Flag*	
number_greater_limit	*number*	Entero mayor que 0.
enable_smallest_cluster_limit	*Flag*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*integer*	Entero mayor que 0.
enable_largest_cluster_limit	*Flag*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percentage	*number*	

Tabla 115. propiedades de nodo de agrupación en clústeres automática (continuación)		
Propiedades de autoclusternode	**Valores**	**Descripción de la propiedad**
largest_cluster_limit_count	*integer*	
enable_smallest_largest_limit	*Flag*	
smallest_largest_limit	*number*	
enable_importance_limit	*Flag*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	Entero entre 0 y 100.
importance_limit_less_than	*number*	Entero entre 0 y 100.
<algorithm>	*Flag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*serie*	Define un valor de propiedad para un algoritmo específico. Para obtener más información, consulte el tema “Propiedades de ajustes de algoritmo” en la página 228.

propiedades de autonumericnode

El nodo Autonumérico calcula y compara modelos para resultados de rango numérico continuo utilizando cierto número de métodos diferentes. El nodo funciona de la misma manera que el nodo Clasificador automático, lo que le permite seleccionar los algoritmos que desee utilizar y experimentar con varias combinaciones de opciones en una única pasada de modelado. Los algoritmos admitidos incluyen redes neuronales, C&RT, CHAID, regresión lineal, regresión lineal generalizada y máquinas de vectores de soporte (SVM). Los modelos se pueden comparar en función de la correlación, el error relativo o el número de variables utilizadas.

Ejemplo

```
node = stream.create("autonumeric", "My node")
node.setPropertyValue("ranking_measure", "Correlation")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_correlation_limit", True)
node.setPropertyValue("correlation_limit", 0.8)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("neuralnetwork", True)
node.setPropertyValue("chaid", False)
```

Tabla 116. propiedades de autonumericnode		
Propiedades de autonumericnode	**Valores**	**Descripción de la propiedad**
custom_fields	*Flag*	Si es verdadero, se utilizará la configuración de campos personalizada en lugar de la configuración del nodo Tipo.
target	*campo*	El nodo Autonumérico requiere un único campo objetivo y uno o más campos de entrada. También se puede especificar campos de ponderación y frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
inputs	*[field1 ... field2]*	
partition	*campo*	
use_frequency	*Flag*	
frequency_field	*campo*	
use_weight	*Flag*	
weight_field	*campo*	
use_partitioned_data	*Flag*	Si se ha definido un campo de partición, sólo se utilizarán los datos de entrenamiento para la generación del modelo.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*integer*	Número de modelos que se incluirán en el nugget de modelo. Especifique un entero entre 1 y 100.
calculate_variable_importance	*Flag*	
enable_correlation_limit	*Flag*	
correlation_limit	*integer*	
enable_number_of_fields_limit	*Flag*	
number_of_fields_limit	*integer*	
enable_relative_error_limit	*Flag*	
relative_error_limit	*integer*	

Tabla 116. propiedades de autonumericnode (continuación)		
Propiedades de autonumericnode	**Valores**	**Descripción de la propiedad**
enable_model_build_time_limit	*Flag*	
model_build_time_limit	*integer*	
enable_stop_after_time_limit	*Flag*	
stop_after_time_limit	*integer*	
stop_if_valid_model	*Flag*	
<algorithm>	*Flag*	Activa o desactiva el uso de un determinado algoritmo.
<algorithm>.<property>	*serie*	Define un valor de propiedad para un algoritmo específico. Para obtener más información, consulte el tema "Propiedades de ajustes de algoritmo" en la página 228.

Propiedades de bayesnetnode

El nodo Red bayesiana le permite crear un modelo de probabilidad combinando pruebas observadas y registradas con conocimiento del mundo real para establecer la probabilidad de instancias. El nodo se centra en las redes de árbol de bayetas naïve aumentada (TAN) y Markov que se utilizan principalmente para la clasificación.

Ejemplo

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Tabla 117. Propiedades de bayesnetnode		
Propiedades de bayesnetnode	**Valores**	**Descripción de la propiedad**
inputs	*[field1 ... fieldn]*	Los modelos de red bayesiana utilizan un único campo objetivo y uno o más campos de entrada. Los campos continuos se establecen en intervalos automáticamente. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
continue_training_existing_model	*Flag*	

Tabla 117. Propiedades de bayesnetnode (continuación)

Propiedades de bayesnetnode	Valores	Descripción de la propiedad
structure_type	TAN MarkovBlanket	Seleccione la estructura que desea utilizar cuando cree la red bayesiana.
use_feature_selection	*Flag*	
parameter_learning_method	Likelihood Bayes	Especifica el método utilizado para calcular las tablas de probabilidad condicional entre nodos donde se conocen los valores de los elementos padre.
mode	Expert Simple	
missing_values	*Flag*	
all_probabilities	*Flag*	
independence	Likelihood Pearson	Especifica el método utilizado para determinar si las observaciones relacionadas de dos variables son independientes entre sí.
significance_level	*number*	Especifica el valor de corte para determinar la independencia.
maximal_conditioning_set	*number*	Establece el número máximo de variables de condición que se utilizarán para la comprobación de la independencia.
inputs_always_selected	*[field1 ... fieldn]*	Especifica qué campos del conjunto de datos se deben utilizar siempre al generar la red bayesiana. **Nota:** El campo objetivo siempre está seleccionado.
maximum_number_inputs	*number*	Especifica el número máximo de campos de entrada que se deben utilizar al generar la red bayesiana.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

propiedades de buildr

El nodo Generación de R le permite especificar script R personalizado para realizar la creación de modelos y la puntuación de modelos desplegados en IBM SPSS Modeler.

Ejemplo

```
node = stream.create("buildr", "My node")
node.setPropertyValue("score_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-
c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabla 118. propiedades de buildr

Propiedades de buildr	Valores	Descripción de la propiedad
build_syntax	*serie*	Sintaxis de scripts R para la creación del modelo.
score_syntax	*serie*	Sintaxis de scripts R para la puntuación del modelo.
convert_flags	StringsAndDoubles LogicalValues	Opción para convertir campos de distintivos.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
convert_missing	*Flag*	Opción para convertir valores perdidos en valor R NA.
output_html	*Flag*	Opción para visualizar gráficos en una pestaña en el nugget de modelo R.
output_text	*Flag*	Opción para escribir salida de texto de la consola R en una pestaña del modelo R.

propiedades de c50node

El nodo C5.0 genera un árbol de decisión o un conjunto de reglas. El modelo divide la muestra basándose en el campo que ofrece la máxima ganancia de información en cada nivel. El campo objetivo debe ser categórico. Se permiten varias divisiones en más de dos subgrupos.

Ejemplo

```
node = stream.create("c50", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tab
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Tabla 119. propiedades de c50node

Propiedad de c50node	Valores	Descripción de la propiedad
target	*campo*	Los modelos C50 utilizan un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de ponderación. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
output_type	DecisionTree RuleSet	
group_symbolics	*Flag*	
use_boost	*Flag*	
boost_num_trials	*number*	
use_xval	*Flag*	
xval_num_folds	*number*	
mode	Simple Expert	
favor	Accuracy Generality	Generalización o precisión de favor.
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*Flag*	
costs	*structured*	Ésta es una propiedad estructurada.
use_winnowing	*Flag*	
use_global_pruning	*Flag*	Activado (True) de forma predeterminada.

Tabla 119. propiedades de c50node (continuación)		
Propiedad de c50node	**Valores**	**Descripción de la propiedad**
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

propiedades de carmanode

El modelo CARMA extrae un conjunto de reglas de los datos sin necesidad de especificar campos de entrada ni de objetivo. En contraste con Apriori , el nodo CARMA ofrece valores de construcción para el soporte de reglas (soporte para el antecedente y consecuente) en lugar de sólo soporte para antecedentes. Esto significa que las reglas generadas se pueden utilizar en una gama de aplicaciones más amplia, por ejemplo, para buscar una lista de productos o servicios (antecedentes) cuyo consecuente es el elemento que se desea promocionar durante esta temporada de vacaciones.

Ejemplo

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
node.setPropertyValue("vary_support", True)
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

Tabla 120. propiedades de carmanode		
Propiedades de carmanode	**Valores**	**Descripción de la propiedad**
inputs	*[field1 ... fieldn]*	Los modelos CARMA utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
id_field	*campo*	Campo utilizado como el campo de ID para la generación del modelo.
contiguous	*Flag*	Se utiliza para especificar si los ID del campo de ID son contiguos.
use_transactional_data	*Flag*	
content_field	*campo*	
min_supp	*number(percent)*	Está relacionado con el soporte de regla en lugar de con el soporte de antecedentes. El valor por omisión es 20%.
min_conf	*number(percent)*	El valor por omisión es 20%.
max_size	*number*	El valor por omisión es 10.
mode	Simple Expert	El valor predeterminado es Simple.
exclude_multiple	*Flag*	Excluye las reglas con varios consecuentes. El valor predeterminado es False.
use_pruning	*Flag*	El valor predeterminado es False.
pruning_value	*number*	El valor predeterminado es 500.
vary_support	*Flag*	
estimated_transactions	*integer*	
rules_without_antecedents	*Flag*	

propiedades de cartnode

El nodo de árbol de clasificación y regresión (C&R) genera un árbol de decisión que permite predecir o clasificar observaciones futuras. El método utiliza la partición reiterada para dividir los registros de entrenamiento en segmentos minimizando las impurezas en cada paso, donde un nodo se considera "puro" si el 100% de los casos del nodo corresponden a una categoría específica del campo objetivo. Los campos de destino y de entrada pueden ser rangos numéricos o categóricos (nominales, ordinales o distintivos); todas las divisiones son binarias (sólo dos subgrupos).

Ejemplo

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4""")
# "Build Options" tab, "Basics" panel
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# "Model Options" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Tabla 121. propiedades de cartnode

Propiedades de cartnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de árbol C&R requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
continue_training_existing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	

Tabla 121. propiedades de cartnode (continuación)		
Propiedades de cartnode	**Valores**	**Descripción de la propiedad**
tree_directives	*serie*	Especifique directivas para desarrollar el árbol. Las directivas se pueden escribir entre comillas triples para evitar comillas o saltos de línea no deseados. Recuerde que las directivas pueden ser muy sensibles a las pequeñas modificaciones de las opciones de modelado o los datos y es posible que no se puedan generalizar para otros conjuntos de datos.
use_max_depth	Default Custom	
max_depth	*integer*	Profundidad máxima del árbol, de 0 a 1000. Se utiliza sólo si use_max_depth = Custom.
prune_tree	*Flag*	Poda del árbol para evitar sobreajustes.
use_std_err	*Flag*	Use la diferencia máxima en riesgos (en errores estándar).
std_err_multiplier	*number*	Diferencia máxima.
max_surrogates	*number*	Número máximo de sustitutos.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*structured*	Propiedad estructurada.
priors	Data Equal Custom	
custom_priors	*structured*	Propiedad estructurada.
adjust_priors	*Flag*	
trails	*number*	Número de modelos de componente para un aumento o agregación autodocimante.

Tabla 121. propiedades de cartnode (continuación)		
Propiedades de cartnode	**Valores**	**Descripción de la propiedad**
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Mean Median	Regla de combinación predeterminada para objetivos continuos.
large_boost	*Flag*	Aplicar aumento a conjunto de datos muy grandes.
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	
train_pct	*number*	Conjunto de prevención sobreajustado.
set_random_seed	*Flag*	Opción replicar resultados.
seed	*number*	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

propiedades de chaidnode

El nodo CHAID genera árboles de decisión utilizando estadísticos de chi-cuadrado para identificar las divisiones óptimas. A diferencia de los nodos C&RT y Árbol y QUEST, CHAID puede generar árboles no binarios, lo que significa que algunas divisiones generarán más de dos ramas. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos. CHAID exhaustivo es una modificación de CHAID que hace un trabajo más minucioso de examinar todas las divisiones posibles, pero tarda más en computar.

Ejemplo

```
filenode = stream.createAt("variablefile", "My node", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("chaid", "My node", 200, 100)
```

```
stream.link(filenode, node)

node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "CHAID")
node.setPropertyValue("method", "Chaid")
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "Test")
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("merge_alpha", 0.04)
node.setPropertyValue("chi_square", "Pearson")
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("epsilon", 0.003)
node.setPropertyValue("max_iterations", 75)
node.setPropertyValue("split_merged_categories", True)
node.setPropertyValue("bonferroni_adjustment", True)
```

Tabla 122. propiedades de chaidnode

Propiedad de chaidnode	Valores	Descripción de la propiedad
target	*campo*	Los modelos CHAID requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.
continue_training_existing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	
tree_directives	*serie*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	

Tabla 122. propiedades de chaidnode (continuación)		
Propiedad de chaidnode	**Valores**	**Descripción de la propiedad**
max_depth	*integer*	Profundidad máxima del árbol, de 0 a 1000. Se utiliza sólo si use_max_depth = Custom.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*structured*	Propiedad estructurada.
trails	*number*	Número de modelos de componente para un aumento o agregación autodocimante.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Mean Median	Regla de combinación predeterminada para objetivos continuos.
large_boost	*Flag*	Aplicar aumento a conjunto de datos muy grandes.
split_alpha	*number*	Nivel de significancia para división.
merge_alpha	*number*	Nivel de significancia para fusión.
bonferroni_adjustment	*Flag*	Los valores de significancia de ajuste utilizando el método de Bonferroni.
split_merged_categories	*Flag*	Permitir segunda división de categorías fusionadas.
chi_square	Pearson LR	Método usado para calcular la estadística de chi cuadrado: Pearson o Razón de verosimilitud
epsilon	*number*	Cambio mínimo en frecuencias de casillas esperadas.
max_iterations	*number*	Número máximo de iteraciones para la convergencia.
set_random_seed	*integer*	
seed	*number*	

Tabla 122. propiedades de chaidnode (continuación)		
Propiedad de chaidnode	**Valores**	**Descripción de la propiedad**
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*integer*	

propiedades de coxregnode

El nodo Regresión de Cox le permite crear un modelo de supervivencia para datos de tiempo hasta el evento en presencia de registros censurados. El modelo produce una función de supervivencia que predice la probabilidad de que el evento de interés se haya producido en un momento dado (*t*) para los valores dados de las variables de entrada.

Ejemplo

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Tabla 123. propiedades de coxregnode		
Propiedades de coxregnode	**Valores**	**Descripción de la propiedad**
survival_time	*campo*	Los modelos de regresión de Cox requieren un único campo con los tiempos de supervivencia.
target	*campo*	Los modelos de regresión de Cox requieren un único campo objetivo y uno o más campos de entrada. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.
method	Enter Stepwise BackwardsStepwise	
groups	*campo*	

Tabla 123. propiedades de coxregnode (continuación)		
Propiedades de coxregnode	**Valores**	**Descripción de la propiedad**
model_type	MainEffects Custom	
custom_terms	["*BP*Sexo*" "*BP*Edad*"]	
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	

Tabla 123. propiedades de coxregnode (continuación)		
Propiedades de coxregnode	**Valores**	**Descripción de la propiedad**
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*Flag*	
ci_value	90 95 99	
correlation	*Flag*	
display_baseline	*Flag*	
survival	*Flag*	
hazard	*Flag*	
log_minus_log	*Flag*	
one_minus_survival	*Flag*	
separate_line	*campo*	
value	*número* o *cadena*	Si no se especifica ningún valor para un campo, se utilizará la opción predeterminada "Mean" para dicho campo.

Propiedades de decisionlistnode

El nodo Lista de decisiones identifica subgrupos, o segmentos, que muestran una mayor o menor posibilidad de proporcionar un resultado binario relacionado con la población global. Por ejemplo, puede buscar clientes que tengan menos posibilidades de abandonar o más posibilidades de responder favorablemente a una campaña. Puede incorporar su conocimiento empresarial al modelo añadiendo sus propios segmentos personalizados y previsualizando modelos alternativos uno junto a otro para comparar los resultados. Los modelos de listas de decisiones constan de una lista de reglas en las que cada regla tiene una condición y un resultado. Las reglas se aplican en orden, y la primera regla que coincide determina el resultado.

Ejemplo

```
node = stream.create("decisionlist", "My node")
node.setPropertyValue("search_direction", "Down")
```

```
node.setPropertyValue("target_value", 1)
node.setPropertyValue("max_rules", 4)
node.setPropertyValue("min_group_size_pct", 15)
```

Tabla 124. Propiedades de decisionlistnode

Propiedades de decisionlistnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de listas de decisiones utilizan un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	Hace referencia a la localización de segmentos, donde Up es el equivalente a Alta probabilidad y Down es el equivalente a Baja probabilidad.
target_value	*serie*	Si no se especifica, se supondrá el valor true para las marcas.
max_rules	*integer*	Número máximo de segmentos sin incluir el resto.
min_group_size	*integer*	Tamaño mínimo del segmento.
min_group_size_pct	*number*	Tamaño mínimo del segmento como porcentaje.
confidence_level	*number*	Umbral mínimo que un campo de entrada tiene que mejorar la probabilidad de la respuesta (aumentar la elevación) para que merezca la pena añadirlo a la definición de un segmento.
max_segments_per_rule	*integer*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*integer*	Amplitud de búsqueda de las listas.
max_rules_per_cycle	*integer*	Amplitud de búsqueda de las reglas de segmentación.
segment_growth	*number*	

Tabla 124. Propiedades de decisionlistnode (continuación)		
Propiedades de decisionlistnode	**Valores**	**Descripción de la propiedad**
include_missing	*Flag*	
final_results_only	*Flag*	
reuse_fields	*Flag*	Permite la reutilización de los atributos (los campos de entrada que aparecen en las reglas).
max_alternatives	*integer*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

propiedades de discriminantnode

El análisis discriminante hace suposiciones más estrictas que la regresión logística, pero puede ser una alternativa valiosa o complementar un análisis de regresión logística cuando se cumplen esos supuestos.

Ejemplo

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tabla 125. propiedades de discriminantnode		
Propiedades de discriminantnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos discriminantes requieren un único campo objetivo y uno o más campos de entrada. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
method	Enter Stepwise	
mode	Simple Expert	

Tabla 125. propiedades de discriminantnode (continuación)		
Propiedades de discriminantnode	**Valores**	**Descripción de la propiedad**
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*Flag*	Opciones de estadísticos del cuadro de diálogo Salida avanzada.
univariate_anovas	*Flag*	
box_m	*Flag*	
within_group_covariance	*Flag*	
within_groups_correlation	*Flag*	
separate_groups_covariance	*Flag*	
total_covariance	*Flag*	
fishers	*Flag*	
unstandardized	*Flag*	
casewise_results	*Flag*	Opciones de clasificación del cuadro de diálogo Salida avanzada.
limit_to_first	*number*	El valor predeterminado es 10.
summary_table	*Flag*	
leave_one_classification	*Flag*	
combined_groups	*Flag*	
separate_groups_covariance	*Flag*	Opción de matrices **Covarianza de grupos separados**.
territorial_map	*Flag*	
combined_groups	*Flag*	Opción de gráfico **Grupos combinados**.
separate_groups	*Flag*	Opción de gráfico **Grupos separados**.
summary_of_steps	*Flag*	
F_pairwise	*Flag*	

Tabla 125. propiedades de discriminantnode (continuación)		
Propiedades de discriminantnode	**Valores**	**Descripción de la propiedad**
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	
F_value_entry	*number*	El valor predeterminado es 3.84.
F_value_removal	*number*	El valor predeterminado es 2.71.
probability_entry	*number*	El valor predeterminado es 0.05.
probability_removal	*number*	El valor predeterminado es 0,10.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de extensionmodelnode

Con el nodo de Modelo de extensión, puede ejecutar scripts R o Python para Spark para crear y puntuar resultados.

Ejemplo de Python para Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_build", "extension_build")
node.setPropertyValue("syntax_type", "Python")

build_script = """
import json
import spss.pyspark.runtime
```

```
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTree

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]

target = "Drug"
predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

def metaMap(row,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(set([row[col]]))
        else:
            meta.append((row[col],row[col]))
        col += 1
    return meta

def metaReduce(meta1,meta2,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(meta1[col].union(meta2[col]))
        else:
            meta.append((min(meta1[col][0],meta2[col][0]),max(meta1[col][1],meta2[col][1])))
        col += 1
    return meta

metadata = df.rdd.map(lambda row: metaMap(row,schema)).reduce(lambda x,y:metaReduce(x,y,schema))

def setToList(v):
    if isinstance(v,set):
        return list(v)
    return v

metadata = map(lambda x: setToList(x), metadata)
print metadata

lookup = {}
for i in range(0,len(schema)):
    lookup[schema[i][0]] = i

def row2LabeledPoint(dm,lookup,target,predictors,row):
    target_index = lookup[target]
    tval = dm[target_index].index(row[target_index])
    pvals = []
    for predictor in predictors:
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            pval = dm[predictor_index].index(row[predictor_index])
        else:
            pval = row[predictor_index]
        pvals.append(pval)
    return LabeledPoint(tval,DenseVector(pvals))

# count number of target classes
predictorClassCount = len(metadata[lookup[target]])

# define function to extract categorical predictor information from datamodel
def getCategoricalFeatureInfo(dm,lookup,predictors):
    info = {}
    for i in range(0,len(predictors)):
        predictor = predictors[i]
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            info[i] = len(dm[predictor_index])
    return info

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))

treeModel = DecisionTree.trainClassifier(
    lps,
    numClasses=predictorClassCount,
    categoricalFeaturesInfo=getCategoricalFeatureInfo(metadata, lookup, predictors),
    impurity='gini',
    maxDepth=5,
```

```
    maxBins=100)

_outputPath = cxt.createTemporaryFolder()
treeModel.save(cxt.getSparkContext(), _outputPath)
cxt.setModelContentFromPath("TreeModel", _outputPath)
cxt.setModelContentFromString("model.dm",json.dumps(metadata), mimeType="application/json")\
   .setModelContentFromString("model.structure",treeModel.toDebugString())

"""

node.setPropertyValue("python_build_syntax", build_script)
```

Ejemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <-
lm(modelerData$Na~modelerData$K,modelerData)
modelerDataModel
modelerModel
""")
```

Tabla 126. Propiedades de extensionmodelnode

Propiedades de extensionmodelnode	**Valores**	**Descripción de la propiedad**
syntax_type	*E* *Python*	Especifica qué script se ejecuta: R o Python (R es el valor predeterminado).
r_build_syntax	*serie*	Sintaxis de scripts R para la creación de modelos.
r_score_syntax	*serie*	Sintaxis de scripts R para la puntuación de modelos.
python_build_syntax	*serie*	Sintaxis de scripts Python para la creación de modelos.
python_score_syntax	*serie*	Sintaxis de scripts Python para la puntuación de modelos.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opción para convertir campos de distintivos.
convert_missing	*Flag*	Opción para convertir valores perdidos en valor R NA.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
output_html	*Flag*	Opción para visualizar gráficos en una pestaña en el nugget de modelo R.
output_text	*Flag*	Opción para escribir salida de texto de la consola R en una pestaña del modelo R.

propiedades de factornode

El nodo PCA/Factorial proporciona técnicas eficaces de reducción de datos para reducir la complejidad de los datos. Análisis de componentes principales (PCA) busca combinaciones lineales de los campos de entrada que realizan el mejor trabajo a la hora de capturar la varianza en todo el conjunto de campos, en el que los componentes son ortogonales (perpendiculares) entre ellos. Análisis factorial intenta identificar factores subyacentes que expliquen el patrón de correlaciones dentro de un conjunto de campos observados. Para los dos métodos, el objetivo es encontrar un número pequeño de campos derivados que resuma de forma eficaz la información del conjunto original de campos.

Ejemplo

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" section
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Tabla 127. propiedades de factornode

Propiedad de `factornode`	Valores	Descripción de la propiedad
`inputs`	[*field1 ... fieldN*]	Los modelos PCA/Factorial utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.

Tabla 127. propiedades de factornode (continuación)		
Propiedad de factornode	**Valores**	**Descripción de la propiedad**
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*Flag*	
matrix	Correlation Covariance	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	*number*	Si selecciona DirectOblimin como el tipo de datos de rotación, podrá especificar un valor para delta. Si no especifica ningún valor, se utilizará el valor predeterminado para delta.

Tabla 127. propiedades de factornode (continuación)		
Propiedad de factornode	**Valores**	**Descripción de la propiedad**
kappa	*number*	Si selecciona Promax como el tipo de datos de rotación, podrá especificar un valor para kappa. Si no especifica ningún valor, se utilizará el valor predeterminado para kappa.
sort_values	*Flag*	
hide_values	*Flag*	
hide_below	*number*	

propiedades de featureselectionnode

El nodo Selección de características filtra los campos de entrada para su eliminación en función de un conjunto de criterios (como el porcentaje de valores perdidos); a continuación, clasifica el grado de importancia del resto de entradas de acuerdo con un objetivo específico. Por ejemplo, a partir de un conjunto de datos dado con cientos de entradas potenciales, ¿cuáles tienen mayor probabilidad de ser útiles para el modelado de resultados de pacientes?

Ejemplo

```
node = stream.create("featureselection", "My node")
node.setPropertyValue("screen_single_category", True)
node.setPropertyValue("max_single_category", 95)
node.setPropertyValue("screen_missing_values", True)
node.setPropertyValue("max_missing_values", 80)
node.setPropertyValue("criteria", "Likelihood")
node.setPropertyValue("unimportant_below", 0.8)
node.setPropertyValue("important_above", 0.9)
node.setPropertyValue("important_label", "Check Me Out!")
node.setPropertyValue("selection_mode", "TopN")
node.setPropertyValue("top_n", 15)
```

Para obtener un ejemplo más detallado que cree y aplique un modelo de Selección de características, consulte en.

Tabla 128. propiedades de featureselectionnode		
Propiedades de featureselectionnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de selección de características ordenan predictores por rangos con respecto a su objetivo específico. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.

Tabla 128. propiedades de featureselectionnode (continuación)

Propiedades de featureselectionnode	**Valores**	**Descripción de la propiedad**
screen_single_category	*Flag*	En caso de establecer True, filtra campos que tienen demasiados registros dentro de la misma categoría respecto al número total de registros.
max_single_category	*number*	Especifica el umbral que se utiliza cuando screen_single_category es True.
screen_missing_values	*Flag*	Si se establece en True, filtra campos con demasiados valores perdidos, expresados como un porcentaje del número total de registros.
max_missing_values	*number*	
screen_num_categories	*Flag*	Si se establece en True, filtra campos con demasiadas categorías respecto al número total de registros.
max_num_categories	*number*	
screen_std_dev	*Flag*	Si se establece en True, filtra campos con una desviación estándar menor o igual que el mínimo especificado.
min_std_dev	*number*	
screen_coeff_of_var	*Flag*	Si se establece en True, filtra campos con un coeficiente de varianza menor o igual que el mínimo especificado.
min_coeff_of_var	*number*	
criteria	Pearson Likelihood CramersV Lambda	Al clasificar los predictores categóricos en función de un objetivo categórico, especifica la medida en la que se basa el valor de importancia.
unimportant_below	*number*	Especifica los valores *p* de umbral utilizados para clasificar las variables como importantes, marginales o sin importancia. Acepta valores de 0,0 a 1,0.
important_above	*number*	Acepta valores de 0,0 a 1,0.

Tabla 128. propiedades de featureselectionnode (continuación)		
Propiedades de `featureselectionnode`	**Valores**	**Descripción de la propiedad**
`unimportant_label`	*serie*	Especifica la etiqueta para la clasificación como 'Sin importancia'.
`marginal_label`	*serie*	
`important_label`	*serie*	

`selection_mode`	`ImportanceLevel`
	
	`ImportanceValue`
	
	`TopN`	

`select_important`	*Flag*	Si `selection_mode` se establece en `ImportanceLevel`, determina si se seleccionan los campos importantes.
`select_marginal`	*Flag*	Si `selection_mode` se establece en `ImportanceLevel`, determina si se seleccionan los campos marginales.
`select_unimportant`	*Flag*	Si `selection_mode` se establece en `ImportanceLevel`, determina si se seleccionan los campos sin importancia.
`importance_value`	*number*	Si `selection_mode` se establece en `ImportanceValue`, determina el valor de corte que se va a usar. Acepta valores de 0 a 100.
`top_n`	*integer*	Si `selection_mode` se establece en `TopN`, determina el valor de corte que se va a usar. Acepta valores de 0 a 1000.

propiedades de genlinnode

El modelo lineal generalizado amplía el modelo lineal general, de manera que la variable dependiente está relacionada linealmente con los factores y las covariables mediante una determinada función de enlace. Además, el modelo permite que la variable dependiente tenga una distribución no normal. Cubre la funcionalidad de un amplio número de modelos estadísticos, incluyendo regresión lineal, regresión logística, modelos loginear para datos de recuento y modelos de supervivencia censurados por intervalos.

Ejemplo

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabla 129. propiedades de genlinnode		
Propiedades de genlinnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos lineales generalizados requieren un único campo objetivo, que debe ser un campo nominal o marca, y uno o más campos de entrada. También se puede especificar un campo de ponderación. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
use_weight	*Flag*	
weight_field	*campo*	El tipo de campo es únicamente continuo.
target_represents_trials	*Flag*	
trials_type	Variable FixedValue	
trials_field	*campo*	El tipo de campo es continuo, marca u ordinal.
trials_number	*number*	El valor predeterminado es 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*campo*	El tipo de campo es únicamente continuo.
offset_value	*number*	Debe ser un número real.
base_category	Last First	
include_intercept	*Flag*	
mode	Simple Expert	

Tabla 129. propiedades de genlinnode (continuación)

Propiedades de genlinnode	Valores	Descripción de la propiedad
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: De Gauss inversa. NEGBIN: Negative binomial.
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	El valor predeterminado es 1. Debe contener un número real no negativo.
tweedie_parameter	*number*	

Tabla 129. propiedades de genlinnode (continuación)

Propiedades de genlinnode	Valores	Descripción de la propiedad
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: log-log complementario. LOGC: complemento log. NEGBIN: Negative binomial. NLOGLOG: Log-log negativo. CUMCAUCHIT: Cauchit acumulada. CUMCLOGLOG: Log-log complementario acumulado. CUMLOGIT: Logit acumulado. CUMNLOGLOG: Log-log negativo acumulado. CUMPROBIT: Probit acumulado.
power	*number*	El valor debe ser real y distinto de cero.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*number*	El valor predeterminado es 1; sólo se admiten enteros positivos.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	

Tabla 129. propiedades de genlinnode (continuación)		
Propiedades de genlinnode	**Valores**	**Descripción de la propiedad**
scale_value	*number*	El valor predeterminado es 1; debe ser mayor que 0.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*number*	El valor predeterminado es 100; sólo enteros no negativos.
max_step_halving	*number*	El valor predeterminado es 5; sólo enteros positivos.
check_separation	*Flag*	
start_iteration	*number*	El valor predeterminado es 20; sólo se admiten enteros positivos.
estimates_change	*Flag*	
estimates_change_min	*number*	El valor predeterminado es 1E-006; sólo se admiten números positivos.
estimates_change_type	Absolute Relative	
loglikelihood_change	*Flag*	
loglikelihood_change_min	*number*	Sólo se admiten números positivos.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*Flag*	
hessian_convergence_min	*number*	Sólo se admiten números positivos.
hessian_convergence_type	Absolute Relative	
case_summary	*Flag*	
contrast_matrices	*Flag*	
descriptive_statistics	*Flag*	
estimable_functions	*Flag*	
model_info	*Flag*	
iteration_history	*Flag*	
goodness_of_fit	*Flag*	
print_interval	*number*	El valor predeterminado es 1; debe ser un entero positivo.
model_summary	*Flag*	

Tabla 129. propiedades de genlinnode (continuación)

Propiedades de genlinnode	Valores	Descripción de la propiedad
lagrange_multiplier	*Flag*	
parameter_estimates	*Flag*	
include_exponential	*Flag*	
covariance_estimates	*Flag*	
correlation_estimates	*Flag*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*number*	El valor predeterminado es 0.0001.
confidence_interval	*number*	El valor predeterminado es 95.
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	

Tabla 129. propiedades de genlinnode (continuación)		
Propiedades de genlinnode	**Valores**	**Descripción de la propiedad**
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de glmmnode

Un modelo lineal mixto generalizado (GLMM) amplía el modelo lineal de modo que el objetivo pueda tener una distribución no normal, esté linealmente relacionado con los factores y covariables mediante una función de enlace especificada y las observaciones se puedan correlacionar. Los modelos lineales mixtos generalizados cubren una amplia variedad de modelos, desde modelos de regresión lineal simple hasta modelos multinivel complejos para datos longitudinales no normales.

Tabla 130. Propiedades de glmmnode		
Propiedades de glmmnode	**Valores**	**Descripción de la propiedad**
residual_subject_spec	*structured*	La combinación de valores de los campos categóricos especificados que definen de forma exclusiva los sujetos del conjunto de datos.
repeated_measures	*structured*	Campos utilizados para identificar observaciones repetidas.
residual_group_spec	[*field1 ... fieldN*]	Campos que definen conjuntos independientes de parámetros de covarianza de efectos repetidos.
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Especifica la estructura de covarianza de residuos.
custom_target	*Flag*	Indica si se puede utilizar un objetivo definido en el nodo anterior (false) o un objetivo personalizado especificado por target_field (true).

Tabla 130. Propiedades de glmmnode (continuación)		
Propiedades de glmmnode	**Valores**	**Descripción de la propiedad**
target_field	*campo*	Campo a utilizar como objetivo si custom_target es true.
use_trials	*Flag*	Indica si hay que utilizar el campo adicional o el valor que especifica el número de ensayos cuando la respuesta objetivo es un número de eventos que tienen lugar en un conjunto de ensayos. El valor predeterminado es false.
use_field_or_value	Field Value	Indica si se utiliza el campo (valor predeterminado) o valor para especificar el número de ensayos.
trials_field	*campo*	Campo a utilizar para especificar el número de ensayos.
trials_value	*integer*	Valor a utilizar para especificar el número de ensayos. Si se especifica, el valor mínimo es 1.
use_custom_target_reference	*Flag*	Indica si hay que utilizar la categoría de referencia personalizada para un objetivo categórico. El valor predeterminado es false.
target_reference_value	*serie*	Categoría de referencia a utilizar si use_custom_target_reference es true.
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Modelos comunes para la distribución de valores de objetivo. Seleccione Custom para especificar una distribución de la lista proporcionada por target_distribution.

Tabla 130. Propiedades de glmmnode (continuación)

Propiedades de glmmnode	Valores	Descripción de la propiedad
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson	La distribución de valores de objetivo cuando dist_link_combination es Custom.
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Función de enlace para relacionar los valores del objetivo a los predictores. Si target_distribution es Binomial podrá utilizarse cualquiera de las funciones de enlace listadas. Si target_distribution es Multinomial podrán utilizarse CLOGLOG, CAUCHIT, LOGIT, NLOGLOG o PROBIT. Si target_distribution es cualquier cosa distinta de Binomial o Multinomial podrán utilizarse IDENTITY, LOG o POWER.
link_function_param	*number*	Valor del parámetro de función de enlace que hay que utilizar. Sólo es aplicable si normal_link_function o link_function_type es POWER.
use_predefined_inputs	*Flag*	Indica si los campos de efecto fijo son los definidos en sentido ascendente como campos de entrada (true) o los de fixed_effects_list (false). El valor predeterminado es false.
fixed_effects_list	*structured*	Si use_predefined_inputs es false, especifica los campos de entrada para utilizarlos como campos de efecto fijo.
use_intercept	*Flag*	Si es true, el valor predeterminado. incluye la interceptación en el modelo.

Tabla 130. Propiedades de glmmnode (continuación)

Propiedades de glmmnode	Valores	Descripción de la propiedad
random_effects_list	*structured*	Lista de campos para especificar como efectos aleatorios.
regression_weight_field	*campo*	Campo a utilizar como campo de ponderación de análisis.
use_offset	None offset_value offset_field	Indica cómo se especifica la compensación. El valor None significa que no se ha utilizado compensación.
offset_value	*number*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_value.
offset_field	*campo*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_field.
target_category_order	Ascending Descending Data	Orden de clasificación para objetivos categóricos. El valor Data especifica que se utiliza el orden de clasificación de los datos. El valor predeterminado es Ascending.
inputs_category_order	Ascending Descending Data	Orden de clasificación para predictores categóricos. El valor Data especifica que se utiliza el orden de clasificación de los datos. El valor predeterminado es Ascending.
max_iterations	*integer*	Número máximo de iteraciones que ejecutará el algoritmo. Un número entero no negativo; el valor predeterminado es 100.
confidence_level	*integer*	Nivel de confianza utilizado para calcular estimaciones de intervalo de los coeficientes del modelo. Un número entero no negativo; el valor máximo es 100, el valor predeterminado es 95.
degrees_of_freedom_method	Fixed Varied	Especifica cómo se calculan los grados de libertad para la prueba de significación.
test_fixed_effects_coefficients	Model Robust	Método para calcular la matriz de covarianza de las estimaciones de los parámetros.
use_p_converge	*Flag*	Opción para la convergencia de parámetros.
p_converge	*number*	Blanco, o cualquier valor positivo.

Tabla 130. Propiedades de glmmnode (continuación)		
Propiedades de glmmnode	**Valores**	**Descripción de la propiedad**
p_converge_type	Absolute Relative	
use_l_converge	*Flag*	Opción para la convergencia log-likelihood.
l_converge	*number*	Blanco, o cualquier valor positivo.
l_converge_type	Absolute Relative	
use_h_converge	*Flag*	Opción para la convergencia hessiana.
h_converge	*number*	Blanco, o cualquier valor positivo.
h_converge_type	Absolute Relative	
max_fisher_steps	*integer*	
singularity_tolerance	*number*	
use_model_name	*Flag*	Indica si se debe especificar un nombre personalizado para el modelo (true) o utilizar el nombre generado por el sistema (false). El valor predeterminado es false.
model_name	*serie*	Si use_model_name es true, especifica el nombre de modelo que se va a utilizar.
confidence	onProbability onIncrease	Base para calcular el valor de confianza de la puntuación: probabilidad más alta predicha, o la diferencia entre la probabilidad más alta predicha y la segunda probabilidad más alta.
score_category_probabilities	*Flag*	Si es true, genera las probabilidades predichas para objetivos categóricos. El valor predeterminado es false.
max_categories	*integer*	Si score_category_probabilities es true, especifica el número máximo de categorías que se han de guardar.
score_propensity	*Flag*	Si es true, produce puntuaciones de propensión para campos de objetivo de marca que indican la probabilidad del resultado "true" para el campo.
emeans	*estructura*	Para cada campo categórico de la lista de efectos fijos, especifica si hay que producir medias marginales estimadas.

Tabla 130. Propiedades de glmmnode (continuación)		
Propiedades de glmmnode	**Valores**	**Descripción de la propiedad**
covariance_list	*estructura*	Para cada campo continuo de la lista de efectos fijos, especifica si hay que usar la media o un valor personalizado al calcular medias marginales estimadas.
mean_scale	Original Transformed	Especifica si las medias marginales estimadas se calculan basándose en la escala original del objetivo (valor predeterminado) o en la transformación de la función de enlace.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Método de ajuste que hay que utilizar al realizar pruebas de hipótesis con varios contrastes.

Propiedades de gle

Un GLE amplía el modelo lineal de forma que el objetivo puede tener una distribución no normal, está relacionado de forma lineal con los factores y las covariables a través de una función de enlace especificada y las observaciones se pueden correlacionar. Los modelos lineales mixtos generalizados cubren una amplia variedad de modelos, desde modelos de regresión lineal simple hasta modelos multinivel complejos para datos longitudinales no normales.

Tabla 131. Propiedades de gle		
Propiedades de gle	**Valores**	**Descripción de la propiedad**
custom_target	*Flag*	Indica si se puede utilizar un objetivo definido en el nodo anterior (false) o un objetivo personalizado especificado por target_field (true).
target_field	*campo*	Campo a utilizar como objetivo si custom_target es true.
use_trials	*Flag*	Indica si hay que utilizar el campo adicional o el valor que especifica el número de ensayos cuando la respuesta objetivo es un número de eventos que tienen lugar en un conjunto de ensayos. El valor predeterminado es false.
use_trials_field_or_value	Field Value	Indica si se utiliza el campo (valor predeterminado) o valor para especificar el número de ensayos.
trials_field	*campo*	Campo a utilizar para especificar el número de ensayos.

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
trials_value	*integer*	Valor a utilizar para especificar el número de ensayos. Si se especifica, el valor mínimo es 1.
use_custom_target_reference	*Flag*	Indica si hay que utilizar la categoría de referencia personalizada para un objetivo categórico. El valor predeterminado es false.
target_reference_value	*serie*	Categoría de referencia a utilizar si use_custom_target_reference es true.
dist_link_combination	NormalIdentity GammaLog PoissonLog NegbinLog TweedieIdentity NominalLogit BinomialLogit BinomialProbit BinomialLogC CUSTOM	Modelos comunes para la distribución de valores de objetivo. Elija CUSTOM para especificar una distribución en la lista proporcionada por target_distribution.
target_distribution	Normal Binomial Multinomial Gamma INVERSE_GAUSS NEG_BINOMIAL Poisson TWEEDIE UNKNOWN	La distribución de valores de objetivo cuando dist_link_combination es Custom.

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
link_function_type	UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER NEG_BINOMIAL GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT	Función de enlace para relacionar valores de objetivo a los predictores. Si target_distribution es Binomial puede utilizar: UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER Si target_distribution es NEG_BINOMIAL puede utilizar: NEG_BINOMIAL. Si target_distribution es UNKNOWN, puede utilizar: GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
link_function_param	*number*	Valor del parámetro Tweedie que hay que utilizar. Sólo es aplicable si normal_link_function o link_function_type es POWER.
tweedie_param	*number*	Valor del parámetro de función de enlace que hay que utilizar. Solo es aplicable si dist_link_combination está establecido en TweedieIdentity, o link_function_type es TWEEDIE.
use_predefined_inputs	*Flag*	Indica si los campos de efectos del modelo deben ser aquellos definidos anteriormente como campos de entrada (true) o han de ser los campos fixed_effects_list (false).
model_effects_list	*structured*	Si use_predefined_inputs es false, especifica los campos de entrada que se utilizarán como campos de efecto de modelo.
use_intercept	*Flag*	Si es true, el valor predeterminado. incluye la interceptación en el modelo.
regression_weight_field	*campo*	Campo a utilizar como campo de ponderación de análisis.
use_offset	None Value Variable	Indica cómo se especifica la compensación. El valor None significa que no se ha utilizado compensación.
offset_value	*number*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_value.
offset_field	*campo*	El valor que se ha de utilizar para desplazamiento si use_offset se establece en offset_field.
target_category_order	Ascending Descending	Orden de clasificación para objetivos categóricos. El valor predeterminado es Ascending.
inputs_category_order	Ascending Descending	Orden de clasificación para predictores categóricos. El valor predeterminado es Ascending.
max_iterations	*integer*	Número máximo de iteraciones que ejecutará el algoritmo. Un número entero no negativo; el valor predeterminado es 100.
confidence_level	*number*	Nivel de confianza utilizado para calcular estimaciones de intervalo de los coeficientes del modelo. Un número entero no negativo; el valor máximo es 100, el valor predeterminado es 95.

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
test_fixed_effects_coefficients	Model Robust	Método para calcular la matriz de covarianza de las estimaciones de los parámetros.
detect_outliers	*Flag*	Si es verdadero, el algoritmo encuentra valores atípicos de influencia para todas las distribuciones excepto para la distribución multinomial.
conduct_trend_analysis	*Flag*	Si el verdadero, el algoritmo realiza un análisis de tendencias para el diagrama de dispersión.
estimation_method	FISHER_SCORING NEWTON_RAPHSON HYBRID	Especifica el algoritmo de estimación de máxima verosimilitud.
max_fisher_iterations	*integer*	Si se utiliza FISHER_SCORING estimation_method, el número máximo de iteraciones. Mínimo 0, máximo 20.
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	Especifique el método que se va a utilizar para la estimación del parámetro de escala.
scale_value	*number*	Solo está disponible si scale_parameter_method está establecido en Fixed.
negative_binomial_method	MLE FIXED	Especifique el método que se va a utilizar para la estimación del parámetro auxiliar binomial negativo.
negative_binomial_value	*number*	Solo está disponible si negative_binomial_method está establecido en Fixed.
non_neg_least_squares	*Flag*	Si se deben realizar mínimos cuadrados negativos. El valor predeterminado es false.
use_p_converge	*Flag*	Opción para la convergencia de parámetros.
p_converge	*number*	Blanco, o cualquier valor positivo.
p_converge_type	*Flag*	Verdadero = Absoluto, Falso = Relativo
use_l_converge	*Flag*	Opción para la convergencia log-likelihood.
l_converge	*number*	Blanco, o cualquier valor positivo.
l_converge_type	*Flag*	Verdadero = Absoluto, Falso = Relativo
use_h_converge	*Flag*	Opción para la convergencia hessiana.
h_converge	*number*	Blanco, o cualquier valor positivo.

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
h_converge_type	*Flag*	Verdadero = Absoluto, Falso = Relativo
max_iterations	*integer*	Número máximo de iteraciones que ejecutará el algoritmo. Un número entero no negativo; el valor predeterminado es 100.
sing_tolerance	*integer*	
use_model_selection	*Flag*	Habilita los controles de umbral de parámetro y de método de selección de modelos.

method	`LASSO`
	`ELASTIC_NET`
	`FORWARD_STEPWISE`
	`RIDGE`
		Determina el método de selección de modelos, o si está utilizando `Ridge`, el método de regularización utilizado.

detect_two_way_interactions	*Flag*	Si es `True`, el modelo detectará automáticamente interacciones bidireccionales entre campos de entrada. Este control solo debería estar habilitado si el modelo es solo efectos principales (es decir, donde el usuario no ha creado ningún efecto de orden superior) y si el `method` seleccionado es Paso adelante, Lazo, o Red elástica.
automatic_penalty_params	*Flag*	Solo está disponible si la selección del modelo `method` es Lazo o red elástica. Utilice esta función para especificar los parámetros de penalización asociados a los métodos de selección de variables Lasso o Red elástica. Si es `True`, se utilizan los valores predeterminados. Si es `False`, se pueden especificar los parámetros de penalización y los valores personalizados habilitados.
lasso_penalty_param	*number*	Solo está disponible si la selección del modelo `method` es Lazo o Red elástica y `automatic_penalty_params` es `False`. Especifique el valor del parámetro de penalización para Lasso.
elastic_net_penalty_param1	*number*	Solo está disponible si la selección del modelo `method` es Lazo o Red elástica y `automatic_penalty_params` es `False`. Especifique el valor del parámetro de penalización para el parámetro 1 de Red elástica.

Tabla 131. Propiedades de gle (continuación)

Propiedades de gle	Valores	Descripción de la propiedad
elastic_net_penalty_param2	*number*	Solo está disponible si la selección del modelo method es Lazo o Red elástica y automatic_penalty_params es False. Especifique el valor del parámetro de penalización para el parámetro 2 de Red elástica.
probability_entry	*number*	Solo está disponible si el method seleccionado es Paso adelante. Especifique el nivel de significación del criterio estadístico F para la inclusión de efectos.
probability_removal	*number*	Solo está disponible si el method seleccionado es Paso adelante. Especifique el nivel de significación del criterio estadístico F para la eliminación de efectos.
use_max_effects	*Flag*	Solo está disponible si el method seleccionado es Paso adelante. Habilita el control max_effects. Si es False, el número de efectos predeterminado incluido debería ser igual al número total de efectos proporcionados al modelo, menos la interceptación.
max_effects	*integer*	Especifique el número máximo de efectos cuando se utiliza el método de creación de paso adelante.
use_max_steps	*Flag*	Habilita el control max_steps. Si es False, el número predeterminado de pasos debe ser igual a tres veces el número de efectos proporcionado al modelo, excluyendo la interceptación.
max_steps	*integer*	Especifique el número máximo de pasos que se van a realizar cuando se utiliza el method de creación de paso adelante.
use_model_name	*Flag*	Indica si se debe especificar un nombre personalizado para el modelo (true) o utilizar el nombre generado por el sistema (false). El valor predeterminado es false.
model_name	*serie*	Si use_model_name es true, especifica el nombre de modelo que se va a utilizar.
usePI	*Flag*	Si true, se calcula la importancia de predictor.

propiedades de kmeansnode

El nodo K-medias agrupa conjuntos de datos en grupos distintos (o clústeres). El método define un número fijo de clústeres, de forma iterativa asigna registros a los clústeres y ajusta los centros de los clústeres hasta que no se pueda mejorar el modelo. En lugar de intentar predecir un resultado, los modelos de *k*-medias utilizan un proceso conocido como aprendizaje no supervisado para revelar los patrones del conjunto de campos de entrada.

Ejemplo

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K",
"Age"])
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

Tabla 132. propiedades de kmeansnode

Propiedad de kmeansnode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos de K-Medias realizan un análisis de clústeres en un conjunto de campos de entrada pero no utilizan ningún campo objetivo. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.
num_clusters	*number*	
gen_distance	*Flag*	
cluster_label	String Number	
label_prefix	*serie*	
mode	Simple Expert	

Tabla 132. propiedades de kmeansnode (continuación)		
Propiedad de kmeansnode	**Valores**	**Descripción de la propiedad**
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.

Propiedades kmeansasnode

k-medias es uno de los algoritmos de agrupación en clúster utilizados con más frecuencia. Agrupa en clúster puntos de datos en una serie de clústeres predefinidos. El nodo K-Medias-AS en SPSS Modeler se implementa en Spark. Si desea más detalles sobre algoritmos de k-medias, consulte https://spark.apache.org/docs/2.2.0/ml-clustering.html. Tenga en cuenta que el nodo K-Medias-AS realiza una codificación "one-hot" automáticamente para variables categóricas.

Tabla 133. Propiedades kmeansasnode		
Propiedades kmeansasnode	**Valores**	**Descripción de la propiedad**
roleUse	*serie*	Especifique predefined para utilizar roles predefinidos, o custom para utilizar asignaciones de campo personalizado. El valor predeterminado es predefined.
autoModel	*boolean*	Especifique true para utilizar el nombre predeterminado ($S-prediction) para el nuevo campo de puntuación generado, o false para utilizar un nombre personalizado. El valor predeterminado es true.
features	*campo*	Lista de los nombres de campo para la entrada de información cuando la propiedad roleUse está establecida en custom.
name	*serie*	El nombre del nuevo campo de puntuación generado cuando la propiedad autoModel está establecida en false.
clustersNum	*integer*	El número de clústeres para crear. El valor predeterminado es 5.

Tabla 133. Propiedades kmeansasnode (continuación)		
Propiedades kmeansasnode	**Valores**	**Descripción de la propiedad**
initMode	*serie*	El algoritmo de inicialización. Los valores posibles son k-means|| o random. El valor predeterminado es k-means||.
initSteps	*integer*	El número de pasos de inicialización cuando initMode está establecido en k-means||. El valor predeterminado es 2.
advancedSettings	*boolean*	Especifique true para que estén disponibles las cuatro propiedades siguientes. El valor predeterminado es false.
maxIteration	*integer*	Número máximo de iteraciones para la agrupación en clúster. El valor predeterminado es 20.
tolerance	*serie*	La tolerancia para detener las iteraciones. Los valores posibles son 1.0E-1, 1.0E-2, ..., 1.0E-6. El valor predeterminado es 1.0E-4.
setSeed	*boolean*	Especifique true para utilizar una semilla aleatoria personalizada. El valor predeterminado es false.
randomSeed	*integer*	La semilla aleatoria personalizada cuando la propiedad setSeed es true.

propiedades de knnnode

El nodo *k* de modelado de vecino (KNN) asocia el nuevo caso con la categoría o valor de los objetos *k* junto a él en el espacio de predictores, donde *k* es un entero. Los casos parecidos están próximos y los que no lo son están alejados entre sí.

Ejemplo

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

Tabla 134. propiedades de knnnode		
Propiedades de knnnode	**Valores**	**Descripción de la propiedad**
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*Flag*	
use_case_labels	*Flag*	Seleccione esta casilla de verificación para activar la siguiente opción.
case_labels_field	*campo*	
identify_focal_cases	*Flag*	Seleccione esta casilla de verificación para activar la siguiente opción.
focal_cases_field	*campo*	
automatic_k_selection	*Flag*	
fixed_k	*integer*	Habilitado sólo si automatic_k_selectio es False.
minimum_k	*integer*	Habilitado sólo si automatic_k_selectio es True.
maximum_k	*integer*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*Flag*	
range_predictions	Mean Median	
perform_feature_selection	*Flag*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*Flag*	
number_to_select	*integer*	
minimum_change	*number*	
validation_fold_assign_by_field	*Flag*	

Tabla 134. propiedades de knnnode (continuación)		
Propiedades de knnnode	**Valores**	**Descripción de la propiedad**
number_of_folds	*integer*	Sólo se activa si el valor de validation_fold_assign_by_field es False
set_random_seed	*Flag*	
random_seed	*number*	
folds_field	*campo*	Sólo se activa si el valor de validation_fold_assign_by_field es True
all_probabilities	*Flag*	
save_distances	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

propiedades de kohonennode

El nodo Kohonen genera un tipo de red neuronal que se puede usar para agrupar un conjunto de datos en grupos distintos. Cuando la red se termina de entrenar, los registros que son similares se deberían cerrar juntos en el mapa de resultados, mientras que los registros que son diferentes aparecerían aparte. Puede observar el número de observaciones capturadas por cada unidad en el nugget de modelo para identificar unidades fuertes. Esto le proporcionará una idea del número apropiado de clústeres.

Ejemplo

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

Tabla 135. propiedades de kohonennode		
Propiedad de kohonennode	**Valores**	**Descripción de la propiedad**
inputs	[*field1 ... fieldN*]	Los modelos Kohonen utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de frecuencia y ponderación no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
continue	*Flag*	
show_feedback	*Flag*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.
cluster_label	*Flag*	
mode	Simple Expert	
width	*number*	
length	*number*	
decay_style	Linear Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

Propiedades de linearnode

Los modelos de regresión lineal predicen un objetivo continuo basado en relaciones lineales entre el objetivo y uno o más predictores.

Ejemplo

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabla 136. Propiedades de linearnode

Propiedades de `linearnode`	**Valores**	**Descripción de la propiedad**
`target`	*campo*	Especifica un campo de objetivo único.
`inputs`	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
`continue_training_existing_model`	*Flag*	

`objective`	`Standard`
	
	`Bagging`
	
	`Boosting`
	
	`psm`
		`psm` se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.

`use_auto_data_preparation`	*Flag*	
`confidence_level`	*number*	

`model_selection`	`ForwardStepwise`
	
	`BestSubsets`
	
	`None`	

`criteria_forward_stepwise`	`AICC`
	
	`Fstatistics`
	
	`AdjustedRSquare`
	
	`ASE`	

`probability_entry`	*number*	
`probability_removal`	*number*	
`use_max_effects`	*Flag*	
`max_effects`	*number*	
`use_max_steps`	*Flag*	

Tabla 136. Propiedades de linearnode (continuación)		
Propiedades de linearnode	**Valores**	**Descripción de la propiedad**
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
use_random_seed	*Flag*	
random_seed	*number*	
use_custom_model_name	*Flag*	
custom_model_name	*serie*	
use_custom_name	*Flag*	
custom_name	*serie*	
tooltip	*serie*	
keywords	*serie*	
annotation	*serie*	

Propiedades de linearasnode

Los modelos de regresión lineal predicen un objetivo continuo basado en relaciones lineales entre el objetivo y uno o más predictores.

Tabla 137. Propiedades de linearasnode		
Propiedades de linearasnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Especifica un campo de objetivo único.
inputs	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
weight_field	*campo*	Campo de análisis usado por el modelo.
custom_fields	*Flag*	El valor predeterminado es TRUE.
intercept	*Flag*	El valor predeterminado es TRUE.
detect_2way_interaction	*Flag*	Indica si debe tenerse en cuenta la interacción bidireccional. El valor predeterminado es TRUE.

Tabla 137. Propiedades de linearasnode (continuación)

Propiedades de linearasnode	**Valores**	**Descripción de la propiedad**
cin	*number*	Intervalo de confianza usado para calcular las estimaciones de los coeficientes del modelo. Especifique un valor mayor que 0 y menor que 100. El valor predeterminado es 95.
factor_order	ascending descending	Orden de clasificación para predictores categóricos. El valor predeterminado es ascending.
var_select_method	ForwardStepwise BestSubsets none	El método de selección de modelo que se va a usar. El valor predeterminado es ForwardStepwise.
criteria_for_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	La estadística utilizada para determinar si un efecto debe añadirse o eliminarse del modelo. El valor predeterminado es AdjustedRSquare.
pin	*number*	El efecto con el valor p más pequeño con respecto a este umbral de pin especificado se añade al modelo. El valor predeterminado es 0.05.
pout	*number*	Cualquier efecto del modelo con un valor p superior a este umbral de pout especificado se eliminará. El valor predeterminado es 0.10.
use_custom_max_effects	*Flag*	Indica si debe utilizarse el número máximo de efectos en el modelo final. El valor predeterminado es FALSE.
max_effects	*number*	Número máximo de efectos por usar en el modelo final. El valor predeterminado es 1.
use_custom_max_steps	*Flag*	Indica si debe utilizarse el número máximo de pasos. El valor predeterminado es FALSE.
max_steps	*number*	El número máximo de pasos antes de que se detenga el algoritmo escalonado. El valor predeterminado es 1.

Tabla 137. Propiedades de linearasnode (continuación)		
Propiedades de linearasnode	**Valores**	**Descripción de la propiedad**
criteria_for_best_subsets	AICC AdjustedRSquare ASE	La modalidad de criterios a utilizar. El valor predeterminado es AdjustedRSquare.

Propiedades de logregnode

La regresión logística es una técnica de estadístico para clasificar los registros en función los valores de los campos de entrada. Es análoga a la regresión lineal, pero utiliza un campo de objetivo categórico en lugar de un rango numérico.

Ejemplo multinomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" options
node.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
```

```
node.setPropertyValue("max_terms", 10)
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

Ejemplo binomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Tabla 138. Propiedades de logregnode

Propiedades de logregnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de regresión logística requieren un único campo objetivo y uno o más campos de entrada. Los campos de frecuencia y ponderación no se usan. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.
logistic_procedure	Binomial Multinomial	
include_constant	*Flag*	

Tabla 138. Propiedades de logregnode (continuación)

Propiedades de logregnode	Valores	Descripción de la propiedad
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	
model_type	MainEffects FullFactorial Custom	Si FullFactorial se especifica como el tipo de modelo, no se ejecutarán los métodos por pasos, aunque así se indique. En su lugar, el método utilizado será Enter. Si el tipo de modelo se establece en Custom pero no se ha especificado ningún campo personalizado, se generará un modelo de efectos principales.
custom_terms	[*[BP Sexo][BP][Edad]*]	
multinomial_base_category	*serie*	Especifica cómo se determina la categoría de referencia.
binomial_categorical_input	*serie*	

Tabla 138. Propiedades de logregnode (continuación)

Propiedades de logregnode	Valores	Descripción de la propiedad
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Propiedad con clave para la entrada categórica que especifica cómo se determina el contraste.
binomial_input_category	First Last	Propiedad con clave para la entrada categórica que especifica cómo se determina la categoría de referencia.
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*Flag*	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	*Flag*	
max_terms	*number*	
entry_criterion	Score LR	

Tabla 138. Propiedades de logregnode (continuación)

Propiedades de logregnode	Valores	Descripción de la propiedad
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*number*	
max_steps	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	*Flag*	
history_steps	*number*	

Tabla 138. Propiedades de logregnode (continuación)		
Propiedades de logregnode	**Valores**	**Descripción de la propiedad**
summary	*Flag*	
likelihood_ratio	*Flag*	
asymptotic_correlation	*Flag*	
goodness_fit	*Flag*	
parameters	*Flag*	
confidence_interval	*number*	
asymptotic_covariance	*Flag*	
classification_table	*Flag*	
stepwise_summary	*Flag*	
info_criteria	*Flag*	
monotonicity_measures	*Flag*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*Flag*	
binomial_parameters	*Flag*	
binomial_iteration_history	*Flag*	
binomial_classification_plots	*Flag*	
binomial_ci_enable	*Flag*	
binomial_ci	*number*	
binomial_residual	outliers all	
binomial_residual_enable	*Flag*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*Flag*	

Tabla 138. Propiedades de logregnode (continuación)		
Propiedades de logregnode	**Valores**	**Descripción de la propiedad**
calculate_raw_propensities	*Flag*	

propiedades de lsvmnode

El nodo Máquina de vectores de soporte lineal (LSVM) le permite clasificar datos en uno o dos grupos sin que haya un ajuste por exceso. LSVM es lineal y funciona bien con conjuntos de datos grandes, como aquellos con un gran número de registros.

Tabla 139. propiedades de lsvmnode		
Propiedades lsvmnode	**Valores**	**Descripción de la propiedad**
intercept	*Flag*	Incluye la interceptación del modelo. El valor predeterminado es True.
target_order	Ascending Descending	Especifica el orden de clasificación del objetivo categórico. Si ignora para objetivos continuos. El valor predeterminado es Ascending.
precision	*number*	Se utiliza sólo si el nivel de medición del campo de destino es Continuous. Especifica el parámetro relacionado con la sensibilidad de la pérdida de regresión. El mínimo es 0 y no hay máximo. El valor predeterminado es 0.1.
exclude_missing_values	*Flag*	Si es True, se excluye un registro si falta algún valor único. El valor predeterminado es False.
penalty_function	L1 L2	Especifica el tipo de función de penalización utilizado. El valor predeterminado es L2.
lambda	*number*	Parámetro de penalización (regularización).

Tabla 139. propiedades de lsvmnode (continuación)		
Propiedades `lsvmnode`	**Valores**	**Descripción de la propiedad**
`calculate_variable_importance`	*Flag*	En modelos que producen una medida adecuada de importancia, esta opción muestra un gráfico que indica la importancia relativa de cada predictor al estimar el modelo.En modelos que producen una medida adecuada de importancia, esta opción muestra un gráfico que indica la importancia relativa de cada predictor al estimar el modelo. Tenga en cuenta que la importancia variable puede ser tardar más en calcularse en algunos modelos, especialmente al trabajar con grandes conjuntos de datos y está desactivada de forma predeterminada para algunos modelos como resultado. La importancia variable no está disponible en los modelos de lista de decisiones.

propiedades de neuralnetnode

Importante: En este release está disponible una versión más reciente del nodo de modelado Red neuronal, con características mejoradas, y se describe en la siguiente sección (*red neuronal*). Aunque aún puede generar y puntuar un modelo con la versión anterior, recomendamos que actualice sus scripts para que se use la nueva versión. Los detalles de la versión anterior se conservan aquí como referencia.

Ejemplo

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Tabla 140. propiedades de neuralnetnode		
Propiedad de neuralnetnode	**Valores**	**Descripción de la propiedad**
targets	[*field1 ... fieldN*]	El nodo Red neuronal espera uno o varios campos objetivo y uno o más campos de entrada. Los campos de frecuencia y ponderación se omiten. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*Flag*	
train_pct	*number*	
set_random_seed	*Flag*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Time	Modo de parada.
accuracy	*number*	Precisión de parada.
cycles	*number*	Ciclos para entrenar.
time	*number*	Tiempo para entrenar (minutos).
continue	*Flag*	
show_feedback	*Flag*	
binary_encode	*Flag*	
use_last_model	*Flag*	
gen_logfile	*Flag*	

Tabla 140. propiedades de neuralnetnode (continuación)

Propiedad de neuralnetnode	Valores	Descripción de la propiedad
logfile_name	*serie*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*serie*	
m_non_pyramids	*Flag*	
m_persistence	*number*	
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*Flag*	
r_alpha	*number*	

Tabla 140. propiedades de neuralnetnode (continuación)		
Propiedad de neuralnetnode	**Valores**	**Descripción de la propiedad**
r_eta	*number*	
optimize	Speed Memory	Se utiliza para especificar si la generación del modelo se debe optimizar para la velocidad o la memoria.
calculate_variable_importance	*Flag*	Nota: La propiedad sensitivity_analysis utilizada en versiones anteriores se ha desaprobado en favor de esta propiedad. La propiedad anterior se sigue admitiendo, pero se recomienda calculate_variable_importance.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de neuralnetwork

El nodo Red neuronal utiliza un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades simples de procesamiento interconectadas que parecen versiones abstractas de neuronas. Las redes neuronales son dispositivos eficaces de cálculo de funciones generales y requieren un conocimiento matemático o estadístico mínimo para entrenarlas o aplicarlas.

Ejemplo

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabla 141. Propiedades de neuralnetwork		
Propiedades de neuralnetworknode	**Valores**	**Descripción de la propiedad**
targets	[*field1 ... fieldN*]	Especifica campos objetivo.
inputs	[*field1 ... fieldN*]	Campos de predictor utilizados por el modelo.
splits	[*field1 ... fieldN*	Especifica el campo o campos para utilizar en el modelado de divisiones.

Tabla 141. Propiedades de neuralnetwork (continuación)		
Propiedades de neuralnetworknode	**Valores**	**Descripción de la propiedad**
use_partition	*Flag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
continue	*Flag*	Continuar entrenando modelo existente.
objective	Standard Bagging Boosting psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*Flag*	
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*Flag*	
max_time	*number*	
use_max_cycles	*Flag*	
max_cycles	*number*	
use_min_accuracy	*Flag*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*Flag*	
random_seed	*number*	

Tabla 141. Propiedades de neuralnetwork (continuación)		
Propiedades de neuralnetworknode	**Valores**	**Descripción de la propiedad**
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*Boolean*	
model_name	*serie*	
confidence	onProbability onIncrease	
score_category_probabilities	*Flag*	
max_categories	*number*	
score_propensity	*Flag*	
use_custom_name	*Flag*	
custom_name	*serie*	
tooltip	*serie*	
keywords	*serie*	
annotation	*serie*	

propiedades de questnode

El nodo QUEST proporciona un método de clasificación binario para generar árboles de decisión; está diseñado para reducir el tiempo de procesamiento necesario para realizar los análisis de C&RT y reducir la tendencia de los métodos de clasificación de árboles para favorecer a las entradas que permitan realizar más divisiones. Los campos de entrada pueden ser continuos (rango numérico), sin embargo el campo objetivo debe ser categórico. Todas las divisiones son binarias.

Ejemplo

```
node = stream.create("quest", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("max_surrogates", 5)
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("prune_tree", True)
node.setPropertyValue("use_std_err", True)
node.setPropertyValue("std_err_multiplier", 3)
```

Tabla 142. propiedades de questnode		
Propiedad de questnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos QUEST requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
continue_training_existing_model	*Flag*	
objective	Standard Boosting Bagging psm	psm se utiliza para conjuntos de datos de grandes dimensiones y requiere una conexión al Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*Flag*	
tree_directives	*serie*	
use_max_depth	Default Custom	
max_depth	*integer*	Profundidad máxima del árbol, de 0 a 1000. Se utiliza sólo si use_max_depth = Custom.
prune_tree	*Flag*	Poda del árbol para evitar sobreajustes.
use_std_err	*Flag*	Use la diferencia máxima en riesgos (en errores estándar).
std_err_multiplier	*number*	Diferencia máxima.
max_surrogates	*number*	Número máximo de sustitutos.
use_percentage	*Flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*Flag*	
costs	*structured*	Propiedad estructurada.

Tabla 142. propiedades de questnode (continuación)		
Propiedad de questnode	**Valores**	**Descripción de la propiedad**
priors	Data Equal Custom	
custom_priors	*structured*	Propiedad estructurada.
adjust_priors	*Flag*	
trails	*number*	Número de modelos de componente para un aumento o agregación autodocimante.
set_ensemble_method	Voting HighestProbability HighestMeanProbabilit y	Regla de combinación predeterminada para objetivos categóricos.
range_ensemble_method	Mean Median	Regla de combinación predeterminada para objetivos continuos.
large_boost	*Flag*	Aplicar aumento a conjunto de datos muy grandes.
split_alpha	*number*	Nivel de significancia para división.
train_pct	*number*	Conjunto de prevención sobreajustado.
set_random_seed	*Flag*	Opción replicar resultados.
seed	*number*	
calculate_variable_impor tance	*Flag*	
calculate_raw_propensiti es	*Flag*	
calculate_adjusted_prope nsities	*Flag*	
adjusted_propensity_part ition	Test Validation	

propiedades randomtrees

El nodo Árboles aleatorios es similar al nodo C & RT existente; sin embargo, el nodo Árboles aleatorios está diseñado para procesar datos grandes para crear un solo árbol y muestra el modelo resultante en el visor de salida que se ha añadido en SPSS Modeler versión 17. El nodo del árbol Árboles aleatorios genera un árbol de decisiones que se utiliza para predecir o clasificar observaciones futuras. El método utiliza la partición reiterada para dividir los registros de entrenamiento en segmentos minimizando las impurezas en cada paso, donde un nodo se considera *puro* si el 100% de los casos del nodo corresponden a una categoría específica del campo objetivo. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos (nominal, ordinal o marca). Todas las divisiones son binarias (sólo se crean dos subgrupos).

Tabla 143. propiedades randomtrees

Propiedades randomtrees	**Valores**	**Descripción de la propiedad**
target	*campo*	En el nodo Árboles aleatorios, los modelos requieren un único objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
number_of_models	*integer*	Determina el número de modelos a construir como parte del modelado de conjunto.
use_number_of_predictors	*Flag*	Determina si se utiliza number_of_predictors.
number_of_predictors	*integer*	Especifica el número de predictores que se van a utilizar al crear modelos de división.
use_stop_rule_for_accuracy	*Flag*	Determina si la creación del modelo se detiene cuando no se puede mejorar la precisión.
sample_size	*number*	Reduzca este valor para mejorar el rendimiento del procesar conjuntos de datos muy grandes.
handle_imbalanced_data	*Flag*	Si el objetivo del modelo es un resultado de marca determinado, y la proporción del resultado deseado con respecto a un resultado no deseado es muy pequeña, los datos se desequilibran y el muestreo de simulación realizado por el modelo puede afectar a la exactitud del mismo. Habilite el manejo de datos desequilibrados para que el modelo capture una proporción mayor del resultado deseado y genere un modelo más sólido.

Tabla 143. propiedades randomtrees (continuación)		
Propiedades randomtrees	**Valores**	**Descripción de la propiedad**
use_weighted_sampling	*Flag*	Si se establece en *False*, las variables para cada nodo se seleccionan aleatoriamente con la misma probabilidad. Si es *True*, las variables se ponderan y se seleccionan de acuerdo con ello.
max_node_number	*integer*	Número máximo de nodos permitidos en árboles individuales. Si el número va a superarse en la división siguiente, el crecimiento del árbol se detiene.
max_depth	*integer*	Máxima profundidad del árbol antes de que se detenga el crecimiento.
min_child_node_size	*integer*	Determina el número mínimo de registros permitidos en un nodo hijo después de que se divida el nodo padre. Si un nodo hijo va a contener menos registros de los especificados aquí, el nodo padre no se dividirá.
use_costs	*Flag*	
costs	*structured*	Propiedad estructurada. El formato es una lista de 3 valores: el valor real, el valor predicho y el coste si dicha predicción es errónea. Por ejemplo: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square custom	**Nota:** sólo se habilita para objetivos ordinales. Establezca valores predeterminados en la matriz de costes.
max_pct_missing	*integer*	Si el porcentaje de valores faltantes en cualquier entrada es mayor que el valor especificado aquí, la entrada se excluye. Mínimo 0, máximo 100.
exclude_single_cat_pct	*integer*	Si un valor de categoría representa un porcentaje mayor de los registros que el especificado aquí, todo el campo se excluye de la creación de modelos. Mínimo 1, máximo 99.
max_category_number	*integer*	Si el número de categorías de un campo supera este valor, el campo se excluye de la creación de modelos. El mínimo es 2.

Tabla 143. propiedades randomtrees (continuación)		
Propiedades randomtrees	**Valores**	**Descripción de la propiedad**
min_field_variation	*number*	Si el coeficiente de variación de un campo continuo es inferior a este valor, el campo se excluye de la creación de modelos.
num_bins	*integer*	Sólo se utiliza si los datos están formados por entradas continuas. Establezca el número de intervalos de frecuencia que deben utilizarse para las entradas; las opciones son 2, 4, 5, 10, 20, 25, 50 o 100.
topN	*integer*	Especifica el número de reglas para informar. El valor predeterminado es 50, con un mínimo de 1 y un máximo de 1000.

Propiedades de regressionnode

La regresión lineal es una técnica estadística común para resumir datos y hacer predicciones ajustando una línea recta o superficie que minimice las discrepancias entre los valores de salida previstos y reales.

Nota: El nodo Lineal reemplazará al nodo Regresión en una versión futura. Recomendamos que a partir de ahora utilice modelos lineales para la regresión lineal.

Ejemplo

```
node = stream.create("regression", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
node.setPropertyValue("weight_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
```

```
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

Tabla 144. Propiedades de regressionnode

Propiedad de regressionnode	**Valores**	**Descripción de la propiedad**
target	*campo*	Los modelos de regresión requieren un único campo objetivo y uno o más campos de entrada. También se puede especificar un campo de ponderación. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
method	Enter Stepwise Backwards Forwards	
include_constant	*Flag*	
use_weight	*Flag*	
weight_field	*campo*	
mode	Simple Expert	
complete_records	*Flag*	

Tabla 144. Propiedades de regressionnode (continuación)		
Propiedad de regressionnode	**Valores**	**Descripción de la propiedad**
tolerance	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10 1.0E-11 1.0E-12	Utilice comillas dobles para los argumentos.
stepping_method	useP useF	useP : utilizar la probabilidad de F useF: utilizar el valor F
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*Flag*	
confidence_interval	*Flag*	
covariance_matrix	*Flag*	
collinearity_diagnostics	*Flag*	
regression_coefficients	*Flag*	
exclude_fields	*Flag*	
durbin_watson	*Flag*	
model_fit	*Flag*	
r_squared_change	*Flag*	
p_correlations	*Flag*	

Tabla 144. Propiedades de regressionnode (continuación)		
Propiedad de regressionnode	**Valores**	**Descripción de la propiedad**
descriptives	*Flag*	
calculate_variable_importance	*Flag*	

propiedades de sequencenode

El nodo Secuencia encuentra reglas de asociación en datos secuenciales o en datos ordenados en el tiempo. Una secuencia es una lista de conjuntos de elementos que tiende a producirse en un orden previsible. Por ejemplo, si un cliente compra una cuchilla y una loción para después del afeitado, probablemente comprará crema para afeitar la próxima vez que vaya a comprar. El nodo Secuencia se basa en el algoritmo de reglas de asociación de CARMA, que utiliza un método de dos pasos para encontrar las secuencias.

Ejemplo

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

Tabla 145. propiedades de sequencenode		
Propiedad de sequencenode	**Valores**	**Descripción de la propiedad**
id_field	*campo*	Para crear un modelo de secuencias, es necesario especificar un campo de ID, un campo de tiempo opcional y uno o varios campos de contenido. Los campos de ponderación y frecuencia no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
time_field	*campo*	
use_time_field	*Flag*	
content_fields	[*field1 ... fieldn*]	
contiguous	*Flag*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*Flag*	
max_duration	*number*	
use_gaps	*Flag*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*Flag*	
pruning_value	*number*	
set_mem_sequences	*Flag*	
mem_sequences	*integer*	

propiedades de slrmnode

El nodo Modelo de respuesta de autoaprendizaje (SLRM) permite crear un modelo en el que un solo caso nuevo o un pequeño número de casos nuevos se pueden utilizar para volver a calcular el modelo sin tener que entrenar de nuevo el modelo utilizando todos los datos.

Ejemplo

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
```

```
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tabla 146. propiedades de slrmnode

Propiedades de slrmnode	Valores	Descripción de la propiedad
target	*campo*	El campo objetivo debe ser un campo nominal o marca. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
target_response	*campo*	El tipo debe ser marca.
continue_training_existing_model	*Flag*	
target_field_values	*Flag*	Utilizar todos: Usar todos los valores del origen. Especifique: Son necesarios determinados valores.
target_field_values_specify	*[field1 ... fieldn]*	
include_model_assessment	*Flag*	
model_assessment_random_seed	*number*	Debe ser un número real.
model_assessment_sample_size	*number*	Debe ser un número real.
model_assessment_iterations	*number*	Número de iteraciones.
display_model_evaluation	*Flag*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Ascending Descending	Especifica si se mostrarán primero las ofertas con las puntuaciones más altas o más bajas.
model_reliability	*Flag*	
calculate_variable_importance	*Flag*	

propiedades de statisticsmodelnode

El nodo Modelo Statistics permite analizar y trabajar con sus datos ejecutando los procedimientos de IBM SPSS Statistics que producen PMML. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Las propiedades de este nodo están descritas en “propiedades de statisticsmodelnode” en la página 436.

propiedades de stpnode

El nodo Predicción espacio-temporal (STP) utiliza datos que contienen datos de ubicación, campos de entrada para la predicción (predictores), un campo de hora y un campo de objetivo. Cada ubicación tiene muchas filas de datos que representan los valores de cada predictor para cada tiempo de medición. Después de analizar los datos, se pueden utilizar para predecir valores en cualquier ubicación dentro de los datos de forma utilizados en el análisis.

Tabla 147. propiedades de stpnode

Propiedades de stpnode	**Tipo de datos**	**Descripción de la propiedad**
Pestaña **Campos**		
`target`	*campo*	Este es el campo de destino.
`location`	*campo*	Campo de ubicación del modelo. Sólo se permiten campos geoespaciales.
`location_label`	*campo*	Campo categórico a utilizar en la salida para etiquetar las ubicaciones elegidas en `location`
`time_field`	*campo*	Campo de hora del modelo. Sólo se permiten campos con medición continua, y el tipo de almacenamiento debe ser hora, fecha, indicación de fecha y hora o entero.
`inputs`	*[field1 ... fieldn]*	Lista de campos de entrada.
Pestaña **Intervalos de tiempo**		

`interval_type_timestamp`	`Years`
	`Quarters`
	`Months`
	`Weeks`
	`Days`
	`Hours`
	`Minutes`
	`Seconds`	

Tabla 147. propiedades de stpnode (continuación)		
Propiedades de stpnode	**Tipo de datos**	**Descripción de la propiedad**
interval_type_date	Years Quarters Months Weeks Days	
interval_type_time	Hours Minutes Seconds	Limita el número de días por semana que se tienen en cuenta al crear el índice de hora que utiliza STP para el cálculo
interval_type_integer	Periods (Sólo campos de índice de hora, almacenamiento Entero)	Intervalo en el que se convertirá el conjunto de datos. La selección disponible depende del tipo de almacenamiento del campo elegido como time_field para el modelo.
period_start	*integer*	
start_month	January February March April May June July August September October November December	El mes desde el que el modelo empezará a indexar (por ejemplo, si se establece en March pero el primer registro del conjunto de datos es January, el modelo omitirá los primeros dos registros y comenzará a indexar en marzo.

Tabla 147. propiedades de stpnode (continuación)

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	Punto de partida para el índice temporal creado por STP a partir de los datos
days_per_week	*integer*	Mínimo 1, máximo 7, con incrementos de 1
hours_per_day	*integer*	El número de horas que el modelo cuenta en un día. Si se establece en 10, el modelo empezará a indexar en la hora day_begins_at y continuará indexando durante 10 horas, y luego saltará al siguiente valor que coincida con el valor day_begins_at, etc.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Establece el valor de hora desde el que el modelo inicia la indexación.

Tabla 147. propiedades de stpnode (continuación)

Propiedades de stpnode	Tipo de datos	Descripción de la propiedad
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Este valor de incremento es de minutos o segundos. Determina dónde el modelo crea índices a partir de los datos. Así que con un incremento de 30 y el intervalo de tipo seconds, el modelo crea un índice a partir de los datos cada 30 segundos.
data_matches_interval	*boolean*	Si se establece en N, la conversión de los datos al interval_type normal se produce antes de que se construya el modelo. Si los datos ya tiene el formato correcto, e interval_type y sus valores asociados coinciden con sus datos, establézcalo en Y para evitar la conversión o la agregación de los datos. Si lo establece en Y, se inhabilitan todos los controles de agregación.

Tabla 147. propiedades de stpnode (continuación)		
Propiedades de stpnode	**Tipo de datos**	**Descripción de la propiedad**
agg_range_default	Sum Mean Min Max Median 1stQuartile 3rdQuartile	Determina el método de agregación predeterminado que se utiliza para los campos continuos. Los campos continuos que no estén incluidos específicamente en la agregación predeterminada se agregarán usando el método aquí indicado.
custom_agg	[[field, aggregation method],[]..] Demo: [['x5' 'FirstQuartile']['x4' 'Sum']]	Propiedad estructurada: Parámetro de script: custom_agg Por ejemplo: set :stpnode.custom_agg = [[field1 function] [field2 function]] Donde function es la función de agregación que se utilizará con ese campo.
Pestaña **Procedimientos básicos**		
include_intercept	*Flag*	
max_autoregressive_lag	*integer*	Mínimo 1, máximo 5, en incrementos de 1. Es el número de registros anteriores necesarios para una predicción. Por lo tanto, si se establece en 5, por ejemplo, entonces los 5 registros se utilizan para crear un nuevo pronóstico. El número de registros especificado aquí a partir de los datos de construcción se incorporan en el modelo y, por lo tanto, el usuario no necesita proporcionar los datos de nuevo al puntuar el modelo.

Tabla 147. propiedades de stpnode (continuación)		
Propiedades de stpnode	**Tipo de datos**	**Descripción de la propiedad**
estimation_method	Parametric Nonparametric	Método para modelar la matriz de covarianzas espacial
parametric_model	Gaussian Exponential PoweredExponential	Parámetro de orden para el modelo de covarianza espacial de tipo Parametric
exponential_power	*number*	Nivel alimentación para el modelo PoweredExponential. Mínimo 1, máximo 2.
Pestaña **Avanzada**		
max_missing_values	*integer*	Porcentaje máximo de registros con valores faltantes que se permite en el modelo.
significance	*number*	Nivel de significación para pruebas de hipótesis en la construcción del modelo. Especifica el valor de significación para todas las pruebas en la estimación del modelo STP, incluidas dos pruebas de Bondad de ajuste, pruebas F de efectos y pruebas T de coeficiente.
Pestaña **Salida**		
model_specifications	*Flag*	
temporal_summary	*Flag*	
location_summary	*Flag*	Determina si la tabla Resumen de ubicación se incluye en la salida del modelo.
model_quality	*Flag*	
test_mean_structure	*Flag*	
mean_structure_coefficients	*Flag*	
autoregressive_coefficients	*Flag*	
test_decay_space	*Flag*	
parametric_spatial_covariance	*Flag*	
correlations_heat_map	*Flag*	
correlations_map	*Flag*	
location_clusters	*Flag*	

Tabla 147. propiedades de stpnode (continuación)		
Propiedades de stpnode	**Tipo de datos**	**Descripción de la propiedad**
similarity_threshold	*number*	Umbral en el cual los clústeres de salida se consideran lo suficientemente parecidos para que se fusionen en un único clúster.
max_number_clusters	*integer*	Límite superior para el número de clústeres que se pueden incluir en la salida del modelo.
Pestaña **Opciones de modelo**		
use_model_name	*Flag*	
model_name	*serie*	
uncertainty_factor	*number*	Mínimo 0, máximo 100. Determina el aumento de la incertidumbre (error) aplicado a las predicciones en el futuro. Es el límite superior e inferior para las predicciones.

propiedades de svmnode

El nodo Máquina de vectores de soporte (SVM) le permite clasificar datos en uno o dos grupos sin que haya un ajuste por exceso. SVM funciona bien con conjuntos de datos grandes, como aquellos con un gran número de campos de entrada.

Ejemplo

```
node = stream.create("svm", "My node")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Tabla 148. propiedades de svmnode		
Propiedades de svmnode	**Valores**	**Descripción de la propiedad**
all_probabilities	*Flag*	

Tabla 148. propiedades de svmnode (continuación)

Propiedades de svmnode	Valores	Descripción de la propiedad
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (valor predeterminado) 1.0E-4 1.0E-5 1.0E-6	Determina cuándo detener el algoritmo de optimización.
regularization	*number*	También se conoce como el parámetro C.
precision	*number*	Se utiliza sólo si el nivel de medición del campo de destino es Continuous.
kernel	RBF (valor predeterminado) Polynomial Sigmoid Linear	Tipo de función kernel utilizada para la transformación.
rbf_gamma	*number*	Sólo se utiliza si kernel es RBF.
gamma	*number*	Se utiliza sólo si kernel es Polynomial o Sigmoid.
bias	*number*	
degree	*number*	Sólo se utiliza si kernel es Polynomial.
calculate_variable_importance	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	
adjusted_propensity_partition	Test Validation	

Propiedades de tcmnode

El modelado causal temporal intenta descubrir relaciones causales clave en datos de series temporales. En el modelado causal temporal, especifique un conjunto de series de objetivos y un conjunto de entradas candidato para estos objetivos. El procedimiento crea un modelo de serie temporal autorregresivo para cada objetivo e incluye solo estas entradas que tienen la relación causal más significativa con el objetivo.

Tabla 149. propiedades de tcmnode

Propiedades de tcmnode	**Valores**	**Descripción de la propiedad**
custom_fields	*boolean*	
dimensionlist	[*dimension1 ... dimensionN*]	
data_struct	Multiple Single	
metric_fields	*Campos*	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Timestamp Period	

Tabla 149. propiedades de tcmnode (continuación)		
Propiedades de tcmnode	**Valores**	**Descripción de la propiedad**
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*serie*	
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	
timestamp_increments	*integer*	

Tabla 149. propiedades de tcmnode (continuación)

Propiedades de tcmnode	**Valores**	**Descripción de la propiedad**
cyclic_increments	*integer*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	Same Notsame	
cross_hour	*boolean*	
aggregate_and_distribute	*lista*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	

Tabla 149. propiedades de tcmnode (continuación)

Propiedades de tcmnode	Valores	Descripción de la propiedad
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend None	
k_mean_param	*integer*	
k_median_param	*integer*	
missing_value_threshold	*integer*	
conf_level	*integer*	
max_num_predictor	*integer*	
max_lag	*integer*	
epsilon	*number*	
threshold	*integer*	
is_re_est	*boolean*	
num_targets	*integer*	
percent_targets	*integer*	
fields_display	*lista*	
series_display	*lista*	
network_graph_for_target	*boolean*	
sign_level_for_target	*number*	
fit_and_outlier_for_target	*boolean*	
sum_and_para_for_target	*boolean*	
impact_diag_for_target	*boolean*	

Tabla 149. propiedades de tcmnode (continuación)

Propiedades de tcmnode	Valores	Descripción de la propiedad
impact_diag_type_for_target	Effect Cause Both	
impact_diag_level_for_target	*integer*	
series_plot_for_target	*boolean*	
res_plot_for_target	*boolean*	
top_input_for_target	*boolean*	
forecast_table_for_target	*boolean*	
same_as_for_target	*boolean*	
network_graph_for_series	*boolean*	
sign_level_for_series	*number*	
fit_and_outlier_for_series	*boolean*	
sum_and_para_for_series	*boolean*	
impact_diagram_for_series	*boolean*	
impact_diagram_type_for_series	Effect Cause Both	
impact_diagram_level_for_series	*integer*	
series_plot_for_series	*boolean*	
residual_plot_for_series	*boolean*	
forecast_table_for_series	*boolean*	
outlier_root_cause_analysis	*boolean*	
causal_levels	*integer*	
outlier_table	Interactive Pivot Both	
rmsp_error	*boolean*	

Tabla 149. propiedades de tcmnode (continuación)		
Propiedades de tcmnode	**Valores**	**Descripción de la propiedad**
bic	*boolean*	
r_square	*boolean*	
outliers_over_time	*boolean*	
series_transormation	*boolean*	
use_estimation_period	*boolean*	
estimation_period	Times Observation	
observations	*lista*	
observations_type	Latest Earliest	
observations_num	*integer*	
observations_exclude	*integer*	
extend_records_into_future	*boolean*	
forecastperiods	*integer*	
max_num_distinct_values	*integer*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*boolean*	
aic	*boolean*	
rmse	*boolean*	

propiedades ts

El nodo Serie temporal estima modelos de suavizado exponencial, modelos autorregresivos integrados de media móvil (ARIMA) univariados y modelos ARIMA (o de función de transferencia) multivariados para series temporales y genera previsiones. Este nodo Serie temporal es similar al nodo Serie temporal anterior que estaba en desuso en SPSS Modeler versión 18. Sin embargo, este nodo Serie temporal más reciente se ha diseñado para emplear la potencia de IBM SPSS Analytic Server para procesar grandes cantidades de datos y mostrar el modelo resultante en el visor de resultados que se ha añadido en SPSS Modeler versión 17.

Tabla 150. propiedades ts		
Propiedades ts	**Valores**	**Descripción de la propiedad**
targets	*campo*	El nodo Serie temporal prevé uno o más objetivos, utilizando opcionalmente uno o más campos de entrada como predictores. Los campos de frecuencia y ponderación no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
candidate_inputs	[*field1 ... fieldN*]	Campos de entrada o predictor utilizados por el modelo.
use_period	*Flag*	
date_time_field	*campo*	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*campo*	

Tabla 150. propiedades ts (continuación)

Propiedades ts	**Valores**	**Descripción de la propiedad**
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	
timestamp_increments	*integer*	
cyclic_increments	*integer*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*Flag*	
cross_hour	*Flag*	
aggregate_and_distribute	*lista*	

Tabla 150. propiedades ts (continuación)

Propiedades ts	**Valores**	**Descripción de la propiedad**
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*integer*	
use_estimation_period	*Flag*	
estimation_period	Observations Times	
date_estimation	*lista*	Solo está disponible si se utiliza date_time_field
period_estimation	*lista*	Solo está disponible si se utiliza use_period
observations_type	Latest Earliest	

Tabla 150. propiedades ts (continuación)		
Propiedades ts	**Valores**	**Descripción de la propiedad**
observations_num	*integer*	
observations_exclude	*integer*	
method	ExpertModeler Exsmooth Arima	
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*Flag*	
detect_outliers	*Flag*	
expert_outlier_additive	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_innovational	*Flag*	
expert_outlier_level_shift	*Flag*	
expert_outlier_transient	*Flag*	
expert_outlier_seasonal_additive	*Flag*	
expert_outlier_local_trend	*Flag*	
expert_outlier_additive_patch	*Flag*	
consider_newesmodels	*Flag*	

Tabla 150. propiedades ts (continuación)

Propiedades ts	**Valores**	**Descripción de la propiedad**
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	Especifica el método de Suavizado exponencial. El valor predeterminado es Simple.

Tabla 150. propiedades ts (continuación)

Propiedades ts	**Valores**	**Descripción de la propiedad**

futureValue_type_method	Compute
	
	specify
		Si se utiliza Compute, el sistema calcula los futuros valores para el período de predicción para cada predictor.
		
		Para cada predictor, puede elegir en una lista de funciones (blanco, media de puntos recientes, valor más reciente) o utilizar specify para especificar valores manualmente. Para especificar campos y propiedades individuales, utilice la propiedad extend_metric_values. Por ejemplo:
		
		`set :ts.futureValue_type_method="specify"`
		`set :ts.extend_metric_values=[{'Market_1', 'USER_SPECIFY', [1,2,3]}, {'Market_2','MOST_RECENT_VALUE', ''}, {'Market_3','RECENT_POINTS_MEAN', ''}]`

exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*integer*	
arima.d	*integer*	
arima.q	*integer*	
arima.sp	*integer*	
arima.sd	*integer*	
arima.sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*Flag*	

Tabla 150. propiedades ts (continuación)

Propiedades ts	**Valores**	**Descripción de la propiedad**
tf_arima.p. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.d. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.q. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sp. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sd. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.sq. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.delay. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima.transformation_type. *fieldname*	None SquareRoot NaturalLog	Para funciones de transferencia.
arima_detect_outliers	*Flag*	
arima_outlier_additive	*Flag*	
arima_outlier_level_shift	*Flag*	
arima_outlier_innovational	*Flag*	
arima_outlier_transient	*Flag*	
arima_outlier_seasonal_additive	*Flag*	
arima_outlier_local_trend	*Flag*	
arima_outlier_additive_patch	*Flag*	
max_lags	*integer*	
cal_PI	*Flag*	
conf_limit_pct	*REAL*	
events	*Campos*	
continue	*Flag*	
scoring_model_only	*Flag*	Se utiliza para modelos con cifras muy grandes (cientos de miles) o series temporales.
forecastperiods	*integer*	
extend_records_into_future	*Flag*	

Tabla 150. propiedades ts (continuación)		
Propiedades ts	**Valores**	**Descripción de la propiedad**
extend_metric_values	*Campos*	Le permite proporcionar valores futuros para los predictores.
conf_limits	*Flag*	
noise_res	*Flag*	
max_models_output	*integer*	Controla cuántos modelos se muestran en el resultado. El valor predeterminado es 10. Los modelos no se muestran en el resultado si el número total de modelos creados supera este valor. Los modelos aún están disponibles para la puntuación.

Propiedades timeseriesnode (en desuso)

Nota: Esta nodo Serie temporal original estaba en desuso en la versión 18 de SPSS Modeler y se sustituye por el nuevo nodo Serie temporal que se ha designado para emplear la potencia de IBM SPSS Analytic Server y procesar grandes cantidades de datos (Big Data).

El nodo Serie temporal estima modelos de suavizado exponencial, modelos autorregresivos integrados de media móvil (ARIMA) univariados y modelos ARIMA (o de función de transferencia) multivariados para series temporales y genera previsiones. Un nodo Serie temporal debe ir siempre precedido por un nodo Intervalos de tiempo.

Ejemplo

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tabla 151. propiedades de timeseriesnode		
Propiedades de `timeseriesnode`	**Valores**	**Descripción de la propiedad**
`targets`	*campo*	El nodo Serie temporal prevé uno o más objetivos, utilizando opcionalmente uno o más campos de entrada como predictores. Los campos de frecuencia y ponderación no se usan. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.
`continue`	*Flag*	

`method`	`ExpertModeler`
	
	`Exsmooth`
	
	`Arima`
	
	`Reuse`	

`expert_modeler_method`	*Flag*	
`consider_seasonal`	*Flag*	
`detect_outliers`	*Flag*	
`expert_outlier_additive`	*Flag*	
`expert_outlier_level_shift`	*Flag*	
`expert_outlier_innovational`	*Flag*	
`expert_outlier_level_shift`	*Flag*	
`expert_outlier_transient`	*Flag*	
`expert_outlier_seasonal_additive`	*Flag*	
`expert_outlier_local_trend`	*Flag*	
`expert_outlier_additive_patch`	*Flag*	

Tabla 151. propiedades de timeseriesnode (continuación)		
Propiedades de timeseriesnode	**Valores**	**Descripción de la propiedad**
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	
arima_d	*integer*	
arima_q	*integer*	
arima_sp	*integer*	
arima_sd	*integer*	
arima_sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*Flag*	
tf_arima_p. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_d. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_q. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_sp. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_sd. *fieldname*	*integer*	Para funciones de transferencia.

Tabla 151. propiedades de timeseriesnode (continuación)		
Propiedades de timeseriesnode	**Valores**	**Descripción de la propiedad**
tf_arima_sq. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_delay. *fieldname*	*integer*	Para funciones de transferencia.
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	Para funciones de transferencia.
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*Flag*	
arima_outlier_level_shift	*Flag*	
arima_outlier_innovational	*Flag*	
arima_outlier_transient	*Flag*	
arima_outlier_seasonal_additive	*Flag*	
arima_outlier_local_trend	*Flag*	
arima_outlier_additive_patch	*Flag*	
conf_limit_pct	*REAL*	
max_lags	*integer*	
events	*Campos*	
scoring_model_only	*Flag*	Se utiliza para modelos con cifras muy grandes (cientos de miles) o series temporales.

propiedades de treeas

El nodo Tree-AS es similar al nodo CHAID existente; sin embargo, el nodo Tree-AS se ha designado para procesar grandes cantidades de datos (Big Data) para crear un solo árbol y mostrar el modelo resultante en el visor de salida que se ha añadido en SPSS Modeler versión 17. El nodo genera un árbol de decisiones utilizando estadísticas de chi-cuadrado (CHAID) para identificar divisiones opcionales. Este uso de CHAID puede generar árboles no binarios, lo que significa que algunas divisiones generarán más de dos ramas. Los campos de entrada y objetivo pueden ser continuos (rango numérico) o categóricos. CHAID exhaustivo es una modificación de CHAID que examina con mayor precisión todas las divisiones posibles, aunque necesita más tiempo para realizar los cálculos.

Tabla 152. propiedades de treeas		
Propiedades de `treeas`	**Valores**	**Descripción de la propiedad**
`target`	*campo*	En el nodo Tree-AS, los modelos CHAID requieren un único objetivo y uno o más campos de entrada. También se puede especificar un campo de frecuencia. Para obtener más información, consulte el tema "Propiedades comunes de nodos de modelado" en la página 219.

`method`	`chaid`
	
	`exhaustive_chaid`	

`max_depth`	*integer*	Profundidad máxima del árbol, de 0 a 20. El valor predeterminado es 5.
`num_bins`	*integer*	Sólo se utiliza si los datos están formados por entradas continuas. Establezca el número de intervalos de frecuencia que deben utilizarse para las entradas; las opciones son 2, 4, 5, 10, 20, 25, 50 o 100.
`record_threshold`	*integer*	El número de registros en los que el modelo pasa de usar valores p a tamaños del efecto mientras se construye el árbol. El valor predeterminado es de 1.000.000; se incrementa o decrementa de 10.000 en 10.000.
`split_alpha`	*number*	Nivel de significancia para división. El valor debe estar entre 0,01 y 0,99.
`merge_alpha`	*number*	Nivel de significancia para fusión. El valor debe estar entre 0,01 y 0,99.
`bonferroni_adjustment`	*Flag*	Los valores de significancia de ajuste utilizando el método de Bonferroni.
`effect_size_threshold_cont`	*number*	Establece el umbral del tamaño del efecto cuando se dividen los nodos y fusionan las categorías al usar un objetivo continuo. El valor debe estar entre 0,01 y 0,99.
`effect_size_threshold_cat`	*number*	Establezca el umbral de tamaño de efecto al dividir nodos y fusionar categorías al utilizar un objetivo categórico. El valor debe estar entre 0,01 y 0,99.
`split_merged_categories`	*Flag*	Permitir segunda división de categorías fusionadas.
`grouping_sig_level`	*number*	Se usa para determinar cómo se forman los grupos de nodos o cómo se identifican los nodos inusuales.

Tabla 152. propiedades de treeas (continuación)

Propiedades de treeas	Valores	Descripción de la propiedad
chi_square	pearson likelihood_ratio	Método usado para calcular la estadística de chi cuadrado: Pearson o Razón de verosimilitud
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*number*	El valor predeterminado es 2. Mínimo 1, máximo 100, e incrementos de 1. El valor de la rama padre debe ser mayor que la rama hijo.
min_child_records_pc	*number*	El valor predeterminado es 1. Mínimo 1, máximo 100, e incrementos de 1.
min_parent_records_abs	*number*	El valor predeterminado es 100. Mínimo 1, máximo 100, e incrementos de 1. El valor de la rama padre debe ser mayor que la rama hijo.
min_child_records_abs	*number*	El valor predeterminado es 50. Mínimo 1, máximo 100, e incrementos de 1.
epsilon	*number*	Cambio mínimo en frecuencias de casillas esperadas.
max_iterations	*number*	Número máximo de iteraciones para la convergencia.
use_costs	*Flag*	
costs	*structured*	Propiedad estructurada. El formato es una lista de 3 valores: el valor real, el valor predicho y el coste si dicha predicción es errónea. Por ejemplo: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square custom	**Nota:** sólo se habilita para objetivos ordinales. Establezca valores predeterminados en la matriz de costes.
calculate_conf	*Flag*	
display_rule_id	*Flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.

Propiedades de twostepnode

El nodo Bietápico es un método de agrupación en clústeres de dos pasos. El primer paso es hacer una única pasada por los datos para comprimir los datos de entrada de la fila en un conjunto de subclústeres administrable. El segundo paso utiliza un método de agrupación en clústeres jerárquica para fundir progresivamente los subclústeres en clústeres cada vez más grandes. El bietápico tiene la ventaja de estimar automáticamente el número óptimo de clústeres para los datos de entrenamiento. Puede gestionar tipos de campos mixtos y grandes conjuntos de datos eficazmente.

Ejemplo

```
node = stream.create("twostep", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Age", "K", "Na", "BP"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "TwoStep_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("exclude_outliers", True)
node.setPropertyValue("cluster_label", "String")
node.setPropertyValue("label_prefix", "TwoStep_")
node.setPropertyValue("cluster_num_auto", False)
node.setPropertyValue("max_num_clusters", 9)
node.setPropertyValue("min_num_clusters", 3)
node.setPropertyValue("num_clusters", 7)
```

Tabla 153. propiedades de twostepnode

Propiedad de twostepnode	Valores	Descripción de la propiedad
inputs	[*field1 ... fieldN*]	Los modelos bietápicos utilizan una lista de campos de entrada, pero no de campos objetivo. Los campos de ponderación y frecuencia no se reconocen. Para obtener más información, consulte el tema “Propiedades comunes de nodos de modelado” en la página 219.
standardize	*Flag*	
exclude_outliers	*Flag*	
percentage	*number*	
cluster_num_auto	*Flag*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String Number	
label_prefix	*serie*	

Tabla 153. propiedades de twostepnode (continuación)		
Propiedad de twostepnode	**Valores**	**Descripción de la propiedad**
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Propiedades de twostepAS

El clúster bietápico es una herramienta de exploración diseñada para descubrir las agrupaciones naturales (o clústeres) de un conjunto de datos que, de otra manera, no sería posible detectar. El algoritmo que emplea este procedimiento incluye varias atractivas características que lo hacen diferente de las técnicas de agrupación en clústeres tradicionales, como el tratamiento de variables categóricas y continuas, la selección automática de número de clústeres y la escalabilidad.

Tabla 154. propiedades de twostepAS		
Propiedades de twostepAS	**Valores**	**Descripción de la propiedad**
inputs	[*f1 ... fN*]	Los modelos bietápicos utilizan una lista de campos de entrada, pero no de objetivos. Los campos de ponderación y frecuencia no se reconocen.
use_predefined_roles	Booleano	Default=True
use_custom_field_assignments	Booleano	Default=False
cluster_num_auto	Booleano	Default=True
min_num_clusters	integer	Default=2
max_num_clusters	integer	Default=15
num_clusters	integer	Default=5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Minimum Maximum	

Tabla 154. propiedades de twostepAS (continuación)

Propiedades de twostepAS	**Valores**	**Descripción de la propiedad**
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	Booleano	
random_seed	integer	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	Booleano	Default=True
num_cases_in_feature_tree_leaf_is_less_than	integer	Default=10
top_perc_outliers	integer	Default=5
initial_dist_change_threshold	integer	Default=0
leaf_node_maximum_branches	integer	Default=8
non_leaf_node_maximum_branches	integer	Default=8
max_tree_depth	integer	Default=3
adjustment_weight_on_measurement_level	integer	Default=6
memory_allocation_mb	number	Default=512
delayed_split	Booleano	Default=True
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	Booleano	Default=True
featureMisPercent	integer	Default=70
coefRange	number	Default=0.05
percCasesSingleCategory	integer	Default=95
numCases	integer	Default=24
include_model_specifications	Booleano	Default=True
include_record_summary	Booleano	Default=True
include_field_transformations	Booleano	Default=True
excluded_inputs	Booleano	Default=True
evaluate_model_quality	Booleano	Default=True
show_feature_importance bar chart	Booleano	Default=True
show_feature_importance_ word_cloud	Booleano	Default=True

Tabla 154. propiedades de twostepAS (continuación)

Propiedades de twostepAS	**Valores**	**Descripción de la propiedad**
show_outlier_clusters interactive_table_and_chart	Booleano	Default=True
show_outlier_clusters_pivot_table	Booleano	Default=True
across_cluster_feature_importance	Booleano	Default=True
across_cluster_profiles_pivot_table	Booleano	Default=True
withinprofiles	Booleano	Default=True
cluster_distances	Booleano	Default=True
cluster_label	String Number	
label_prefix	String	

Capítulo 14. Propiedades del nodo de nugget de modelo

Los nodos de nugget de modelo comparten las mismas propiedades comunes que los otros nodos. Para obtener más información, consulte el tema "Propiedades de nodos comunes" en la página 76.

Propiedades de applyanomalydetectionnode

Los nodos de modelado Detección de anomalías pueden utilizarse para generar un nugget de modelo Detección de anomalías. El nombre de script de este nugget de modelo es *applyanomalydetectionnode*. Para obtener más información sobre la creación de scripts, el propio nodo de modelado, "propiedades de anomalydetectionnode" en la página 220

Tabla 155. propiedades de applyanomalydetectionnode

Propiedades de applyanomalydetectionnode	**Valores**	**Descripción de la propiedad**
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Determina que resultados se crean para puntuación.
num_fields	*integer*	Campos para informar.
discard_records	*Flag*	Indica si los registros se descartan del resultado o no.
discard_anomalous_records	*Flag*	Indicador de cuando descartar los registros anómalos o *no* anómalos. El valor predeterminado es off, que significa que se descartan los registros *no* anómalos. En caso contrario, si es on, se descartan los registros anómalos. Esta propiedad se activa sólo si la propiedad discard_records se activa.

Propiedades de applyapriorinode

Los nodos de modelado Apriori pueden utilizarse para generar un nugget de modelo Apriori. El nombre de script de este nugget de modelo es *applyapriorinode*. Para obtener más información sobre la creación de scripts, el propio nodo de modelado, "propiedades de apriorinode" en la página 221

Tabla 156. propiedades de applyapriorinode

Propiedades de applyapriorinode	**Valores**	**Descripción de la propiedad**
max_predictions	*número (entero)*	
ignore_unmatached	*Flag*	
allow_repeats	*Flag*	

Tabla 156. propiedades de applyapriorinode (continuación)		
Propiedades de applyapriorinode	**Valores**	**Descripción de la propiedad**
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Deployability	

Propiedades de applyassociationrulesnode

El nodo de modelado de reglas de asociación se puede utilizar para generar un nugget de modelo de reglas de asociación. El nombre de script de este nugget de modelo es *applyassociationrulesnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades associationrulesnode" en la página 223.

Tabla 157. propiedades de applyassociationrulesnode		
Propiedades de applyassociationrulesnode	**Tipo de datos**	**Descripción de la propiedad**
max_predictions	*integer*	El número máximo de reglas que se pueden aplicar a cada entrada de la puntuación.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Seleccione la medida utilizada para determinar la fuerza de las reglas.
allow_repeats	*boolean*	Determine si las reglas con la misma predicción se incluyen en la puntuación.
check_input	NoPredictions Predictions NoCheck	

Propiedades de applyautoclassifiernode

Los nodos de modelado de clasificador automático se pueden utilizar para crear un nugget de modelo Clasificador automático. El nombre de script de este nugget de modelo es *applyautoclassifiernode*. Para obtener más información sobre los scripts para el propio nodo de modelado, "propiedades de autoclassifiernode" en la página 226.

Tabla 158. propiedades de applyautoclassifiernode

Propiedades de applyautoclassifiernode	**Valores**	**Descripción de la propiedad**
flag_ensemble_method	Voting EvaluationWeightedVoting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
flag_evaluation_selection	Accuracy AUC_ROC	Esta opción es solo para el objetivo de marca, para decidir la medida de evaluación que se elige para la votación ponderada de evaluación.
filter_individual_model_output	*Flag*	Especifica si los resultados de puntuación de los modelos individuales se deben eliminar.
is_ensemble_update	*Flag*	Habilita la modalidad de aprendizaje automático de las máquinas automático continuo, que añade nuevos modelos de componentes a un conjunto de modelos automático existente en lugar de sustituir el modelo automático existente, y vuelve a evaluar las medidas de los modelos de componentes existentes utilizando datos nuevos disponibles.
is_auto_ensemble_weights_reevaluation	*Flag*	Habilita la reevaluación de pesos del modelo automático.
use_accumulated_factor	*Flag*	El factor acumulado se utiliza para calcular las medidas acumuladas.
accumulated_factor	*número (double)*	El valor máximo es 0.99 y el valor mínimo es 0.85.

Tabla 158. propiedades de applyautoclassifiernode (continuación)		
Propiedades de applyautoclassifiernode	**Valores**	**Descripción de la propiedad**
use_accumulated_reducing	*Flag*	Realiza la reducción del modelo en función del límite acumulado durante la renovación del modelo.
accumulated_reducing_limit	*número (double)*	El valor máximo es 0.7 y el valor mínimo es 0.1.
use_accumulated_weighted_evaluation	*Flag*	La medida de evaluación acumulada se utiliza para votar cuando se selecciona el método de votación ponderado de evaluación para el método de conjunto.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de marca.
set_ensemble_method	Voting EvaluationWeightedVoting ConfidenceWeightedVoting HighestConfidence	Especifica el método utilizado para determinar la puntuación del conjunto. Este conjunto sólo se aplica si el objetivo seleccionado es un campo de conjunto.
set_voting_tie_selection	Random HighestConfidence	Si se selecciona un método de votación, especifica cómo se resolverán los empates. Este conjunto sólo se aplica si el objetivo seleccionado es un campo nominal.

Propiedades de applyautoclusternode

Los nodos de modelado de Clúster automático se pueden utilizar para crear un nugget de modelo Clúster automático. El nombre de script de este nugget de modelo es *applyautoclusternode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de nodo de agrupación en clústeres automática" en la página 229.

Propiedades de applyautonumericnode

Los nodos de modelado autonumérico se pueden utilizar para crear un nugget de modelo Autonumérico. El nombre de script de este nugget de modelo es *applyautonumericnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, "propiedades de autonumericnode" en la página 231.

Tabla 159. propiedades de applyautonumericnode		
Propiedades de applyautonumericnode	**Valores**	**Descripción de la propiedad**
calculate_standard_error	*Flag*	

Propiedades de applybayesnetnode

Los nodos de modelado de red bayesiana pueden utilizarse para generar un nugget de modelo de red bayesiana. El nombre de script de este nugget de modelo es *applybayesnetnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de bayesnetnode" en la página 233.

Tabla 160. propiedades de applybayesnetnode		
Propiedades de applybayesnetnode	**Valores**	**Descripción de la propiedad**
all_probabilities	*Flag*	
raw_propensity	*Flag*	
adjusted_propensity	*Flag*	
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applyc50node

Los nodos de modelado C5.0 pueden utilizarse para generar un nugget de modelo C5.0. El nombre de script de este nugget de modelo es *applyc50node*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de c50node" en la página 235.

Tabla 161. propiedades de applyc50node		
Propiedades de applyc50node	**Valores**	**Descripción de la propiedad**
sql_generate	udf Never NoMissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas. El valor predeterminado es udf.
calculate_conf	*Flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applycarmanode

Los nodos de modelado CARMA pueden utilizarse para generar un nugget de modelo CARMA. El nombre de script de este nugget de modelo es *applycarmanode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de carmanode" en la página 237.

Propiedades de applycartnode

Se pueden utilizar los nodos de modelado C&RT para generar un nugget de modelo C&RT. El nombre de script de este nugget de modelo es *applycartnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de cartnode" en la página 238.

Tabla 162. propiedades de applycartnode

Propiedades de applycartnode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	Never MissingValues NoMissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas.
calculate_conf	*Flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
display_rule_id	*Flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applychaidnode

Los nodos de modelado CHAID pueden utilizarse para generar un nugget de modelo CHAID. El nombre de script de este nugget de modelo es *applychaidnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de chaidnode" en la página 241.

Tabla 163. propiedades de applychaidnode

Propiedades de applychaidnode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	Never MissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas.
calculate_conf	*Flag*	

Tabla 163. propiedades de applychaidnode (continuación)		
Propiedades de applychaidnode	**Valores**	**Descripción de la propiedad**
display_rule_id	*Flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applycoxregnode

Los nodos de modelado Cox pueden utilizarse para generar un nugget de modelo Cox. El nombre de script de este nugget de modelo es *applycoxregnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de coxregnode" en la página 244.

Tabla 164. propiedades de applycoxregnode		
Propiedades de applycoxregnode	**Valores**	**Descripción de la propiedad**
future_time_as	Intervals Fields	
time_interval	*number*	
num_future_times	*integer*	
time_field	*campo*	
past_survival_time	*campo*	
all_probabilities	*Flag*	
cumulative_hazard	*Flag*	

Propiedades de applydecisionlistnode

Los nodos de modelado Lista de decisiones pueden utilizarse para generar un nugget de modelo Lista de decisiones. El nombre de script de este nugget de modelo es *applydecisionlistnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de decisionlistnode" en la página 246.

Tabla 165. propiedades de applydecisionlistnode		
Propiedades de applydecisionlistnode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	*Flag*	Cuando se establece en true, IBM SPSS Modeler intenta enviar el modelo Lista de decisiones a SQL.
calculate_raw_propensities	*Flag*	

Tabla 165. propiedades de applydecisionlistnode (continuación)		
Propiedades de applydecisionlistnode	**Valores**	**Descripción de la propiedad**
calculate_adjusted_propensities	*Flag*	

Propiedades de applydiscriminantnode

Los nodos de modelado Discriminante pueden utilizarse para generar un nugget de modelo Discriminante. El nombre de script de este nugget de modelo es *applydiscriminantnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de discriminantnode" en la página 248.

Tabla 166. propiedades de applydiscriminantnode		
Propiedades de applydiscriminantnode	**Valores**	**Descripción de la propiedad**
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applyextension

Los nodos de Modelo de extensión se pueden utilizar para generar un nugget de modelo de Extensión. El nombre de script de este nugget de modelo es *applyextension*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de extensionmodelnode" en la página 250.

Ejemplo de Python para Spark

```
#### script example for Python for Spark
applyModel = stream.findByType("extension_apply", None)

score_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTreeModel
from pyspark.sql.types import StringType, StructField

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()

    _modelPath = cxt.getModelContentToPath("TreeModel")
    metadata = json.loads(cxt.getModelContentToString("model.dm"))

    schema = df.dtypes[:]
    target = "Drug"
    predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

    lookup = {}
```

```
    for i in range(0,len(schema)):
        lookup[schema[i][0]] = i

    def row2LabeledPoint(dm,lookup,target,predictors,row):
        target_index = lookup[target]
        tval = dm[target_index].index(row[target_index])
        pvals = []
        for predictor in predictors:
            predictor_index = lookup[predictor]
            if isinstance(dm[predictor_index],list):
                pval = row[predictor_index] in dm[predictor_index] and
dm[predictor_index].index(row[predictor_index]) or -1
            else:
                pval = row[predictor_index]
            pvals.append(pval)
        return LabeledPoint(tval, DenseVector(pvals))

    # convert dataframe to an RDD containing LabeledPoint
    lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
    treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
    # score the model, produces an RDD containing just double values
    predictions = treeModel.predict(lps.map(lambda lp: lp.features))

    def addPrediction(x,dm,lookup,target):
        result = []
        for _idx in range(0, len(x[0])):
            result.append(x[0][_idx])
        result.append(dm[lookup[target]][int(x[1])])
        return result

    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    rdd2 = df.rdd.zip(predictions).map(lambda x:addPrediction(x, metadata, lookup, target))
    outDF = cxt.getSparkSQLContext().createDataFrame(rdd2, _schema)

    cxt.setSparkOutputData(outDF)
"""
applyModel.setPropertyValue("python_syntax", score_script)
```

Ejemplo de R

```
#### script example for R
applyModel.setPropertyValue("r_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabla 167. Propiedades de applyextension

Propiedades de applyextension	Valores	Descripción de la propiedad
r_syntax	*serie*	Sintaxis de scripts R para la puntuación del modelo.
python_syntax	*serie*	Sintaxis de scripts Python para la puntuación de modelo.
use_batch_size	*Flag*	Habilitar uso de proceso por lotes.
batch_size	*integer*	Especifique el número de registros de datos que se deben incluir en cada lote.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opción para convertir campos de distintivos.

Tabla 167. Propiedades de applyextension (continuación)		
Propiedades de applyextension	**Valores**	**Descripción de la propiedad**
convert_missing	*Flag*	Opción para convertir valores perdidos en el valor R NA.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.

Propiedades de applyfactornode

Los nodos de modelado PCA/Factorial pueden utilizarse para generar un nugget de modelo PCA/Factorial. El nombre de script de este nugget de modelo es *applyfactornode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de factornode" en la página 253.

Propiedades de applyfeatureselectionnode

Los nodos de modelado Selección de características pueden utilizarse para generar un nugget de modelo Selección de características. El nombre de script de este nugget de modelo es *applyfeatureselectionnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de featureselectionnode" en la página 255.

Tabla 168. propiedades de applyfeatureselectionnode		
Propiedades de applyfeatureselectionnode	**Valores**	**Descripción de la propiedad**
selected_ranked_fields		especifica qué campos clasificados se comprueban en el explorador de modelos.
selected_screened_fields		Especifica qué campos filtrados se comprueban en el explorador de modelos.

Propiedades de applygeneralizedlinearnode

Los nodos de modelado lineal generalizado (genlin) pueden utilizarse para generar un nugget de modelo lineal generalizado. El nombre de script de este nugget de modelo es *applygeneralizedlinearnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de genlinnode" en la página 257.

Tabla 169. propiedades de applygeneralizedlinearnode		
Propiedades de applygeneralizedlinearnode	**Valores**	**Descripción de la propiedad**
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applyglmmnode

Los nodos de modelado GLMM pueden utilizarse para generar un nugget de modelo GLMM. El nombre de script de este nugget de modelo es *applyglmmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de glmmnode" en la página 263.

Tabla 170. Propiedades de applyglmmnode		
Propiedades de applyglmmnode	**Valores**	**Descripción de la propiedad**
confidence	onProbability onIncrease	Base para calcular el valor de confianza de la puntuación: probabilidad más alta predicha, o la diferencia entre la probabilidad más alta predicha y la segunda probabilidad más alta.
score_category_probabilities	*Flag*	Si es True, genera las probabilidades predichas para objetivos categóricos. Se crea un campo para cada categoría. El valor predeterminado es False.
max_categories	*integer*	Número máximo de categorías para el que se van a predecir las probabilidades. Sólo se utiliza si score_category_probabilities es True.
score_propensity	*Flag*	Si se establece en True, genera puntuaciones de propensión en bruto (probabilidad de resultado "True") para modelos con objetivos de marca. Si las particiones están en vigor, también genera puntuaciones de propensión ajustadas en función de la partición de prueba. El valor predeterminado es False.

Tabla 170. Propiedades de applyglmmnode (continuación)		
Propiedades de applyglmmnode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	udf native	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Las opciones son para retrotraer a la base de datos y puntuar utilizando un adaptador de puntuación de SPSS® Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado) o para puntuar en SPSS Modeler. El valor predeterminado es udf.

Propiedades de applygle

Los nodos de modelado GLE pueden utilizarse para generar un nugget de modelo GLE. El nombre de script de este nugget de modelo es *applygle*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de gle" en la página 268.

Tabla 171. Propiedades de applygle		
Propiedades de applygle	**Valores**	**Descripción de la propiedad**
enable_sql_generation	udf native	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Elija si desea realizar una retrotracción a la base de datos y puntuar utilizando un adaptador de puntuación de SPSS Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado) o puntuar dentro de SPSS Modeler.

Propiedades applygmm

El nodo de mezcla gausiana se puede utilizar para generar un nugget de modelo de mezcla gausiana. El nombre de script de este nugget de modelo es *applygmm*. Las propiedades de la siguiente tabla están disponibles en la versión 18.2.1.1 y posterior. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades gmm" en la página 439.

Tabla 172. Propiedades applygmm		
Propiedades applygmm	**Tipo de datos**	**Descripción de la propiedad**
centers		
item_count		
total		
dimension		
components		
partition		

Propiedades de applykmeansnode

Los nodos de modelado K-medias pueden utilizarse para generar un nugget de modelo K-medias. El nombre de script de este nugget de modelo es *applykmeansnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de kmeansnode" en la página 275.

Propiedades de applyknnnode

Los nodos de modelado KNN pueden utilizarse para generar un nugget de modelo KNN. El nombre de script de este nugget de modelo es *applyknnnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de knnnode" en la página 277.

Tabla 173. propiedades de applyknnnode

Propiedades de applyknnnode	**Valores**	**Descripción de la propiedad**
`all_probabilities`	*Flag*	
`save_distances`	*Flag*	

Propiedades de applykohonennode

Los nodos de modelado Kohonen pueden utilizarse para generar un nugget de modelo Kohonen. El nombre de script de este nugget de modelo es *applykohonennode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de c50node" en la página 235.

Propiedades de applylinearnode

Los nodos de modelado lineal pueden utilizarse para generar un nugget de modelo lineal. El nombre de script de este nugget de modelo es *applylinearnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de linearnode" en la página 280.

Tabla 174. Propiedades de applylinearnode

Propiedades de linear	**Valores**	**Descripción de la propiedad**
`use_custom_name`	*Flag*	
`custom_name`	*serie*	

`enable_sql_generation`	`udf`
	
	`native`
	
	`puresql`
		Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Las opciones se van a retrotraer a la base de datos y la puntuación mediante el adaptador de puntuación SPSS® Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado), para puntuar en SPSS Modeler, o para retrotraer a la base de datos y puntuar mediante SQL.
		
		El valor predeterminado es `udf`.

Propiedades de applylinearasnode

Los nodos de modelado Linear-AS pueden utilizarse para generar un nugget de modelo Linear-AS. El nombre de script de este nugget de modelo es *applylinearasnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de linearasnode" en la página 282.

Tabla 175. Propiedades de applylinearasnode

Propiedad de applylinearasnode	Valores	Descripción de la propiedad
enable_sql_generation	udf native	El valor predeterminado es udf.

Propiedades de applylogregnode

Los nodos de modelado Regresión logística pueden utilizarse para generar un nugget de modelo Regresión logística. El nombre de script de este nugget de modelo es *applylogregnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de logregnode" en la página 284.

Tabla 176. propiedades de applylogregnode

Propiedades de applylogregnode	Valores	Descripción de la propiedad
calculate_raw_propensities	*Flag*	
calculate_conf	*Flag*	
enable_sql_generation	*Flag*	

Propiedades de applylsvmnode

Los nodos de modelado LSVM pueden utilizarse para generar un nugget de modelo LSVM. El nombre de script de este nugget de modelo es *applylsvmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de lsvmnode" en la página 290.

Tabla 177. propiedades de applylsvmnode

Propiedades de applylsvmnode	Valores	Descripción de la propiedad
calculate_raw_propensities	*Flag*	Especifica si deben calcularse puntuaciones de propensión en bruto.
enable_sql_generation	udf native	Especifica si debe puntuarse utilizando el adaptador de puntuación (si está instalado) o en proceso, o puntuar fuera de la base de datos.

Propiedades de applyneuralnetnode

Los nodos de modelado Red neuronal pueden utilizarse para generar un nugget de modelo Red neuronal. El nombre de script de este nugget de modelo es *applyneuralnetnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de neuralnetnode" en la página 291.

Precaución: Una versión más reciente del nodo de modelado Red neural, con características mejoradas, está disponible en esta versión y se describe en la sección siguiente (*applyneuralnetwork*). Aunque la versión anterior sigue estando disponible, le recomendamos actualizar sus scripts para que se usen la nueva versión. En este documento se incluyen detalles de la versión anterior como referencia, pero en versiones futuras dejará de ser compatible.

Tabla 178. propiedades de applyneuralnetnode

Propiedades de applyneuralnetnode	**Valores**	**Descripción de la propiedad**
`calculate_conf`	*Flag*	Disponible cuando la generación de SQL está activada. Esta propiedad incluye los cálculos de confianza en el árbol generado.
`enable_sql_generation`	*Flag*	

`nn_score_method`	`Difference`
	
	`SoftMax`	

`calculate_raw_propensities`	*Flag*	
`calculate_adjusted_propensities`	*Flag*	

propiedades de applyneuralnetworknode

Los nodos de modelado Red neuronal pueden utilizarse para generar un nugget de modelo Red neuronal. El nombre de script de este nugget de modelo es *applyneuralnetworknode*. Si desea más información sobre cómo crear scripts para el propio nodo de modelado, consulte Propiedades neuralnetworknode

Tabla 179. propiedades de applyneuralnetworknode

Propiedades de applyneuralnetworknode	**Valores**	**Descripción de la propiedad**
`use_custom_name`	*Flag*	
`custom_name`	*serie*	

`confidence`	`onProbability`
	
	`onIncrease`	

`score_category_probabilities`	*Flag*	
`max_categories`	*number*	
`score_propensity`	*Flag*	

Tabla 179. propiedades de applyneuralnetworknode (continuación)		
Propiedades de applyneuralnetworknode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	udf native puresql	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Las opciones se van a retrotraer a la base de datos y la puntuación mediante el adaptador de puntuación SPSS® Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado), para puntuar en SPSS Modeler, o para retrotraer a la base de datos y puntuar mediante SQL. El valor predeterminado es udf.

Propiedades de applyocsvmnode

Los nodos SVM de una clase se pueden utilizar para generar un nugget de modelo SVM de una clase. El nombre de script de este nugget de modelo es *applyocsvmnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "ocsvmnode, propiedades" en la página 445.

Propiedades de applyquestnode

Los nodos de modelado QUEST pueden utilizarse para generar un nugget de modelo QUEST. El nombre de script de este nugget de modelo es *applyquestnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de questnode" en la página 296.

Tabla 180. propiedades de applyquestnode		
Propiedades de applyquestnode	**Valores**	**Descripción de la propiedad**
enable_sql_generation	Never MissingValues NoMissingValues	Se utiliza para establecer las opciones de generación de SQL durante la ejecución del conjunto de reglas.
calculate_conf	*Flag*	
display_rule_id	*Flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
calculate_raw_propensities	*Flag*	
calculate_adjusted_propensities	*Flag*	

Propiedades de applyr

Los nodos de modelado R pueden utilizarse para generar un nugget de modelo R. El nombre de script de este nugget de modelo es *applyr*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de buildr" en la página 235.

Tabla 181. propiedades de applyr

Propiedades de applyr	Valores	Descripción de la propiedad
score_syntax	*serie*	Sintaxis de scripts R para la puntuación del modelo.
convert_flags	StringsAndDoubles LogicalValues	Opción para convertir campos de distintivos.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	POSIXct POSIXlt	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
convert_missing	*Flag*	Opción para convertir valores perdidos en valor R NA.
use_batch_size	*Flag*	Habilite el uso del proceso por lotes
batch_size	*integer*	Especifique el número de registros de datos que se incluirán en cada lote

Propiedades applyrandomtrees

El nodo de modelador Árboles aleatorios se puede utilizar para generar un nugget del modelo Árboles aleatorios. El nombre de script de este nugget de modelo es *applyrandomtrees*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades randomtrees" en la página 299.

Tabla 182. propiedades applyrandomtrees

Propiedades applyrandomtrees	Valores	Descripción de la propiedad
calculate_conf	*Flag*	Esta propiedad incluye cálculos de confianza en el árbol generado.
enable_sql_generation	udf native	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Elija si desea realizar una retrotracción a la base de datos y puntuar utilizando un adaptador de puntuación de SPSS Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado) o puntuar dentro de SPSS Modeler.

Propiedades de applyregressionnode

Los nodos de modelado Regresión lineal pueden utilizarse para generar un nugget de modelo Regresión lineal. El nombre de script de este nugget de modelo es *applyregressionnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de regressionnode" en la página 301.

propiedades de applyselflearningnode

Los nodos de modelado de modelo de respuesta de autoaprendizaje (SLRM) pueden utilizarse para generar un nugget de modelo SLRM. El nombre de script de este nugget de modelo es *applyselflearningnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de slrmnode" en la página 305.

Tabla 183. propiedades de applyselflearningnode

Propiedades de applyselflearningnode	**Valores**	**Descripción de la propiedad**
`max_predictions`	*number*	
`randomization`	*number*	
`scoring_random_seed`	*number*	

`sort`	`ascending`
	
	`descending`
		Especifica si se mostrarán primero las ofertas con las puntuaciones más altas o más bajas.

`model_reliability`	*Flag*	Tiene en cuenta la opción de fiabilidad del modelo de la pestaña Configuración.

Propiedades de applysequencenode

Los nodos de modelado Secuencia pueden utilizarse para generar un nugget de modelo Secuencia. El nombre de script de este nugget de modelo es *applysequencenode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de sequencenode" en la página 304.

Propiedades de applysvmnode

Los nodos de modelado SVM pueden utilizarse para generar un nugget de modelo SVM. El nombre de script de este nugget de modelo es *applysvmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de svmnode" en la página 313.

Tabla 184. propiedades de applysvmnode

Propiedades de applysvmnode	**Valores**	**Descripción de la propiedad**
`all_probabilities`	*Flag*	
`calculate_raw_propensities`	*Flag*	
`calculate_adjusted_propensities`	*Flag*	

Propiedades de applystpnode

El nodo de modelado STP puede utilizarse para generar un nugget de modelo asociado, que muestra la salida del modelo en el Visor de salida. El nombre de script de este nugget de modelo es *applystpnode*.

Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de stpnode" en la página 307.

Tabla 185. propiedades de applystpnode

Propiedades de applystpnode	Tipo de datos	Descripción de la propiedad
uncertainty_factor	*boolean*	Mínimo 0, máximo 100.

Propiedades de applytcmnode

Los nodos de modelado temporal causal (TCM) pueden usarse para generar un nugget de modelo TCM. El nombre de script de este nugget de modelo es *applytcmnode*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de tcmnode" en la página 315.

Tabla 186. propiedades de applytcmnode

Propiedades de applytcmnode	Valores	Descripción de la propiedad
ext_future	*Boolean*	
ext_future_num	*integer*	
noise_res	*Boolean*	
conf_limits	*Boolean*	
target_fields	*lista*	
target_series	*lista*	

Propiedades applyts

El nodo de modelado Serie temporal se puede utilizar para generar un nugget de modelo Serie temporal. El nombre de scripts de este nugget de modelo es *applyts*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades ts" en la página 320.

Tabla 187. propiedades applyts

Propiedades applyts	Valores	Descripción de la propiedad
extend_records_into_future	*boolean*	
ext_future_num	*integer*	
compute_future_values_input	*boolean*	
forecastperiods	*integer*	
noise_res	*Boolean*	
conf_limits	*Boolean*	
target_fields	*lista*	
target_series	*lista*	
includeTargets	*campo*	

Propiedades applytimeseriesnode (en desuso)

El nodo de modelado Serie temporal se puede utilizar para generar un nugget de modelo Serie temporal. El nombre de script de este nugget de modelo es *applytimeseriesnode*. Para obtener más información

sobre los scripts para el propio nodo de modelado, consulte "Propiedades timeseriesnode (en desuso)" en la página 328.

Tabla 188. propiedades de applytimeseriesnode

Propiedades de applytimeseriesnode	Valores	Descripción de la propiedad
calculate_conf	*Flag*	
calculate_residuals	*Flag*	

Propiedades de applytreeas

Los nodos de modelado Tree-AS pueden utilizarse para generar un nugget de modelo Tree-AS. El nombre de script de este nugget de modelo es *applytreenas*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "propiedades de treeas" en la página 331.

Tabla 189. propiedades de applytreeas

Propiedades de applytreeas	Valores	Descripción de la propiedad
calculate_conf	*Flag*	Esta propiedad incluye cálculos de confianza en el árbol generado.
display_rule_id	*Flag*	Añade un campo en el resultado de puntuación que indica el ID para el nodo terminal al que se asigna cada registro.
enable_sql_generation	udf native	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Elija si desea realizar una retrotracción a la base de datos y puntuar utilizando un adaptador de puntuación de SPSS Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado) o puntuar dentro de SPSS Modeler.

Propiedades de applytwostepnode

Los nodos de modelado Bietápico pueden utilizarse para generar un nugget de modelo Bietápico. El nombre de script de este nugget de modelo es *applytwostepnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de twostepnode" en la página 334.

Propiedades de applytwostepAS

Los nodos de modelado Bietápico AS pueden utilizarse para generar un nugget de modelo Bietápico AS. El nombre de script de este nugget de modelo es *applytwostepAS*. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de twostepAS" en la página 335.

Tabla 190. Propiedades de applytwostepAS		
Propiedades de applytwostepAS	**Valores**	**Descripción de la propiedad**
enable_sql_generation	udf native	Se utiliza para establecer las opciones de generación de SQL durante la ejecución de ruta. Las opciones son para retrotraer a la base de datos y puntuar utilizando un adaptador de puntuación de SPSS® Modeler Server (si está conectado a una base de datos con un adaptador de puntuación instalado) o para puntuar en SPSS Modeler. El valor predeterminado es udf.

Propiedades de applyxgboosttreenode

El nodo XGBoost Tree se puede utilizar para generar un nugget de modelo de XGBoost Tree. El nombre de script de este nugget de modelo es *applyxgboosttreenode*. Las propiedades de la siguiente tabla se han añadido en 18.2.1.1. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades de xgboosttreenode" en la página 454.

Tabla 191. Propiedades de applyxgboosttreenode		
Propiedades de applyxgboosttreenode	**Tipo de datos**	**Descripción de la propiedad**
use_model_name		
model_name		

propiedades de applyxgboostlinearnode

Los nodos XGBoost Linear se pueden utilizar para generar un nugget de modelo de XGBoost Linear. El nombre de script de este nugget de modelo es *applyxgboostlinearnode*. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades xgboostlinearnode" en la página 453.

Propiedades de hdbscannugget

El nodo HDBSCAN se puede utilizar para generar un nugget de modelo HDBSCAN. El nombre de scripts de este nugget de modelo es hdbscannugget. No existe ninguna otra propiedad para este nugget de modelo. Para obtener más información sobre los scripts para el propio nodo de modelado, consulte "Propiedades hdbscannode" en la página 440.

propiedades de kdeapply

El nodo de modelado KDE se puede utilizar para generar un nugget de modelo KDE. El nombre de scripts de este nugget de modelo es kdeapply. Si desea más información sobre los scripts del propio nodo de modelado, consulte "Propiedades kdemodel" en la página 442.

Tabla 192. propiedades de kdeapply		
Propiedades kdeapply	**Tipo de datos**	**Descripción de la propiedad**
`outLogDensity` Renombrado como `out_log_density` a partir de la versión 18.2.1.1	*Boolean*	Especifique `True` o `False` para incluir o excluir el valor de densidad de logaritmo en el resultado. El valor predeterminado es `False`.

Capítulo 15. Propiedades del nodo de modelado de base de datos

IBM SPSS Modeler da soporte a la integración con herramientas de modelado y minería de datos disponibles de proveedores de bases de datos, incluidos Microsoft SQL Server Analysis Services, Oracle Data Mining, y IBM Netezza Analytics. Podrá crear y almacenar modelos mediante algoritmos nativos de bases de datos, todo desde la aplicación IBM SPSS Modeler. Los modelos de base de datos también se pueden crear y manipular a través de scripts utilizando las propiedades descritas en esta sección.

Por ejemplo, el siguiente fragmento de script muestra la creación de un modelo de árboles de decisión de Microsoft mediante la interfaz de scripts de IBM SPSS Modeler:

```
stream = modeler.script.stream()
msbuilder = stream.createAt("mstreenode", "MSBuilder", 200, 200)

msbuilder.setPropertyValue("analysis_server_name", 'localhost')
msbuilder.setPropertyValue("analysis_database_name", 'TESTDB')
msbuilder.setPropertyValue("mode", 'Expert')
msbuilder.setPropertyValue("datasource", 'LocalServer')
msbuilder.setPropertyValue("target", 'Drug')
msbuilder.setPropertyValue("inputs", ['Age', 'Sex'])
msbuilder.setPropertyValue("unique_field", 'IDX')
msbuilder.setPropertyValue("custom_fields", True)
msbuilder.setPropertyValue("model_name", 'MSDRUG')

typenode = stream.findByType("type", None)
stream.link(typenode, msbuilder)
results = []
msbuilder.run(results)
msapplier = stream.createModelApplierAt(results[0], "Drug", 200, 300)
tablenode = stream.createAt("table", "Results", 300, 300)
stream.linkBetween(msapplier, typenode, tablenode)
msapplier.setPropertyValue("sql_generate", True)
tablenode.run([])
```

Propiedades de nodos de modelado de Microsoft

Propiedades de nodos de modelado de Microsoft

Propiedades comunes

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de Microsoft.

Tabla 193. Propiedades comunes de nodos de Microsoft

Propiedades comunes de nodo de Microsoft	Valores	Descripción de la propiedad
analysis_database_name	*serie*	Nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host de Analysis Services.
use_transactional_data	*Flag*	Especifica si los datos de entrada están en formato tabular o transaccional.
inputs	*lista*	Campos de entrada de datos tabulares.

Tabla 193. Propiedades comunes de nodos de Microsoft (continuación)		
Propiedades comunes de nodo de Microsoft	**Valores**	**Descripción de la propiedad**
target	*campo*	Campo predicho (no aplicable a nodo Clúster de MS o nodos de Agrupación en clústeres de secuencias).
unique_field	*campo*	Campos clave.
msas_parameters	*structured*	Parámetros del algoritmo. Para obtener más información, consulte el tema "Parámetros del algoritmo" en la página 363.
with_drillthrough	*Flag*	Opción Con exploración.

Árbol de decisión de MS

No hay propiedades específicas para los nodos del tipo mstreenode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Clúster de MS

No hay propiedades específicas para los nodos del tipo msclusternode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Reglas de asociación de MS

Las siguientes propiedades específicas están disponibles para los nodos del tipo msassocnode:

Tabla 194. propiedades de msassocnode		
Propiedades de msassocnode	**Valores**	**Descripción de la propiedad**
id_field	*campo*	Identifica todas las transacciones en los datos.
trans_inputs	*lista*	Los campos de entrada de datos transaccionales.
transactional_target	*campo*	Campo predicho (datos transaccionales).

Bayesiano ingenuo de MS

No hay propiedades específicas para los nodos del tipo msbayesnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Regresión lineal de MS

No hay propiedades específicas para los nodos del tipo msregressionnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Red neuronal de MS

No hay propiedades específicas para los nodos del tipo msneuralnetworknode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Regresión logística de MS

No hay propiedades específicas para los nodos del tipo mslogisticnode. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Series temporales de MS

No hay propiedades específicas para los nodos del tipo `mstimeseriesnode`. Consulte las propiedades comunes de Microsoft que se indican al comienzo de esta sección.

Clúster de secuencias de MS

Las siguientes propiedades específicas están disponibles para los nodos del tipo `mssequenceclusternode`:

Tabla 195. propiedades de mssequenceclusternode

Propiedades de `mssequenceclusternode`	**Valores**	**Descripción de la propiedad**
`id_field`	*campo*	Identifica todas las transacciones en los datos.
`input_fields`	*lista*	Los campos de entrada de datos transaccionales.
`sequence_field`	*campo*	Identificador de secuencia.
`target_field`	*campo*	Campo predicho (datos tabulares).

Parámetros del algoritmo

Cada tipo de modelo de base de datos de Microsoft tiene parámetros específicos que se pueden establecer utilizando la propiedad `msas_parameters` -- por ejemplo:

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters",
[["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Estos parámetros se derivan de SQL Server. Para ver los parámetros relevantes para cada nodo:

1. Coloque un nodo de origen de base de datos en el lienzo.
2. Abra el nodo de origen de base de datos.
3. Seleccione un origen válido en la lista desplegable **Origen de datos**.
4. Seleccione una tabla válida en la lista **Nombre de tabla**.
5. Pulse **Aceptar** para cerrar el nodo de origen de base de datos.
6. Conecte un nodo de modelado de bases de datos de Microsoft cuyas propiedades desee conocer.
7. Abra el nodo de modelado de bases de datos.
8. Seleccione la pestaña **Experto**.

Aparecerán las propiedades `msas_parameters` disponibles de este nodo.

Propiedades de nugget de modelo de Microsoft

Las siguientes propiedades son para los nugget de modelo creados mediante los nodos de modelado de bases de datos de Microsoft.

Árbol de decisión de MS

Tabla 196. Propiedades de Árbol de decisión de MS

Propiedades de applymstreenode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.
datasource	*serie*	Nombre del origen de datos (DSN) ODBC de SQL Server.
sql_generate	*Flag* udf	Activa la generación de SQL.

Regresión lineal de MS

Tabla 197. Propiedades de Regresión lineal de MS

Propiedades de applymsregressionnode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.

Red neuronal de MS

Tabla 198. Propiedades de Red neuronal de MS

Propiedades de applymsneuralnetworknode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.

Regresión logística de MS

Tabla 199. Propiedades de Regresión logística de MS		
Propiedades de applymslogisticnode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.

Series temporales de MS

Tabla 200. Propiedades de MS Time Series		
Propiedades de applymstimeseriesnode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.
start_from	new_prediction `historical_prediction`	Especifica si se realizarán predicciones futuras o históricas.
new_step	*number*	Define el período de tiempo inicial de predicciones futuras.
historical_step	*number*	Define el período de tiempo inicial de predicciones históricas.
end_step	*number*	Define el período de tiempo final de las predicciones.

Clúster de secuencias de MS

Tabla 201. Propiedades de Agrupación en clústeres de secuencias de MS		
Propiedades de applymssequenceclusternode	**Valores**	**Descripción**
analysis_database_name	*serie*	Este nodo se puede puntuar directamente en una ruta. Esta propiedad se utiliza para identificar el nombre de la base de datos de Analysis Services.
analysis_server_name	*serie*	Nombre del host del servidor de análisis.

Propiedades de nodos de modelado de Oracle

Propiedades de nodos de modelado de Oracle

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de Oracle.

Tabla 202. Propiedades comunes de nodos de Oracle

Propiedades comunes de nodos de Oracle	**Valores**	**Descripción de la propiedad**
target	*campo*	
inputs	*Lista de campos*	
partition	*campo*	Campo usado para dividir los datos en muestras independientes para las fases de entrenamiento, comprobación y validación en la generación del modelo.
datasource		
username		
password		
epassword		
use_model_name	*Flag*	
model_name	*serie*	Nombre personalizado para nuevo modelo.
use_partitioned_data	*Flag*	Si se ha definido un campo de partición, esta opción garantiza que sólo se utilizarán los datos de la partición de entrenamiento para la generación del modelo.
unique_field	*campo*	
auto_data_prep	*Flag*	Activa o desactiva la característica de preparación de datos automática de Oracle (solamente para bases de datos 11g).
costs	*structured*	Propiedad estructurada de la siguiente forma: `[[drugA drugB 1.5] [drugA drugC 2.1]]`, donde los argumentos en [] son costes pronosticados reales.

mode	`Simple`
	
	`Expert`
		Hace que se ignoren ciertas propiedades si se establece como `Simple`, como se indica en las propiedades de nodos individuales.

use_prediction_probability	*Flag*	
prediction_probability	*serie*	
use_prediction_set	*Flag*	

Bayesiano ingenuo de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo `oranbnode`.

Tabla 203. Propiedades de oranbnode		
Propiedades de oranbnode	**Valores**	**Descripción de la propiedad**
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	*structured*	Propiedad estructurada de la siguiente forma: set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Se ignora la propiedad si mode se establece en Simple.

Bayesiano adaptativo de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraabnnode.

Tabla 204. Propiedades de oraabnnode		
Propiedades de oraabnnode	**Valores**	**Descripción de la propiedad**
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*Flag*	*
execution_time_limit	*integer*	El valor debe ser mayor que 0.*
max_naive_bayes_predictors	*integer*	El valor debe ser mayor que 0.*
max_predictors	*integer*	El valor debe ser mayor que 0.*
priors	Data Equal Custom	
custom_priors	*structured*	Propiedad estructurada de la siguiente forma: set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Se ignora la propiedad si mode se establece en Simple.

Máquinas de vectores de soporte de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo orasvmnode.

Tabla 205. Propiedades de orasvmnode

Propiedades de orasvmnode	Valores	Descripción de la propiedad
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*integer*	Solamente kernel gaussiano. El valor debe ser mayor que 0.*
convergence_tolerance	*number*	El valor debe ser mayor que 0.*
use_standard_deviation	*Flag*	Solamente kernel gaussiano.*
standard_deviation	*number*	El valor debe ser mayor que 0.*
use_epsilon	*Flag*	Solamente modelos de regresión.*
epsilon	*number*	El valor debe ser mayor que 0.*
use_complexity_factor	*Flag*	*
complexity_factor	*number*	*
use_outlier_rate	*Flag*	Solamente variantes de una clase.*
outlier_rate	*number*	Solamente variantes de una clase. 0.0–1.0.*
weights	Data Equal Custom	
custom_weights	*structured*	Propiedad estructurada de la siguiente forma: set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4] [drugY 5]]

* Se ignora la propiedad si mode se establece en Simple.

Modelos lineales generalizados de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraglmnode.

Tabla 206. Propiedades de oraglmnode

Propiedades de oraglmnode	Valores	Descripción de la propiedad
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*Flag*	*
row_weights_field	*campo*	*
save_row_diagnostics	*Flag*	*
row_diagnostics_table	*serie*	*
coefficient_confidence	*number*	*
use_reference_category	*Flag*	*
reference_category	*serie*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*Flag*	*

* Se ignora la propiedad si mode se establece en Simple.

Árbol de decisión de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oradecisiontreenode.

Tabla 207. Propiedades de oradecisiontreenode

Propiedades de oradecisiontreenode	Valores	Descripción de la propiedad
use_costs	*Flag*	
impurity_metric	Entropy Gini	
term_max_depth	*integer*	2–20.*

Tabla 207. Propiedades de oradecisiontreenode (continuación)		
Propiedades de oradecisiontreenode	**Valores**	**Descripción de la propiedad**
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*
term_minrec_node	*integer*	El valor debe ser mayor que 0.*
term_minrec_split	*integer*	El valor debe ser mayor que 0.*
display_rule_ids	*Flag*	*

* Se ignora la propiedad si mode se establece en Simple.

O-clúster de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraoclusternode.

Tabla 208. Propiedades de oraoclusternode		
Propiedades de oraoclusternode	**Valores**	**Descripción de la propiedad**
max_num_clusters	*integer*	El valor debe ser mayor que 0.
max_buffer	*integer*	El valor debe ser mayor que 0.*
sensitivity	*number*	0.0–1.0.*

* Se ignora la propiedad si mode se establece en Simple.

K-medias de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo orakmeansnode.

Tabla 209. Propiedades de orakmeansnode		
Propiedades de orakmeansnode	**Valores**	**Descripción de la propiedad**
num_clusters	*integer*	El valor debe ser mayor que 0.
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
iterations	*integer*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	El valor predeterminado es Variance.*

Tabla 209. Propiedades de orakmeansnode (continuación)		
Propiedades de orakmeansnode	**Valores**	**Descripción de la propiedad**
num_bins	*integer*	El valor debe ser mayor que 0.*
block_growth	*integer*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* Se ignora la propiedad si mode se establece en Simple.

NMF de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oranmfnode.

Tabla 210. Propiedades de oranmfnode		
Propiedades de oranmfnode	**Valores**	**Descripción de la propiedad**
normalization_method	minmax none	
use_num_features	*Flag*	*
num_features	*integer*	0–1. El algoritmo estima el valor predeterminado a partir de los datos.*
random_seed	*number*	*
num_iterations	*integer*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*Flag*	*

* Se ignora la propiedad si mode se establece en Simple.

Apriori de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo oraapriorinode.

Tabla 211. Propiedades de oraapriorinode		
Propiedades de oraapriorinode	**Valores**	**Descripción de la propiedad**
content_field	*campo*	
id_field	*campo*	
max_rule_length	*integer*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*Flag*	

Longitud mínima de la descripción de Oracle (LMD)

No hay propiedades específicas para los nodos del tipo oramdlnode. Consulte las propiedades comunes de Oracle que se indican al comienzo de esta sección.

Importancia del atributo de Oracle (AI)

Las siguientes propiedades están disponibles para los nodos del tipo oraainode.

Tabla 212. Propiedades de oraainode

Propiedades de oraainode	Valores	Descripción de la propiedad
custom_fields	*Flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*Flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos importantes.
important_label	*serie*	Especifica la etiqueta para la clasificación como "important".
select_marginal	*Flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos marginales.
marginal_label	*serie*	Especifica la etiqueta para la clasificación como "marginal".
important_above	*number*	0.0–1.0.
select_unimportant	*Flag*	Si selection_mode se establece en ImportanceLevel, determina si se seleccionan los campos sin importancia.
unimportant_label	*serie*	Especifica la etiqueta para la clasificación como "unimportant".
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	Si selection_mode se establece en ImportanceValue, determina el valor de corte que se va a usar. Acepta valores de 0 a 100.
top_n	*number*	Si selection_mode se establece en TopN, determina el valor de corte que se va a usar. Acepta valores de 0 a 1000.

Propiedades de nugget de modelo de Oracle

Las siguientes propiedades son para los nugget de modelo creados mediante los modelos de Oracle.

Bayesiano ingenuo de Oracle

No hay propiedades específicas para los nodos del tipo applyoranbnode.

Bayesiano adaptativo de Oracle

No hay propiedades específicas para los nodos del tipo `applyoraabnnode`.

Máquinas de vectores de soporte de Oracle

No hay propiedades específicas para los nodos del tipo `applyorasvmnode`.

Árbol de decisión de Oracle

Las siguientes propiedades están disponibles para los nodos del tipo `applyoradecisiontreenode`.

Tabla 213. propiedades de applyoradecisiontreenode

Propiedades de `applyoradecisiontreenode`	Valores	Descripción de la propiedad
`use_costs`	*Flag*	
`display_rule_ids`	*Flag*	

O-clúster de Oracle

No hay propiedades específicas para los nodos del tipo `applyoraoclusternode`.

K-medias de Oracle

No hay propiedades específicas para los nodos del tipo `applyorakmeansnode`.

NMF de Oracle

La siguiente propiedad está disponible para los nodos del tipo `applyoranmfnode`.

Tabla 214. propiedades de applyoranmfnode

Propiedades de `applyoranmfnode`	Valores	Descripción de la propiedad
`display_all_features`	*Flag*	

Apriori de Oracle

Este nugget de modelo no se puede aplicar en los scripts.

LMD de Oracle

Este nugget de modelo no se puede aplicar en los scripts.

Propiedades de nodos de modelado de IBM Netezza Analytics

Propiedades de nodos de modelado de Netezza

Las siguientes propiedades son comunes a los nodos de modelado de bases de datos de IBM Netezza.

Tabla 215. Propiedades comunes de nodos de Netezza		
Propiedades comunes de nodos de Netezza	**Valores**	**Descripción de la propiedad**
custom_fields	*Flag*	Si es verdadero, le permite especificar el objetivo, la entrada y otros campos del nodo actual. Si es falso, se utiliza la configuración actual de un nodo Tipo situado en un punto anterior de la ruta.
inputs	*[field1 ... fieldn]*	Campos de entrada o predictor utilizados por el modelo.
target	*campo*	Campo de destino (continuo o categórico).
record_id	*campo*	El campo que se debe utilizar como identificador de registros exclusivo.
use_upstream_connection	*Flag*	Si es verdadero (valor predeterminado), los detalles de conexión especificados en un nodo anterior. No se utiliza si se especifica move_data_to_connection.
move_data_connection	*Flag*	Si es verdadero, transfiere los datos a la base de datos especificada mediante connection. No se utiliza si se especifica use_upstream_connection .
connection	*structured*	La cadena de conexión para la base de datos de Netezza donde se almacena el modelo. Propiedad estructurada de la siguiente forma: ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' [true|false]] donde: <dsn> es el nombre del origen de datos <username> y <psw> son el nombre de usuario y la contraseña para la base de datos <catname> es el nombre de catálogo <conn_attribs> son los atributos de conexión true | false indica si la contraseña es necesaria.
table_name	*serie*	Nombre de la base de datos donde se debe almacenar el modelo.
use_model_name	*Flag*	Si es true, utiliza el nombre que especifica mediante model_name como el nombre del modelo, de lo contrario el sistema crea el nombre del modelo.
model_name	*serie*	Nombre personalizado para nuevo modelo.

Tabla 215. Propiedades comunes de nodos de Netezza (continuación)		
Propiedades comunes de nodos de Netezza	**Valores**	**Descripción de la propiedad**
include_input_fields	*Flag*	Si es verdadero, transmite todos los campos de entrada siguientes, de lo contrario solo transmite record_id y los campos generados por el modelo.

Árbol de decisión de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzadectreenode.

Tabla 216. Propiedades de netezzadectreenode		
Propiedades de netezzadectreenode	**Valores**	**Descripción de la propiedad**
impurity_measure	Entropy Gini	La medición de impureza, utilizada para valorar la mejor ubicación para dividir el árbol.
max_tree_depth	*integer*	Número máximo de niveles que puede alcanzar el crecimiento de árbol. El valor predeterminado es 62 (el máximo posible).
min_improvement_splits	*number*	Mejoras mínimas que se pueden realizar en la impureza de la división. El valor predeterminado es 0.01.
min_instances_split	*integer*	Número mínimo de registros por dividir antes de realizar la división. El valor predeterminado es 2 (el mínimo posible).
weights	*structured*	Ponderaciones relativas para clases. Propiedad estructurada de la siguiente forma: set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]] El valor predeterminado es la ponderación de 1 para todas las clases.
pruning_measure	Acc wAcc	El valor predeterminado es Acc (precisión). wAcc alternativo (precisión ponderada) tiene en cuenta las ponderaciones de clase mientras se aplica la poda.

Tabla 216. Propiedades de netezzadectreenode (continuación)		
Propiedades de netezzadectreenode	**Valores**	**Descripción de la propiedad**
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	El valor predeterminado es utilizar allTrainingData para calcular la precisión del modelo. Utilice partitionTrainingData para especificar un porcentaje de datos de prueba por utilizar, o useOtherTable para utilizar un conjunto de datos de prueba desde una tabla específica de la base de datos.
perc_training_data	*number*	Si prune_tree_options se establece en partitionTrainingData, especifica el porcentaje de datos que se utilizará para entrenamiento.
prune_seed	*integer*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis si prune_tree_options se establece en partitionTrainingData; el valor predeterminado es 1.
pruning_table	*serie*	Nombre de tabla de un conjunto de datos de poda separado para estimar la precisión del modelo.
compute_probabilities	*Flag*	Si es verdadero, produce un campo de nivel de confianza (probabilidad) y el campo de predicción.

K-medias de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzakmeansnode.

Tabla 217. Propiedades de netezzakmeansnode		
Propiedades de netezzakmeansnode	**Valores**	**Descripción de la propiedad**
distance_measure	Euclidean Manhattan Canberra maximum	Método que se debe utilizar para medir la distancia entre puntos de datos.
num_clusters	*integer*	Número de clústeres que se deben crear; el valor predeterminado es 3.

Tabla 217. Propiedades de netezzakmeansnode (continuación)		
Propiedades de netezzakmeansnode	**Valores**	**Descripción de la propiedad**
max_iterations	*integer*	Número de iteraciones de algoritmos después de los cuáles se debe detener la prueba del modelo; el valor predeterminado es 5.
rand_seed	*integer*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis; el valor predeterminado es 12345.

Red bayesiana de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzabayesnode.

Tabla 218. Propiedades de netezzabayesnode		
Propiedades de netezzabayesnode	**Valores**	**Descripción de la propiedad**
base_index	*integer*	Identificador numérico asignado al primer campo de entrada de gestión interna; el valor predeterminado es 777.
sample_size	*integer*	Tamaño de la muestra que se tomará si el número de atributos es muy grande; el valor predeterminado es 10.000.
display_additional_information	*Flag*	Si es verdadero, muestra información adicional sobre el progreso en un cuadro de diálogo de mensaje.
type_of_prediction	best neighbors nn-neighbors	Tipo de algoritmo de predicción que se utilizará: el mejor (vecino más correlacionado), vecinos (predicción ponderada de vecinos), o vecinos NN (vecinos no nulos).

Bayesiano ingenuo de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzanaivebayesnode.

Tabla 219. Propiedades de netezzanaivebayesnode		
Propiedades de netezzanaivebayesnode	**Valores**	**Descripción de la propiedad**
compute_probabilities	*Flag*	Si es verdadero, produce un campo de nivel de confianza (probabilidad) y el campo de predicción.
use_m_estimation	*Flag*	Si es verdadero, utiliza la técnica m-estimation para evitar probabilidades de cero durante el cálculo.

KNN de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzaknnnode.

Tabla 220. Propiedades de netezzaknnnode		
Propiedades de netezzaknnnode	**Valores**	**Descripción de la propiedad**
weights	*structured*	Propiedad estructurada que se utiliza para asignar ponderaciones a clases individuales. Ejemplo: `set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]`
distance_measure	Euclidean Manhattan Canberra Maximum	Método que se utiliza para medir la distancia entre puntos de datos.
num_nearest_neighbors	*integer*	Número de vecinos más próximos de un caso concreto; el valor predeterminado es 3.
standardize_measurements	*Flag*	Si es verdadero, estandariza las mediciones de campos de entrada continuos antes de calcular los valores de distancia.
use_coresets	*Flag*	Si es verdadero, utiliza el muestreo del conjunto principal para acelerar el cálculo de conjuntos de datos grandes.

Clúster divisivo de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzadivclusternode.

Tabla 221. Propiedades de netezzadivclusternode		
Propiedades de netezzadivclusternode	**Valores**	**Descripción de la propiedad**
distance_measure	Euclidean Manhattan Canberra Maximum	Método que se utiliza para medir la distancia entre puntos de datos.
max_iterations	*integer*	Número máximo de iteraciones de algoritmo que se ejecutarán antes de detener el entrenamiento del modelo; el valor predeterminado es 5.
max_tree_depth	*integer*	El número máximo de niveles en los que se puede subdividir el conjunto de datos; el valor predeterminado es 3.
rand_seed	*integer*	Semilla aleatoria, se utiliza para replicar los análisis; el valor predeterminado es 12345.
min_instances_split	*integer*	El número mínimo de registros que se pueden dividir, el valor predeterminado es 5.

Tabla 221. Propiedades de netezzadivclusternode (continuación)		
Propiedades de netezzadivclusternode	**Valores**	**Descripción de la propiedad**
level	*integer*	El nivel de jerarquía en el que se guardan los registros; el valor predeterminado es -1.

PCA de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzapcanode.

Tabla 222. Propiedades de netezzapcanode		
Propiedades de netezzapcanode	**Valores**	**Descripción de la propiedad**
center_data	*Flag*	Si es verdadero (opción predeterminada), ejecuta el centrado de datos (también conocido como "sustracción de media") antes del análisis.
perform_data_scaling	*Flag*	Si es verdadero, ejecuta la adaptación de los datos antes del análisis. De esta forma el análisis será menos arbitrario si las diferentes variables se miden en unidades diferentes.
force_eigensolve	*Flag*	Si es verdadero, utiliza un método menos preciso, pero más rápido para encontrar componentes principales.
pc_number	*integer*	El número principal de componentes al que se reducirá el conjunto de datos; el valor predeterminado es 1.

Árbol de regresión de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzaregtreenode.

Tabla 223. Propiedades de netezzaregtreenode		
Propiedades de netezzaregtreenode	**Valores**	**Descripción de la propiedad**
max_tree_depth	*integer*	Número máximo de niveles que puede crecer el árbol por debajo del nodo raíz; el valor predeterminado es 10.
split_evaluation_measure	Variance	Medida de clase de impureza, se utiliza para evaluar la mejor ubicación para dividir el árbol; el valor predeterminado (y la única opción actualmente) es Variance.
min_improvement_splits	*number*	Cantidad mínima para reducir la impureza antes de que se cree la nueva división en el árbol.
min_instances_split	*integer*	El número mínimo de registros que se pueden dividir.

Tabla 223. Propiedades de netezzaregtreenode (continuación)		
Propiedades de netezzaregtreenode	**Valores**	**Descripción de la propiedad**
pruning_measure	mse r2 pearson spearman	Método que se utilizará para la poda.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	El valor predeterminado es utilizar allTrainingData para calcular la precisión del modelo. Utilice partitionTrainingData para especificar un porcentaje de datos de prueba por utilizar, o useOtherTable para utilizar un conjunto de datos de prueba desde una tabla específica de la base de datos.
perc_training_data	*number*	Si prune_tree_options se establece en PercTrainingData, especifica el porcentaje de datos que se utilizará para entrenamiento.
prune_seed	*integer*	Semilla aleatoria que se debe utilizar para replicar los resultados del análisis si prune_tree_options se establece en PercTrainingData; el valor predeterminado es 1.
pruning_table	*serie*	Nombre de tabla de un conjunto de datos de poda separado para estimar la precisión del modelo.
compute_probabilities	*Flag*	Si es verdadero, especifica que las varianzas de las clases asignadas se deben incluir en el resultado.

Regresión lineal de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzalineregressionnode.

Tabla 224. Propiedades de netezzalineregressionnode		
Propiedades de netezzalineregressionnode	**Valores**	**Descripción de la propiedad**
use_svd	*Flag*	Si es verdadero, utiliza la matriz de descomposición de valores singulares en lugar de la matriz original, para mayor velocidad y precisión numérica.
include_intercept	*Flag*	Si es verdadero (valor predeterminado), aumenta la precisión global de la solución.
calculate_model_diagnostics	*Flag*	Si es verdadero, calcula el diagnóstico del modelo.

Series temporales de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzatimeseriesnode.

Tabla 225. Propiedades de netezzatimeseriesnode		
Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
time_points	*campo*	El campo de entrada que contiene los valores de fecha u hora de la serie temporal.
time_series_ids	*campo*	El campo de entrada que contiene diversos ID de series temporales; utilice esta opción si la entrada contiene más de una serie temporal.
model_table	*campo*	Nombre de la tabla de base de datos en la que se guardará el modelo de series temporales de Netezza.
description_table	*campo*	Nombre de la tabla de entrada que contiene los nombres y las descripciones de las series temporales.
seasonal_adjustment_table	*campo*	Nombre de la tabla de salida en la que se guardarán los valores ajustados calculados por los algoritmos de suavizado exponencial o de descomposición de tendencia estacional.

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)		
Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
algorithm_name	SpectralAnalysis o spectral ExponentialSmoothing o esmoothing ARIMA SeasonalTrendDecomposition o std	Algoritmo que hay que utilizar para el modelado de series temporales.
trend_name	N A DA M DM	Tipo de tendencia del suavizado exponencial: N - none A - aditivo DA -aditivo amortiguado M - multiplicativo DM - multiplicativo amortiguado
seasonality_type	N A M	Tipo de estacionalidad del suavizado exponencial: N - none A - aditivo M - multiplicativo
interpolation_method	linear cubicspline exponentialspline	Método de interpolación que hay que utilizar.

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
timerange_setting	SD SP	Valor de rango de tiempo que se debe utilizar: SD - determinado por el sistema (utiliza el rango completo de datos de series de tiempo) SP - especificado por el usuario mediante earliest_time y latest_time

earliest_time
 	
latest_time
	integer
	
	Date
	
	Tiempo
	
	Indicación de fecha y hora
		Valores de inicio y finalización si timerange_setting es SP.
		
		El formato debe seguir el valor time_points.
		
		Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha.
		
		Ejemplo:
		
		`set NZ_DT1.timerange_setting = 'SP'`
		
		`set NZ_DT1.earliest_time = '1921-01-01'`
		
		`set NZ_DT1.latest_time = '2121-01-01'`

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
arima_setting	SD SP	Valor para el algoritmo ARIMA (sólo se utiliza si algorithm_name se establece en ARIMA): SD - determinado por el sistema SP - especificado por el usuario Si arima_setting = SP, utilice los parámetros siguientes para establecer los valores estacionales y no estacionales. Ejemplo (solo no estacionales): set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol d_symbol q_symbol sp_symbol sd_symbol sq_symbol	less eq lesseq	ARIMA - operador para los parámetros p, d, q, sp, sd y sq: less - menor que eq - igual a lesseq - menor o igual que
p	*integer*	ARIMA: grados no estacionales de autocorrelación.

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
q	*integer*	ARIMA: valor de derivación no estacional.
d	*integer*	ARIMA: número no estacional de órdenes de media móvil presentes en el modelo.
sp	*integer*	ARIMA: grados estacionales de autocorrelación.
sq	*integer*	ARIMA: valor de derivación estacional.
sd	*integer*	ARIMA: número estacional de órdenes de media móvil presentes en el modelo.
advanced_setting	SD SP	Determina cómo se manejan los valores avanzados: SD - determinado por el sistema SP - especificado por el usuario mediante period, units_period y forecast_setting. Ejemplo: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
period	*integer*	Longitud de ciclo estacional, especificado junto con units_period. No aplicable para análisis espectrales.

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)

Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
units_period	ms s min h d wk q y	Unidades en que se expresa period: ms - milisegundos s - segundos min - minutos h - horas d - días wk - semanas q - trimestres y - años Por ejemplo, para una serie temporal semanal utilice 1 para period y wk para units_period.
forecast_setting	forecasthorizon forecasttimes	Especifica cómo se han de realizar las previsiones.
forecast_horizon	*integer* *Date* *Tiempo* *Indicación de fecha y hora*	Si forecast_setting = forecasthorizon, especifica el valor de punto final para la previsión. El formato debe seguir el valor time_points. Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha.

Tabla 225. Propiedades de netezzatimeseriesnode (continuación)		
Propiedades de netezzatimeseriesnode	**Valores**	**Descripción de la propiedad**
forecast_times	*integer* *Date* *Tiempo* *Indicación de fecha y hora*	Si forecast_setting = forecasttimes, especifica los valores que se utilizarán para hacer previsiones. El formato debe seguir el valor time_points. Por ejemplo, si el campo time_points contiene una fecha, éste también debería ser una fecha.
include_history	*Flag*	Indica si se deben incluir los valores históricos en los resultados.
include_interpolated_values	*Flag*	Indica si se deben incluir los valores interpolados en los resultados. No es aplicable si include_history es false.

Lineal generalizado de Netezza

Las siguientes propiedades están disponibles para los nodos del tipo netezzaglmnode.

Tabla 226. Propiedades de netezzaglmnode		
Propiedades de netezzaglmnode	**Valores**	**Descripción de la propiedad**
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	El tipo de distribución. El valor predeterminado es bernoulli.
dist_params	*number*	Valor del parámetro de distribución que hay que utilizar. Sólo se aplica si distribution es Negativebinomial.

Tabla 226. Propiedades de netezzaglmnode (continuación)

Propiedades de netezzaglmnode	**Valores**	**Descripción de la propiedad**
`trials`	*integer*	Sólo se aplica si `distribution` es `Binomial`. Cuando la respuesta objetivo es un número de eventos que tienen lugar en un conjunto de ensayos, el campo `target` contiene el número de eventos, y el campo `trials` contiene el número de ensayos.
`model_table`	*campo*	Nombre de la tabla de base de datos en la que se guardará el modelo lineal generalizado de Netezza.
`maxit`	*integer*	Número máximo de iteraciones que debe ejecutar el algoritmo; el valor predeterminado es 20.
`eps`	*number*	Valor del error máximo (en notación científica) en el que el algoritmo debería dejar de buscar el modelo de mejor ajuste. El valor predeterminado es -3, lo que significa 1E-3 o 0,001.
`tol`	*number*	El valor (en notación científica) por debajo del que los errores se tratan como si su valor fuera cero. El valor predeterminado es -7, lo que significa que los valores de error por debajo de 1E-7 (o 0,0000001) se cuentan como insignificantes.

Tabla 226. Propiedades de netezzaglmnode (continuación)

Propiedades de netezzaglmnode	**Valores**	**Descripción de la propiedad**
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Función de enlace que se ha de utilizar; el valor predeterminado es logit.
link_params	*number*	Valor del parámetro de función de enlace que hay que utilizar. Sólo se aplica si link_function es power u oddspower.

Tabla 226. Propiedades de netezzaglmnode (continuación)		
Propiedades de netezzaglmnode	**Valores**	**Descripción de la propiedad**
interaction	[[[*colnames1*],[*levels1*]], [[*colnames2*],[*levels2*]], ...,[[*colnamesN*],[*nivelesN*]],]	Especifica las interacciones entre los campos. *colnames* es una lista de campos de entrada, y *level* es siempre 0 para cada campo. Ejemplo: `[[["K","BP","Sex","K"], [0,0,0,0]], [["Age","Na"],[0,0]]]`
intercept	*Flag*	Si es true, incluye la interceptación en el modelo.

Propiedades de nugget de modelo de Netezza

Las siguientes propiedades son comunes a los nuggets del modelo de la base de datos de Netezza.

Tabla 227. Propiedades comunes de nugget de nodos de Netezza		
Propiedades comunes de nugget de modelo de Netezza	**Valores**	**Descripción de la propiedad**
connection	*serie*	La cadena de conexión para la base de datos de Netezza donde se almacena el modelo.
table_name	*serie*	Nombre de la base de datos donde se almacenará el modelo.

Otras las propiedades del nugget de modelo son las mismas que las del nodo de modelado correspondiente.

Los nombres de script de los nuggets de modelo son los siguientes.

Tabla 228. Nombres de script de nuggets de modelos Netezza	
Nugget de modelo	**Nombre de script**
Árbol de decisiones	applynetezzadectreenode
K-medias	applynetezzakmeansnode
Red bayesiana	applynetezzabayesnode
Bayesiano ingenuo	applynetezzanaivebayesnode
KNN	applynetezzaknnnode
Clúster divisivo	applynetezzadivclusternode
PCA	applynetezzapcanode
Árbol de regresión	applynetezzaregtreenode
Regresión lineal	applynetezzalineregressionnode
Serie temporal	applynetezzatimeseriesnode

Tabla 228. Nombres de script de nuggets de modelos Netezza (continuación)

Nugget de modelo	**Nombre de script**
Lineal generalizado	`applynetezzaglmnode`

Capítulo 16. Propiedades del nodo de resultados

Las propiedades de nodos de resultados se diferencian un poco de las de otros tipos de nodos. En lugar de hacer referencia a una opción determinada de nodo, las propiedades de nodos de resultados almacenan una referencia en el objeto de resultado. Esto resulta útil al tomar un valor de una tabla y establecerlo como un parámetro de ruta.

Esta sección describe las propiedades de scripts disponibles para los nodos de resultados.

propiedades de analysisnode

El nodo Análisis evalúa la capacidad de los modelos predictivos para generar predicciones precisas. Los nodos Análisis realizan varias comparaciones entre los valores pronosticados y los valores reales para uno o más nugget de modelo. También pueden comparar modelos predictivos entre sí.

Ejemplo

```
node = stream.create("analysis", "My node")
# "Analysis" tab
node.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# "Define User Measure..."
node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# "Output" tab
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Tabla 229. propiedades de analysisnode

Propiedad de analysisnode	**Tipo de datos**	**Descripción de la propiedad**
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.

Tabla 229. propiedades de analysisnode (continuación)		
Propiedad de analysisnode	**Tipo de datos**	**Descripción de la propiedad**
by_fields	*lista*	
full_filename	*serie*	Nombre del archivo de resultados, si se trata de resultados HTML, de datos o de disco.
coincidence	*Flag*	
performance	*Flag*	
evaluation_binary	*Flag*	
confidence	*Flag*	
threshold	*number*	
improve_accuracy	*number*	
field_detection_method	Metadata Name	Determina cómo se establece la correspondencia entre los campos predichos y el campo objetivo original. Especifique Metadata o Name.
inc_user_measure	*Flag*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	
user_compute	[Mean Sum Min Max SDev]	

propiedades de dataauditnode

El nodo Auditoría de datos permite echar un primer vistazo exhaustivo a los datos, incluyendo estadísticos de resumen, histogramas y distribución para cada campo, así como información sobre valores atípicos, valores perdidos y extremos. Los resultados se muestran en una matriz fácil de leer que se puede ordenar y utilizar para generar gráficos de tamaño completo y nodos de preparación de datos.

Ejemplo

```
filenode = stream.createAt("variablefile", "File", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("dataaudit", "My node", 196, 100)
stream.link(filenode, node)
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("fields", ["Age", "Na", "K"])
node.setPropertyValue("display_graphs", True)
node.setPropertyValue("basic_stats", True)
node.setPropertyValue("advanced_stats", True)
node.setPropertyValue("median_stats", False)
node.setPropertyValue("calculate", ["Count", "Breakdown"])
node.setPropertyValue("outlier_detection_method", "std")
node.setPropertyValue("outlier_detection_std_outlier", 1.0)
```

```
node.setPropertyValue("outlier_detection_std_extreme", 3.0)
node.setPropertyValue("output_mode", "Screen")
```

Tabla 230. propiedades de dataauditnode

Propiedad de dataauditnode	Tipo de datos	Descripción de la propiedad
custom_fields	*Flag*	
fields	*[campo1 ... campoN]*	
overlay	*campo*	
display_graphs	*Flag*	Se utiliza para activar o desactivar la representación de gráficos en la matriz de resultados.
basic_stats	*Flag*	
advanced_stats	*Flag*	
median_stats	*Flag*	
calculate	Count Breakdown	Se utiliza para calcular valores perdidos. Seleccione uno de los métodos de cálculo, ambos o ninguno.
outlier_detection_method	std iqr	Se utiliza para especificar el método de detección de valores atípicos y extremos.
outlier_detection_std_outlier	*number*	Si outlier_detection_method es std, especifica el número que se utilizará para definir los valores atípicos.
outlier_detection_std_extreme	*number*	Si outlier_detection_method es std, especifica el número que se utilizará para definir los valores extremos.
outlier_detection_iqr_outlier	*number*	Si outlier_detection_method es iqr, especifica el número que se utilizará para definir los valores atípicos.
outlier_detection_iqr_extreme	*number*	Si outlier_detection_method es iqr, especifica el número que se utilizará para definir los valores extremos.
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.

Tabla 230. propiedades de dataauditnode (continuación)		
Propiedad de dataauditnode	**Tipo de datos**	**Descripción de la propiedad**
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
paginate_output	*Flag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*number*	Si se usa con paginate_output, especifica las líneas por página del resultado.
full_filename	*serie*	

Propiedades de extensionoutputnode

El nodo de Resultados de extensión le permite analizar los datos y los resultados de la puntuación de modelo utilizando su propio script R o Python para Spark personalizado. El resultado del análisis puede ser en texto o en gráficos. El resultado se añade a la pestaña **Resultado** del panel de administrador o también puede redireccionar el resultado a un archivo.

Ejemplo de Python para Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_output", "extension_output")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import json
import spss.pyspark.runtime

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]
print df
"""

node.setPropertyValue("python_syntax", python_script)
```

Ejemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

Tabla 231. Propiedades de extensionoutputnode

Propiedades de extensionoutputnode	Tipo de datos	Descripción de la propiedad
syntax_type	*E* *Python*	Especifica qué script se ejecuta: R o Python (R es el valor predeterminado).
r_syntax	*serie*	Sintaxis de scripts R para la puntuación del modelo.
python_syntax	*serie*	Sintaxis de scripts Python para la puntuación de modelo.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opción para convertir campos de distintivos.
convert_missing	*Flag*	Opción para convertir valores perdidos en el valor R `NA`.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.
output_to	`Screen` `File`	Especifique el tipo de salida (`Screen` o `File`).
output_type	`Graph` `Text`	Especifique si se debe producir salida gráfica o de texto.
full_filename	*serie*	Nombre de archivo a utilizar para la salida generada.
graph_file_type	`HTML` `COU`	Tipo de archivo para el archivo de salida (`.html` o `.cou`).

text_file_type	`HTML`
	`TEXT`
	`COU`
		Especifique el tipo de archivo para la salida de texto (`.html`, `.txt` o `.cou`).

Propiedades kdeexport

La estimación de densidad de kernel (KDE)© utiliza los algoritmos del árbol de métrica o del árbol KD para unas consultas eficientes y combina los conceptos de aprendizaje no supervisado, ingeniería de características y modelado de datos. Los enfoques basados en vecinos como, por ejemplo, KDE, son algunas de las técnicas de estimación de densidad más populares y útiles. Los nodos de modelado de KDE y de simulación de KDE en SPSS Modeler exponen las características principales y los parámetros utilizados con más frecuencia de la biblioteca KDE. Los nodos se implementan en Python.

Tabla 232. Propiedades kdeexport

Propiedades kdeexport	Tipo de datos	Descripción de la propiedad
`bandwidth`	*DOUBLE*	El valor predeterminado es 1.
`kernel`	*serie*	El kernel para utilizar: `gaussian` o `tophat`. El valor predeterminado es `gaussian`.
`algorithm`	*serie*	El algoritmo de árbol para utilizar: `kd_tree`, `ball_tree` o `auto`. El valor predeterminado es `auto`.
`metric`	*serie*	La métrica para utilizar al calcular distancias. Para el algoritmo `kd_tree`, elija entre: `Euclidean`, `Chebyshev`, `Cityblock`, `Minkowski`, `Manhattan`, `Infinity`, `P`, `L2` o `L1`. Para el algoritmo `ball_tree`, elija entre: `Euclidian`, `Braycurtis`, `Chebyshev`, `Canberra`, `Cityblock`, `Dice`, `Hamming`, `Infinity`, `Jaccard`, `L1`, `L2`, `Minkowski`, `Matching`, `Manhattan`, `P`, `Rogersanimoto`, `Russellrao`, `Sokalmichener`, `Sokalsneath` o `Kulsinski`. El valor predeterminado es `Euclidean`.
`atol`	*Float*	La tolerancia absoluta deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es `0.0`.
`rtol`	*Float*	La tolerancia relativa deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es `1E-8`.
`breadthFirst`	*Boolean*	Establezca en `True` para utilizar un enfoque de amplitud. Establezca en `False` para utilizar un enfoque de búsqueda en profundidad. El valor predeterminado es `True`.

Tabla 232. Propiedades kdeexport (continuación)		
Propiedades kdeexport	**Tipo de datos**	**Descripción de la propiedad**
LeafSize	*integer*	El tamaño de hoja del árbol subyacente. El valor predeterminado es 40. Cambiar este valor podría afectar de forma significativa al rendimiento.
pValue	*DOUBLE*	Especifique el Valor P para utilizar si está utilizando Minkowski para la métrica. El valor predeterminado es 1.5.

propiedades de matrixnode

El nodo Matriz crea una tabla que muestra las relaciones entre campos. Se suele utilizar para mostrar la relación entre dos campos simbólicos, pero también puede mostrar relaciones entre campos de bandera o campos numéricos.

Ejemplo

```
node = stream.create("matrix", "My node")
# "Settings" tab
node.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
# "Appearance" tab
node.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# "Output" tab
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabla 233. propiedades de matrixnode		
Propiedades de matrixnode	**Tipo de datos**	**Descripción de la propiedad**
fields	Selected Flags Numerics	
row	*campo*	
column	*campo*	

Tabla 233. propiedades de matrixnode (continuación)

Propiedades de matrixnode	Tipo de datos	Descripción de la propiedad
include_missing_values	*Flag*	Determina si los valores perdidos por el sistema (nulos) o no especificados por el usuario (vacíos) se incluyen en los resultados de fila y columna.
cell_contents	CrossTabs Function	
function_field	*serie*	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Ascending Descending	
highlight_top	*number*	Si no es cero, es verdadero.
highlight_bottom	*number*	Si no es cero, es verdadero.
display	[Counts Expected Residuals RowPct ColumnPct TotalPct]	
include_totals	*Flag*	
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.

Tabla 233. propiedades de matrixnode (continuación)		
Propiedades de matrixnode	**Tipo de datos**	**Descripción de la propiedad**
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado. Los formatos Formatted y Delimited aceptan el modificador transposed, mediante el cual se transponen filas y columnas en la tabla.
paginate_output	*Flag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*number*	Si se usa con paginate_output, especifica las líneas por página del resultado.
full_filename	*serie*	

propiedades de meansnode

El nodo Medias compara las medias de grupos independientes o de pares de campos relacionados para probar si existen diferencias significativas. Por ejemplo, puede comparar los ingresos medios antes y después de poner en marcha una promoción o comparar los ingresos de los clientes que no recibieron esa promoción con los que sí lo hicieron.

Ejemplo

```
node = stream.create("means", "My node")
node.setPropertyValue("means_mode", "BetweenFields")
node.setPropertyValue("paired_fields", [["OPEN_BAL", "CURR_BAL"]])
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("output_view", "Advanced")
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/means_output.html")
```

Tabla 234. propiedades de meansnode		
Propiedad de meansnode	**Tipo de datos**	**Descripción de la propiedad**
means_mode	BetweenGroups BetweenFields	Especifica el tipo de estadístico de las medias que se va a ejecutar en los datos.
test_fields	[field1 ... fieldn]	Especifica el campo de prueba si means_mode se establece en BetweenGroups.

Tabla 234. propiedades de meansnode (continuación)

Propiedad de meansnode	Tipo de datos	Descripción de la propiedad
grouping_field	*campo*	Especifica el campo de agrupación.
paired_fields	[[field1 field2] [field3 field4] ...]	Especifica los pares de campos que se usan si means_mode se establece en BetweenFields.
label_correlations	*Flag*	Determina si las etiquetas de correlación se muestran en el resultado. Esta configuración se aplica únicamente si means_mode se establece en BetweenFields.
correlation_mode	Probability Absolute	Determina si las correlaciones deben etiquetarse según la probabilidad o según el valor absoluto.
weak_label	*serie*	
medium_label	*serie*	
strong_label	*serie*	
weak_below_probability	*number*	Si correlation_mode se establece en Probability, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_probability	*number*	Valor de corte para correlaciones fuertes.
weak_below_absolute	*number*	Si correlation_mode se establece en Absolute, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_absolute	*number*	Valor de corte para correlaciones fuertes.
unimportant_label	*serie*	
marginal_label	*serie*	
important_label	*serie*	

Tabla 234. propiedades de meansnode (continuación)		
Propiedad de meansnode	**Tipo de datos**	**Descripción de la propiedad**
unimportant_below	*number*	Valor de corte para una importancia del campo baja. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
important_above	*number*	
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Nombre que se va a usar.
output_mode	Screen File	Especifica la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Especifica el tipo de resultado.
full_filename	*serie*	
output_view	Simple Advanced	Determina si el resultado muestra la vista simple o la avanzada.

propiedades de reportnode

El nodo Informe crea informes con formato que contienen texto fijo, así como datos y otras expresiones derivadas de los datos. Puede especificar el formato del informe utilizando plantillas de texto para definir el texto fijo y las construcciones de resultados de datos. Puede proporcionar formato de texto personalizado utilizando etiquetas HTML de la plantilla y configurando opciones en la pestaña Resultado. Puede incluir valores de datos y otra salida condicional utilizando expresiones CLEM en la plantilla.

Ejemplo

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

Tabla 235. propiedades de reportnode		
Propiedad de reportnode	**Tipo de datos**	**Descripción de la propiedad**
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Se utiliza para especificar el tipo de salida de archivo.
format	Auto Custom	Se utiliza para elegir si la salida se formatea automáticamente o si se formatea utilizando el HTML incluido en la plantilla. Para utilizar el formato HTML en la plantilla, especifique Custom.
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
text	*serie*	
full_filename	*serie*	
highlights	*Flag*	
title	*serie*	
lines_per_page	*number*	

propiedades de routputnode

El nodo Routput permite analizar datos y resultados de la puntuación del modelo utilizando su propio script R personalizado. El resultado del análisis puede ser en texto o en gráficos. El resultado se añade a la pestaña **Resultado** del panel de administrador o también puede redireccionar el resultado a un archivo.

Tabla 236. propiedades de routputnode		
Propiedades de routputnode	**Tipo de datos**	**Descripción de la propiedad**
syntax	*serie*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*Flag*	

Tabla 236. propiedades de routputnode (continuación)		
Propiedades de routputnode	**Tipo de datos**	**Descripción de la propiedad**
convert_datetime_class	`POSIXct` `POSIXlt`	
convert_missing	*Flag*	
output_name	`Auto` `Custom`	
custom_name	*serie*	
output_to	`Screen` `File`	
output_type	`Graph` `Text`	
full_filename	*serie*	
graph_file_type	`HTML` `COU`	

text_file_type	`HTML`
	`TEXT`
	`COU`	

propiedades de setglobalsnode

El nodo Val. globales explora los datos y calcula los valores de resumen que se pueden utilizar en expresiones CLEM. Por ejemplo, puede utilizar este nodo para calcular estadísticas para un campo denominado *edad* y, a continuación, utilizar la media general de *edad* en expresiones CLEM insertando la función `@GLOBAL_MEAN(age)`.

Ejemplo

```
node = stream.create("setglobals", "My node")
node.setKeyedPropertyValue("globals", "Na", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "K", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
node.setPropertyValue("clear_first", False)
node.setPropertyValue("show_preview", True)
```

Tabla 237. propiedades de setglobalsnode		
Propiedad de setglobalsnode	**Tipo de datos**	**Descripción de la propiedad**
globals	[Sum Mean Min Max SDev]	Propiedad estructurada en la que los campos que se van a establecer deben hacer referencia a la sintaxis siguiente: node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	*Flag*	
show_preview	*Flag*	

propiedades de simevalnode

El nodo Evaluación de simulación evalúa un campo de destino especificado predicho, y presenta la información de correlación y distribución sobre el campo de destino.

Tabla 238. propiedades de simevalnode		
propiedades de simevalnode	**Tipo de datos**	**Descripción de la propiedad**
target	*campo*	
iteration	*campo*	
presorted_by_iteration	*Boolean*	
max_iterations	*number*	
tornado_fields	*[field1...fieldN]*	
plot_pdf	*Boolean*	
plot_cdf	*Boolean*	
show_ref_mean	*Boolean*	
show_ref_median	*Boolean*	
show_ref_sigma	*Boolean*	
num_ref_sigma	*number*	
show_ref_pct	*Boolean*	
ref_pct_bottom	*number*	
ref_pct_top	*number*	
show_ref_custom	*Boolean*	
ref_custom_values	*[number1...numberN]*	

Tabla 238. propiedades de simevalnode (continuación)		
propiedades de simevalnode	**Tipo de datos**	**Descripción de la propiedad**
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	*Boolean*	
pct_table	Quartiles Intervals Custom	
pct_intervals_num	*number*	
pct_custom_values	*[number1...numberN]*	

propiedades de simfitnode

El nodo Ajustar simulación examina la distribución estadística de los datos en cada campo y genera (o actualiza) un nodo Generar simulación, asignando a cada campo la mejor distribución de ajuste. El nodo Generar simulación podrá entonces utilizarse para generar datos simulados.

Tabla 239. propiedades de simfitnode		
propiedades de simfitnode	**Tipo de datos**	**Descripción de la propiedad**
build	Node XMLExport Both	
use_source_node_name	*Boolean*	
source_node_name	*serie*	El nombre personalizado del nodo de origen que se está generando o actualizando.
use_cases	All LimitFirstN	
use_case_limit	*integer*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*integer*	
parameter_xml_filename	*serie*	
generate_parameter_import	*Boolean*	

propiedades de statisticsnode

El nodo Estadísticos ofrece información básica de resumen acerca de los campos numéricos. Calcula estadísticos de resumen para campos individuales y correlaciones entre campos.

Ejemplo

```
node = stream.create("statistics", "My node")
# "Settings" tab
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
node.setPropertyValue("strong_label", "upper quartile")
# "Output" tab
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Tabla 240. propiedades de statisticsnode

Propiedad de statisticsnode	**Tipo de datos**	**Descripción de la propiedad**
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
full_filename	*serie*	
examine	*lista*	
correlate	*lista*	
statistics	[count mean sum min max range variance sdev semean median mode]	

Tabla 240. propiedades de statisticsnode (continuación)		
Propiedad de statisticsnode	**Tipo de datos**	**Descripción de la propiedad**
correlation_mode	Probability Absolute	Determina si las correlaciones deben etiquetarse según la probabilidad o según el valor absoluto.
label_correlations	*Flag*	
weak_label	*serie*	
medium_label	*serie*	
strong_label	*serie*	
weak_below_probability	*number*	Si correlation_mode se establece en Probability, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_probability	*number*	Valor de corte para correlaciones fuertes.
weak_below_absolute	*number*	Si correlation_mode se establece en Absolute, determina el valor de corte para las correlaciones débiles. Debe tratarse de un valor comprendido entre 0 y 1; por ejemplo, 0,90.
strong_above_absolute	*number*	Valor de corte para correlaciones fuertes.

Propiedades de statisticsoutputnode

El nodo Resultados de Statistics le permite llamar a un procedimiento de IBM SPSS Statistics para analizar los datos de IBM SPSS Modeler. Se puede acceder a una gran variedad de procedimientos analíticos de IBM SPSS Statistics. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Las propiedades de este nodo están descritas en “Propiedades de statisticsoutputnode” en la página 437.

propiedades de tablenode

El nodo Tabla muestra los datos en formato de tabla, que también se puede escribir en un archivo. Esto es útil en cualquier momento en que necesite inspeccionar sus valores de datos o exportarlos en un formato fácilmente legible.

Ejemplo

```
node = stream.create("table", "My node")
node.setPropertyValue("highlight_expr", "Age > 30")
node.setPropertyValue("output_format", "HTML")
```

```
node.setPropertyValue("transpose_data", True)
node.setPropertyValue("full_filename", "C:/output/table_output.htm")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabla 241. propiedades de tablenode

Propiedad de tablenode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	Nombre del archivo de resultados, si se trata de resultados HTML, de datos o de disco.
use_output_name	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
output_name	*serie*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.
output_mode	Screen File	Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Se utiliza para especificar el tipo de resultado.
transpose_data	*Flag*	Transpone los datos antes de exportarlos de manera que las filas representan campos y las columnas, registros.
paginate_output	*Flag*	Si output_format es HTML, los resultados se separarán por páginas.
lines_per_page	*number*	Si se usa con paginate_output, especifica las líneas por página del resultado.
highlight_expr	*serie*	
output	*serie*	Propiedad de sólo lectura que mantiene una referencia en la última tabla creada por el nodo.
value_labels	*[[Valor CadenaEtiqueta]* *[Valor CadenaEtiqueta] ...]*	Se utiliza para especificar etiquetas para los pares de valores.
display_places	*integer*	Establece el número de cifras decimales para el campo cuando se muestra (sólo se aplica a campos con almacenamiento REAL). Un valor de -1 utilizará el valor predeterminado de la ruta.

Tabla 241. propiedades de tablenode (continuación)		
Propiedad de tablenode	**Tipo de datos**	**Descripción de la propiedad**
export_places	*integer*	Establece el número de cifras decimales para el campo cuando se exporta (sólo se aplica a campos con almacenamiento REAL). Un valor de -1 utilizará el valor predeterminado de la ruta.
decimal_separator	DEFAULT PERIOD COMMA	Establece el separador decimal para el campo (sólo se aplica a campos con almacenamiento REAL).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Establece el formato de fecha del campo (sólo se aplica a los campos con almacenamiento DATE o TIMESTAMP).

Tabla 241. propiedades de tablenode (continuación)		
Propiedad de tablenode	**Tipo de datos**	**Descripción de la propiedad**
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Establece el formato de hora para el campo (sólo se aplica a campos con almacenamiento TIME o TIMESTAMP).
column_width	*integer*	Establece el ancho de columna para el campo. Un valor de -1 establecerá el ancho de columna en Auto.
justify	AUTO CENTER LEFT RIGHT	Establece la justificación de columna para el campo.

propiedades de transformnode

El nodo Transformar permite seleccionar y previsualizar visualmente los resultados de las transformaciones antes de aplicarlas a los campos seleccionados.

Ejemplo

```
node = stream.create("transform", "My node")
node.setPropertyValue("fields", ["AGE", "INCOME"])
node.setPropertyValue("formula", "Select")
node.setPropertyValue("formula_log_n", True)
node.setPropertyValue("formula_log_n_offset", 1)
```

Tabla 242. propiedades de transformnode

propiedades de `transformnode`	**Tipo de datos**	**Descripción de la propiedad**
`fields`	[*field1... fieldn*]	Campos que se utilizarán en la transformación.

`formula`	`All`
	
	`Select`
		Indica si se deben calcular todas las transformaciones o sólo las seleccionadas.

`formula_inverse`	*Flag*	Indica si se debe utilizar la transformación inversa.
`formula_inverse_offset`	*number*	Indica el desplazamiento de los datos que se utilizará en la fórmula. De forma predeterminada es 0, a menos que el usuario especifique un valor.
`formula_log_n`	*Flag*	Indica si debe utilizarse la transformación $\log_n$.
`formula_log_n_offset`	*number*	
`formula_log_10`	*Flag*	Indica si debe utilizarse la transformación $\log_{10}$.
`formula_log_10_offset`	*number*	
`formula_exponential`	*Flag*	Indica si se debe utilizar la transformación exponencial (e^x).
`formula_square_root`	*Flag*	Indica si se debe utilizar la transformación de raíz cuadrada.
`use_output_name`	*Flag*	Especifica si se utiliza un nombre de resultado personalizado.
`output_name`	*string*	Si use_output_name es verdadero, especifica el nombre que se va a utilizar.

`output_mode`	`Screen`
	
	`File`
		Se utiliza para especificar la ubicación objetivo para el resultado generado desde el nodo de resultados.

`output_format`	`HTML` (*.html*) `Output` (*.cou*)	Se utiliza para especificar el tipo de resultado.

Tabla 242. propiedades de transformnode (continuación)		
propiedades de `transformnode`	**Tipo de datos**	**Descripción de la propiedad**
`paginate_output`	*Flag*	Si `output_format` es `HTML`, los resultados se separarán por páginas.
`lines_per_page`	*number*	Si se usa con `paginate_output`, especifica las líneas por página del resultado.
`full_filename`	*string*	Indica el nombre de archivo que se utilizará para el resultado de archivo.

Capítulo 17. Propiedades de nodos Exportar

Propiedades de nodos Exportar comunes

Las siguientes propiedades son comunes a todos los nodos de exportación:

Tabla 243. Propiedades comunes de nodos de exportación

Propiedad	Valores	Descripción de la propiedad
publish_path	*serie*	Introduzca el nombre raíz que se utilizará para la imagen publicada y los archivos de parámetros.
publish_metadata	*Flag*	Especifica si un archivo de metadatos se crea y que describe las entradas y los resultados de la imagen y sus modelos de datos.
publish_use_parameters	*Flag*	Especifica si se incluyen parámetros de ruta en el archivo *.par.
publish_parameters	*string list*	Especifica los parámetros que se van a incluir.
execute_mode	export_data publish	Especifica si el nodo se ejecuta sin publicar la ruta, o si la ruta se publica automáticamente cuando se ejecuta el nodo.

Propiedades de asexport

La exportación de Analytic Server permite ejecutar una ruta en el sistema de archivos distribuido de Hadoop (HDFS).

Ejemplo

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false
","","","","",""])
```

Tabla 244. Propiedades de asexport

Propiedades de asexport	Tipo de datos	Descripción de la propiedad
data_source	*serie*	Nombre del origen de datos.
export_mode	*serie*	Especifica si se deben **append** exportar datos al origen de datos existente o a **overwrite** el origen de datos existente.

Tabla 244. Propiedades de asexport (continuación)		
Propiedades de asexport	**Tipo de datos**	**Descripción de la propiedad**
use_default_as	*Boolean*	Si se establece en True, utiliza la conexión Analytic Server predeterminada configurada en el archivo options.cfg del servidor. Si se establece en False, utiliza la conexión de este nodo.
connection	["string","string","string", "string","string","string","string", "string" ,"string","string", "string" ,"string"]	Una propiedad de la lista que contiene los detalles de conexión de Analytic Server. El formato es: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] Donde: is_secure_connect: indica si se utiliza una conexión segura, y es true o false. use-kerberos-auth: indica si se utiliza autenticación Kerberos y es true o false. enable-kerberos-debug: indica si se utiliza el modo de depuración de la autenticación Kerberos y es true o false.

Propiedades del nodo de exportación Cognos

El nodo Exportar IBM Cognos exporta datos en un formato que pueden leer las bases de datos de Cognos.

Para este nodo, debe definir una conexión de Cognos y una conexión ODBC.

Conexión de Cognos

Las propiedades de la conexión de Cognos son las siguientes.

Tabla 245. Propiedades del nodo de exportación Cognos

Propiedades de cognosexportnode	**Tipo de datos**	**Descripción de la propiedad**
`cognos_connection`	*["cadena","distintivo","cadena","cadena", "cadena"]*	Una propiedad de la lista que contiene los detalles de conexión para el servidor de Cognos. El formato es: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` donde: `Cognos_server_URL` es la URL del servidor de Cognos que contiene el origen. `login_mode` indica si se utiliza el inicio de sesión anónimo, y es `true` o `false`; si se establece en `true`, los campos siguientes deben establecerse en `""`. `namespace` especifica el proveedor de autenticación de seguridad utilizado para registrarse en el servidor. `username` y `password` son los utilizados para registrarse en el servidor de Cognos. En lugar de `login_mode`, también hay disponibles las modalidades siguientes: • `anonymousMode`. For example: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. For example: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]`

Tabla 245. Propiedades del nodo de exportación Cognos (continuación)		
Propiedades de cognosexportnode	**Tipo de datos**	**Descripción de la propiedad**
		• storedCredentialMode. For example: ['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"] Donde stored_credential_name es el nombre de una credencial de Cognos del repositorio.
cognos_package_name	*serie*	La ruta y el nombre del paquete de Cognos al que está exportando datos, por ejemplo: /Public Folders/MyPackage
cognos_datasource	*serie*	
cognos_export_mode	Publish ExportFile	
cognos_filename	*serie*	

Conexión ODBC

Las propiedades de la conexión ODBC son idénticas a las indicadas para databaseexportnode en la sección siguiente, a excepción de la propiedad datasource, que no es válida.

propiedades de databaseexportnode

El nodo Exportar base de datos escribe datos en orígenes de datos relacionales compatibles con ODBC. Para poder escribir en un origen de datos ODBC, el origen de datos debe existir y debe tener permiso de escritura para el mismo.

Ejemplo

```
'''
Assumes a datasource named "MyDatasource" has been configured
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tab
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
```

```
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialog
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialog
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP
INDEX <index-name>
 ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id",
"region"]])
```

Tabla 246. propiedades de databaseexportnode

Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
datasource	*serie*	
username	*serie*	
password	*serie*	
epassword	*serie*	Este intervalo es de sólo lectura durante la ejecución. Para generar una contraseña codificada, utilice la herramienta Contraseña disponible del menú Herramientas. Para obtener más información, consulte el tema "Generación de una contraseña codificada" en la página 54.
table_name	*serie*	
write_mode	Create Append Merge	
map	*serie*	Correlaciona un nombre de campo de ruta a un nombre de columna de la base de datos (sólo es válido si write_mode es Merge). Para una fusión, todos los campos se deben estar correlacionados para que se exporten. Los nombres de campos que no existen en la base de datos se añaden como nuevas columnas.

Tabla 246. propiedades de databaseexportnode (continuación)

Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
key_fields	*lista*	Especifica el campo de ruta que se utiliza para la clave; la propiedad map muestra los elementos que se corresponden con la base de datos.
join	Database Add	
drop_existing_table	*Flag*	
delete_existing_rows	*Flag*	
default_string_size	*integer*	
type		Propiedad estructurada que se utiliza para establecer el tipo de esquema.
generate_import	*Flag*	
use_custom_create_table_command	*Flag*	Utilice el intervalo *custom_create_table* para modificar el comando de SQL estándar CREATE TABLE.
custom_create_table_command	*serie*	Especifica el comando de cadena a utilizar en lugar del comando de SQL estándar CREATE TABLE.
use_batch	*Flag*	Las siguientes propiedades son opciones avanzadas para la carga masiva de la base de datos. Un valor verdadero para Use_batch desactiva fila a fila las confirmaciones en la base de datos.
batch_size	*number*	Especifica el número de registros para enviar a la base de datos antes de confirmar en la memoria.
bulk_loading	Off ODBC External	Especifica el tipo de carga masiva. A continuación se muestran las opciones adicionales para ODBC y External.
not_logged	*Flag*	
odbc_binding	Row Column	Especifique el enlace a lo largo de las filas o de las columnas para la carga masiva a través de ODBC.

Tabla 246. propiedades de databaseexportnode (continuación)

Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
loader_delimit_mode	Tab Space Other	Especifique el tipo de delimitador para la carga masiva a través de un programa externo. Seleccione Other junto con la propiedad loader_other_delimiter para especificar los delimitadores, como la coma (,).
loader_other_delimiter	*serie*	
specify_data_file	*Flag*	Una marca verdadera activa la siguiente propiedad data_file, en la que puede especificar el nombre de archivo y la vía de acceso en la que se va a escribir al realizar la carga masiva en la base de datos.
data_file	*serie*	
specify_loader_program	*Flag*	Una marca verdadera activa la siguiente propiedad loader_program, en la que puede especificar el nombre y la ubicación de un programa o script del cargador externo.
loader_program	*serie*	
gen_logfile	*Flag*	Una marca verdadera activa la siguiente propiedad logfile_name, en la que puede especificar el nombre de un archivo en el servidor para generar un registro de errores.
logfile_name	*serie*	
check_table_size	*Flag*	Una marca verdadera permite la comprobación de la tabla para garantizar que el aumento del tamaño de la tabla de la base de datos se corresponde con el número de filas exportadas desde IBM SPSS Modeler.
loader_options	*serie*	Especifique los argumentos adicionales, como -comment y -specialdir, en el programa cargador.
export_db_primarykey	*Flag*	Determina si el campo especificado es una clave primaria.

Tabla 246. propiedades de databaseexportnode (continuación)		
Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
use_custom_create_index_command	*Flag*	Si se establece true, se activa SQL personalizado para todos los índices.
custom_create_index_command	*serie*	Especifica el comando de SQL empleado para crear índices cuando SQL personalizado está activado. (Este valor puede anularse para determinados índices, tal y como se indica a continuación.)
indexes.INDEXNAME.fields		Crea el índice especificado si procede y enumera los nombres de campos que se van a incluir en él.
INDEXNAME "use_custom_create_index_command"	*Flag*	Se usa para activar o desactivar el SQL personalizado para un índice específico. Consulte los ejemplos situados a continuación de la siguiente tabla.
INDEXNAME "custom_create_index_command"	*serie*	Especifica el SQL personalizado que se usa para el índice específico. Consulte los ejemplos situados a continuación de la siguiente tabla.
indexes.INDEXNAME.remove	*Flag*	Si se establece True, se elimina el índice específico del grupo de índices.
table_space	*serie*	Especifica el espacio de tabla que se creará.
use_partition	*Flag*	Especifica que se utilizará el campo Distribuir Hash.
partition_field	*serie*	Especifica el contenido del campo Distribuir Hash.

Nota: Para algunas bases de datos, puede especificar que las tablas de base de datos se creen para la exportación con compresión (por ejemplo, el equivalente de `CREATE TABLE MYTABLE (...) COMPRESS YES;` en SQL). Las propiedades `use_compression` y `compression_mode` se proporcionan para dar soporte a esta característica, como se indica a continuación.

Tabla 247. Propiedades de databaseexportnode utilizando funciones de compresión		
Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
use_compression	*boolean*	Si se establece en True, crea tablas para la exportación con compresión.

Tabla 247. Propiedades de databaseexportnode utilizando funciones de compresión (continuación)		
Propiedades de databaseexportnode	**Tipo de datos**	**Descripción de la propiedad**
compression_mode	Row Page	Establece el nivel de compresión de las bases de datos de SQL Server.
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	Establece el nivel de compresión de las bases de datos de Oracle. Tenga en cuenta que los valores OLTP, Query_High, Query_Low, Archive_High y Archive_Low requieren un mínimo de Oracle 11gR2.

Ejemplo que muestra cómo cambiar el comando CREATE INDEX para un índice específico:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["custom_create_index_command",
 "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

De forma alternativa, esto puede hacerse a través de una tabla hash:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id",
"region"],
 "use_custom_create_index_command":True,
"custom_create_index_command":"CREATE INDEX <index-name> ON
 <table-name> <(index-columns)>"])
```

Propiedades de datacollectionexportnode

El nodo de exportación Data Collection abre los datos en el formato utilizado por el software de investigación de mercados Data Collection. Se debe instalar una biblioteca de datos de Data Collection para utilizar este nodo.

Ejemplo

```
stream = modeler.script.stream()
datacollectionexportnode = stream.createAt("datacollectionexport", "Data
Collection", 200, 200)
```

```
datacollectionexportnode.setPropertyValue("metadata_file", "c:\\museums.mdd")
datacollectionexportnode.setPropertyValue("merge_metadata", "Overwrite")
datacollectionexportnode.setPropertyValue("casedata_file", "c:\
\museumdata.sav")
datacollectionexportnode.setPropertyValue("generate_import", True)
datacollectionexportnode.setPropertyValue("enable_system_variables", True)
```

Tabla 248. propiedades de datacollectionexportnode

propiedades de datacollectionexportnode	**Tipo de datos**	**Descripción de la propiedad**
metadata_file	*serie*	Nombre del archivo de metadatos que se va a exportar.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*Flag*	Especifica si el archivo *.mdd* exportado debe incluir las variables de sistema de Data Collection.
casedata_file	*serie*	Nombre del archivo *.sav* donde se exportan los datos de casos.
generate_import	*Flag*	

Propiedades de excelexportnode

El nodo de exportación de Excel genera datos en Microsoft Excel.Formato de archivo `xlsx` . Si lo desea, puede elegir iniciar automáticamente Excel y abrir el archivo exportado cuando se ejecute el nodo.

Ejemplo

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xlsx")
excelexportnode.setPropertyValue("excel_file_type", "Excel2007")
excelexportnode.setPropertyValue("inc_field_names", True)
excelexportnode.setPropertyValue("inc_labels_as_cell_notes", False)
excelexportnode.setPropertyValue("launch_application", True)
excelexportnode.setPropertyValue("generate_import", True)
```

Tabla 249. Propiedades de excelexportnode

Propiedad de excelexportnode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	
excel_file_type	Excel2007	
export_mode	Create Append	

Tabla 249. Propiedades de excelexportnode (continuación)		
Propiedad de excelexportnode	**Tipo de datos**	**Descripción de la propiedad**
inc_field_names	*Flag*	Determina si los nombres de campos deben incluirse en la primera fila de la hoja de trabajo.
start_cell	*serie*	Especifica la casilla de inicio de la exportación.
worksheet_name	*serie*	Nombre de la hoja de trabajo que se va a escribir.
launch_application	*Flag*	Determina si Excel debe invocarse para el archivo resultante. Tenga en cuenta que deberá especificar la ruta para iniciar Excel en el cuadro de diálogo Aplicaciones de ayuda (menú Herramientas, Aplicaciones de ayuda).
generate_import	*Flag*	Determina si debe crearse un nodo Importar a Excel que leerá el archivo de datos exportado.

Propiedades de extensionexportnode

Con el nodo de Exportación de extensión, puede ejecutar scripts R o Python para Spark para exportar datos.

Ejemplo de Python para Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_export", "extension_export")
node.setPropertyValue("syntax_type", "Python")

python_script = """import spss.pyspark.runtime
from pyspark.sql import SQLContext
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
print df.dtypes[:]
_newDF = df.select("Age","Drug")
print _newDF.dtypes[:]

df.select("Age", "Drug").write.save("c:/data/ageAndDrug.json", format="json")
"""

node.setPropertyValue("python_syntax", python_script)
```

Ejemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

Tabla 250. Propiedades de extensionexportnode

Propiedades de extensionexportnode	Tipo de datos	Descripción de la propiedad
syntax_type	*E* *Python*	Especifica qué script se ejecuta: R o Python (R es el valor predeterminado).
r_syntax	*serie*	Sintaxis de scripts R a ejecutar.
python_syntax	*serie*	Sintaxis de scripts Python a ejecutar.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opción para convertir campos de distintivos.
convert_missing	*Flag*	Opción para convertir valores perdidos en el valor R NA.
convert_datetime	*Flag*	Opción para convertir las variables con los formatos de fecha o de fecha y hora para formatos de fecha/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opciones para especificar a qué formato se convierten las variables con los formatos de fecha o de fecha y hora.

Propiedades jsonexportnode

El nodo de exportación JSON genera datos en formato JSON.

Tabla 251. Propiedades jsonexportnode

Propiedades jsonexportnode	Tipo de datos	Descripción de la propiedad
full_filename	*serie*	El nombre completo del archivo, incluyendo la vía de acceso.
string_format	*Registros* *Valores*	Especifique el formato de la cadena JSON. El valor predeterminado es `records`.
generate_import	*Flag*	Especifica si se debe generar un nodo de importación JSON que leerá el archivo de datos exportado. El valor predeterminado es `False`.

Propiedades de outputfilenode

El nodo Archivo sin formato produce datos en un archivo de texto delimitado. Es útil para exportar datos que pueden ser leídos por otro análisis o software de hoja de cálculo.

Ejemplo

```
stream = modeler.script.stream()
outputfile = stream.createAt("outputfile", "File Output", 200, 200)
outputfile.setPropertyValue("full_filename", "c:/output/flatfile_output.txt")
outputfile.setPropertyValue("write_mode", "Append")
outputfile.setPropertyValue("inc_field_names", False)
outputfile.setPropertyValue("use_newline_after_records", False)
outputfile.setPropertyValue("delimit_mode", "Tab")
outputfile.setPropertyValue("other_delimiter", ",")
outputfile.setPropertyValue("quote_mode", "Double")
outputfile.setPropertyValue("other_quote", "*")
outputfile.setPropertyValue("decimal_symbol", "Period")
outputfile.setPropertyValue("generate_import", True)
```

Tabla 252. propiedades de outputfilenode

propiedades de outputfilenode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	Nombre del archivo de resultados.
write_mode	Overwrite Append	
inc_field_names	*Flag*	
use_newline_after_records	*Flag*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*carácter*	
quote_mode	None Single Double Other	
other_quote	*Flag*	
generate_import	*Flag*	

Tabla 252. propiedades de outputfilenode (continuación)		
propiedades de outputfilenode	**Tipo de datos**	**Descripción de la propiedad**
encoding	StreamDefault SystemDefault "UTF-8"	

Propiedades de sasexportnode

El nodo Exportar SAS produce datos en formato SAS, para leerlos en SAS o en un paquete de software compatible con SAS. Hay tres formatos de archivo SAS disponibles: SAS para Windows/OS2, SAS para UNIX o SAS Versión 7/8.

Ejemplo

```
stream = modeler.script.stream()
sasexportnode = stream.createAt("sasexport", "SAS Export", 200, 200)
sasexportnode.setPropertyValue("full_filename", "c:/output/
SAS_output.sas7bdat")
sasexportnode.setPropertyValue("format", "SAS8")
sasexportnode.setPropertyValue("export_names", "NamesAndLabels")
sasexportnode.setPropertyValue("generate_import", True)
```

Tabla 253. propiedades de sasexportnode		
Propiedad de sasexportnode	**Tipo de datos**	**Descripción de la propiedad**
format	Windows UNIX SAS7 SAS8	Campos de etiquetas de propiedad de variantes.
full_filename	*serie*	
export_names	NamesAndLabels NamesAsLabels	Se utiliza para correlacionar los nombres de campos de IBM SPSS Modeler que se vayan a exportar a nombres de variables de IBM SPSS Statistics o SAS.
generate_import	*Flag*	

Propiedades de statisticsexportnode

El nodo Exportar Statistics ofrece los resultados en formato IBM SPSS Statistics *.sav* o *.zsav*. IBM SPSS Statistics Base y otros productos pueden leer los archivos `.sav` o `.zsav` . Este es también el formato utilizado para los archivos de memoria caché en IBM SPSS Modeler.

Las propiedades de este nodo están descritas en "Propiedades de statisticsexportnode" en la página 437.

Propiedades del nodo tm1odataexport

El nodo Exportar IBM Cognos TM1 exporta datos en un formato que pueden leer las bases de datos de Cognos TM1.

Tabla 254. Propiedades del nodo tm1odataexport

Propiedades del nodo `tm1odataexport`	**Tipo de datos**	**Descripción de la propiedad**
`credential_type`	inputCredential o storedCredential	Se utiliza para indicar el tipo de credencial.
`input_credential`	*lista*	Cuando `credential_type` es `inputCredential`; especifique el dominio, el nombre de usuario y la contraseña.
`stored_credential_name`	*serie*	Cuando `credential_type` es `storedCredential`; especifique el nombre de la credencial en el servidor C & DS.
`selected_cube`	*campo*	El nombre del cubo al que está exportando datos. Por ejemplo: `TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")`

Tabla 254. Propiedades del nodo tm1odataexport (continuación)		
Propiedades del nodo `tm1odataexport`	**Tipo de datos**	**Descripción de la propiedad**
`spss_field_to_tm1_element_mapping`	*lista*	El elemento de tm1 al que hay que correlacionar debe formar parte de la dimensión de columna para la vista de cubo seleccionada. El formato es: `[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]` Hay 2 listas para describir la información de correlación. La correlación de un elemento de hoja con una dimensión corresponde al ejemplo 2 siguiente: Ejemplo 1: La primera lista: `([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...])` se utiliza para la información de correlación de TM1 Dimension. Cada lista de 3 valores indica información de correlación de dimensiones. El tercer valor booleano se utiliza para indicar si selecciona un elemento de una dimensión. Por ejemplo: `"[Field_1, Dimension_1, False]"` significa que `Field_1` está correlacionado con `Dimension_1`; `"[Element_1, Dimension_2, True]"` significa que `Element_1` está seleccionado para `Dimension_2`. Ejemplo 2: La segunda lista: `([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...])` se utiliza para la información de correlación de elementos de dimensión de medida de TM1 . Cada lista de 3 valores indica información de correlación de elementos de medida. El tercer valor booleano se utiliza para indicar la necesidad de crear un elemento nuevo. `"[Field_2, ExistMeasureElement, False]"` significa que `Field_2` está correlacionado con `ExistMeasureElement`; `"[Field_3, NewMeasureElement, True]"` significa que `NewMeasureElement` necesita ser la dimensión de medida elegida en `selected_measure` y que `Field_3` se correlaciona con ella.
`selected_measure`	*serie*	Especifique la dimensión de medida. Ejemplo: `setPropertyValue("selected_measure", "Measures")`

Tabla 254. Propiedades del nodo tm1odataexport (continuación)		
Propiedades del nodo `tm1odataexport`	**Tipo de datos**	**Descripción de la propiedad**
`connection_type`	`AdminServer` `TM1Server`	Indica el tipo de conexión. El valor predeterminado es `AdminServer`.
`admin_host`	*serie*	El URL del nombre de host de la API REST. Obligatorio si `connection_type` es `AdminServer`.
`server_name`	*serie*	El nombre del servidor TM1 seleccionado de `admin_host`. Obligatorio si `connection_type` es `AdminServer`.
`server_url`	*serie*	El URL de la API REST del Servidor TM1. Obligatorio si `connection_type` es `TM1Server`.

Propiedades del nodo tm1export (en desuso)

El nodo Exportar IBM Cognos TM1 exporta datos en un formato que pueden leer las bases de datos de Cognos TM1.

Nota: Este nodo estaba en desuso en Modeler 18.0. El nombre del script del nodo de sustitución es *tm1odataexport*.

Tabla 255. Propiedades del nodo tm1export		
Propiedades del nodo `tm1export`	**Tipo de datos**	**Descripción de la propiedad**
`pm_host`	*serie*	**Nota:** sólo para la versión 16.0 y 17.0 Nombre del host. Por ejemplo: `TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
`tm1_connection`	*["campo", "campo", ..., "campo"]*	**Nota:** sólo para la versión 16.0 y 17.0 Una propiedad de la lista que contiene los detalles de conexión para el servidor de TM1. El formato es: `[ "TM1_Server_Name", "tm1_ username", "tm1_password"]` Por ejemplo: `TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])`
`selected_cube`	*campo*	El nombre del cubo al que está exportando datos. Por ejemplo: `TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")`

Tabla 255. Propiedades del nodo tm1export (continuación)

Propiedades del nodo `tm1export`	**Tipo de datos**	**Descripción de la propiedad**
`spssfield_tm1element_mapping`	*lista*	El elemento de tm1 al que hay que correlacionar debe formar parte de la dimensión de columna para la vista de cubo seleccionada. El formato es: `[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]` Existen 2 listas para describir la información de correlación. La correlación de un elemento de hoja a una dimensión se corresponde al ejemplo 2 siguiente: Ejemplo 1: La primera lista: `([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...])` se utiliza para la información de correlación de TM1 Dimension. Cada lista de 3 valores indica información de correlación de dimensiones. El tercer valor booleano se utiliza para indicar si selecciona un elemento de una dimensión. Por ejemplo: `"[Field_1, Dimension_1, False]"` significa que `Field_1` está correlacionado con `Dimension_1`; `"[Element_1, Dimension_2, True]"` significa que `Element_1` está seleccionado para `Dimension_2`. Ejemplo 2: La segunda lista: `([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...])` se utiliza para la información de correlación de elementos de dimensión de medida de TM1 . Cada lista de 3 valores indica información de correlación de elementos de medida. El tercer valor booleano se utiliza para indicar la necesidad de crear un elemento nuevo. `"[Field_2, ExistMeasureElement, False]"` significa que `Field_2` está correlacionado con `ExistMeasureElement`; `"[Field_3, NewMeasureElement, True]"` significa que `NewMeasureElement` necesita ser la dimensión de medida elegida en `selected_measure` y que `Field_3` se correlaciona con ella.

Tabla 255. Propiedades del nodo tm1export (continuación)		
Propiedades del nodo `tm1export`	**Tipo de datos**	**Descripción de la propiedad**
`selected_measure`	*serie*	Especifique la dimensión de medida. Ejemplo:`setPropertyValue("selected_measure", "Measures")`

Propiedades de xmlexportnode

El nodo de exportación XML exporta datos a un archivo en formato XML. También puede crear un nodo de origen XML para leer los datos exportados a la ruta.

Ejemplo

```
stream = modeler.script.stream()
xmlexportnode = stream.createAt("xmlexport", "XML Export", 200, 200)
xmlexportnode.setPropertyValue("full_filename", "c:/export/data.xml")
xmlexportnode.setPropertyValue("map", [["/catalog/book/genre", "genre"], ["/
catalog/book/title", "title"]])
```

Tabla 256. propiedades de xmlexportnode		
propiedades de `xmlexportnode`	**Tipo de datos**	**Descripción de la propiedad**
`full_filename`	*serie*	(obligatorio) Ruta completa y nombre de archivo del archivo XML para exportar.
`use_xml_schema`	*Flag*	Especifica si utilizar un esquema XML (archivo XSD o DTD) para controlar la estructura de los datos exportados.
`full_schema_filename`	*serie*	Ruta completa y nombre de archivo del archivo XSD o DTD que se quiere utilizar. Es obligatorio si `use_xml_schema` está establecido como true.
`generate_import`	*Flag*	Genera un nodo de origen XML que leerá el archivo de datos exportados de nuevo en la ruta.
`records`	*serie*	Expresión de XPath que denota el límite de registro.
`map`	*serie*	Correlaciona el nombre de campo a la estructura XML.

Capítulo 18. Propiedades de nodos de IBM SPSS Statistics

Propiedades de statisticsimportnode

El nodo Archivo de estadísticas lee los datos desde un formato de archivo *.sav* o *.zsav* que utiliza IBM SPSS Statistics y archivos caché guardados en IBM SPSS Modeler, que también puede utilizar el mismo formato.

Ejemplo

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import",
200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drug1n.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

Tabla 257. propiedades de statisticsimportnode

propiedades de `statisticsimportnode`	**Tipo de datos**	**Descripción de la propiedad**
`full_filename`	*serie*	El nombre completo del archivo, incluyendo la vía de acceso.
`password`	*serie*	La contraseña. El parámetro `password` se debe establecer antes que el parámetro `file_encrypted`.
`file_encrypted`	*Flag*	Indica si el archivo está protegido con contraseña.

`import_names`	`NamesAndLabels`
	
	`LabelsAsNames`
		Método para gestionar nombres y etiquetas de variables.

`import_data`	`DataAndLabels`
	
	`LabelsAsData`
		Método para gestionar valores y etiquetas.

`use_field_format_for_storage`	*boolean*	Especifica si se utiliza la información de formato de campo de IBM SPSS Statistics al importar.

propiedades de statisticstransformnode

El nodo Transformación Statistics ejecuta una selección de comandos de sintaxis de IBM SPSS Statistics en los orígenes de datos de IBM SPSS Modeler. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Ejemplo

```
stream = modeler.script.stream()
statisticstransformnode = stream.createAt("statisticstransform",
"Transform", 200, 200)
statisticstransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na +
K.")
statisticstransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed
Drugs")
statisticstransformnode.setPropertyValue("check_before_saving", True)
```

Tabla 258. propiedades de statisticstransformnode

propiedades de statisticstransformnode	**Tipo de datos**	**Descripción de la propiedad**
syntax	*serie*	
check_before_saving	*Flag*	Valida la sintaxis introducida antes de guardar las entradas. Muestra un mensaje de error si la sintaxis no es válida.
default_include	*Flag*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.
include	*Flag*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.
new_name	*serie*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.

propiedades de statisticsmodelnode

El nodo Modelo Statistics permite analizar y trabajar con sus datos ejecutando los procedimientos de IBM SPSS Statistics que producen PMML. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Ejemplo

```
stream = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed
Drugs")
```

propiedades de statisticsmodelnode	**Tipo de datos**	**Descripción de la propiedad**
syntax	*serie*	
default_include	*Flag*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.
include	*Flag*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.

propiedades de statisticsmodelnode	Tipo de datos	Descripción de la propiedad
new_name	*serie*	Para obtener más información, consulte el tema “propiedades de filternode” en la página 168.

Propiedades de statisticsoutputnode

El nodo Resultados de Statistics le permite llamar a un procedimiento de IBM SPSS Statistics para analizar los datos de IBM SPSS Modeler. Se puede acceder a una gran variedad de procedimientos analíticos de IBM SPSS Statistics. Este nodo requiere una copia con licencia de IBM SPSS Statistics.

Ejemplo

```
stream = modeler.script.stream()
statisticsoutputnode = stream.createAt("statisticsoutput", "Output", 200,
200)
statisticsoutputnode.setPropertyValue("syntax", "SORT CASES BY Age(A) Sex(A)
BP(A) Cholesterol(A)")
statisticsoutputnode.setPropertyValue("use_output_name", False)
statisticsoutputnode.setPropertyValue("output_mode", "File")
statisticsoutputnode.setPropertyValue("full_filename", "Cases by Age, Sex
and Medical History")
statisticsoutputnode.setPropertyValue("file_type", "HTML")
```

Tabla 259. propiedades de statisticsoutputnode

propiedades de statisticsoutputnode	Tipo de datos	Descripción de la propiedad
mode	Dialog Syntax	Selecciona la opción "cuadro de diálogo de IBM SPSS Statistics" o el editor Sintaxis
syntax	*serie*	
use_output_name	*Flag*	
output_name	*serie*	
output_mode	Screen File	
full_filename	*serie*	
file_type	HTML SPV SPW	

Propiedades de statisticsexportnode

El nodo Exportar Statistics ofrece los resultados en formato IBM SPSS Statistics *.sav* o *.zsav*. IBM SPSS Statistics Base y otros productos pueden leer los archivos `.sav` o `.zsav` . Este es también el formato utilizado para los archivos de memoria caché en IBM SPSS Modeler.

Ejemplo

```
stream = modeler.script.stream()
statisticsexportnode = stream.createAt("statisticsexport", "Export", 200,
200)
statisticsexportnode.setPropertyValue("full_filename", "c:/output/
SPSS_Statistics_out.sav")
statisticsexportnode.setPropertyValue("field_names", "Names")
statisticsexportnode.setPropertyValue("launch_application", True)
statisticsexportnode.setPropertyValue("generate_import", True)
```

Tabla 260. propiedades de statisticsexportnode

Propiedades de statisticsexportnode	**Tipo de datos**	**Descripción de la propiedad**
full_filename	*serie*	
file_type	sav zsav	Guardar el archivo en formato *sav* o *zsav*. Por ejemplo: statisticsexportnode.setPropertyValue("file_type","sav")
encrypt_file	*Flag*	Indica si el archivo está protegido con contraseña.
password	*serie*	La contraseña.
launch_application	*Flag*	
export_names	NamesAndLabels NamesAsLabels	Se utiliza para correlacionar los nombres de campos de IBM SPSS Modeler que se vayan a exportar a nombres de variables de IBM SPSS Statistics o SAS.
generate_import	*Flag*	

Capítulo 19. Propiedades de nodo Python

Propiedades gmm

Un modelo de Gaussian Mixture© es un modelo probabilístico que presupone que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gausianas con parámetros desconocidos. Se puede pensar que los modelos de mezcla son generalizadores de la agrupación en clúster de k-medias para incorporar información sobre la estructura de covarianzas de los datos, así como los centros de valores gausianos latentes. El nodo de mezcla gausiana en SPSS Modeler expone las características principales y los parámetros utilizados con más frecuencia de la biblioteca de la mezcla gausiana. El nodo se implementa en Python.

Tabla 261. Propiedades gmm

Propiedades gmm	Tipo de datos	Descripción de la propiedad
use_partition	*Boolean*	Establezca en True o False para especificar si se van a utilizar datos particionados. El valor predeterminado es False.
covariance_type	*serie*	Especifique Full, Tied, Diag o Spherical para establecer el tipo de covarianza.
number_component	*integer*	Especifique un entero para el número de componentes de mezcla. El valor mínimo es 1. El valor predeterminado es 2.
component_lable	*Boolean*	Especifique True para establecer la etiqueta de clúster en una cadena o False para establecer la etiqueta del clúster en un número. El valor predeterminado es False.
label_prefix	*serie*	Si está utilizando una etiqueta de clúster de cadena, puede especificar un prefijo.
enable_random_seed	*Boolean*	Especifique True si desea utilizar una semilla aleatoria. El valor predeterminado es False.
random_seed	*integer*	Si está utilizando una semilla aleatoria, especifique un entero que se utilizará para generar muestras aleatorias.
tol	*double*	Especifique el umbral de convergencia. El valor predeterminado es 0.000.1.
max_iter	*integer*	Especifique el número máximo de iteraciones para realizar. El valor predeterminado es 100.
init_params	*serie*	Establezca el parámetro de inicialización para utilizar. Las opciones son Kmeans o Random.

Tabla 261. Propiedades gmm (continuación)		
Propiedades gmm	**Tipo de datos**	**Descripción de la propiedad**
warm_start	*Boolean*	Especifique True para utilizar la solución del último ajuste como la inicialización para la siguiente llamada de ajuste. El valor predeterminado es False.

Propiedades hdbscannode

La agrupación en clúster espacial basada en densidad jerárquica (HDBSCAN)© utiliza el aprendizaje no supervisado para encontrar clústeres, o regiones densas, de un conjunto de datos. El nodo HDBSCAN en SPSS Modeler expone las características principales y los parámetros utilizados con más frecuencia de la biblioteca HDBSCAN. El nodo se implementa en Python, y puede utilizarlo para agrupar en clúster el conjunto de datos en grupos distintos cuando no sabe que grupos son al principio.

Tabla 262. Propiedades hdbscannode		
Propiedades hdbscannode	**Tipo de datos**	**Descripción de la propiedad**
inputs	*campo*	Campos de entrada para la agrupación en clúster.
useHPO	*Boolean*	Especifique true o false para habilitar o inhabilitar la optimización HPO (Hyper-Parameter Optimization) basada en Rbfopt, que descubre automáticamente la combinación óptima de parámetros, de forma que el modelo conseguirá el índice de error previsto o inferior en las muestras. El valor predeterminado es false.
min_cluster_size	*integer*	El tamaño mínimo de clústeres. Especifique un entero. El valor predeterminado es 5.
min_samples	*integer*	El número de muestras en un vecindario para un punto que se va a considerar un punto central. Especifique un entero. Si se establece en 0, se utiliza min_cluster_size. El valor predeterminado es 0.
algorithm	*serie*	Especifique qué algoritmo utilizar: best, generic, prims_kdtree, prims_balltree, boruvka_kdtree o boruvka_balltree. El valor predeterminado es best.

Tabla 262. Propiedades hdbscannode (continuación)

Propiedades hdbscannode	Tipo de datos	Descripción de la propiedad
metric	*serie*	Especifique qué métrica utilizar al calcular la distancia entre instancias en una matriz de características: euclidean, cityblock, L1, L2, manhattan, braycurtis, canberra, chebyshev, correlation, minkowski o sqeuclidean. El valor predeterminado es euclidean.
useStringLabel	*Boolean*	Especifique true para utilizar una etiqueta de clúster de serie, o false para utilizar una etiqueta de clúster de número. El valor predeterminado es false.
stringLabelPrefix	*serie*	Si el parámetro useStringLabel está establecido en true, especifique un valor para el prefijo de etiqueta de serie. El prefijo predeterminado es cluster.
approx_min_span_tree	*Boolean*	Especifique true para aceptar un árbol de expansión mínima aproximado, o false si desea sacrificar la velocidad por la precisión. El valor predeterminado es true.
cluster_selection_method	*serie*	Especifique el método para utilizar para seleccionar clústeres en el árbol condensado: eom o leaf. El valor predeterminado es eom (algoritmo de exceso de masa).
allow_single_cluster	*Boolean*	Especifique true si desea permitir resultados de clúster único. El valor predeterminado es false.
p_value	*DOUBLE*	Especifique el p value para utilizar si está utilizando minkowski para la métrica. El valor predeterminado es 1.5.
leaf_size	*integer*	Si utiliza un algoritmo de árbol de espacio (boruvka_kdtree, o boruvka_balltree), especifique el número de puntos en un nodo de hoja del árbol. El valor predeterminado es 40.
outputValidity	*Boolean*	Especifique true o false para controlar si el gráfico de índice de validez está incluido en la salida del modelo.
outputCondensed	*Boolean*	Especifique true o false para controlar si el gráfico de árbol condensado está incluido en la salida del modelo.
outputSingleLinkage	*Boolean*	Especifique true o false para controlar si el gráfico de árbol de enlace único está incluido en la salida del modelo.

Tabla 262. Propiedades hdbscannode (continuación)		
Propiedades hdbscannode	**Tipo de datos**	**Descripción de la propiedad**
outputMinSpan	*Boolean*	Especifique `true` o `false` para controlar si el gráfico de árbol de expansión mínima está incluido en la salida del modelo.
is_split		Se ha añadido en la versión 18.2.1.1.

Propiedades kdemodel

La estimación de densidad de kernel (KDE)© utiliza los algoritmos del árbol de métrica o del árbol KD para unas consultas eficientes y combina los conceptos de aprendizaje no supervisado, ingeniería de características y modelado de datos. Los enfoques basados en vecinos como, por ejemplo, KDE, son algunas de las técnicas de estimación de densidad más populares y útiles. Los nodos de modelado de KDE y de simulación de KDE en SPSS Modeler exponen las características principales y los parámetros utilizados con más frecuencia de la biblioteca KDE. Los nodos se implementan en Python.

Tabla 263. Propiedades kdemodel		
Propiedades kdemodel	**Tipo de datos**	**Descripción de la propiedad**
bandwidth	*DOUBLE*	El valor predeterminado es 1.
kernel	*serie*	El kernel para utilizar: `gaussian`, `tophat`, `epanechnikov`, `exponential`, `linear` o `cosine`. El valor predeterminado es `gaussian`.
algorithm	*serie*	El algoritmo de árbol para utilizar: `kd_tree`, `ball_tree` o `auto`. El valor predeterminado es `auto`.
metric	*serie*	La métrica para utilizar al calcular distancias. Para el algoritmo `kd_tree`, elija entre: `Euclidean`, `Chebyshev`, `Cityblock`, `Minkowski`, `Manhattan`, `Infinity`, `P`, `L2` o `L1`. Para el algoritmo `ball_tree`, elija entre: `Euclidian`, `Braycurtis`, `Chebyshev`, `Canberra`, `Cityblock`, `Dice`, `Hamming`, `Infinity`, `Jaccard`, `L1`, `L2`, `Minkowski`, `Matching`, `Manhattan`, `P`, `Rogersanimoto`, `Russellrao`, `Sokalmichener`, `Sokalsneath` o `Kulsinski`. El valor predeterminado es `Euclidean`.
atol	*Float*	La tolerancia absoluta deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es `0.0`.

Tabla 263. Propiedades kdemodel (continuación)		
Propiedades kdemodel	**Tipo de datos**	**Descripción de la propiedad**
rtol	*Float*	La tolerancia relativa deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es 1E-8.
breadthFirst Renombrado como breadth_first a partir de la versión 18.2.1.1	*Boolean*	Establezca en True para utilizar un enfoque de amplitud. Establezca en False para utilizar un enfoque de búsqueda en profundidad. El valor predeterminado es True.
LeafSize Renombrado como leaf_size a partir de la versión 18.2.1.1	*integer*	El tamaño de hoja del árbol subyacente. El valor predeterminado es 40. Cambiar este valor podría afectar de forma significativa al rendimiento.
pValue	*DOUBLE*	Especifique el Valor P para utilizar si está utilizando Minkowski para la métrica. El valor predeterminado es 1.5.
custom_name		
default_node_name		
use_HPO		

Propiedades kdeexport

La estimación de densidad de kernel (KDE)© utiliza los algoritmos del árbol de métrica o del árbol KD para unas consultas eficientes y combina los conceptos de aprendizaje no supervisado, ingeniería de características y modelado de datos. Los enfoques basados en vecinos como, por ejemplo, KDE, son algunas de las técnicas de estimación de densidad más populares y útiles. Los nodos de modelado de KDE y de simulación de KDE en SPSS Modeler exponen las características principales y los parámetros utilizados con más frecuencia de la biblioteca KDE. Los nodos se implementan en Python.

Tabla 264. Propiedades kdeexport		
Propiedades kdeexport	**Tipo de datos**	**Descripción de la propiedad**
bandwidth	*DOUBLE*	El valor predeterminado es 1.
kernel	*serie*	El kernel para utilizar: gaussian o tophat. El valor predeterminado es gaussian.
algorithm	*serie*	El algoritmo de árbol para utilizar: kd_tree, ball_tree o auto. El valor predeterminado es auto.

Tabla 264. Propiedades kdeexport (continuación)		
Propiedades kdeexport	**Tipo de datos**	**Descripción de la propiedad**
metric	*serie*	La métrica para utilizar al calcular distancias. Para el algoritmo kd_tree, elija entre: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2 o L1. Para el algoritmo ball_tree, elija entre: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath o Kulsinski. El valor predeterminado es Euclidean.
atol	*Float*	La tolerancia absoluta deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es 0.0.
rtol	*Float*	La tolerancia relativa deseada del resultado. Por regla general, una tolerancia mayor conducirá a una ejecución más rápida. El valor predeterminado es 1E-8.
breadthFirst	*Boolean*	Establezca en True para utilizar un enfoque de amplitud. Establezca en False para utilizar un enfoque de búsqueda en profundidad. El valor predeterminado es True.
LeafSize	*integer*	El tamaño de hoja del árbol subyacente. El valor predeterminado es 40. Cambiar este valor podría afectar de forma significativa al rendimiento.
pValue	*DOUBLE*	Especifique el Valor P para utilizar si está utilizando Minkowski para la métrica. El valor predeterminado es 1.5.

Propiedades gmm

Un modelo de Gaussian Mixture© es un modelo probabilístico que presupone que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gausianas con parámetros desconocidos. Se puede pensar que los modelos de mezcla son generalizadores de la agrupación en clúster de k-medias para incorporar información sobre la estructura de covarianzas de los datos, así como los centros de valores gausianos latentes. El nodo de mezcla gausiana en SPSS Modeler expone las características principales y los parámetros utilizados con más frecuencia de la biblioteca de la mezcla gausiana. El nodo se implementa en Python.

Tabla 265. Propiedades gmm		
Propiedades gmm	**Tipo de datos**	**Descripción de la propiedad**
use_partition	*Boolean*	Establezca en True o False para especificar si se van a utilizar datos particionados. El valor predeterminado es False.
covariance_type	*serie*	Especifique Full, Tied, Diag o Spherical para establecer el tipo de covarianza.
number_component	*integer*	Especifique un entero para el número de componentes de mezcla. El valor mínimo es 1. El valor predeterminado es 2.
component_lable	*Boolean*	Especifique True para establecer la etiqueta de clúster en una cadena o False para establecer la etiqueta del clúster en un número. El valor predeterminado es False.
label_prefix	*serie*	Si está utilizando una etiqueta de clúster de cadena, puede especificar un prefijo.
enable_random_seed	*Boolean*	Especifique True si desea utilizar una semilla aleatoria. El valor predeterminado es False.
random_seed	*integer*	Si está utilizando una semilla aleatoria, especifique un entero que se utilizará para generar muestras aleatorias.
tol	*double*	Especifique el umbral de convergencia. El valor predeterminado es 0.000.1.
max_iter	*integer*	Especifique el número máximo de iteraciones para realizar. El valor predeterminado es 100.
init_params	*serie*	Establezca el parámetro de inicialización para utilizar. Las opciones son Kmeans o Random.
warm_start	*Boolean*	Especifique True para utilizar la solución del último ajuste como la inicialización para la siguiente llamada de ajuste. El valor predeterminado es False.

ocsvmnode, propiedades

El nodo SVM de una clase utiliza un algoritmo de aprendizaje no supervisado. El nodo se puede utilizar para la detección de novedad. Detectará el límite flexible de un conjunto de muestras proporcionado, para clasificar a continuación los puntos nuevos como pertenecientes o no a dicho conjunto. Este nodo de modelado SVM de una clase en SPSS Modeler se implementa en Python y requiere la biblioteca scikit-learn© Python .

Tabla 266. ocsvmnode, propiedades

Propiedades de ocsvmnode	Tipo de datos	Descripción de la propiedad
role_use Renombrado como custom_fields a partir de la versión 18.2.1.1	*serie*	Especifique predefined para utilizar roles predefinidos o custom para utilizar asignaciones de campo personalizado. El valor predeterminado es predefined.
splits	*campo*	Lista de nombre de campo para división.
use_partition	*boolean*	Especifique true o false. El valor predeterminado es true. Si se establece en true, sólo se utilizarán datos de entrenamiento al generar el modelo.
mode_type	*serie*	Modo. Los valores posibles son simple o expert. Todos los parámetros de la pestaña Experto estarán inhabilitados si se especifica simple.
stopping_criteria	*serie*	Serie de notación científica. Los valores posibles son 1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5 o 1.0E-6. El valor predeterminado es 1.0E-3.
precision	*Float*	Precisión de regresión (nu). Límite en la fracción de errores de entrenamiento y vectores de soporte. Especifique un número mayor que 0 y menor o igual que 1.0. El valor predeterminado es 0.1.
kernel	*serie*	Tipo de kernel a utilizar en el algoritmo. Los valores posibles son linear, poly, rbf, sigmoid o precomputed. El valor predeterminado es rbf.
enable_gamma	*boolean*	Habilita el parámetro gamma. Especifique true o false. El valor predeterminado es true.
gamma	*Float*	Este parámetro sólo está habilitado para los kernels rbf, poly y sigmoid. Si el parámetro enable_gamma se establece en false, este parámetro se establecerá en auto. Si se establece en true, el valor predeterminado es 0.1.
coef0	*Float*	Término independiente en la función de kernel. Este parámetro sólo está habilitado para el kernel poly y el kernel sigmoid. El valor predeterminado es 0.0.
degree	*integer*	Grado de función de kernel polinómica. Este parámetro sólo está habilitado para el kernel poly. Especifique cualquier entero. El valor predeterminado es 3.

Tabla 266. ocsvmnode, propiedades (continuación)

Propiedades de ocsvmnode	Tipo de datos	Descripción de la propiedad
shrinking	*boolean*	Especifica si se debe utilizar la opción heurística de reducción. Especifique true o false. El valor predeterminado es false.
enable_cache_size	*boolean*	Habilita el parámetro cache_size. Especifique true o false. El valor predeterminado es false.
cache_size	*Float*	Tamaño de caché de kernel en MB. El valor predeterminado es 200.
pc_type	*serie*	Tipo del gráfico de coordenadas paralelas. Los valores posibles son independent o general.
lines_amount	*integer*	Número máximo de líneas a incluir en el gráfico. Especifique un entero entre 1 y 1000.
lines_fields_custom	*boolean*	Habilita el parámetro lines_fields, que le permite especificar campos personalizados para mostrarlos en el resultado gráfico. Si se establece en false, se mostrarán todos los campos. Si se establece en true, sólo se mostrarán los campos especificados con el parámetro lines_fields. Por razones de rendimiento, se visualizará un máximo de 20 campos.
lines_fields	*campo*	Lista de nombres de campo a incluir en el gráfico como ejes verticales.
enable_graphic	*boolean*	Especifique true o false. Habilita el resultado gráfico (inhabilite esta opción si desea ahorrar tiempo y reducir el tamaño de archivo de ruta).
enable_hpo	*boolean*	Especifique true o false para habilitar o inhabilitar las opciones de HPO. Si está establecido en true, Rbfopt se aplicará para descubrir el "mejor" modelo SVM de una clase automáticamente, que alcanza el valor de objetivo de destino que ha definido el usuario con el parámetro target_objval siguiente.
target_objval	*Float*	El valor de función de objetivo (índice de error del modelo en las muestras) que se desea alcanzar (por ejemplo, el valor del óptimo desconocido). Establezca este parámetro en el valor apropiado si el óptimo es desconocido (por ejemplo, 0.01).

Tabla 266. ocsvmnode, propiedades (continuación)		
Propiedades de ocsvmnode	**Tipo de datos**	**Descripción de la propiedad**
max_iterations	*integer*	Número máximo de iteraciones para intentar el modelo. El valor predeterminado es 1000.
max_evaluations	*integer*	Número máximo de evaluaciones de función para intentar el modelo, donde el foco es la precisión en la velocidad. El valor predeterminado es 300.

Propiedades rfnode

El nodo Bosque aleatorio utiliza una implementación avanzada de un algoritmo de agregación autodocimante con un modelo de árbol como modelo base. Este nodo de modelado de bosque aleatorio en SPSS Modeler se implementa en Python y requiere la biblioteca scikit-learn© Python .

Tabla 267. Propiedades rfnode		
Propiedades rfnode	**Tipo de datos**	**Descripción de la propiedad**
role_use	*serie*	Especifique predefined para utilizar roles predefinidos o custom para utilizar asignaciones de campo personalizado. El valor predeterminado es predefined.
inputs	*campo*	Lista de los nombres de campo para entrada.
splits	*campo*	Lista de nombre de campo para división.
n_estimators	*integer*	Número de árboles para crear. El valor predeterminado es 10.
specify_max_depth	*boolean*	Especifique la profundidad máxima personalizada. Si es false, los nodos se expanden hasta que todas las hojas están vacía o hasta que todas las hojas contienen menos que min_samples_split muestras. El valor predeterminado es false.
max_depth	*integer*	La profundidad máxima del árbol. El valor predeterminado es 10.
min_samples_leaf	*integer*	Tamaño mínimo de nodo de hoja. El valor predeterminado es 1.

Tabla 267. Propiedades rfnode (continuación)

Propiedades rfnode	Tipo de datos	Descripción de la propiedad
max_features	*serie*	El número de características para tener en cuenta al buscar la mejor división: • Si auto, max_features=sqrt(n_features) para clasificador y para regresión. • Si sqrt, max_features=sqrt(n_features). • Si log2, max_features=log2 (n_features). El valor predeterminado es auto.
bootstrap	*boolean*	Utilice las muestras de programa de arranque al crear árboles. El valor predeterminado es true.
oob_score	*boolean*	Utilice muestras aleatorias para estimar la precisión de la generalización. El valor predeterminado es false.
extreme	*boolean*	Utilice árboles extremadamente aleatorizados. El valor predeterminado es false.
use_random_seed	*boolean*	Especifique esto para duplicar los resultados. El valor predeterminado es false.
random_seed	*integer*	La semilla de aleatorización para utilizar al crear árboles. Especifique cualquier entero.
cache_size	*Float*	Tamaño de caché de kernel en MB. El valor predeterminado es 200.
enable_random_seed	*boolean*	Habilita el parámetro random_seed. Especifique true o false. El valor predeterminado es false.
enable_hpo	*boolean*	Especifique true o false para habilitar o inhabilitar las opciones de HPO. Si está establecido en true, Rbfopt se aplicará para determinar el "mejor" modelo de bosque aleatorio automáticamente, que alcanza el valor del objetivo de destino que ha definido el usuario con el parámetro target_objval siguiente.
target_objval	*Float*	El valor de función de objetivo (índice de errores del modelo en las muestras) que desea alcanzar (por ejemplo, el valor del óptimo conocido). Establezca este parámetro en el valor apropiado si el óptimo es desconocido (por ejemplo, 0.01).

Tabla 267. Propiedades rfnode (continuación)		
Propiedades rfnode	**Tipo de datos**	**Descripción de la propiedad**
max_iterations	*integer*	Número máximo de iteraciones para intentar el modelo. El valor predeterminado es 1000.
max_evaluations	*integer*	Número máximo de evaluaciones de función para intentar el modelo, donde el foco es la precisión en la velocidad. El valor predeterminado es 300.

Propiedades de smotenode

El nodo SMOTE (Synthetic Minority Over-sampling Technique) proporciona un algoritmo de sobremuestreo para tratar con conjuntos de datos desequilibrados. Proporciona un método avanzado para equilibrar datos. El nodo de proceso SMOTE en SPSS Modeler se implementa en Python y requiere la biblioteca imbalanced-learn© Python .

Tabla 268. smotenode, propiedades		
Propiedades de smotenode	**Tipo de datos**	**Descripción de la propiedad**
target_field Renombrado como target a partir de la versión 18.2.1.1	*campo*	Campo objetivo.
sample_ratio	*serie*	Habilita un valor de cociente personalizado. Las dos opciones son Automático (sample_ratio_auto) o Establecer cociente (sample_ratio_manual).
sample_ratio_value	*Float*	La razón es el número de muestras en la clase minoritaria sobre el número de muestras de la clase mayoritaria. Debe ser mayor que 0 y menor que o igual a 1. El valor predeterminado es auto.
enable_random_seed	*boolean*	Si está establecido en true, se habilitará la propiedad random_seed.
random_seed	*integer*	Semilla utilizada por el generador de números aleatorios.
k_neighbours	*integer*	Número de vecinos más cercanos a utilizar para construir muestras sintéticas. El valor predeterminado es 5.
m_neighbours	*integer*	Número de vecinos más cercanos a utilizar para determinar si una muestra de minoría está en peligro. Esta opción sólo se habilita con los tipos de algoritmo de SMOTE borderline1 y borderline2. El valor predeterminado es 10.

Tabla 268. smotenode, propiedades (continuación)		
Propiedades de smotenode	**Tipo de datos**	**Descripción de la propiedad**
algorithm_kind Renombrado como algorithm a partir de la versión 18.2.1.1	*serie*	Tipo de algoritmo de SMOTE: regular, borderline1 o borderline2.
usepartition Renombrado como use_partition a partir de la versión 18.2.1.1	*boolean*	Si se establece en true, sólo se utilizarán datos de entrenamiento para la generación de modelos. El valor predeterminado es true.

Propiedades tsnenode

t-SNE (vecino estocástico con t distribuida incorporado) es una herramienta para visualizar datos de alta dimensión. Convierte afinidades de puntos de datos a probabilidades. Este nodo t-SNE en SPSS Modeler se implementa en Python y requiere la biblioteca scikit-learn© Python.

Tabla 269. propiedades tsnenode		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
mode_type	*serie*	Especifique la modalidad simple o expert .
n_components	*serie*	Dimensión del espacio incorporado (2D o 3D). Especifique 2 o 3. El valor predeterminado es 2.
method	*serie*	Especifique barnes_hut o exact. El valor predeterminado es barnes_hut.
init	*serie*	Inicialización de la incorporación. Especifique random o pca. El valor predeterminado es random.
target_field Renombrado como target a partir de la versión 18.2.1.1	*serie*	El nombre del campo objetivo. Será un mapa de colores en el gráfico de salida. El gráfico utilizará un color si no se especifica ningún campo objetivo.
perplexity	*Float*	La perplejidad está relacionada con el número de los vecinos más cercanos utilizados en otros algoritmos de aprendizaje de colector. Normalmente, los conjuntos de datos más grandes requieren una mayor perplejidad. Considere seleccionar un valor entre 5 y 50. El valor predeterminado es 30.

Tabla 269. propiedades tsnenode (continuación)		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
early_exaggeration	*Float*	Controla cómo de ajustados están los clústeres naturales del espacio original en el espacio incorporado, y cuánto espacio habrá entre ellos. El valor predeterminado es 12.0.
learning_rate	*Float*	El valor predeterminado es 200.
n_iter	*integer*	Número máximo de iteraciones para la optimización. Se establece en 250, como mínimo. El valor predeterminado es 1000.
angle	*Float*	El tamaño angular del nodo distante como se ha medido a partir de un punto. Especifique un valor dentro del rango de 0-1. El valor predeterminado es 0.5.
enable_random_seed	*boolean*	Establezca el valor en true para habilitar el parámetro random_seed. El valor predeterminado es false.
random_seed	*integer*	La semilla de aleatorización para utilizar. El valor predeterminado es None.
n_iter_without_progress	*integer*	Máximo de iteraciones sin progreso. El valor predeterminado es 300.
min_grad_norm	*serie*	Si la norma gradiente está por debajo de este umbral, la optimización se detendrá. El valor predeterminado es 1.0E-7. Los valores posibles son: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*boolean*	Se establece en true para realizar t-SNE con varias perplejidades distintas. El valor predeterminado es false.
output_Rename	*boolean*	Especifique true si desea proporcionar un nombre personalizado, o false para denominar la salida automáticamente. El valor predeterminado es false.
output_to	*serie*	Especifique Screen o Output. El valor predeterminado es Screen.
full_filename	*serie*	Especifique el nombre de archivo de salida.

Tabla 269. propiedades tsnenode (continuación)		
Propiedades tsnenode	**Tipo de datos**	**Descripción de la propiedad**
output_file_type	*serie*	Formato de archivo de salida. Especifique HTML o Output object. El valor predeterminado es HTML.

Propiedades xgboostlinearnode

XGBoost Linear© es una implementación avanzada de un algoritmo de aumento de gradiente con un modelo lineal como modelo base. Los algoritmos de aumento conocen de forma iterativa los clasificadores débiles y, a continuación, los añaden a un clasificador fuerte final. El nodo XGBoost Linear en SPSS Modeler se implementa en Python.

Tabla 270. xgboostlinearnode, propiedades		
Propiedades de xgboostlinearnode	**Tipo de datos**	**Descripción de la propiedad**
TargetField Renombrado como target a partir de la versión 18.2.1.1	*campo*	
InputFields Renombrado como inputs a partir de la versión 18.2.1.1	*campo*	
alpha	*double*	Parámetro de aumento lineal alfa. Especifique cualquier número 0 o superior. El valor predeterminado es 0.
lambda	*double*	Parámetro de aumento lineal lambda. Especifique cualquier número 0 o superior. El valor predeterminado es 1.
lambdaBias	*double*	Parámetro de aumento lineal lambda bias. Especifique cualquier número. El valor predeterminado es 0.
numBoostRound Renombrado como num_boost_round a partir de la versión 18.2.1.1	*integer*	Valor de redondeo de aumento de número para la generación de modelos. Especifique un valor entre 1 y 1000. El valor predeterminado es 10.

Tabla 270. xgboostlinearnode, propiedades (continuación)		
Propiedades de xgboostlinearnode	**Tipo de datos**	**Descripción de la propiedad**
objectiveType	*serie*	Tipo de objetivo para la tarea de aprendizaje. Los valores posibles son reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic o multi. Tenga en cuenta que para objetivos de marca, sólo se puede utilizar binary:logistic o multi. Si se utiliza multi, el resultado de puntuación mostrará los tipos de objetivo de XGBoost multi:softmax y multi:softprob.
random_seed	*integer*	Semilla de aleatorización. Cualquier número entre 0 y 9999999. El valor predeterminado es 0.
useHPO	*boolean*	Especifique true o false para habilitar o inhabilitar las opciones de HPO. Si está establecido en true, Rbfopt se aplicará para descubrir el "mejor" modelo SVM de una clase automáticamente, que alcanza el valor de objetivo de destino que ha definido el usuario con el parámetro target_objval.

Propiedades de xgboosttreenode

XGBoost Tree© es una implementación avanzada de un algoritmo de aumento de gradiente con un modelo de árbol como modelo base. Los algoritmos de aumento conocen de forma iterativa los clasificadores débiles y, a continuación, los añaden a un clasificador fuerte final. XGBoost Tree es muy flexible y proporciona muchos parámetros que pueden ser abrumadores para la mayoría de usuarios, de modo que el nodo XGBoost Tree en SPSS Modeler expone las características principales y los parámetros utilizados comúnmente. El nodo se implementa en Python.

Tabla 271. xgboosttreenode, propiedades		
Propiedades de xgboosttreenode	**Tipo de datos**	**Descripción de la propiedad**
TargetField Renombrado como target a partir de la versión 18.2.1.1	*campo*	Campos objetivo.
InputFields Renombrado como inputs a partir de la versión 18.2.1.1	*campo*	Campos de entrada.

Tabla 271. xgboosttreenode, propiedades (continuación)

Propiedades de xgboosttreenode	**Tipo de datos**	**Descripción de la propiedad**
treeMethod Renombrado como tree_method a partir de la versión 18.2.1.1	*serie*	Método de árbol para la generación de modelos. Los valores posibles son auto, exact o approx. El valor predeterminado es auto.
numBoostRound Renombrado como num_boost_round a partir de la versión 18.2.1.1	*integer*	Valor de redondeo de aumento de número para la generación de modelos. Especifique un valor entre 1 y 1000. El valor predeterminado es 10.
maxDepth Renombrado como max_depth a partir de la versión 18.2.1.1	*integer*	Profundidad máxima para el crecimiento del árbol. Especifique un valor de 1 o superior. El valor predeterminado es 6.
minChildWeight Renombrado como min_child_weight a partir de la versión 18.2.1.1	*double*	Ponderación hijo mínima para el crecimiento del árbol. Especifique un valor de 0 o superior. El valor predeterminado es 1.
maxDeltaStep Renombrado como max_delta_step a partir de la versión 18.2.1.1	*double*	Paso delta máximo para el crecimiento del árbol. Especifique un valor de 0 o superior. El valor predeterminado es 0.
objectiveType Renombrado como objective_type a partir de la versión 18.2.1.1	*serie*	Tipo de objetivo para la tarea de aprendizaje. Los valores posibles son reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic o multi. Tenga en cuenta que para objetivos de marca, sólo se puede utilizar binary:logistic o multi. Si se utiliza multi, el resultado de puntuación mostrará los tipos de objetivo de XGBoost multi:softmax y multi:softprob.
earlyStopping Renombrado como early_stopping a partir de la versión 18.2.1.1	*boolean*	Indica si debe utilizarse la función de detención temprana. El valor predeterminado es False.
earlyStoppingRounds Renombrado como early_stopping_rounds a partir de la versión 18.2.1.1	*integer*	El error de validación debe disminuir al menos cada ronda de detención temprana para continuar el entrenamiento. El valor predeterminado es 10.

Tabla 271. xgboosttreenode, propiedades (continuación)		
Propiedades de xgboosttreenode	**Tipo de datos**	**Descripción de la propiedad**
evaluationDataRatio Renombrado como evaluation_data_ratio a partir de la versión 18.2.1.1	*double*	La proporción de datos de entrada utilizados para errores de validación. El valor predeterminado es 0.3.
random_seed	*integer*	Semilla de aleatorización. Cualquier número entre 0 y 9999999. El valor predeterminado es 0.
sampleSize Renombrado como sample_size a partir de la versión 18.2.1.1	*double*	La submuestra para el sobreajuste de control. Especifique un valor entre 0.1 y 1.0. El valor predeterminado es 0.1.
eta	*double*	Eta para el sobreajuste de control. Especifique un valor entre 0 y 1. El valor predeterminado es 0.3.
gamma	*double*	Gamma para el sobreajuste de control. Especifique cualquier número 0 o superior. El valor predeterminado es 6.
colsSampleRatio Renombrado como col_sample_ratio a partir de la versión 18.2.1.1	*double*	Muestra de columna por árbol para el sobreajuste de control. Especifique un valor entre 0.01 y 1. El valor predeterminado es 1.
colsSampleLevel Renombrado como col_sample_level a partir de la versión 18.2.1.1	*double*	Muestra de columna por nivel para el sobreajuste de control. Especifique un valor entre 0.01 y 1. El valor predeterminado es 1.
lambda	*double*	Lambda para el sobreajuste de control. Especifique cualquier número 0 o superior. El valor predeterminado es 1.
alpha	*double*	Alfa para el sobreajuste de control. Especifique cualquier número 0 o superior. El valor predeterminado es 0.
scalePosWeight Renombrado como scale_pos_weight a partir de la versión 18.2.1.1	*double*	Ponderación de posición de escala para manejar conjuntos de datos desequilibrados. El valor predeterminado es 1.
use_HPO Se ha añadido para la versión 18.2.1.1		

Capítulo 20. Propiedades de nodo Spark

Propiedades isotonicasnode

La regresión isotónica pertenece a la familia de algoritmos de regresión. El nodo Isotónica-AS en SPSS Modeler se implementa en Spark. Si desea detalles sobre algoritmos de regresión isotónica, consulte https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html.

Tabla 272. propiedades isotonicasnode

Propiedades isotonicasnode	Tipo de datos	Descripción de la propiedad
label	*serie*	Esta propiedad es una variable dependiente para la cual se calcula la regresión isotónica.
features	*serie*	Esa propiedad es una variable independiente.
weightCol	*serie*	La ponderación representa una serie de medidas. El valor predeterminado es 1.
isotonic	*boolean*	Esta propiedad indica si el tipo es isotonic o antitonic.
featureIndex	*integer*	Esta propiedad es para el índice de la característica, si featuresCol es una columna de vector. El valor predeterminado es 0.

Propiedades kmeansasnode

k-medias es uno de los algoritmos de agrupación en clúster utilizados con más frecuencia. Agrupa en clúster puntos de datos en una serie de clústeres predefinidos. El nodo K-Medias-AS en SPSS Modeler se implementa en Spark. Si desea más detalles sobre algoritmos de k-medias, consulte https://spark.apache.org/docs/2.2.0/ml-clustering.html. Tenga en cuenta que el nodo K-Medias-AS realiza una codificación "one-hot" automáticamente para variables categóricas.

Tabla 273. Propiedades kmeansasnode

Propiedades kmeansasnode	Valores	Descripción de la propiedad
roleUse	*serie*	Especifique predefined para utilizar roles predefinidos, o custom para utilizar asignaciones de campo personalizado. El valor predeterminado es predefined.

Tabla 273. Propiedades kmeansasnode (continuación)

Propiedades kmeansasnode	Valores	Descripción de la propiedad
autoModel	*boolean*	Especifique true para utilizar el nombre predeterminado ($S-prediction) para el nuevo campo de puntuación generado, o false para utilizar un nombre personalizado. El valor predeterminado es true.
features	*campo*	Lista de los nombres de campo para la entrada de información cuando la propiedad roleUse está establecida en custom.
name	*serie*	El nombre del nuevo campo de puntuación generado cuando la propiedad autoModel está establecida en false.
clustersNum	*integer*	El número de clústeres para crear. El valor predeterminado es 5.
initMode	*serie*	El algoritmo de inicialización. Los valores posibles son k-means|| o random. El valor predeterminado es k-means||.
initSteps	*integer*	El número de pasos de inicialización cuando initMode está establecido en k-means||. El valor predeterminado es 2.
advancedSettings	*boolean*	Especifique true para que estén disponibles las cuatro propiedades siguientes. El valor predeterminado es false.
maxIteration	*integer*	Número máximo de iteraciones para la agrupación en clúster. El valor predeterminado es 20.
tolerance	*serie*	La tolerancia para detener las iteraciones. Los valores posibles son 1.0E-1, 1.0E-2, ..., 1.0E-6. El valor predeterminado es 1.0E-4.
setSeed	*boolean*	Especifique true para utilizar una semilla aleatoria personalizada. El valor predeterminado es false.
randomSeed	*integer*	La semilla aleatoria personalizada cuando la propiedad setSeed es true.

Propiedades multilayerperceptronnode

El perceptrón multicapa es un clasificador basado en la red neuronal artificial preventiva y está formado por varias capas. Cada capa está totalmente conectada con la siguiente capa de la red. El nodo Perceptrón multicapa-AS en SPSS Modeler se implementa en Spark. Si desea detalles sobre el clasificador de perceptrón multicapa (MLPC), consulte https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier.

Tabla 274. propiedades multilayerperceptronnode

Propiedades multilayerperceptronnode	**Tipo de datos**	**Descripción de la propiedad**
features	*campo*	Uno o más campos para utilizar como entradas para la predicción.
label	*campo*	El campo para utilizar como objetivo para la predicción.
layers[0]	*integer*	El número de capas del perceptrón para incluir. El valor predeterminado es 1.
layers[1…<latest-1>]	*integer*	El número de capas ocultas. El valor predeterminado es 1.
layers[<latest>]	*integer*	El número de capas de salida. El valor predeterminado es 1.
seed	*integer*	La semilla aleatoria personalizada.
maxiter	*integer*	El número máximo de iteraciones para realizar. El valor predeterminado es 10.

Propiedades xgboostasnode

XGBoost es una implementación avanzada de un algoritmo de aumento de gradiente. Los algoritmos de aumento conocen de forma iterativa los clasificadores débiles y, a continuación, los añaden a un clasificador fuerte final. XGBoost es muy flexible y proporciona muchos parámetros que pueden resultar abrumadores para la mayoría de los usuarios, así que el nodo XGBoost-AS en SPSS Modeler expone las características principales y los parámetros utilizados normalmente. El nodo XGBoost-AS se implementa en Spark.

Tabla 275. propiedades xgboostasnode

Propiedades xgboostasnode	**Tipo de datos**	**Descripción de la propiedad**
target_field	*campo*	Lista de los nombres de campo para el objetivo.
input_fields	*campo*	Lista de los nombres de campo para entradas.
nWorkers	*integer*	El número de trabajadores utilizados para formar el modelo XGBoost. El valor predeterminado es 1.
numThreadPerTask	*integer*	El número de hebras utilizadas por trabajador. El valor predeterminado es 1.

Tabla 275. propiedades xgboostasnode (continuación)		
Propiedades xgboostasnode	**Tipo de datos**	**Descripción de la propiedad**
useExternalMemory	*boolean*	Indica si se va a utilizar memoria externa como memoria caché. El valor predeterminado es **False**.
boosterType	*serie*	El tipo de amplificador para utilizar. Las opciones disponibles son gbtree, gblinear o dart. El valor predeterminado es gbtree.
numBoostRound	*integer*	El número de rondas para aumentar. Especifique un valor de 0 o superior. El valor predeterminado es 10.
scalePosWeight	*double*	Controlar el equilibrio de ponderaciones positivas y negativas. El valor predeterminado es 1.
randomseed	*integer*	Semilla utilizada por el generador de números aleatorios. El valor predeterminado es **0**.
objectiveType	*serie*	El objetivo de aprendizaje. Los valores posibles son reg:linear, reg:logistic, reg:gamma, reg:tweedie,rank:pairwise, binary:logistic o multi. Tenga en cuenta que para objetivos de marca, sólo se puede utilizar binary:logistic o multi. Si se utiliza multi, el resultado de puntuación mostrará los tipos de objetivo de XGBoost multi:softmax y multi:softprob. El valor predeterminado es reg:linear.
evalMetric	*serie*	Métricas de evaluación para datos de validación. Se asignará una métrica predeterminada de acuerdo con el objetivo. Los valores posibles son rmse, mae, logloss, error, merror, mlogloss, auc, ndcg, map o gamma-deviance. El valor predeterminado es rmse.
lambda	*double*	Término de regularización L2 en ponderaciones. Al aumentar este valor, el modelo será más conservador. Especifique cualquier número 0 o superior. El valor predeterminado es 1.
alpha	*double*	Término de regularización L1 sobre ponderaciones. Al aumentar este valor, el modelo será más conservador. Especifique cualquier número 0 o superior. El valor predeterminado es 0.

Tabla 275. propiedades xgboostasnode (continuación)		
Propiedades xgboostasnode	**Tipo de datos**	**Descripción de la propiedad**
lambdaBias	*double*	Término de regularización L2 en sesgo. Si se utiliza el tipo de amplificador gblinear, está disponible este parámetro d e amplificador lineal de sesgo lambda. Especifique cualquier número 0 o superior. El valor predeterminado es 0.
treeMethod	*serie*	Si se utiliza el tipo de amplificador gbtree o dart, está disponible este parámetro de método de árbol para el crecimiento del árbol (y los otros parámetros de árbol que siguen). Especifica el algoritmo de construcción de árbol XGBoost para utilizar. Las opciones disponibles son auto, exact o approx. El valor predeterminado es auto.
maxDepth	*integer*	La profundidad máxima para árboles. Especifique un valor de 2 o superior. El valor predeterminado es 6.
minChildWeight	*double*	La suma mínima de la ponderación de instancias (hessiana) necesaria en un hijo. Especifique un valor de 0 o superior. El valor predeterminado es 1.
maxDeltaStep	*double*	El paso delta máximo para permitir para la estimación ponderada de cada árbol. Especifique un valor de 0 o superior. El valor predeterminado es 0.
sampleSize	*double*	La muestra secundaria es la proporción de la instancia de formación. Especifique un valor entre 0.1 y 1.0. El valor predeterminado es 1.0.
eta	*double*	La reducción de tamaño de paso utilizada durante el paso de actualización para evitar el sobreajuste. Especifique un valor entre 0 y 1. El valor predeterminado es 0.3.
gamma	*double*	La reducción de pérdida mínima necesaria para realizar una partición adicional en un nodo hoja del árbol. Especifique cualquier número 0 o superior. El valor predeterminado es 6.
colsSampleRatio	*double*	La proporción de muestra secundaria de columnas al construir cada árbol. Especifique un valor entre 0.01 y 1. El valor predeterminado es 1.
colsSampleLevel	*double*	La proporción de muestra secundaria de columnas para cada división, en cada nivel. Especifique un valor entre 0.01 y 1. El valor predeterminado es 1.

Tabla 275. propiedades xgboostasnode (continuación)		
Propiedades xgboostasnode	**Tipo de datos**	**Descripción de la propiedad**
normalizeType	*serie*	Si se utiliza el tipo de amplificador dart, están disponibles este parámetro dart y los tres parámetros dart siguientes. Este parámetro define el algoritmo de normalización. Especifique `tree` o `forest`. El valor predeterminado es `tree`.
sampleType	*serie*	El tipo de algoritmo de muestreo. Especifique `uniform` o `weighted`. El valor predeterminado es `uniform`.
rateDrop	*double*	El parámetro de amplificador dart de índice de descarte. Especifique un valor entre `0.0` y `1.0`. El valor predeterminado es `0.0`.
skipDrop	*double*	El parámetro de amplificador dart para la probabilidad de descarte de salto. Especifique un valor entre `0.0` y `1.0`. El valor predeterminado es `0.0`.

Capítulo 21. Propiedades de supernodo

En las siguientes tablas se describen las propiedades específicas de los Supernodos. Tenga en cuenta que las propiedades de nodos comunes se aplican también a los Supernodos.

Tabla 276. Propiedades del supernodo de terminal

Nombre de la propiedad	Tipo de propiedad / Lista de valores	Descripción de la propiedad

`execute_method`	`Script`
	
	`Normal`	

`script`	*serie*	

Parámetros de Supernodos

Puede utilizar scripts para crear o establecer parámetros de Supernodo utilizando el formato general:

```
mySuperNode.setParameterValue("minvalue", 30)
```

Puede recuperar el valor del parámetro con:

```
value mySuperNode.getParameterValue("minvalue")
```

Búsqueda de los supernodos existentes

Puede encontrar supernodos en rutas utilizando la función findByType() :

```
source_supernode = modeler.script.stream().findByType("source_super", None)
process_supernode = modeler.script.stream().findByType("process_super", None)
terminal_supernode = modeler.script.stream().findByType("terminal_super",
None)
```

Configuración de las propiedades de nodos encapsulados

Puede establecer las propiedades de determinados nodos encapsulados dentro un supernodo accediendo al diagrama hijo dentro del Supernodo. Por ejemplo, imaginemos que tiene un Supernodo de origen que incluye un nodo de archivo de variables encapsulado para leer los datos. Puede pasar el nombre del archivo para leer (especificado mediante la propiedad `full_filename`) accediendo al diagrama hijo y buscando el nodo relevante como se indica a continuación:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Creación de supernodos

Si desea crear un supernodo y su contenido de cero, puede hacerlo de forma similar a la creación del supernodo, accediendo al diagrama hijo y creando los nodos que desee. También debe asegurarse de que

los nodos del diagrama del supernodo estén también vinculados a los nodos de los conectores de entrada y/o de salida. Por ejemplo, si desea crear un proceso Supernodo:

```
process_supernode = modeler.script.stream().createAt("process_super", "My
SuperNode", 200, 200)
childDiagram = process_supernode.getChildDiagram()
filternode = childDiagram.createAt("filter", "My Filter", 100, 100)
childDiagram.linkFromInputConnector(filternode)
childDiagram.linkToOutputConnector(filternode)
```

Apéndice A. Referencia de nombres de nodo

Esta sección ofrece una referencia de todos los nombres de script de los nodos de IBM SPSS Modeler.

Nombres de nugget de modelo

Se puede hacer referencia a los nugget de modelo (también denominados modelos generados) según el tipo, como con los objetos de nodo y de resultado. Las siguientes tablas muestran los nombres de referencia de los objetos del modelo.

Tenga en cuenta que estos nombres se utilizan específicamente para hacer referencia a los nugget de modelo en la paleta Modelos (en la esquina superior derecha de la ventana de IBM SPSS Modeler). Para hacer referencia a los nodos de modelo que se han añadido a una ruta para la puntuación, se utiliza un conjunto diferente de nombres con el prefijo `apply...`.

Nota: En circunstancias normales, se recomienda hacer referencia a los modelos por nombre *y* tipo para evitar confusiones.

Tabla 277. Nombres de nugget de modelo (paleta de modelado)

Nombre del modelo	**Modelo**
anomalydetection	Anomalía
a priori	Apriori
autoclassifier	Clasificador automático
autocluster	Clúster automático
autonumeric	Autonumérico
bayesnet	Red bayesiana
c50	C5.0
carma	Carma
árbol cr	Árbol C&R
chaid	CHAID
coxreg	Regresión de Cox
decisionlist	Lista de decisiones
discriminant	Discriminante
factor	PCA/Factorial
featureselection	Selección de características
genlin	Regresión lineal generalizada
glmm	GLMM
kmeans	K-medias
knn	*k*: vecino más cercano
kohonen	Kohonen
lineales	Lineal
logreg	Regresión logística

Tabla 277. Nombres de nugget de modelo (paleta de modelado) (continuación)	
Nombre del modelo	**Modelo**
neuralnetwork	Red neuronal
quest	QUEST
regresión	Regresión lineal
secuencia	Secuencia
slrm	Modelo de respuesta de autoaprendizaje
statisticsmodel	Modelo IBM SPSS Statistics
svm	Máquina de vectores de soporte
timeseries	Serie temporal
twostep	TwoStep

Tabla 278. Nombres de nugget de modelo (paleta de modelado de bases de datos)	
Nombre del modelo	**Modelo**
db2imcluster	Clúster de IBM ISW
db2imlog	Regresión logística de IBM ISW
db2imnb	Bayesiano ingenuo de IBM ISW
db2imreg	Regresión de IBM ISW
db2imtree	Árbol de decisión de IBM ISW
msassoc	Reglas de asociación de MS
msbayes	Bayesiano ingenuo de MS
mscluster	Clúster de MS
mslogistic	Regresión logística de MS
msneuralnetwork	Red neuronal de MS
msregression	Regresión lineal de MS
mssequencecluster	Clúster de secuencias de MS
mstimeseries	Series temporales de MS
mstree	Árbol de decisión de MS
netezzabayes	Red bayesiana de Netezza
netezzadectree	Árbol de decisión de Netezza
netezzadivcluster	Clúster divisivo de Netezza
netezzaglm	Lineal generalizado de Netezza
netezzakmeans	K-medias de Netezza
netezzaknn	KNN de Netezza
netezzalineregression	Regresión lineal de Netezza
netezzanaivebayes	Bayesiano ingenuo de Netezza
netezzapca	PCA de Netezza

Tabla 278. Nombres de nugget de modelo (paleta de modelado de bases de datos) (continuación)	
Nombre del modelo	**Modelo**
netezzaregtree	Árbol de regresión de Netezza
netezzatimeseries	Series temporales de Netezza
oraabn	Bayesiano adaptativo de Oracle
oraai	Oracle AI
oradecisiontree	Árbol de decisión de Oracle
oraglm	GLM de Oracle
orakmeans	*K*-medias de Oracle
oranb	Bayesiano ingenuo de Oracle
oranmf	NMF de Oracle
oraocluster	O-clúster de Oracle
orasvm	SVM de Oracle

Evitar nombres duplicados del modelo

Al utilizar los scripts para manipular los modelos generados, debe tener en cuenta que el hecho de permitir nombres de modelo duplicados puede originar referencias ambiguas. Para evitarlo, resulta útil utilizar nombres exclusivos para los modelos generados en los scripts.

Para configurar las opciones de los nombres de modelo duplicados:

1. En los menús seleccione:

   Herramientas > **Opciones de usuario**
2. Pulse en la pestaña **Notificaciones**.
3. Seleccione **Sustituir modelo anterior** para restringir los nombres duplicados de los modelos generados.

El comportamiento de la ejecución de scripts puede variar entre SPSS Modeler y IBM SPSS Collaboration and Deployment Services cuando haya referencias de modelo ambiguas. El cliente de SPSS Modeler incluye la opción "Reemplazar modelo anterior", que reemplaza automáticamente los modelos que tengan el mismo nombre (por ejemplo, cuando un script se itera a través de un bucle para producir un modelo diferente cada vez). Sin embargo, esta opción no está disponible cuando el mismo script se ejecuta en IBM SPSS Collaboration and Deployment Services. Puede evitar esta situación cambiando el nombre del modelo generado en cada iteración para evitar referencias ambiguas a los modelos o borrando el modelo actual (por ejemplo, añadiendo una instrucción `clear generated palette`) antes del final del bucle.

Nombres de tipo de resultado

La siguiente tabla indica los tipos de objetos de resultados y los nodos que los crean.

Tabla 279. Tipos de objeto de resultado y los nodos que los crean	
Tipo de objeto de resultado	**Nodo**
analysisoutput	Análisis
collectionoutput	Colección
dataauditoutput	Auditoría de datos

Tabla 279. Tipos de objeto de resultado y los nodos que los crean (continuación)	
Tipo de objeto de resultado	**Nodo**
distributionoutput	Distribución
evaluationoutput	Evaluación
histogramoutput	Histograma
matrixoutput	Matriz
meansoutput	Medias
multiplotoutput	G. múltiple
plotoutput	Gráfico
qualityoutput	Calidad
reportdocumentoutput	Este tipo de objeto no es de un nodo, es un resultado creado por un informe de proyecto
reportoutput	Informe
statisticsprocedureoutput	Salida de Estadísticas
statisticsoutput	Estadísticos
tableoutput	Tabla
timeplotoutput	Gráfico de tiempo
weboutput	Malla

Apéndice B. Migración desde scripts de herencia a scripts Python

Visión general de la migración de scripts de herencia

Esta sección proporciona un resumen de las diferencias entre el script de Python y el script de herencia en IBM SPSS Modeler y proporciona información acerca de cómo migrar los scripts de herencia a scripts Python. En esta sección encontrar una lista de los comandos de herencia de SPSS Modeler estándar y los comandos Python equivalentes.

Diferencias generales

Una gran parte del diseño de los scripts de herencia se debe a los scripts de comandos del sistema operativo. Los scripts de herencia están orientados a líneas y, aunque existen algunas estructuras de bloque, por ejemplo `if...then...else...endif` y `for...endfor`, generalmente la indentación no es importante.

En los scripts Python, la indentación es importante y las líneas que pertenecen al mismo bloque lógico se deben indentar en el mismo nivel.

Nota: Debe prestar atención cuando copie y pegue el código Python. En el editor, una línea que se ha indentado utilizando pestañas puede parecer la misma que una línea que se ha indentado utilizando espacios. Sin embargo, el script Python generará un error porque no se considera que la indentación de las líneas sea la misma.

El contexto de los scripts

El contexto de script define el entorno en el que se ejecuta un script como, por ejemplo, la ruta o Supernodo que ejecuta el script. En los scripts heredados el contexto es implícito, lo que significa que, por ejemplo, se asume que toda referencia a un nodo de una ruta está dentro de la ruta que ejecuta el script.

En los scripts Python el contexto de script se proporciona de forma explícita mediante el módulo `modeler.script`. Por ejemplo, un script Python de ruta puede acceder a la ruta que ejecuta el script mediante el código siguiente:

```
s = modeler.script.stream()
```

A continuación podrán invocarse funciones relacionadas con la ruta a través del objeto devuelto.

Comparativa de comandos y funciones

Los scripts heredados están orientados a comando. Esto significa que cada línea del script suele comenzar con el comando a ejecutar seguido de los parámetros, por ejemplo:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python utiliza funciones que suelen invocarse a través de un objeto (módulo, clase u objeto) que define la función, por ejemplo:

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Literales y comentarios

Algunos de los literales y comandos de comentarios que normalmente se utilizan en IBM SPSS Modeler tienen sus equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 280. Correlación de scripts de herencia con scripts Python para literales y comentarios

Scripts de herencia	**scripts Python**
Entero, por ejemplo 4	El mismo
Flotante, por ejemplo `0.003`	El mismo
Series entre comillas simples, por ejemplo `'Hello'`	El mismo **Nota:** Los literales de cadena que contengan caracteres que no sean ASCII deberán tener el prefijo u para garantizar que se representen en Unicode.
Series entre comillas dobles, por ejemplo `"Hello again"`	El mismo **Nota:** Los literales de cadena que contengan caracteres que no sean ASCII deberán tener el prefijo u para garantizar que se representen en Unicode.
Cadenas largas, por ejemplo, `"""This is a string that spans multiple lines"""`	El mismo
Listas, por ejemplo, `[1 2 3]`	`[1, 2, 3]`
Referencia de variable, por ejemplo `set x = 3`	`x = 3`
Continuación de línea (\), por ejemplo, `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
Comentario de bloque, por ejemplo, `/* This is a long comment over a line. */`	`""" This is a long comment over a line. """`
Comentario de línea, por ejemplo `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Operadores

Algunos de los comandos de operadores que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 281. Correlación de scripts de herencia con scripts Python para operadores

Scripts de herencia	scripts Python

`NUM1 + NUM2`
`LIST + ITEM`
`LIST1 + LIST2`
	`NUM1 + NUM2`
	`LIST.append(ITEM)`
	`LIST1.extend(LIST2)`

`NUM1 - NUM2` `LIST - ITEM`	`NUM1 - NUM2` `LIST.remove(ITEM)`
`NUM1 * NUM2`	`NUM1 * NUM2`
`NUM1 / NUM2`	`NUM1 / NUM2`
`=` `==`	`==`
`/=` `/==`	`!=`
`X ** Y`	`X ** Y`

`X < Y`
`X <= Y`
`X > Y`
`X >= Y`
	`X < Y`
	`X <= Y`
	`X > Y`
	`X >= Y`

`X div Y`
`X rem Y`
`X mod Y`
	`X // Y`
	`X % Y`
	`X % Y`

`and`
`or`
`not(EXPR)`
	`and`
	`or`
	`not EXPR`

Comandos condicionales y de bucle

Algunos comandos condicionales y de bucle utilizados habitualmente en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 282. Correspondencia de scripts de herencia con scripts Python en lo referente a comandos condicionales y de bucle

Scripts de herencia	scripts Python

`for VAR from INT1 to INT2`
`  …`
`endfor`
	`for VAR in range(INT1, INT2):`
	`  …`
	
	o
	
	`VAR = INT1`
	`while VAR <= INT2:`
	`  ...`
	`  VAR += 1`

Tabla 282. Correspondencia de scripts de herencia con scripts Python en lo referente a comandos condicionales y de bucle (continuación)

Scripts de herencia	scripts Python

`for VAR in LIST`
`  …`
`endfor`
	`for VAR in LIST:`
	`  …`

`for VAR in_fields_to NODE`
`  …`
`endfor`
	`for VAR in NODE.getInputDataModel():`
	`  ...`

`for VAR in_fields_at NODE`
`  …`
`endfor`
	`for VAR in NODE.getOutputDataModel():`
	`  ...`

`if…then`
`  …`
`elseif…then`
`  …`
`else`
`  …`
`endif`
	`if …:`
	`  …`
	`elif …:`
	`  …`
	`else:`
	`  …`

`with TYPE OBJECT`
`  …`
`endwith`
	Sin equivalente

`var VAR1`	La declaración de variables no es obligatoria

Variables

En los scripts heredados, las variables se declaran antes de ser referenciadas, por ejemplo:

```
var mynode
set mynode = create typenode at 96 96
```

En los scripts Python, las variables se crean la primera vez que se referencian, por ejemplo:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

En los scripts heredados, las referencias a variables deben eliminarse explícitamente mediante el operador ^, por ejemplo:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Al igual que en la mayoría de lenguajes de script, esto no es necesario en los scripts Python, por ejemplo:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Tipos modelo, resultado y nodo

En los scripts heredados, a los distintos tipos de objeto (nodo, resultado y modelo) se les suele añadir el tipo al tipo de objeto. Por ejemplo, el nodo Derivar es de tipo `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

El API de IBM SPSS Modeler en Python no incluye el sufijo node, de modo que el nodo Derivar tiene el tipo `derive`, por ejemplo:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

La única diferencia en los tipos de nombre entre los scripts Python y los heredados es la ausencia del sufijo de tipo.

Nombres de propiedades

Los nombres de las propiedades son los mismos en scripts heredados y en scripts Python. Por ejemplo, en el nodo Archivo variable, la propiedad que define la ubicación del archivo es `full_filename` en ambos entornos de creación de scripts.

Referencias de nodos

Muchos scripts de herencia utilizan una búsqueda implícita para buscar y acceder al nodo que se ha de modificar. Por ejemplo, los comandos siguientes buscan en la ruta actual un nodo Type con la etiqueta "Type", a continuación, establecen la dirección (o el rol de modelado) del campo "Age" como entrada y el campo "Drug" como destino, esto es, el valor predicho:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

En los scripts Python, los objetos de nodo se han de localizar de forma explícita antes de llamar a la función para establecer el valor de propiedad, por ejemplo:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Nota: En este caso, "Target" debe estar encerrado entre comillas en la cadena.

Los scripts Python pueden utilizar de forma alternativa la enumeración `ModelingRole` del paquete `modeler.api`.

Aunque la versión de los scripts Python puede ser más verbosa, el rendimiento de tiempo de ejecución es mejor ya que la búsqueda del nodo generalmente solo se realiza una vez. En el ejemplo de scripts de herencia, la búsqueda del nodo se realiza para cada comando.

También está soportado buscar nodos por ID (el ID de nodo se puede ver en la pestaña Anotaciones del diálogo del nodo). Por ejemplo, en los scripts de herencia:

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

El script siguiente muestra el mismo ejemplo en scripts Python:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Obtener y establecer propiedades

Los scripts de herencia utilizan el comando `set` para asignar un valor. El término que sigue al comando `set` puede ser una definición de propiedad. El script siguiente muestra dos formatos de script posibles para establecer una propiedad:

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

En los scripts Python, se obtiene el mismo resultado utilizando las funciones `setPropertyValue()` y `setKeyedPropertyValue()`, por ejemplo:

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

En los scripts de herencia, se puede acceder a los valores de las propiedades utilizando el comando `get`, por ejemplo:

```
var n v
set n = get node :filternode
set v = ^n.name
```

En los scripts Python, se obtiene el mismo resultado utilizando la función `getPropertyValue()`, por ejemplo:

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Edición de rutas

En los scripts de herencia, se utiliza el comando `create` para crear un nodo nuevo, por ejemplo:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

En los scripts Python, las rutas tienen varios métodos para crear nodos, por ejemplo:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

En los scripts de herencia, se utiliza el comando `connect` para crear enlaces entre nodos, por ejemplo:

```
connect ^agg to ^select
```

En los scripts Python, se utiliza el método `link` para crear enlaces entre nodos, por ejemplo:

```
stream.link(agg, select)
```

En los scripts de herencia, se utiliza el comando `disconnect` para eliminar enlaces entre nodos, por ejemplo:

```
disconnect ^agg from ^select
```

En los scripts Python, se utiliza el método `unlink` para eliminar enlaces entre nodos, por ejemplo:

```
stream.unlink(agg, select)
```

En los scripts de herencia, se utiliza el comando `position` para posicionar los nodos en el lienzo de rutas o entre nodos, por ejemplo:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

En los scripts Python, se obtiene el mismo resultado utilizando dos métodos separados: `setXYPosition` y `setPositionBetween`. Por ejemplo:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Operaciones de nodo

Algunos de los comandos de operaciones de nodo que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 283. Correlación de scripts de herencia con scripts Python para operaciones de nodo

Scripts de herencia	**scripts Python**

`create nodespec at x y`	`stream.create(type, name)`
	`stream.createAt(type, name, x, y)`
	`stream.createBetween(type, name, preNode, postNode)`
	`stream.createModelApplier(model, name)`

`connect fromNode to toNode`	`stream.link(fromNode, toNode)`
`delete node`	`stream.delete(node)`
`disable node`	`stream.setEnabled(node, False)`
`enable node`	`stream.setEnabled(node, True)`
`disconnect fromNode from toNode`	`stream.unlink(fromNode, toNode)` `stream.disconnect(node)`
`duplicate node`	`node.duplicate()`
`execute node`	`stream.runSelected(nodes, results)` `stream.runAll(results)`
`flush node`	`node.flushCache()`
`position node at x y`	`node.setXYPosition(x, y)`
`position node between node1 and node2`	`node.setPositionBetween(node1, node2)`
`rename node as name`	`node.setLabel(name)`

Bucle

En los scripts de herencia, hay dos opciones de bucle principales a las que se da soporte:

- Bucles de *Valor contado*, en los que una variable de índice se mueve entre dos límites de entero.
- Bucles de *secuencia* que avanzan en bucle por una secuencia de valores, enlazando el valor actual con la variable de bucle.

El script siguiente es un ejemplo de un bucle de valor contado en un script de herencia:

```
for i from 1 to 10
    println ^i
endfor
```

El script siguiente es un ejemplo de un bucle de secuencia en un script de herencia:

```
var items
set items = [a b c d]

for i in items
    println ^i
endfor
```

También existen otros tipos de bucles que se pueden utilizar:

- Iteración por los modelos de la paleta de modelos o por los resultados de la paleta de resultados.
- Iteración por los campos de entrada o salida de un nodo.

Los scripts Python también dan soporte a diferentes tipos de bucles. El script siguiente es un ejemplo de un bucle de valor contado en un script Python:

```
i = 1
while i <= 10:
    print i
    i += 1
```

El script siguiente es un ejemplo de un bucle de secuencia en un script Python:

```
items = ["a", "b", "c", "d"]
for i in items:
    print i
```

El bucle de secuencia es muy flexible y cuando se combina con los métodos de la API de IBM SPSS Modeler puede dar soporte a la mayoría de los casos de uso scripts de herencia. El siguiente ejemplo muestra cómo utilizar un bucle de secuencia en scripts Python para iterar por los campos de salida de un nodo:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
    print column.getColumnName()
```

Ejecución de rutas

Durante la ejecución de la ruta, el modelo o los objetos de resultados que se generan se añaden a uno de los gestores de objeto. En el script existente, el script debe localizar los objetos creados desde el gestor de objeto, o acceder al resultado generado más recientemente desde el nodo que ha generado el resultado.

La ejecución de rutas en Python es diferente, ya que cualquier objeto de modelo o resultados que genere la ejecución se devuelve una lista que se pasa a la función de ejecución. Esto hace que resulte más sencillo acceder a los resultados de la ejecución de la ruta.

Los scripts de herencia dan soporte a tres comandos de ejecución de ruta:

- `execute_all` ejecuta todos nodos terminales ejecutables en la ruta.
- `execute_script` ejecuta el script de ruta independientemente del valor de la ejecución del script.
- `execute` *`node`* ejecuta el nodo especificado.

Los scripts Python dan soporte a un conjunto de funciones similares:

- *`stream`*`.runAll(`*`results-list`*`)` ejecuta todos nodos terminales ejecutables en la ruta.
- *`stream`*`.runScript(`*`results-list`*`)` ejecuta el script de ruta independientemente del valor de la ejecución del script.
- *`stream`*`.runSelected(`*`node-array, results-list`*`)` ejecuta el conjunto de nodos especificado en el orden en que se suministran.
- *`node`*`.run(`*`results-list`*`)` ejecuta el nodo especificado.

En los scripts de herencia, la ejecución de ruta se puede finalizar con el comando `exit` con un código de entero opcional, por ejemplo:

```
exit 1
```

En los scripts Python, se puede obtener el mismo resultado con el script siguiente:

```
modeler.script.exit(1)
```

Acceso a objetos mediante el sistema de archivos y el repositorio

En los scripts heredados se puede abrir una ruta, un modelo o un resultado existentes mediante el comando open, por ejemplo:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

En los scripts Python, existe la clase TaskRunner, accesible desde la sesión, que puede utilizarse para realizar tareas similares, por ejemplo:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Para guardar un objeto en los scripts heredados, puede utilizarse el comando save, por ejemplo:

```
save stream s as "c:/my streams/new_modeling.str"
```

El enfoque de un script Python consiste en utilizar la clase TaskRunner, por ejemplo:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Las operaciones basadas en un IBM SPSS Collaboration and Deployment Services Repository se soportan en los scripts heredados mediante los comandos retrieve y store, por ejemplo:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

En los scripts Python, se accede a la funcionalidad equivalente a través del objeto Repository asociado a la sesión, por ejemplo:

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

Nota: El acceso al repositorio exige que la sesión se haya configurado con una conexión de repositorio válida.

Operaciones de ruta

Algunos comandos de operación de ruta que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 284. Correlación de scripts de herencia con scripts Python para operaciones de ruta

Scripts de herencia	**scripts Python**
`create stream` *DEFAULT_FILENAME*	*taskrunner*`.createStream(`*name*`, `*autoConnect*`, `*autoManage*`)`
`close stream`	*stream*`.close()`
`clear stream`	*stream*`.clear()`
`get stream` *stream*	Sin equivalente
`load stream` *path*	Sin equivalente
`open stream` *path*	*taskrunner*`.openStreamFromFile(`*path*`, `*autoManage*`)`

Tabla 284. Correlación de scripts de herencia con scripts Python para operaciones de ruta (continuación)	
Scripts de herencia	**scripts Python**
`save stream as path`	`taskrunner.saveStreamToFile(stream, path)`
`retreive stream path`	`repository.retreiveStream(path, version, label, autoManage)`
`store stream as path`	`repository.storeStream(stream, path, label)`

Operaciones de modelo

Algunos de los comandos de operación de modelo que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 285. Correlación de scripts de herencia con scripts Python para operaciones de modelo	
Scripts de herencia	**scripts Python**
`open model path`	`taskrunner.openModelFromFile(path, autoManage)`
`save model as path`	`taskrunner.saveModelToFile(model, path)`
`retrieve model path`	`repository.retrieveModel(path, version, label, autoManage)`
`store model as path`	`repository.storeModel(model, path, label)`

Operaciones de resultado de documento

Algunos de los comandos de operaciones de resultado de documento que normalmente se utilizan en IBM SPSS Modeler tienen sus comandos equivalentes en los scripts Python. Esto puede ayudarle a convertir los scripts de SPSS Modeler de herencia existentes en scripts Python para utilizarlos en IBM SPSS Modeler 17.

Tabla 286. Correlación de scripts de herencia con scripts Python para operaciones de resultado de documento	
Scripts de herencia	**scripts Python**
`open output path`	`taskrunner.openDocumentFromFile(path, autoManage)`
`save output as path`	`taskrunner.saveDocumentToFile(output, path)`
`retrieve output path`	`repository.retrieveDocument(path, version, label, autoManage)`
`store output as path`	`repository.storeDocument(output, path, label)`

Otras diferencias entre scripts heredados y scripts Python

Los scripts heredados soportan la manipulación de proyectos de IBM SPSS Modeler. Los scripts Python no soportan esto actualmente.

Los scripts heredados proporcionan cierto soporte de carga de objetos de *estado* (combinaciones de rutas y modelos). Los objetos de estado han caído en desuso desde IBM SPSS Modeler 8.0. Los scripts Python no soportan objetos de estado.

Los scripts Python proporcionan las siguientes funciones adicionales no disponibles en los scripts heredados:

- Definiciones de clase y función.
- Manejo de errores.
- Soporte más sofisticado de entrada/salida.
- Módulos externos y de terceros.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en Estados Unidos. Es posible que este material esté disponible en IBM en otros idiomas. Sin embargo, deberá poseer una copia del producto o de la versión del producto en ese idioma para poder acceder a él.

Es posible que IBM no ofrezca los productos, servicios o características que se tratan en este documento en otros países. El representante local de IBM le puede informar sobre los productos y servicios que están actualmente disponibles en su localidad. Las referencias a productos, programas o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

Es posible que IBM tenga patentes o solicitudes de patente pendientes que traten el tema descrito en este documento. La entrega de este documento no le otorga ninguna licencia sobre dichas patentes. Puede enviar preguntas acerca de las licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE.UU.

Para consultas de licencia relativas a la información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país, o envíe las consultas, por escrito, a:

Intellectual Property Licensing
Derecho de propiedad intelectual y legal
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGUNA CLASE, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O ADECUACIÓN A UN FIN DETERMINADO. Algunas jurisdicciones no permiten la renuncia a las garantías explícitas o implícitas en determinadas transacciones; por lo tanto, es posible que esta declaración no sea aplicable en su caso.

Esta información podría contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM se reserva el derecho a realizar en cualquier momento y sin notificación previa, mejoras o modificaciones en los productos y programas que se describen en el presente manual.

Las referencias en esta información a sitios web no IBM se proporcionan solo por comodidad y en modo alguno constituyen un aval de dichos sitios web. El material de esos sitios web no forma parte del material de este producto de IBM y el uso de dichos sitios web es responsabilidad del usuario.

IBM puede utilizar o distribuir cualquier información que proporcione en la forma que crea adecuada sin incurrir por ello en ninguna obligación con el cliente.

Los titulares de licencias de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Director of Licensing
IBM Corporation

North Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE.UU.

Dicha información puede estar disponible, sujeta a los términos y condiciones correspondientes, incluyendo, en algunos casos, el pago de una tarifa.

IBM proporciona el programa bajo licencia descrito en esta información y todo el material bajo licencia disponible para él de acuerdo con los términos del Contrato del cliente de IBM, el Acuerdo internacional de licencia de programas de IBM o cualquier contrato equivalente existente entre las partes.

Los datos de rendimiento y los ejemplos de cliente se presentan únicamente con fines ilustrativos. Los resultados reales del rendimiento pueden variar en función de determinadas configuraciones y condiciones de funcionamiento.

La información acerca de productos que no son de IBM se ha obtenido de los proveedores de dichos productos, sus publicaciones anunciadas u otras fuentes disponibles públicamente. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad ni contemplar ninguna otra reclamación relacionada con los productos que no son de IBM. Las preguntas acerca de las funciones de productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Las declaraciones sobre la intención y el rumbo futuro IBM están sujetas a cambio o retirada sin previo aviso y representan únicamente metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres de personas o empresas es mera coincidencia.

Marcas comerciales

IBM, el logotipo de IBM e ibm.com son marcas comerciales o marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Existe una lista actual de las marcas registradas de IBM en la web bajo "Copyright and trademark information" en www.ibm.com/legal/copytrade.shtml.

Adobe, el logotipo de Adobe, PostScript y el logotipo de PostScript son marcas registradas o marcas comerciales registradas de Adobe Systems Incorporated en Estados Unidos o en otros países.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus filiales en los Estados Unidos de América y en otros países.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos o en otros países.

Microsoft, Windows, Windows NT, y el logotipo de Windows son marcas comerciales de Microsoft Corporation en Estados Unidos, otros países o ambos.

UNIX es una marca registrada de The Open Group en Estados Unidos y en otros países.

Java y todas las marcas registradas y logotipos basados en Java son marcas registradas de Oracle o sus filiales.

Términos y condiciones de la documentación de producto

Los permisos para utilizar estas publicaciones se otorgan de acuerdo con los términos y condiciones siguientes.

Ámbito de aplicación

Estos términos y condiciones son adicionales a los términos de uso del sitio web de IBM.

Uso personal

Puede reproducir estas publicaciones para el uso personal, no comercial puesto que se conservan los avisos de propiedad. No puede distribuir, visualizar o realizar trabajos derivados de estas publicaciones, o de partes de las mismas, sin el consentimiento expreso de IBM.

Uso comercial

Puede reproducir, distribuir y mostrar estas publicaciones exclusivamente en su empresa siempre que se conserven todos los avisos de propiedad. No podrá crear trabajo derivado de estas publicaciones, ni reproducir, distribuir ni visualizar estas publicaciones o cualquier parte de éstas sin el consentimiento expreso de IBM.

Derechos

Salvo que se indique expresamente lo contrario en este permiso, no se otorga ningún otro permiso, licencia o derecho, ya sea de manera implícita o explícita, a las publicaciones ni a cualquier información, datos, software u otros elementos sujetos a la propiedad intelectual contenidos en ella.

IBM se reserva el derecho a retirar los permisos otorgados en ellas siempre que, a su discreción, el uso de las publicaciones vaya en detrimento de sus intereses o no se sigan adecuadamente las instrucciones anteriores, tal como determina IBM.

No puede descargar, exportar ni volver a exportar esta información excepto con plena conformidad de todas las leyes y regulaciones aplicables, incluyendo la legislación y las regulaciones de exportación de EE.UU.

IBM NO GARANTIZA EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN "TAL CUAL" Y SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO PERO SIN LIMITARSE A LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

Índice

Caracteres Especiales

Árbol de decisión de MS
  propiedades de scripts de nodos 361, 363

A

acceder a resultados de ejecución de la ruta
  modelo de contenido de tabla 56
  modelo de contenido JSON 60
  modelo de contenido XML 58
adición de atributos 25
API de creación de scripts
  acceso a objetos generados 43
  buscar 39
  ejemplo 39
  introducción 39
  manejo de errores 44
  metadatos 40
  obtener un directorio 39
  parámetros de ruta 45
  parámetros de sesión 45
  parámetros de Supernodos 45
  scripts autónomos 49
  valores globales 48
  varias rutas 49
applyocsvm, propiedades 354
argumentos
  archivo de comandos 70
  conexión con el servidor 68
  conexión de IBM SPSS Collaboration and Deployment Services Repository 69
  Conexión del repositorio de IBM SPSS Analytic Server 70
  sistema 66
atravesar los nodos 34

B

bloques de código 20
bucles
  uso en scripts 51
bucles en rutas 6, 7
buscar nodos 29

C

cadenas
  cambio entre minúsculas y mayúsculas 51
campos
  desactivación de los scripts 193
caracteres que no son ASCII 23
clave de iteración
  bucle en scripts 8
CLEM
  scripts 1
Clúster de secuencias de MS
  propiedades de scripts de nodos 363
comando clear generated palette 55
comando de conjunto múltiple 73
comando for 51
comando retrieve 51
comando store 51
comentarios 19
comprobación de errores
  scripts 54
configuración de propiedades 30
contraseñas
  adición a scripts 54
  codificadas 68
contraseñas codificadas
  adición a scripts 54
creación de nodos 31–33
crear una clase 25

D

de entorno
  scripts 16
definición de atributos 25
definir métodos 25
definir una clase 24
derive_stbnode
  propiedades 127
Diagramas 27
distintivos
  argumentos de la línea de comandos 65
  combinación de varias marcas 70

E

ejecución condicional de rutas 6, 10
Ejecución de rutas 27
ejecución de scripts 11
ejemplos 21
exportModelToFile 43

F

función lowertoupper 51
funciones
  bucle 471
  comentarios 470
  condicionales 471
  literales 470
  operaciones de modelo 478
  operaciones de nodo 475
  operaciones de resultado de documento 478
  operaciones de ruta 477
  operadores 470
  referencias de objeto 470
funciones de cadena 51

I

IBM SPSS Collaboration and Deployment Services Repository
  argumentos de la línea de comandos 69
  scripts 51
IBM SPSS Modeler
  ejecución desde la línea de comandos 65
identificadores 19
inheritance 26
instrucciones 19
interrupción de scripts 11

J

Jython 15

L

línea de comandos
  ejecutar IBM SPSS Modeler 65
  lista de argumentos 66, 68–70
  parámetros 67
  scripts 55
  varios argumentos 70
listas 16

M

métodos matemáticos 21
migrar
  acceder a objetos 477
  borrar rutas, salida y gestores de modelos 35
  bucle 475
  comandos 469
  configuración de propiedades 473
  contexto de los scripts 469
  diferencias generales 469
  editar rutas 474
  ejecución de rutas 476
  funciones 469
  nombres de propiedad 473
  obtener propiedades 473
  referencias de nodos 473
  repositorio 477
  sistema de archivos 477
  tipos de modelos 472
  tipos de nodo 472
  tipos de resultados 472
  Variables 472
  varios 478
  visión general 469
modelado de bases de datos 361
modelo de contenido de tabla 56
modelo de contenido JSON 60
modelo de contenido XML 58
modelos
  nombres de scripts 465, 467
modelos apriori
  propiedades de scripts de nodos 221, 339
Modelos Apriori de Oracle
  propiedades de scripts de nodos 366, 372
modelos Árboles aleatorios
modelos Árboles aleatorios *(continuación)*
  propiedades de scripts de nodos 299, 355
modelos autonuméricos
  propiedades de scripts de nodos 231
Modelos autonuméricos
  propiedades de scripts de nodos 342
Modelos bayesianos adaptativos de Oracle
  propiedades de scripts de nodos 366, 372
modelos bietápicos
  propiedades de scripts de nodos 334, 358
modelos bietápicos AS
  propiedades de scripts de nodos 335, 358
modelos C5.0
  propiedades de scripts de nodos 235, 343
modelos CARMA
  propiedades de scripts de nodos 237, 344
modelos causales temporales
  propiedades de scripts de nodos 315
modelos CHAID
  propiedades de scripts de nodos 241, 344
Modelos de agrupación en clústeres divisivo de Netezza
  propiedades de scripts de nodos 373, 390
modelos de árbol C&R
  propiedades de scripts de nodos 238, 344
Modelos de árbol de decisión de Oracle
  propiedades de scripts de nodos 366, 372
modelos de árboles de decisión de Netezza
  propiedades de scripts de nodos 373, 390
Modelos de árboles de regresión de Netezza
  propiedades de scripts de nodos 373, 390
modelos de Autoclúster
  propiedades de scripts de nodos 342
Modelos de bayesiano ingenuo de Netezza
  propiedades de scripts de nodos 373, 390
Modelos de bayesiano ingenuo de Oracle
  propiedades de scripts de nodos 366, 372
Modelos de clasificador automático
  propiedades de scripts de nodos 341
modelos de detección de anomalías
  propiedades de scripts de nodos 220, 339
modelos de IBM SPSS Statistics
  propiedades de scripts de nodos 436
modelos de K-Means
  propiedades de scripts de nodos 275, 351
modelos de K-medias de Netezza
  propiedades de scripts de nodos 373, 390
Modelos de K-medias de Oracle
  propiedades de scripts de nodos 366, 372
modelos de la máquina de vectores de soporte
  propiedades de scripts de nodos 313, 356
modelos de la máquina de vectores de soporte lineal
  propiedades de scripts de nodos 290, 352
modelos de listas de decisiones
  propiedades de scripts de nodos 246, 345
Modelos de máquinas de vectores de soporte de Oracle
  propiedades de scripts de nodos 366, 372
modelos de Microsoft
  propiedades de scripts de nodos 361, 363
Modelos de Netezza
  propiedades de scripts de nodos 373
Modelos de NMF de Oracle
  propiedades de scripts de nodos 366, 372
modelos de Oracle
  propiedades de scripts de nodos 366

modelos de red bayesiana
propiedades de scripts de nodos 233
Modelos de red bayesiana de Netezza
propiedades de scripts de nodos 373, 390
modelos de red neuronal
propiedades de scripts de nodos 291, 352
Modelos de redes bayesianas
propiedades de scripts de nodos 343
Modelos de regresión de Cox
propiedades de scripts de nodos 244, 345
modelos de regresión lineal
propiedades de scripts de nodos 301, 355, 356
Modelos de regresión lineal de Netezza
propiedades de scripts de nodos 373, 390
modelos de regresión logística
propiedades de scripts de nodos 284, 352
modelos de respuesta de autoaprendizaje
propiedades de scripts de nodos 305, 356
modelos de secuencias
propiedades de scripts de nodos 304, 356
modelos de selección de características
aplicar 5
propiedades de scripts de nodos 255, 348
scripts 5
modelos de serie temporal
propiedades de scripts de nodos 320, 357
modelos de series temporales
propiedades de scripts de nodos 320, 328, 357
Modelos de series temporales de Netezza
propiedades de scripts de nodos 373
modelos del vecino más próximo
propiedades de scripts de nodos 277
modelos discriminantes
propiedades de scripts de nodos 248, 346
modelos generados
nombres de scripts 465, 467
modelos GLE
propiedades de scripts de nodos 268, 350
Modelos GLMM
propiedades de scripts de nodos 263, 349
modelos K-Medias-AS
propiedades de scripts de nodos 276, 457
modelos KDE
propiedades de scripts de nodos 359
modelos KNN
propiedades de scripts de nodos 351
Modelos KNN de Netezza
propiedades de scripts de nodos 373, 390
modelos kohonen
propiedades de scripts de nodos 279
modelos Kohonen
propiedades de scripts de nodos 351
modelos lineales
propiedades de scripts de nodos 280, 351
modelos lineales AS
propiedades de scripts de nodos 282, 352
modelos lineales generalizados
propiedades de scripts de nodos 257, 348
Modelos lineales generalizados de Netezza
propiedades de scripts de nodos 373
Modelos lineales generalizados de Oracle
propiedades de scripts de nodos 366
modelos LSVM
propiedades de scripts de nodos 290
Modelos Oracle AI
propiedades de scripts de nodos 366
Modelos para LMD de Oracle
propiedades de scripts de nodos 366, 372
modelos PCA
propiedades de scripts de nodos 253, 348
Modelos PCA de Netezza
propiedades de scripts de nodos 373, 390
modelos PCA/Factorial
propiedades de scripts de nodos 253, 348
modelos QUEST
propiedades de scripts de nodos 296, 354
modelos Serie temporal de modalidad continua
propiedades de scripts de nodos 142
modelos SLRM
propiedades de scripts de nodos 305, 356
modelos SVM
propiedades de scripts de nodos 313
modelos tcm
propiedades de scripts de nodos 357
modelos Tree-AS
propiedades de scripts de nodos 331, 358
modificar rutas 31, 34

N

Nodo Adición de RFM
propiedades 134
nodo Agregar
propiedades 121
Nodo Agrupación en clústeres automática
propiedades de scripts de nodos 229
Nodo Análisis
Propiedades 393
Nodo Análisis de RFM
propiedades 174
nodo Anonimizar
propiedades 153
nodo Añadir
propiedades 121
nodo Archivo fijo
propiedades 101
nodo Archivo var.
propiedades 113
Nodo Auditoría de datos
propiedades 394
nodo Base de datos
propiedades 93
nodo binning
propiedades 159
Nodo Bosque aleatorio
propiedades 448
Nodo Cajas-espacio-tiempo
propiedades 127, 140
Nodo Clasificador automático
propiedades de scripts de nodos 226
nodo Colección
propiedades 194
Nodo Conjunto
propiedades 166
nodo de datos de usuario
propiedades 112
Nodo de Exportación de extensión
propiedades 425

Nodo de exportación de IBM SPSS Statistics Collection
propiedades 437
nodo de exportación de Recopilación de datos
propiedades 423
Nodo de exportación Excel
propiedades 426
Propiedades 424
nodo de exportación XML
propiedades 433
Nodo de importación de extensión
propiedades 99
Nodo de mezcla gausiana
propiedades 439, 444
Nodo de Modelo de extensión
propiedades de scripts de nodos 250
Nodo de origen de Excel
propiedades 98
nodo de origen de IBM SPSS Statistics
propiedades 435
Nodo de origen de importación de TWC
propiedades 111
Nodo de origen geoespacial
propiedades 106
nodo de origen JSON
Propiedades 106
nodo de origen Recopilación de datos
propiedades 94
Nodo de origen SAS
propiedades 107
Nodo de origen XML
propiedades 119
nodo de predicción espaciotemporal
propiedades 307
Nodo de reglas de asociación
propiedades 223
Nodo de Resultados de extensión
propiedades 396
nodo de simulación de ajuste
Propiedades 407
nodo de simulación de evaluación
propiedades 406
nodo de simulación de KDE
propiedades 398, 443
nodo de transformación
propiedades 412
Nodo de Transformación de extensión
propiedades 131
Nodo de transformación de IBM SPSS Statistics Collection
propiedades 435
nodo de visualización de mapas
propiedades 204
nodo Derivar
propiedades 162
nodo Distinguir
propiedades 129
nodo Distribución
propiedades 195
nodo Equilibrar
propiedades 123
nodo Estadísticos
propiedades 408
nodo Evaluación
propiedades 196
nodo Exportar base de datos
nodo Exportar base de datos *(continuación)*
propiedades 418
Nodo Exportar SAS
propiedades 428
nodo Filtrar
propiedades 168
nodo flatfilenode
propiedades 427
nodo Fundir
propiedades 132
nodo G. múltiple
propiedades 209
nodo Generación de R
propiedades de scripts de nodos 235
nodo Generar simulación
propiedades 107
nodo Gráfico
propiedades 210
nodo Gráfico de tiempo
propiedades 213
nodo Gráfico electrónico
propiedades 214
nodo HDBSCAN
propiedades 440
nodo Histograma
propiedades 203
nodo Historial
propiedades 169
nodo Informe
propiedades 403
nodo Intervalos de tiempo
propiedades 177
nodo Intervalos de tiempo AS
propiedades 158
Nodo Isotónica-AS
propiedades 457
nodo Malla
propiedades 217
nodo Malla direccional
propiedades 217
Nodo Marcas
propiedades 176
nodo Matriz
Propiedades 399
nodo Medias
propiedades 401
nodo Modelado KDE
propiedades 442
nodo Muestrear
propiedades 137
nodo Ordenar
propiedades 139
nodo origen Analytic Server
propiedades 88
nodo Origen de IBM Cognos
propiedades 89
nodo Origen de IBM Cognos TM1
propiedades 109, 110
nodo Partición
Propiedades 170
Nodo Perceptrón multicapa-
AS
propiedades 459
nodo Reclasificar

nodo Reclasificar *(continuación)*
propiedades 171
nodo Reestructurar
propiedades 173
nodo Rellenar
propiedades 167
nodo Reordenar
propiedades 172
nodo Reorg. campos
propiedades 172
nodo Resultado de R
propiedades 404
nodo Resultados de IBM SPSS Statistics
propiedades 437
nodo Seleccionar
propiedades 139
Nodo Serie temporal de modalidad continua
propiedades 149
nodo Sim Eval
propiedades 406
nodo Sim Fit
Propiedades 407
nodo Sim Gen
propiedades 107
Nodo SMOTE
propiedades 450
nodo STP
propiedades 307
Nodo SVM de una clase
propiedades 445
nodo t-SNE
propiedades 215, 451
nodo Tabla
propiedades 409
Nodo Tablero
propiedades 198
nodo Tipo
propiedades 185
Nodo Transformación de R
Propiedades 136
nodo Transponer
propiedades 183
nodo Val. globales
propiedades 405
Nodo XGBoost Linear
propiedades 453
Nodo XGBoost Tree
propiedades 454
Nodo XGBoost-AS
propiedades 459
nodos
desenlazar nodos 32
enlazar nodos 32
importar 33
información 35
recorrido en bucle en scripts 51
referencia de nombres 465
suprimir 33
sustitución 33
nodos de exportación
propiedades de scripts de nodos 415
nodos de gráficos
propiedades de los scripts 193
nodos de modelado
nodos de modelado *(continuación)*
propiedades de scripts de nodos 219
nodos de origen
Propiedades 81
nodos de resultados
propiedades de los scripts 393
nombres de campos
cambio entre minúsculas y mayúsculas 51
nugget
propiedades de scripts de nodos 339
nugget de nodo de reglas de asociación
Propiedades 340
nugget de nodo STP
propiedades 356
nuggets de modelo
nombres de scripts 465, 467
propiedades de scripts de nodos 339

O

O-clúster de Oracle
propiedades de scripts de nodos 366, 372
objetos de resultados
nombres de scripts 467
objetos del modelo
nombres de scripts 465, 467
ocsvmnode, propiedades 445
operaciones 16
Optimización de CPLEX, nodo
Propiedades 123
orden de ejecución
modificación con scripts 51
orden de ejecución de rutas
modificación con scripts 51
orientado a objetos 24

P

palabra clave generada 55
parámetros
scripts 16
Supernodos 463
parámetros de intervalo 5, 73, 75
pasar argumentos 20
preparación automática de datos
propiedades 154
properties autodataprepnode 154
propiedad stream.nodes 51
propiedades
nodos de modelado de bases de datos 361
nodos Filtrar 73
ruta 77
scripts 73, 75, 219, 415
scripts comunes 76
Supernodos 463
Propiedades
scripts 75, 339
Propiedades applygmm 350
propiedades applyrandomtrees 355
propiedades applyts 357
propiedades associationrulesnode 223
Propiedades cplexoptnode 123
propiedades de aggregatenode 121

propiedades de analysisnode 393
propiedades de anomalydetectionnode 220
propiedades de anonymizenode 153
propiedades de appendnode 121
propiedades de applyanomalydetectionnode 339
propiedades de applyapriorinode 339
propiedades de applyassociationrulesnode 340
propiedades de applyautoclassifiernode 341
Propiedades de applyautoclusternode 342
propiedades de applyautonumericnode 342
propiedades de applybayesnetnode 343
propiedades de applyc50node 343
propiedades de applycarmanode 344
propiedades de applycartnode 344
propiedades de applychaidnode 344
propiedades de applycoxregnode 345
propiedades de applydecisionlistnode 345
propiedades de applydiscriminantnode 346
Propiedades de applyextension 346
propiedades de applyfactornode 348
propiedades de applyfeatureselectionnode 348
propiedades de applygeneralizedlinearnode 348
Propiedades de applygle 350
Propiedades de applyglmmnode 349
propiedades de applykmeansnode 351
propiedades de applyknnnode 351
propiedades de applykohonennode 351
propiedades de applylinearasnode 352
Propiedades de applylinearnode 351
propiedades de applylogregnode 352
propiedades de applylsvmnode 352
propiedades de applymslogisticnode 363
propiedades de applymsneuralnetworknode 363
propiedades de applymsregressionnode 363
propiedades de applymssequenceclusternode 363
propiedades de applymstimeseriesnode 363
propiedades de applymstreenode 363
Propiedades de applynetezzabayesnode 390
Propiedades de applynetezzadectreenode 390
Propiedades de applynetezzadivclusternode 390
Propiedades de applynetezzakmeansnode 390
Propiedades de applynetezzaknnnode 390
Propiedades de applynetezzalineregressionnode 390
Propiedades de applynetezzanaivebayesnode 390
Propiedades de applynetezzapcanode 390
Propiedades de applynetezzaregtreenode 390
propiedades de applyneuralnetnode 352
propiedades de applyneuralnetworknode 353
propiedades de applyoraabnnode 372
propiedades de applyoradecisiontreenode 372
propiedades de applyorakmeansnode 372
propiedades de applyoranbnode 372
propiedades de applyoranmfnode 372
propiedades de applyoraoclusternode 372
propiedades de applyorasvmnode 372
propiedades de applyquestnode 354
propiedades de applyr 355
Propiedades de applyregressionnode 356
propiedades de applyselflearningnode 356
propiedades de applysequencenode 356
propiedades de applystpnode 356
propiedades de applysvmnode 356
propiedades de applytcmnode 357
propiedades de applytimeseriesnode 357
propiedades de applytreeas 358
propiedades de applytwostepAS 358
propiedades de applytwostepnode 358
propiedades de applyxgboostlinearnode 359
Propiedades de applyxgboosttreenode 359
propiedades de apriorinode 221
Propiedades de asexport 415
Propiedades de asimport 88
propiedades de astimeintervalsnode 158
propiedades de autoclassifiernode 226
propiedades de autonumericnode 231
propiedades de balancenode 123
propiedades de bayesnet 233
propiedades de binningnode 159
propiedades de buildr 235
propiedades de c50node 235
propiedades de carmanode 237
propiedades de cartnode 238
propiedades de chaidnode 241
propiedades de collectionnode 194
propiedades de coxregnode 244
propiedades de dataauditnode 394
propiedades de databaseexportnode 418
propiedades de databasenode 93
propiedades de datacollectionexportnode 423
propiedades de datacollectionimportnode 94
propiedades de decisionlist 246
propiedades de derivenode 162
Propiedades de directedwebnode 217
propiedades de discriminantnode 248
propiedades de distinctnode 129
propiedades de distributionnode 195
propiedades de ensemblenode 166
propiedades de evaluationnode 196
Propiedades de excelexportnode 424, 426
propiedades de excelimportnode 98
Propiedades de extensionexportnode 425
Propiedades de extensionimportnode 99
Propiedades de extensionmodelnode 250
Propiedades de extensionoutputnode 396
Propiedades de extensionprocessnode 131
propiedades de factornode 253
propiedades de featureselectionnode 5, 255
propiedades de fillernode 167
propiedades de filternode 168
Propiedades de fixedfilenode 101
propiedades de flatfilenode 427
propiedades de genlinnode 257
Propiedades de gle 268
Propiedades de glmmnode 263
Propiedades de graphboardnode 198
Propiedades de hdbscannugget 359
propiedades de histogramnode 203
propiedades de historynode 169
propiedades de kdeapply 359
propiedades de kmeansnode 275
propiedades de knnnode 277
propiedades de kohonennode 279
Propiedades de logregnode 284
propiedades de lsvmnode 290
Propiedades de mapvisualization 204
propiedades de matrixnode 399
propiedades de meansnode 401
propiedades de mergenode 132

propiedades de msassocnode 361
propiedades de msbayesnode 361
propiedades de msclusternode 361
propiedades de mslogisticnode 361
propiedades de msneuralnetworknode 361
propiedades de msregressionnode 361
propiedades de mssequenceclusternode 361
propiedades de mstimeseriesnode 361
propiedades de mstreenode 361
propiedades de multiplotnode 209
Propiedades de netezzabayesnode 373
Propiedades de netezzadectreenode 373
Propiedades de netezzadivclusternode 373
Propiedades de netezzaglmnode 373
Propiedades de netezzakmeansnode 373
Propiedades de netezzaknnnode 373
Propiedades de netezzalineregressionnode 373
Propiedades de netezzanaivebayesnode 373
Propiedades de netezzapcanode 373
Propiedades de netezzaregtreenode 373
Propiedades de netezzatimeseriesnode 373
propiedades de neuralnetnode 291
Propiedades de neuralnetwork 294
propiedades de nodo de agrupación en clústeres automática 229
propiedades de numericpredictornode 231
Propiedades de oraabnnode 366
Propiedades de oraainode 366
Propiedades de oraapriorinode 366
Propiedades de oradecisiontreenode 366
Propiedades de oraglmnode 366
Propiedades de orakmeansnode 366
Propiedades de oramdlnode 366
Propiedades de oranbnode 366
Propiedades de oranmfnode 366
Propiedades de oraoclusternode 366
Propiedades de orasvmnode 366
propiedades de outputfilenode 427
propiedades de partitionnode 170
propiedades de plotnode 210
propiedades de questnode 296
propiedades de reclassifynode 171
Propiedades de regressionnode 301
propiedades de reordernode 172
propiedades de reportnode 403
propiedades de reprojectnode 173
propiedades de restructurenode 173
propiedades de rfmanalysisnode 174
propiedades de routputnode 404
Propiedades de Rprocessnode 136
propiedades de samplenode 137
propiedades de sasexportnode 428
propiedades de sasimportnode 107
propiedades de scripts de nodos
  nodos de exportación 415
  nodos de modelado 219
  nuggets de modelo 339
propiedades de selectnode 139
propiedades de sequencenode 304
propiedades de setglobalsnode 405
propiedades de settoflagnode 176
propiedades de simevalnode 406
propiedades de simfitnode 407
propiedades de simgennode 107
propiedades de slrmnode 305
propiedades de sortnode 139
Propiedades de spacetimeboxes 140
propiedades de statisticsexportnode 437
propiedades de statisticsimportnode 5, 435
propiedades de statisticsmodelnode 436
propiedades de statisticsnode 408
propiedades de statisticsoutputnode 437
propiedades de statisticstransformnode 435
propiedades de stpnode 307
propiedades de svmnode 313
propiedades de tablenode 409
propiedades de tcmnode 315
propiedades de timeintervalsnode 177
propiedades de timeplotnode 213
propiedades de timeseriesnode 328
propiedades de transformnode 412
propiedades de transposenode 183
propiedades de treeas 331
propiedades de twostepAS 335
propiedades de twostepnode 334
propiedades de typenode 5, 185
propiedades de userinputnode 112
Propiedades de variablefilenode 113
propiedades de webnode 217
propiedades de xmlexportnode 433
propiedades de xmlimportnode 119
Propiedades del nodo Cajas-Espacio-Tiempo 127
propiedades del nodo cognosimport 89
Propiedades del nodo gsdata_import 106
Propiedades del nodo tm1import 110
Propiedades del nodo tm1odataimport 109
Propiedades del nodo twcimport 111
propiedades eplotnode 214
propiedades estructuradas 73
Propiedades gmm 439, 444
Propiedades hdbscannode 440
propiedades isotonicasnode 457
propiedades jsonimportnode 106
Propiedades kdeexport 398, 443
Propiedades kdemodel 442
Propiedades kmeansasnode 276, 457
propiedades lineales 280
propiedades linear-AS 282
propiedades multilayerperceptronnode 459
propiedades randomtrees 299
propiedades rfmaggregatenode 134
Propiedades rfnode 448
propiedades streamingtimeseries 142
propiedades streamingts 149
propiedades ts 320
propiedades tsnenode 215, 451
propiedades xgboostasnode 459
Python
  scripts 16
Python, modelos
  propiedades de script del nodo de mezcla gausiana 350
  propiedades de scripts de nodos 354, 359

R

Red neuronal de MS
  propiedades de scripts de nodos 361, 363
redes neuronales

redes neuronales *(continuación)*
propiedades de scripts de nodos 294, 353
referencia a nodos
buscar nodos 29
configuración de propiedades 30
Regresión lineal de MS
propiedades de scripts de nodos 361, 363
Regresión logística de MS
propiedades de scripts de nodos 361, 363
Repositorio de IBM SPSS Analytic Server Repository
argumentos de la línea de comandos 70
Reproyección, nodo
propiedades 173
resultados de ejecución de la secuencia, acceder
modelo de contenido de tabla 56
modelo de contenido JSON 60
modelo de contenido XML 58
rutas
bucle 6, 7
comando de conjunto múltiple 73
ejecución 27
ejecución condicional 6, 10
modificación 31
propiedades 77
scripts 1, 27

S

scripts
abreviaturas utilizadas 74
almacenamiento 1
bucle 6, 7
bucles visuales 6, 7
clave de iteración 8
compatibilidad con versiones anteriores 55
comprobación de errores 54
contexto 28
desde la línea de comandos 55
Diagramas 27
ejecución 11
ejecución condicional 6, 10
en Supernodos 5
importación desde archivos de texto 1
interfaz de usuario 1, 4, 5
interrupción 11
modelos de selección de características 5
nodos de gráficos 193
nodos de resultados 393
orden de ejecución de rutas 51
propiedades comunes 76
rutas 1, 27
Rutas de Supernodo 27
scripts autónomos 1, 27
scripts de herencia 470, 471, 475, 477, 478
Scripts de Supernodo 1, 27
scripts Python 470, 471, 475, 477, 478
selección de campos 10
sintaxis 16, 17, 19–21, 23–26
variable de iteración 9
visión general 1, 15
scripts autónomos 1, 4, 27
seguridad
contraseñas codificadas 54, 68
Series temporales de MS
Series temporales de MS *(continuación)*
propiedades de scripts de nodos 363
servidor
argumentos de la línea de comandos 68
sistema
argumentos de la línea de comandos 66
sistema de coordenadas de reproyección
propiedades 173
smotenode, propiedades 450
supernodo 73
Supernodo
ruta 27
Supernodos
configuración de propiedades 463
parámetros 463
propiedades 463
rutas 27
scripts 1, 5, 6, 27, 463

V

variable de iteración
bucle en scripts 9
variables ocultas 26

X

xgboostlinearnode, propiedades 453
xgboosttreenode, propiedades 454