Руководство по сценариям Python и автоматизации IBM SPSS Modeler 18.3

Примечание

Прежде чем использовать эту информацию и продукт, описанный в ней, прочтите сведения в разделе “Уведомления” на стр. 499.

Информация о продукте

Это издание применимо к версии 18, выпуск 3, модификация 0 IBM® SPSS Modeler и ко всем последующим версиям и модификациям до тех пор, пока в новых изданиях не будет указано иное.

© Copyright International Business Machines Corporation .

Содержание

Глава 1. Сценарий и язык написания сценариев.....1
Обзор сценариев.....1
Типы сценариев.....1
Сценарии потока.....1
Пример сценария потока: обучение нейронной сети.....3
Предельные размеры кода Jython.....4
Автономные сценарии.....4
Пример автономного сценария: сохранение и загрузка модели.....4
Пример автономного сценария: генерирование модели выбора возможностей.....5
Сценарии надузла.....5
Пример сценария надузла.....6
Потоки с циклами и условиями.....6
Циклы в потоках.....7
Выполнение с условиями в потоках.....10
Выполнение и прерывание сценариев.....12
Найти и заменить.....12

Глава 2. Язык сценариев.....15
Обзор языка сценариев.....15
Python и Jython.....15
Сценарий Python.....16
Операции.....16
Списки.....16
Строки.....17
Комментарии.....19
Синтаксис операторов.....19
Идентификаторы.....19
Блоки кода.....20
Передача аргументов в сценарий.....20
Примеры.....21
Математические методы.....21
Использование символов не из кодового набора ASCII.....23
Объектно-ориентированное программирование.....24
Определение класса.....25
Создание экземпляра класса.....25
Добавление атрибутов к экземпляру класса.....25
Определение атрибутов классов и методов.....26
Скрытые переменные.....26
Наследование.....26

Глава 3. Сценарии в IBM SPSS Modeler.....29
Типы сценариев.....29
Потоки, потоки надузлов и диаграммы.....29
Потоки.....29
Потоки надузлов.....29
Диаграммы.....29
Выполнение потока.....30
Контекст сценариев.....30
Ссылки на существующие узлы.....31
Поиск узлов.....31

Задание свойств.....32
Создание узлов и изменение потоков.....33
Создание узлов.....33
Соединение и отсоединение узлов.....34
Импорт, замена и удаление узлов.....35
Перемещение по узлам в потоке.....36
Элементы, очистка или удаление.....37
Получение информации об узлах.....38

Глава 4. API сценариев.....41
Введение в API сценариев.....41
Пример 1: поиск узлов с помощью пользовательского фильтра.....41
Пример 2: разрешение пользователям получать информацию о каталоге или файле на основе их привилегий.....41
Метаданные: Информация о данных.....42
Доступ к сгенерированным объектам.....45
Обработка ошибок.....46
Параметры потока, сеанса и надузла.....47
Глобальные значения.....51
Работа с несколькими потоками: автономные сценарии.....52

Глава 5. Подсказки для сценариев.....53
Изменение выполнения потока.....53
Циклы по узлам.....53
Доступ к объектам IBM SPSS Collaboration and Deployment Services Repository.....53
Генерирование закодированного пароля.....56
Проверка сценария.....56
Работа со сценариями из командной строки.....57
Совместимость с предыдущими выпусками.....57
Доступ к результатам выполнения потока.....57
Модель табличного содержимого.....58
Модель содержимого XML.....60
Модель содержимого JSON.....62
Модель содержимого статистики столбцов и модель содержимого попарной статистики.....63

Глава 6. Аргументы командной строки.....67
Вызов программного обеспечения.....67
Использование аргументов командной строки.....67
Системные аргументы.....68
Аргументы параметров.....70
Аргументы соединения с сервером.....70
Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository.....72
Аргументы соединения с IBM SPSS Analytic Server.....72
Объединение нескольких аргументов.....73

Глава 7. Справочник по свойствам.....75
Справочный обзор свойств.....75
Синтаксис для свойств.....75
Примера свойств узла и потока.....77
Обзор свойств узлов.....77
Общие свойства узлов.....78

Глава 8. Свойства потока.....79

Глава 9. Свойства узла источника.....85
Общие свойства узлов источников.....85

Свойства asimport ... 93
Свойства узла cognosimport ... 94
Свойства databasenode ... 98
Свойства узла импорта собрания данных (datacollectionimportnode) ... 100
Свойства узла импорта Excel (excelimportnode) ... 104
свойства extensionimportnode ... 105
Свойства fixedfilenode ... 107
Свойства узла gsdata_import ... 112
Свойства jsonimportnode ... 112
Свойства узла импорта SAS (sasimportnode) ... 113
Свойства simgennode ... 113
Свойства узла импорта статистики (statisticsimportnode) ... 115
Свойства узла tm1odataimport ... 115
Свойства узла tm1import (объявлено устаревшим) ... 116
Свойства узла twcimport ... 117
Свойства userinputnode ... 118
Свойства узла файла переменных (variablefilenode) ... 119
Свойства узла импорта XML (xmlimportnode) ... 125

Глава 10. Запись свойств узла операций ... 127
Свойства appendnode ... 127
Свойства aggregatenode ... 127
Свойства balancenode ... 129
Свойства cplexoptnode ... 129
Свойства derive_stbnode ... 133
Свойства distinctnode ... 135
Свойства extensionprocessnode ... 137
Свойства mergenode ... 138
Свойства rfmaggregatenode ... 140
Свойства samplenode ... 143
Свойства selectnode ... 145
Свойства sortnode ... 146
Свойства spacetimeboxes ... 146
Свойства streamingtimeseries ... 148

Глава 11. Поле Свойства узла операций ... 159
Свойства anonymizenode ... 159
Свойства autodataprepnode ... 160
Свойства astimeintervalsnode ... 164
Свойства binningnode ... 165
Свойства derivenode ... 169
Свойства ensemblenode ... 172
Свойства fillernode ... 173
Свойства filternode ... 174
Свойства historynode ... 175
Свойства partitionnode ... 176
Свойства reclassifynode ... 177
Свойства reordernode ... 178
Свойства reprojectnode ... 179
Свойства restructurenode ... 179
Свойства rfmanalysisnode ... 180
Свойства settoflagnode ... 182
Свойства statisticstransformnode ... 183
Свойства timeintervalsnode (объявлено устаревшим) ... 183
Свойства transposenode ... 189
Свойства typenode ... 191

Глава 12. Свойства узла графика.....201
Общие свойства узлов графиков.....201
Свойства узла Собрание (collectionnode).....202
Свойства узла распределения (distributionnode).....203
Свойства узла оценки (evaluationnode).....204
Свойства узла панели выбора диаграмм (graphboardnode).....206
Свойства узла гистограммы (histogramnode).....211
Свойства mapvisualization.....212
Свойства узла Несколько графиков (multiplotnode).....217
Свойства plotnode.....218
Свойства узла График зависимости от времени (timeplotnode).....221
Свойства eplotnode.....222
Свойства tsnenode.....223
Свойства узла Web (webnode).....225

Глава 13. Свойства узла моделирования.....229
Общие свойства узлов моделирования.....229
Свойства anomalydetectionnode.....230
Свойства apriorinode.....232
Свойства associationrulesnode.....233
Свойства autoclassifiernode.....237
Задание свойств алгоритмов.....239
Свойства autoclusternode.....240
Свойства autonumericnode.....242
Свойства bayesnetnode.....244
Свойства c50node.....246
Свойства carmanode.....248
Свойства cartnode.....249
Свойства chaidnode.....252
Свойства coxregnode.....255
Свойства decisionlistnode.....257
Свойства discriminantnode.....259
Свойства extensionmodelnode.....261
Свойства factornode.....263
Свойства featureselectionnode.....266
Свойства genlinnode.....268
Свойства glmmnode.....274
Свойства gle.....280
Свойства kmeansnode.....288
Свойства kmeansasnode.....290
Свойства knnnode.....291
Свойства kohonennode.....293
Свойства linearnode.....294
Свойства linearasnode.....296
Свойства logregnode.....298
Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.....304
Свойства neuralnetnode.....305
Свойства neuralnetworknode.....308
Свойства questnode.....310
Свойства randomtrees.....313
Свойства regressionnode.....315
Свойства sequencenode.....318
Свойства slrmnode.....319
Свойства statisticsmodelnode.....320
Свойства stpnode.....321

Свойства svmnode....327
Свойства tcmnode....329
Свойства ts....334
Свойства treeas....345
Свойства узла двухшаговых моделей (twostepnode)....348
Свойства twostepAS....349

Глава 14. Свойства узлов слепков моделей....353
Свойства применения узла обнаружения аномалий (applyanomalydetectionnode)....353
Свойства применения узла Априори (applyapriorinode)....353
Свойства applyassociationrulesnode....354
Свойство применения узла автоклассификации (applyautoclassifiernode)....355
Свойства применения узла автокластеризации (applyautoclusternode)....357
Свойства узла автонумерации (applyautonumericnode)....357
Свойства применения узла Байесовской сети (applybayesnetnode)....357
Свойство применения узла C5.0 (applyc50node)....358
Свойства применения узла CARMA (applycarmanode)....358
Свойства применения узла CART (applycartnode)....358
Свойства применения узла CHAID (applychaidnode)....359
Свойства узла применения регрессии Кокса (applycoxregnode)....359
Свойства применения узла списка решений (applydecisionlistnode)....360
Свойства применения узла дискриминанта (applydiscriminantnode)....360
Свойства applyextension....361
Свойства применения узла факторов (applyfactornode)....362
Свойства применения узла выбора возможностей (applyfeatureselectionnode)....362
Свойства применения узла обобщенной линейной регрессии (applygeneralizedlinearnode)....363
Свойства применения узла GLMM (applyglmmnode)....363
Свойства applygle....364
Свойства applygmm....364
Свойства применения узла k-средних (applykmeansnode)....365
Свойства применения узла KNN (applyknnnode)....365
Свойства применения узла Коонена (applykohonennode)....365
Свойства применения узла линейных моделей (applylinearnode)....365
Свойства applylinearasnode....366
Свойства применения узла логистической регрессии (applylogregnode)....366
Свойства applylsvmnode....366
Свойства applyneuralnetnode....367
Свойства applyneuralnetworknode....367
Свойства applyocsvmnode....368
Свойства применения узла QUEST (applyquestnode)....368
Свойства applyrandomtrees....369
Свойства применения узла регрессии (applyregressionnode)....370
Свойства applyselflearningnode....370
Свойства применения узла последовательности (applysequencenode)....370
Свойства применения узла SVM (applysvmnode)....371
Свойства applystpnode....371
Свойства applytcmnode....371
Свойства applyts....371
Свойства applytimeseriesnode (объявлено устаревшим)....372
Свойства applytreeas....372
Свойства применения узла двухшаговых моделей (applytwostepnode)....373
Свойства applytwostepAS....373
Свойства applyxgboosttreenode....373
Свойства applyxgboostlinearnode....374
Свойства hdbscannugget....374
Свойства kdeapply....374

Глава 15. Свойства узла моделирования баз данных.....375
Свойства узлов для моделирования Microsoft.....375
Свойства узлов моделирования Microsoft.....375
Свойства слепков моделей Microsoft.....377
Свойства узлов для моделирования Oracle.....380
Свойства узлов моделирования Oracle.....380
Свойства слепков моделей Oracle.....387
Свойства узлов для моделирования IBM Netezza Analytics.....388
Свойства узлов моделирования Netezza.....388
Свойства слепков моделей Netezza.....405

Глава 16. Свойства узлов вывода.....407
Свойства analysisnode.....407
Свойства dataauditnode.....408
Свойства extensionoutputnode.....410
свойства kdeexport.....412
Свойства matrixnode.....413
Свойства meansnode.....416
Свойства reportnode.....418
Свойства setglobalsnode.....420
Свойства simevalnode.....420
Свойства simfitnode.....421
Свойства statisticsnode.....422
Свойства узла выходных данных статистики (statisticsoutputnode).....424
Свойства tablenode.....424
Свойства transformnode.....427

Глава 17. Свойства узла экспорта.....431
Общие свойства узлов экспорта.....431
Свойства asexport.....431
Свойства узла экспорта Cognos (cognosexportnode).....432
Свойства databaseexportnode.....434
Свойства узла экспорта собрания данных (datacollectionexportnode).....440
Свойства узла экспорта Excel (excelexportnode).....440
Свойства extensionexportnode.....441
Свойства jsonexportnode.....443
Свойства узла выходного файла (outputfilenode).....443
Свойства узла экспорта SAS (sasexportnode).....444
Свойства узла экспорта статистики (statisticsexportnode).....445
Свойства узла tm1odataexport.....445
Свойства узла tm1export (объявлено устаревшим).....448
Свойства узла экспорта XML (xmlexportnode).....451

Глава 18. Свойства узла IBM SPSS Statistics.....453
Свойства узла импорта статистики (statisticsimportnode).....453
Свойства statisticstransformnode.....453
Свойства statisticsmodelnode.....454
Свойства узла выходных данных статистики (statisticsoutputnode).....455
Свойства узла экспорта статистики (statisticsexportnode).....456

Глава 19. Свойства узла Python.....457
Свойства gmm.....457
Свойства hdbscannode.....458
Свойства kdemodel.....460
свойства kdeexport.....461
Свойства gmm.....462

свойства ocsvmnode ... 463
Свойства rfnode ... 466
Свойства smotenode ... 468
Свойства tsnenode ... 469
Свойства xgboostlinearnode ... 471
Свойства xgboosttreenode ... 472

Глава 20. Свойства узла Spark ... 475
Свойства isotonicasnode ... 475
Свойства kmeansasnode ... 475
Свойства multilayerperceptronnode ... 477
Свойства xgboostasnode ... 477

Глава 21. Свойства надузлов ... 481

Приложение A. Ссылки на имена узлов ... 483
Имена слепков моделей ... 483
Исключение дублирования имен моделей ... 485
Имена типов вывода ... 485

Приложение B. Перенастройка от унаследованных сценарием к сценариям Python ... 487
Обзор перенастройки унаследованных сценариев ... 487
Общие отличия ... 487
Контекст сценариев ... 487
Команды или функции ... 487
Литералы и комментарии ... 488
Операторы ... 488
Условное выполнение и циклы ... 489
Переменные ... 490
Типы узлов, объектов вывода и моделей ... 490
Имена свойств ... 491
Ссылки на узлы ... 491
Получение и задание свойств ... 491
Редактирование потоков ... 492
Операции с узлами ... 493
Циклы ... 493
Выполнение потоков ... 494
Доступ к объектам через файловую систему и репозиторий ... 495
Операции с потоками ... 495
Операции с моделями ... 496
Операции вывода документов ... 496
Другие различие между унаследованными сценариями и сценариями Python ... 497

Уведомления ... 499
Товарные знаки ... 500
Правила и условия для документации продукта ... 501

Индекс ... 503

Глава 1. Сценарий и язык написания сценариев

Обзор сценариев

Сценарии в IBM SPSS Modeler - это мощный инструмент для автоматизации процессов в пользовательском интерфейсе. Сценарии могут выполнять действия того же типа, которые выполняются с помощью мыши или клавиатуры, и их можно использовать для автоматизации задач, которые при ручном выполнении могли бы быть многократно повторяемыми или требующими большого времени.

Вы можете использовать сценарии для следующего:

- Установить конкретный порядок для выполнений узлов в потоке.
- Задавать свойства для узла, а также выполнять производные действия при помощи подмножества CLEM (языка управления для работы с выражениями).
- Задавать автоматическую последовательность действий, которые обычно выполняются с участием пользователя, например, можно построить модель и затем испытать ее.
- Сконфигурировать сложные процессы, требующие существенных взаимодействий с пользователями, например, процедуры перекрестной проверки, для которых нужны повторяющиеся действия по созданию и испытанию моделей.
- Сконфигурировать процессы, обращающиеся с потоками, например, можно взять поток обучения модели, запустить его, а затем автоматически создать соответствующий поток испытания модели.

В этой части представлены описания высокого уровня и примеры сценариев на уровне потока, автономных сценариев и сценариев на надузлах в интерфейсе IBM SPSS Modeler. Более подробная информация о языке сценариев, синтаксисе и командах представлена в последующих частях .

Прим.:

Нельзя импортировать и запускать сценарии, созданные в IBM SPSS Statistics в составе IBM SPSS Modeler.

Типы сценариев

IBM SPSS Modeler использует три типа сценариев:

- **Потоковые сценарии** хранятся как свойство потока и поэтому сохраняются и загружаются вместе с конкретным потоком. Например, можно написать потоковый сценарий для автоматизации процесса обучения и применения слепка модели. Можно задать также, чтобы при любом выполнении конкретного потока вместо содержимого холста потока запускался сценарий.
- **Автономные сценарии** не связаны с каким-то конкретным потоком и сохраняются во внешних текстовых файлах. Автономный сценарий можно использовать, например, для одновременной работы с несколькими потоками.
- **Сценарии надузла** хранятся как свойство потока надузла. Сценарии надузла доступны только на конечных надузлах. Сценарий надузла можно использовать для управления выполнением последовательности содержимого надузла. Для не конечных надузлов (источников или процессов) непосредственно в вашем потоковом сценарии можно определить свойства для надузла или содержащихся в нем узлов.

Сценарии потока

Сценарии могут использоваться для настройки операций в конкретном потоке, и они сохраняются с этим потоком. Сценарии потоков можно использовать для указания конкретного порядка

выполнения для конечных узлов в потоке. Диалоговое окно сценариев потоков используется для изменения сценария, который сохранен в текущем потоке.

Чтобы получить доступ на вкладку сценариев потоков в диалоговом окне Свойства потоков:

1. В меню **Инструменты** выберите:

   Свойства потоков > **Выполнение**

2. Перейдите на вкладку **Выполнение** для работы со сценариями для текущего потока.

Значки панели инструментов в верхней части диалогового окна сценария потока используются для следующих операций:

- Импортировать в это окно содержимое уже существующего автономного сценария.
- Сохранит сценарий в виде текстового файла.
- Распечатать сценарий.
- Добавить в конец сценарий по умолчанию.
- Отредактировать сценарий (функции откат, вырезать, копировать, вставить и другие обычные функции редактирования).
- Выполнить весь текущий сценарий.
- Выполнить выбранные в сценарии строки.
- Остановить сценарий во время выполнения. (Этот значок доступен, только когда выполняется сценарий.)
- Проверить синтаксис сценария и в случае обнаружения каких-то ошибок вывести их для изучения на нижней панели диалогового окна.

Прим.: Начиная с версии 16.0 и далее, SPSS Modeler использует язык сценариев Python. Во всех версиях до 16.0 использовался язык сценариев, уникальный для SPSS Modeler, сценарии которого теперь называются унаследованными. В зависимости от типа сценария, с которым вы работаете, на вкладке **Выполнение** выберите режим выполнения **По умолчанию (дополнительный сценарий)**, а затем выберите либо **Python**, либо **Унаследованный**.

Можно указать, будет или нет запущен этот сценарий при выполнении потока. Для запуска сценария всякий раз при выполнении потока с учетом порядка выполнения сценария выберите опцию **Запускать этот сценарий**. Этот параметр на уровне потока обеспечивает автоматизацию для ускорения построения модели. Однако по умолчанию этот сценарий игнорируется при выполнении сценариев. Даже если выбрана опция **Игнорировать этот сценарий**, вы всегда можете запустить сценарий непосредственно из этого диалогового окна.

Редактор сценариев поддерживает следующие возможности, помогающие разрабатывать сценарии:

- Выделение синтаксиса; выделяются ключевые слова, литеральные значения (строчные и числовые), комментарии.
- Нумерация строк.
- Выявление парных блоков; когда указатель помещают у начала программного блока, выделяется также соответствующий конечный блок.
- Предлагаемое автозаполнение.

Цвета и стили текста, используемые при выделении синтаксиса, можно настроить в предпочтениях экрана IBM SPSS Modeler. Для обращения к предпочтениям экрана выберите **Инструменты** > **Опции** > **Опции пользователя** и откройте вкладку **Синтаксис**.

Для вывода списка предлагаемых завершений синтаксиса выберите **Автопредложение** в контекстном меню или нажмите клавиши Ctrl + Пробел. При помощи клавиш со стрелками можно перемещаться вверх и вниз по списку; клавиша Enter вставляет выделенный текст. Для выхода из режима автозаполнений без изменения существующего текста нажмите клавишу Esc.

Вкладка **Отладка** содержит сообщения отладки; на ней можно оценить состояние сценария сразу после выполнения. Вкладка **Отладка** состоит из текстовой области только для чтения и

однострочного текстового поля. Текстовая область содержит текст, отправляемый либо на стандартное устройство вывода, либо в стандартный поток ошибок через команду в сценарии, например, через текст сообщения об ошибке. В текстовое поле пишет пользователь. Этот текст оценивается в контексте последнего выполненного сценария в этом диалоговом окне (так называемый *сценарный контекст*). Текстовая область содержит команду и полученный вывод, так что пользователь может просматривать трассировку команд. Текстовое поле вывода всегда содержит приглашение командной строки (для унаследованных сценариев это `-->`).

Новый сценарный контекст создается в следующих ситуациях:

- Запущен некоторый сценарий при помощи либо кнопки **Запустить этот сценарий**, либо кнопки **Запустить выбранные строки**.
- Изменен язык сценариев.

Когда создается новый сценарный контекст, текстовая область очищается.

Прим.: В случае выполнении потока вне панели сценариев контекст сценария панели сценариев не изменяется. Значения переменных, создаваемых в составе этого выполнения, в диалоговом окне сценария не выводятся.

Пример сценария потока: обучение нейронной сети

При выполнении поток можно использовать для обучения модели нейронной сети. Обычно для испытания модели можно запустить узел моделирования, чтобы добавить модель в поток, установить соответствующие соединения и выполнить узел анализа.

Используя сценарий IBM SPSS Modeler, можно автоматизировать процесс испытания слепка модели после того, как вы ее создали. Например, следующий сценарий потока для испытания демонстрационного потока `druglearn.str` (доступного в папке `/Demos/streams/` каталога установки IBM SPSS Modeler) можно запустить из диалогового окна Свойства потока (**Инструменты > Свойства потока > Сценарий**):

```
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([])
```

Следующие маркеры обозначают каждую строку в этом примере сценария.

- Первая строка определяет переменную, указывающую на текущий поток.
- В строке 2 сценарий находит узел построителя нейронных сетей.
- В строке 3 сценарий создает список, в котором можно хранить результаты выполнения.
- В строке 4 создается слепок модели нейронной сети. Он хранится в списке, определенном в строке 3.
- В строке 5 узел применения модели создается для слепка модели и помещается на холст потока.
- В строке 6 создается узел анализа с названием `Drug`.
- В строке 7 сценарий находит узел Тип.
- В строке 8 сценарий сценарий соединяет узел применения модели, созданный в строке 5, с узлом типа и с узлом анализа.
- И наконец, узел анализа выполняется, чтобы создать отчет анализа.

Сценарий можно использовать для построения и запуска потока с нуля, начиная с пустого холста.

Предельные размеры кода Jython

Jython компилирует каждый сценарий в байт-код Java, который затем выполняется виртуальной Java-машиной (Java Virtual Machine, JVM). Однако Java накладывает предел на размер одного файла байт-кода. Если этот предел превышен, когда Jython пытается загрузить байт-код, это может привести к отказу JVM. IBM SPSS Modeler не может предотвратить такие отказы.

Убедитесь, что вы написали сценарии Jython, используя рекомендации по правильному кодированию (такие как минимизация дублированного кода через использование переменных или функций для вычисления общих промежуточных значений). При необходимости вам может потребоваться разделить код на несколько исходных файлов или определить его с использованием модулей, которые будут компилироваться в отдельные файлы байт-кода.

Автономные сценарии

Диалоговое окно Автономный сценарий используется, чтобы создать или изменить сценарий, сохраненный как текстовый файл. Он выводит имя файла и предоставляет возможности для загрузки, сохранения, импорта и выполнения сценариев.

Чтобы получить доступ к диалоговому окну автономного сценария:

В основном меню выберите:

Инструменты > **Автономный сценарий**

Для автономных и потоковых сценариев доступна одинаковая панель инструментов и опции проверки синтаксиса сценариев. Дополнительную информацию смотрите в разделе “Сценарии потока” на стр. 1.

Пример автономного сценария: сохранение и загрузка модели

Автономные сценарии полезны для работы с потоками. Допустим, у вас есть два потока, один, создающий поток, и другой, использующий графики для изучения сгенерированного набора правил из первого потока с существующими потоками данных. Автономный сценарий для каждого сценария может выглядеть примерно так:

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

# Измените здесь папку на правильную папку Demos установки Modeler.
# Обратите внимание на использование начальной и завершающей дробной черты.
installation = "C:/Program Files/IBM/SPSS/Modeler/19/Demos/"

# Сначала загрузите из файла поток построителя моделей и постройте модель
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Сохраните модель в файл
taskrunner.saveModelToFile(results[0], "rule.gm")

# Теперь загрузите поток построения графиков, прочите модель из файла и вставьте ее в поток
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Теперь найдите узел построения моделей, отсоедините его и соединитесь с
# узлом применения модели между узлом получения и узлом построения графиков
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([])
```

Прим.: Чтобы узнать больше о языке сценариев в целом, посмотрите раздел “Обзор языка сценариев” на стр. 15.

Пример автономного сценария: генерирование модели выбора возможностей

Начиная с пустого холста, в этом примере строится поток, генерирующий модель выбора возможностей, применяется эта модель и создается таблица, в которой перечислено 15 наиболее важных полей, относящихся к заданному полю назначения.

```
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)

# Предполагается, что поток автоматически размещает узлы применения моделей
в потоке
applynode = stream.findByType("applyfeatureselection", None)
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96,
applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

Этот сценарий создает узел источника для чтения данных, использует узел Тип для задания роли (направления) от поля `response_01` к `Target` и затем создает и выполняет узел выбора возможностей. Этот сценарий соединяет также узлы и располагает каждый из них на холсте потока для создания читаемого макета. Затем окончательный слепок модели соединяется с узлом Таблица, где перечисляется 15 наиболее важных полей, как определено свойствами `selection_mode` и `top_n`. Дополнительную информацию смотрите в разделе “Свойства featureselectionnode” на стр. 266.

Сценарии надузла

Используя язык сценариев IBM SPSS Modeler можно создавать и сохранять сценарии на любом конечном надузле. Эти сценарии доступны только для конечных надузлов и часто используются при создании потоков шаблонов или для установления особого порядка выполнения для содержимого надузлов. Сценарии надузлов позволяют также одновременный запуск нескольких сценариев в потоке.

Например, допустим, что вам нужно было задать порядок выполнения для сложного потока, а надузел содержит несколько узлов, в том числе узел Задать глобальные значения, который нужно выполнить до получения нового поля, используемого на узле График. В этом случае можно создать сценарий надузла, первоначально выполняющий узел Задать глобальные значения. Вычисленные этим узлом значения, такие как среднее или среднеквадратичное отклонение, могут затем использоваться при выполнении узла График.

Свойства узлов можно задать в сценарии надузла таким же образом, как это делается в других сценариях. Как вариант, изменить или определить свойства для любого надузла или содержащихся в нем узлов можно непосредственно в потоковом сценарии. Дополнительную информацию смотрите в разделе Глава 21, “Свойства надузлов”, на стр. 481. Этот способ работает для надузлов источников и процессов, а также для конечных надузлов.

Прим.: Так как свои собственные сценарии могут выполнять только конечные надузлы, только для них доступна вкладка Сценарии в диалоговом окне Суперузел.

Чтобы открыть диалоговое окно сценария надузла на главном холсте:

Выберите конечный надузел на холсте потоков и в меню Суперузел перейдите к следующему пункту:

Сценарий надузла...

Чтобы открыть диалоговое окно сценария надузла на раскрывающемся холсте надузла:

Щелкните правой кнопкой мыши по холсту надузла и в контекстном меню выберите:

Сценарий надузла...

Пример сценария надузла

Следующий сценарий надузла объявляет порядок, в котором должны выполняться конечные узлы на надузле. Этот порядок обеспечивает, что узел Задать глобальные значения выполняется первым, так что вычисленные этим узлом значения могут затем использоваться при выполнении другого узла.

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

Блокировка и разблокировка надузлов

Следующий пример показывает, как можно блокировать и разблокировать надузел:

```
stream = modeler.script.stream()
superNode=stream.findByID('id854RNTSD5MB')
# разблокировать один надузел
print 'разблокируйте надузел с паролем abcd'
if superNode.unlock('abcd'):
    print 'разблокированд.'
else:
    print 'недопустимый пароль.'
# заблокировать один надузел
print 'заблокируйте надузел с паролем abcd'
superNode.lock('abcd')
```

Потоки с циклами и условиями

Начиная с версии 16.0, простые сценарии в SPSS Modeler можно создавать внутри потока, не вводя сами инструкции на языке сценариев, а выбирая значения в различных диалоговых окнах. Два основных типа сценария, доступных для создания этим способом, - это простые циклы и возможность выполнять узлы в зависимости от выполнения условия.

Возможно сочетание в одном потоке и циклов, и условий. Пусть, например, вы располагаете данными о продажах автомобилей, произведенных по всему миру. Вы можете создать цикл для обработки данных в потоке, задав подробности о стране производителя, и вывести данные на

различные диаграммы, чтобы показать такие подробности, как объем продаж по моделям, уровни эмиссии вредных веществ в выхлопных газах по производителю и рабочему объему двигателя и так далее. Если вам нужна только информация по Европе, вы можете также добавить в цикл условия, чтобы не создавать диаграмм для производителей из Америки и Азии.

Прим.: Поскольку циклы и условия основаны на фоновых сценариях, они применяются только к потоку в целом, при его выполнении.

- **Выполнение в цикле** Циклы служат для автоматизации повторяющихся задач. Например, можно добавить в поток заданное число узлов, изменяя каждый раз один параметр узла. Также можно устроить многократное (с заданным числом раз) выполнение потока или его ветви, как в следующих примерах:
  - Выполнить поток заданное число раз, каждый раз изменяя источник.
  - Выполнить поток заданное число раз, каждый раз изменяя значение некоторой переменной.
  - Выполнить поток заданное число раз, каждый раз вводя дополнительное поле.
  - Построить модель заданное число раз, каждый раз изменяя параметр модели.
- **Выполнение с условиями** С его помощью можно управлять запуском конечных узлов с учетом заданных условий; вот несколько возможных примеров:
  - Управлять запуском узла с учетом полученного значения true или false.
  - Определить, что узлы в цикле должны выполняться параллельно или последовательно.

Циклы и условия задаются на вкладке Выполнение в диалоговом окне Свойства потока. Все узлы, которые используются в требованиях с условиями или циклами, изображаются на холсте с добавлением специального символа.

Есть 3 способа открыть вкладку Выполнение:

- При помощи меню в верхней части главного диалогового окна:
  1. В меню Инструменты выберите:

     Свойства потоков > **Выполнение**
  2. Перейдите на вкладку Выполнение для работы со сценариями для текущего потока.
- Из потока:
  1. Щелкните правой кнопкой по узлу и выберите **С циклами и условиями**.
  2. Выберите нужный пункт подменю.
- На панели инструментов графики в верхней части главного диалогового окна щелкните по значку свойств потока.

Если вы впервые задаете подробности циклов или условий, выберите на вкладке Выполнение режим выполнения **С циклами и условиями**, затем выберите подвкладку **С условием** или **В цикле**.

Циклы в потоках

При помощи циклов можно автоматизировать повторяющиеся задачи в потоках; вот некоторые возможные примеры:

- Выполнить поток заданное число раз, каждый раз изменяя источник.
- Выполнить поток заданное число раз, каждый раз изменяя значение некоторой переменной.
- Выполнить поток заданное число раз, каждый раз вводя дополнительное поле.
- Построить модель заданное число раз, каждый раз изменяя параметр модели.

Необходимые условия задаются на подвкладке **В цикле** на вкладке выполнения потока. Чтобы увидеть эту подвкладку, выберите режим выполнения **С циклами и условиями**.

Все определенные вами требования выполнения в цикле действуют при запуске потока в режиме **С циклами и условиями**. Вы можете сгенерировать код сценария для требований цикла и вставить этот код в редактор сценариев, выбрав **Вставить...** в нижнем правом углу подвкладки В цикле;

основная вкладка выполнения изменится - на ней появится режим выполнения **По умолчанию (дополнительный сценарий)** и сценарий в верхней части вкладки. Таким образом можно определить структуру цикла, пользуясь различными опциями диалогового окна циклов; сгенерированный сценарий потом можно дополнительно изменить в редакторе сценариев. Обратите внимание на то, что при нажатии на кнопку **Вставить...** в сгенерированный сценарий будут вставлены также все требования выполнения с условиями.

Важное замечание: Переменные цикла, заданные в потоке SPSS Modeler, могут быть перезаписаны, если вы запустите поток в задании IBM SPSS Collaboration and Deployment Services. Это связано с тем, что запись редактора заданий IBM SPSS Collaboration and Deployment Services переопределяет запись SPSS Modeler. Например, если вы задаете переменную цикла в потоке для создания различных имен файлов вывода для каждого цикла, эти файлы правильно называются в SPSS Modeler, но перезаписываются фиксированной записью, введенной на вкладке Результаты менеджера внедрений IBM SPSS Collaboration and Deployment Services.

Конфигурирование цикла

1. Создайте ключ итерации, чтобы определить основную структуру цикла, выполняемую в потоке. Дополнительную информацию смотрите в разделе Создать ключ итерации.
2. Если нужно, определите одну или несколько переменных итерации. Дополнительную информацию смотрите в разделе Создать переменную итерации.
3. Итерации и любые созданные вами переменные показаны в основной части подвкладки. По умолчанию итерации выполняются и оцениваются в том порядке, в каком показаны; чтобы изменить порядок, переместите итерацию вверх или вниз в списке, для чего щелкните по ней, чтобы выделить, и затем щелкайте по стрелкам вверх и вниз в столбце в правой части подвкладки, чтобы переместить выделенное.

Создание ключа итерации для циклов в потоках

При помощи ключа итерации определяется основная структура цикла, выполняемая в потоке. Например, если вы анализируете продажи автомобилей, можно создать параметр потока *Страна изготовителя* и использовать его как ключ итерации; при выполнении потока на каждой итерации для этого ключа будет задаваться значение другой страны. Для задания ключа служит диалоговое окно Определить ключ итерации.

Чтобы открыть это диалоговое окно, нажмите кнопку **Ключ итерации...** в нижнем левом углу подвкладки циклов или щелкните правой кнопкой мыши по любому узлу потока и выберите **С циклами и условиями** > **Определить ключ итерации (поля)** или **С циклами и условиями** > **Определить ключ итерации (значения)**. Если открыть диалоговое окно из потока, некоторые поля могут быть заполнены автоматически, например, имя узла.

Чтобы задать ключ итерации, заполните следующие поля:

При итерации перебирать. Можно выбрать один из следующих вариантов:

- **Параметр потока - Поля**. При помощи этой опции можно создать цикл, в котором все заданные поля поочередно используются как значение для существующего параметра потока.
- **Параметр потока - Значения**. При помощи этой опции можно создать цикл, в котором все заданные значения поочередно используются для существующего параметра потока.
- **Свойство узла - Поля**. При помощи этой опции можно создать цикл, в котором все заданные поля поочередно используются как значение для свойства узла.
- **Свойство узла - Значения**. При помощи этой опции можно создать цикл, в котором все заданные значения поочередно используются для свойства узла.

Что задать. Выберите элемент, значение которого будет задаваться при каждом выполнении цикла. Можно выбрать один из следующих вариантов:

- **Параметр**. Доступно, только если выбрать **Параметр потока - Поля** или **Параметр потока - Значения**. Выберите нужный параметр из списка.

- **Узел**. Доступно, только если выбрать **Свойство узла - Поля** или **Свойство узла - Поля**. Выберите узел, для которого хотите задать цикл. Нажмите кнопку просмотра, чтобы открыть диалоговое окно Выбрать узел и выбрать нужный узел; если узлов в этом окне слишком много, можете отфильтровать список, чтобы в нем остались только некоторые типы узлов, для чего выберите одну из следующих категорий: узел источника, узел обработки, узел диаграмм, узел моделирования, узел вывода, узел экспорта или узел применения модели.
- **Свойство**. Доступно, только если выбрать **Свойство узла - Поля** или **Свойство узла - Значения**. Выберите свойство узла из списка.

Используемые поля. Доступно, только если выбрать **Параметр потока - поля** или **Свойство узла - поля**. Выберите одно или несколько полей на узле, которые послужат данными для итерации. Можно выбрать один из следующих вариантов:

- **Узел**. Доступно, только если выбрать **Параметр потока - поля**. Выберите узел, содержащий информацию, для которой вы хотите задать цикл. Нажмите кнопку просмотра, чтобы открыть диалоговое окно Выбрать узел и выбрать нужный узел; если узлов в этом окне слишком много, можете отфильтровать список, чтобы в нем остались только некоторые типы узлов, для чего выберите одну из следующих категорий: узел источника, узел обработки, узел диаграмм, узел моделирования, узел вывода, узел экспорта или узел применения модели.
- **Список полей**. Щелкните по кнопке списка в столбце справа, чтобы открыть диалоговое окно Выбрать поля и выбрать в нем поля узла, которые послужат данными для итерации. Дополнительную информацию смотрите в разделе “Выбор полей для итераций” на стр. 10.

Используемые значения. Доступно, только если выбрать **Параметр потока - значения** или **Свойство узла - значения**. Выберите одно или несколько значений в выбранном поле, которые будут использоваться как значения итерации. Можно выбрать один из следующих вариантов:

- **Узел**. Доступно, только если выбрать **Параметр потока - значения**. Выберите узел, содержащий информацию, для которой вы хотите задать цикл. Нажмите кнопку просмотра, чтобы открыть диалоговое окно Выбрать узел и выбрать нужный узел; если узлов в этом окне слишком много, можете отфильтровать список, чтобы в нем остались только некоторые типы узлов, для чего выберите одну из следующих категорий: узел источника, узел обработки, узел диаграмм, узел моделирования, узел вывода, узел экспорта или узел применения модели.
- **Список полей**. Выберите поле на узле, из которого нужно брать данные для итерации.
- **Список значений**. Нажмите кнопку списка в столбце справа, чтобы открыть диалоговое окно Выбрать значения и выбрать в нем значения поля, которые будут перебираться при итерации.

Создание переменной итерации для циклов в потоках

С помощью переменных итерации можно при каждом выполнении цикла изменять значения параметров потока или свойств выбранных узлов в этом потоке. Например, пусть в вашем цикле анализируются данные о продажах автомобилей и в качестве ключа итерации используется *Страна изготовителя*, и пусть у вас на одной выходной диаграмме показаны продажи по моделям и на другой - информация о выхлопных газах. В таком случае вы можете создать переменные итерации, которыми будут создаваться заголовки для новых графов, например, *Выхлоп шведских автомобилей* и *Продажи японских машин по модели*. Все нужные переменные можно задать в диалоговом окне Определить переменную итерации.

Чтобы открыть это диалоговое окно, нажмите кнопку **Добавить переменную...** в нижнем левом углу подвкладки циклов или щелкните правой кнопкой по любому узлу потока и выберите: **С циклами и условиями** > **Определить переменную итерации**.

Чтобы задать переменную итерации, заполните следующие поля:

Изменить. Выберите тип атрибута, который хотите корректировать. Доступные варианты - **Параметр потока** и **Свойство узла**.

- Если вы выбрали **Параметр потока**, выберите нужный параметр, затем воспользуйтесь одной из описанных ниже опций, если она доступна в вашем потоке, чтобы определить значение параметра, задаваемое на каждой итерации цикла:

  - **Глобальная переменная**. Выберите глобальную переменную, значение которой будет использовано, чтобы задать значение параметра потока.
  - **Ячейка вывода таблицы**. Чтобы задать для параметра потока значение из ячейки табличного вывода, выберите таблицу в списке и введите нужные **Строку** и **Столбец**.
  - **Ввести вручную**. Выберите этот вариант, если нужно вручную вводить значение этого параметра на каждой итерации. Когда вы вернетесь на подвкладку циклов, будет создан новый столбец, в который вы введете нужные тексты.
- Если вы выбрали **Свойство узла**, выберите нужный узел и одно из его свойств, затем задайте значение для этого свойства. При задании нового значения свойства используйте одну из следующих возможностей:
  - **Отдельно**. В качестве значения свойства будет использоваться значение ключа итерации. Дополнительную информацию смотрите в разделе “Создание ключа итерации для циклов в потоках” на стр. 8.
  - **Как префикс для основы**. Значение ключа итерации используется как префикс перед значением, которое вы введете в поле **Основа**.
  - **Как суффикс для основы**. Значение ключа итерации используется как суффикс после значения, которое вы введете в поле **Основа**.

  Если вы выбрали вариант с префиксом или суффиксом, появится приглашение ввести также текст в поле **Основа**. Например, пусть у вас значение ключа итерации - *Страна производителя* и вы выбрали опцию **Как префикс для основы**, тогда в это поле можно ввести, например, такой текст: - *продажи по модели*.

Выбор полей для итераций

Создавая итерацию, вы можете выбрать одно или несколько полей в диалоговом окне Выбрать поля.

Сортировать по Доступные поля можно отсортировать для просмотра, выбрав одну из следующих опций:

- **Естественный** Просмотр полей в порядке их поступления через поток данных на текущий узел.
- **Имя** Использовать алфавитный порядок сортировки полей для просмотра.
- **Тип** Просмотр полей, отсортированных по их шкалам измерений. Эта опция полезна при выборе полей с конкретной шкалой измерений.

Выберите поля в списке по одному или сразу несколько, удерживая при их выборе нажатой клавишу Shift или Ctrl. При помощи опций под этим списком можно также выбрать группы полей на основе их шкалы измерений, а также выбрать сразу все поля или отменить выбор всех полей в таблице.

Обратите внимание на то, что доступные для выбора поля фильтруются: показаны только те поля, которые допустимы для выбранного параметра потока или свойства узла. Например, если используется параметр потока со строковым типом хранения, показаны только поля со строковым типом хранения.

Выполнение с условиями в потоках

При помощи выполнения с условиями можно управлять запуском конечных узлов с учетом того, соответствует ли содержимое потока задаваемым вами условиям; вот несколько возможных примеров:

- Управлять запуском узла с учетом полученного значения true или false.
- Определить, что узлы в цикле должны выполняться параллельно или последовательно.

Необходимые условия задаются на подвкладке **С условиями** на вкладке выполнения потока. Чтобы увидеть эту подвкладку, выберите режим выполнения **С циклами и условиями**.

Все определенные вами требования выполнения с условиями действуют при запуске потока в режиме **С циклами и условиями**. Вы можете сгенерировать код сценария для требований условного выполнения и вставить этот код в редактор сценариев, выбрав **Вставить...** в нижнем правом углу подвкладки С условиями; основная вкладка выполнения изменится - на ней появится режим выполнения **По умолчанию (дополнительный сценарий)** и сценарий в верхней части вкладки. Таким образом можно определить условия, пользуясь различными опциями диалогового окна циклов; сгенерированный сценарий потом можно дополнительно изменить в редакторе сценариев. Обратите внимание на то, что при нажатии на кнопку **Вставить...** в сгенерированный сценарий будут вставлены также все требования циклов.

Чтобы задать условие:

1. В правом столбце на подвкладке С условиями нажмите кнопку Добавить новое условие , чтобы открыть диалоговое окно **Добавить оператор условного выполнения**. В этом диалоговом окне можно задать условие, только при выполнении которого выполняется узел.
2. В диалоговом окне **Добавить оператор выполнения с условиями** укажите следующее:
   a. **Узел**. Выберите узел, для которого хотите задать условное выполнение. Нажмите кнопку просмотра, чтобы открыть диалоговое окно Выбрать узел и выбрать нужный узел; если узлов в этом окне слишком много, можете отфильтровать список по одной из следующих категорий: узел экспорта, узел диаграмм, узел моделирования или узел вывода.
   b. **Условие на основе**. Задайте условие, только при выполнении которого выполняется узел. Можно выбрать один из четырех вариантов: **Параметр потока**, **Глобальная переменная**, **Ячейка табличного вывода** или **Всегда true**. Подробности, которые вводятся в нижней части диалогового окна, зависят от выбранного условия.
      - **Параметр потока**. Выберите параметр в списке, затем для этого параметра выберите **операцию**, например, Более, Равен, Меньше, Между и так далее. Затем, в зависимости от выбранной операции, введите одно **Значение** или минимальное и максимальное значения.
      - **Глобальная переменная**. Выберите переменную в списке, например, Среднее, Сумма, Минимальное значение, Максимальное значение, Среднеквадратичное отклонение. Затем выберите **Операцию** и задайте необходимые значения.
      - **Ячейка вывода таблицы**. Выберите табличный узел в списке, затем выберите **строку** и **столбец** таблицы. Затем выберите **Операцию** и задайте необходимые значения.
      - **Всегда true**. Выберите эту опцию, если узел нужно выполнять всегда. Если вы выбрали эту опцию, никакие другие параметры задавать не нужно.
3. Повторите шаги 1 и 2, сколько требуется, чтобы задать все нужные условия. Выбранный узел и условие выполнения этого узла появятся в основной области подвкладки в столбцах **Выполнить узел** и **При этом условии** соответственно.
4. По умолчанию узлы и условия выполняются и оцениваются в том порядке, в каком показаны; чтобы изменить порядок, переместите узел и условие вверх или вниз в списке, для чего щелкните по ним, чтобы выделить, и затем щелкайте по стрелкам вверх и вниз в столбце в правой части подвкладки, чтобы переместить выделенное.

Кроме того, в нижней части вкладки С условиями можно задать следующие опции:

- **Оценивать все по порядку**. Выберите эту опцию, чтобы оценивать все условия в том порядке, в каком они показаны на этой подвкладке. После оценки всех условий будут выполнены те узлы, для которых при оценке условия было получено значение "True".
- **Выполнять по очереди**. Эта опция доступна, только если выбрано **Оценивать все по порядку**. Выбор этой опции означает, что если при оценке очередного условия получено значение "True", то сначала выполняется соответствующий узел, а затем оценивается следующее условие.
- **Оценивать до первого совпадения**. Выбор этой опции означает, что выполняется только первый узел, для которого при оценке заданных вами условий возвращено значение "True".

Выполнение и прерывание сценариев

Доступно несколько способов выполнения сценариев. Например, в диалоговом окне потокового или автономного сценария есть кнопка "Запустить этот сценарий", после нажатия которой выполняется полный сценарий:

Рисунок 1. Кнопка Запустить этот сценарий

При нажатии кнопки "Запустить выбранные строки" выполняется одна строка или блок соседних строк, которые вы выбрали в сценарии:

Рисунок 2. Кнопка Запустить выбранные строки

Сценарий можно выполнить с использованием любого из следующих способов:

- Нажмите кнопку "Запустить этот сценарий" или "Запустить выбранные строки" в диалоговом окне потокового или автономного сценария.
- Запустить поток, когда в качестве способа выполнения по умолчанию задано **Запустить этот сценарий**.
- Используйте флаг `-execute` для запуска в интерактивном режиме. Дополнительную информацию смотрите в разделе “Использование аргументов командной строки” на стр. 67.

Прим.: Сценарий суперузла выполняется, когда выполняется сам суперузел, а также в диалоговом окне сценария суперузла выбрана опция **Запустить этот сценарий**.

Прерывание выполнения сценария

В диалоговом окне потокового сценария красная кнопка остановки активируется во время выполнения сценария. При нажатии этой кнопки прерывается выполнение сценария и любого текущего потока.

Найти и заменить

Диалоговое окно Найти/заменить доступно там, где вы редактируете текст сценария или выражения, в том числе в редакторе сценариев, в построителе выражений CLEM или при определении шаблонов на узле Отчет. При изменении текста в любой из этих областей нажмите комбинацию клавиш `Ctrl+F` для перехода в диалоговое окно изменений и убедитесь, что указатель мыши сфокусирован в области текста. Например, при работе на узле Заполнитель можно получить доступ к диалоговому окну из любой текстовой области на вкладке Параметры или из поля текста в построителе выражений.

1. При указателе в области текста нажмите клавиши Ctrl+F для доступа к диалоговому окну Найти/заменить.
2. Введите текст для поиска или выберите из раскрывающегося списка элементы недавнего поиска.
3. Введите текст для замещения, если такой есть.
4. Нажмите кнопку **Найти далее** для запуска поиска.
5. Нажмите кнопку **Заменить** для замены текущего выбранного фрагмента или кнопку **Заменить все** для замены всех или выбранных экземпляров.
6. Диалоговое окно закрывается после каждой операции. Находясь в любой области текста, нажмите F3 для повторения последней операции поиска или Ctrl+F для повторного доступа к диалоговому окну.

Опции поиска

Учитывать регистр. Задает, учитывается ли при поиске регистр символов; например, совпадает ли *myvar* с *myVar*. Текст замещения всегда вставляется так, как он введен, независимо от этого параметра.

Только слова целиком. Задает, учитывает ли операция поиска совпадения текста внутри слов. При выборе этой опции поиск для *spider* не найдет соответствий с вариантами *spiderman* или *spider-man*.

Регулярные выражения. Задает, используется ли синтаксис регулярных выражений (смотрите следующий раздел). При выборе этой опции отключается опция **Только слова целиком** и ее значение игнорируется.

Только выбранный текст. Управляет областью выполнения поиска при использовании опции **Заменить все**.

Синтаксис регулярных выражений

Регулярные выражения позволяют искать специальные символы, такие как символы табуляции или перехода на новую строку, классы или диапазоны символов, такие как от *a* до *d* при любых цифровых или нецифровых параметрах и граничные положения, такие как начало или конец строки. Поддерживаются следующие типы выражений.

Таблица 1. Совпадения символов

Символы	**Совпадает**
x	Символ x
\\	Символ обратной дробной черты
\0n	Символ с восьмеричным значением 0n (0 <= n <= 7)
\0nn	Символ с восьмеричным значением 0nn (0 <= n <= 7)
\0mnn	Символ с восьмеричным значением (0 <= m <= 3, 0 <= n <= 7)
\xhh	Символ с шестнадцатиричным значением 0xhh
\uhhhh	Символ с шестнадцатиричным значением 0xhhhh
\t	Символ табуляции ('\u0009')
\n	Символ новой строки (начало строки) ('\u000A')
\r	Символ возврата каретки ('\u000D')
\f	Символ перехода к новой странице ('\u000C')
\a	Символ предупреждения (звонок) ('\u0007')
\e	Символ перехода ESC ('\u001B')
\cx	Управляющий символ, соответствующий x

Таблица 2. Совпадения классов символов

Классы символов	**Совпадает**
[abc]	a, b или c (простой класс)
[^abc]	Любой символ, кроме a, b или c (вычитание)
[a-zA-Z]	От a до z или от A до Z, включительно (диапазон)
[a-d[m-p]]	От a до d или от m до p (объединение). Альтернативно это можно задать как [a-dm-p]

Таблица 2. Совпадения классов символов (продолжение)	
Классы символов	**Совпадает**
[a-z&&[def]]	От a до z и d, e или f (пересечение)
[a-z&&[^bc]]	От a до z, кроме b и c (вычитание). Альтернативно это можно задать как [ad-z]
[a-z&&[^m-p]]	От a до z, но не от m до p (вычитание). Альтернативно это можно задать как [a-lq-z]

Таблица 3. Предварительно определенные классы символов	
Предварительно определенные классы символов	**Совпадает**
.	Любой символ (разделители строк могут и совпадать, и не совпадать)
\d	Любая цифра: [0-9]
\D	Не цифра: [^0-9]
\s	Пробельный символ: [\t\n\x0B\f\r]
\S	Не пробельный символ: [^\s]
\w	Символ слова: [a-zA-Z_0-9]
\W	Символ не слова: [^\w]

Таблица 4. Совпадения границ	
Обнаружители совпадений границ	**Совпадает**
^	Начало строки
$	Конец строки
\b	Граница слова
\B	Граница не слова
\A	Начало ввода
\Z	Конец ввода кроме заключительного разделителя, если он есть
\z	Конец ввода

Глава 2. Язык сценариев

Обзор языка сценариев

Возможность написания сценариев для IBM SPSS Modeler позволяет вам создавать сценарии, работающие в пользовательском интерфейсе SPSS Modeler, манипулирующие объектами вывода и запускающие командный синтаксис. Запустить сценарии можно непосредственно изнутри SPSS Modeler.

Сценарии в IBM SPSS Modeler пишутся на языке сценариев Python. Используемая IBM SPSS Modeler реализация Python на основе Java называется Jython. Язык написания сценариев состоит из следующих возможностей:

- Формат для ссылки на узлы, потоки, проекты, выходные данные и другие объекты IBM SPSS Modeler.
- Набор операторов сценариев или команд, которые можно задать для манипулирования этими объектами.
- Язык выражений сценариев для задания значений переменных, параметров и других объектов.
- Поддержка комментариев, продолжений и блоков литерального текста.

В следующих разделах описывается язык сценариев, его конкретная реализация Jython и основной синтаксис для начала работы со сценариями в IBM SPSS Modeler. Информация о конкретных свойствах и командах представлена в последующем разделе.

Python и Jython

Jython - это реализация языка сценариев Python, написанная на языке Java и интегрированная с платформой Java. Python - богатый объектно-ориентированный язык сценариев. Jython полезен тем, что предлагает повышающие производительность труда программистов возможности развитого языка сценариев, но, в отличие от языка Python, работает в любой среде, поддерживающей виртуальную Java-машину (JVM). Это значит, что при написании программ доступны библиотеки Java для JVM. Работая на языке Jython, можно пользоваться этим преимуществом и вместе с тем синтаксисом и большинством возможностей языка Python

Как язык сценариев, Python (и его реализация Jython) удобен для изучения, эффективен при программировании и дает возможность создать работающую программу при минимальных требованиях к структуре. Код можно вводить интерактивно, по одной строке. Python работает как интерпретатор языка сценариев, не требуя стадии компиляции, которая необходима в Java. Программы Python - это просто текстовые файлы, которые интерпретируются по мере ввода (после синтаксического анализа для выявления ошибок). Простые выражения, такие как задание значений, и более сложные действия, такие как определение функций, выполняются немедленно, и их результаты доступны для использования. Все изменения, вносимые в код, можно быстро протестировать. Однако у интерпретаторов сценариев есть свои недостатки. Так, использование переменной до ее определения не считается ошибкой для компилятора и обнаруживается только при попытке выполнить оператор, содержащий такую переменную. В этом случае программу можно редактировать и запускать для исправления ошибок.

Python рассматривает все как объекты, например, объекты данных и объекты кода. Со всеми объектами можно работать, редактируя строки кода. Некоторые типы выбора, такие как числа и строки, удобнее считать не объектами, а значениями, и это поддерживается в Python. Поддерживается одно пустое значение. Ему присвоено зарезервированное имя None.

Более подробное введение в сценарии Python и Jython с примерами сценариев есть в разделах http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html и http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html.

Сценарий Python

Это руководству по языку написания сценариев Python представляет собой введение в компоненты, которые чаще всего используются для сценариев в IBM SPSS Modeler, а также в концепции и основы программирования. Вы получите достаточные знания для начала разработки собственных сценариев Python и их использования в IBM SPSS Modeler.

Операции

Назначение производится с помощью знака равенства (=). Например, чтобы назначить значение "3" переменной с именем "x", используется следующий оператор:

```
x = 3
```

Знак равенства используется также для назначения переменной строковых значений. Например, чтобы назначить значение "строковое значение" переменной "y", используется следующий оператор:

```
y = "строковое значение"
```

В следующей таблице перечисляются общие операции с числами и операции сравнения, а также их описания.

Таблица 5. Общие операции с числами и операции сравнения

Операция	**Описание**
`x < y`	x меньше y?
`x > y`	x больше y?
`x <= y`	x меньше или равно y?
`x >= y`	x больше или равно y?
`x == y`	x равно y?
`x != y`	x не равно y?
`x <> y`	x не равно y?
`x + y`	Сложить y и x
`x - y`	Вычесть y из x
`x * y`	Умножить x на y
`x / y`	Разделить x на y
`x ** y`	Возвести x в степень y

Списки

Перечисляет последовательность элементов. Список может содержать любое количество элементов, и эти элементы списка могут быть любых типов объектов. Списки можно рассматривать как массивы. Число элементов в списке может увеличиваться или уменьшаться при добавлении, удалении или замене элементов.

Примеры

`[]`	Любой пустой список.
`[1]`	Список с одним элементом, целое число.

`["Mike", 10, "Don", 20]`	Список с четырьмя элементами - двумя строковыми и двумя целочисленными.
`[[],[7],[8,9]]`	Список списков. Каждый подсписок - это или пустой список, или список целочисленных элементов.
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	Список целых чисел. В этом примере демонстрируется использование переменных и выражений.

Список можно назначить переменной, например:

```
mylist1 = ["один", "два", "три"]
```

Затем вы можете обратиться к любому конкретному элементу в этом списке, например:

```
mylist[0]
```

При этом вывод будет следующим:

```
один
```

Число в прямых скобках (`[]`) называют *индексом*; оно указывает на конкретный элемент в списке. Элементы в списке индексируются, начиная с нуля.

Вы можете выбрать также диапазон элементов в списке; такой диапазон называется *срезом*. Например, `x[1:3]` выбирает второй и третий элемент `x`. Обозначенный конечный индекс - это индекс следующего элемента после последнего выбранного.

Строки

Строка - это неизменная последовательность символов, которая рассматривается как значение. Строки поддерживают все функции и операторы с неизменными последовательностями, которые дают на выходе новую строку. Например, `"abcdef"[1:4]` дает в результате `"bcd"`.

В Python символы представлены строками единичной длины.

Строковые литералы определяются использованием одинарных или тройных кавычек. Строки, определенные одинарными кавычками, не могут распространяться на несколько строк кода, а строки с тройными кавычками могут. Строка может заключаться в одинарные (') или в двойные кавычки ("). Между символами кавычек могут содержаться другие символы кавычек, не обозначающие переход, или символы кавычек с переходом, перед которыми должен стоять знак обратной дробной черты (\).

Примеры

```
"Это строка"
'Это тоже строка'
"It's a string"
'Эта книга называется "Руководство по автоматизации и сценариям Python".'
"Это кавычка, заданная с использованием эскейп-символа (\") в закавыченной строке"
```

Несколько строк, разделенных пробельным символом, автоматически объединяются синтаксическим анализатором Python. Так проще вводить длинные строки и совместно использовать разные типы кавычек в одной строке, например:

```
"В этой строке используется ', а " 'в этой строке используется ".'
```

Это приводит к следующему выводу:

```
В этой строке используется ', а в этой строке используется ".
```

Строки поддерживают несколько полезных методов. Некоторые из них приведены в следующей таблице.

Таблица 6. Строковые методы

Метод	**Использование**
`s.capitalize()`	Переводит в верхний регистр начальные символы s
`s.count(ss {,start {,end}})`	Считает количество ss в `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Проверяет, начинается ли s с `str` Проверяет, заканчивается ли s на `str`
`s.expandtabs({size})`	Заменяет символы табуляции на пробелы, по умолчанию значение `size` - 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Ищет первое вхождение `str` в s; если не находит, возвращает -1. `rfind` ищет справа налево.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Находит первое вхождение `str` в s; если не находит, выводит ошибку `ValueError`. `rindex` ищет справа налево.
`s.isalnum`	Проверяет, состоит ли строка из букв и цифр
`s.isalpha`	Проверяет, состоит ли строка из букв
`s.isnum`	Проверяет, состоит ли строка из цифр
`s.isupper`	Проверяет, используется ли в строке только верхний регистр
`s.islower`	Проверяет, используется ли в строке только нижний регистр
`s.isspace`	Проверяет, состоит ли строка только из пробельных символов
`s.istitle`	Проверяет, состоит ли строка из последовательности алфавитно-цифровых строк, начинающихся символом капители

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Преобразует все в нижний регистр
	Преобразует все в верхний регистр
	Преобразует все в противоположный регистр
	Преобразует первую букву каждого слова в верхний
	регистр, а прочие буквы - в нижний регистр

`s.join(seq)`	Объединяет строки в `seq`, используя s как разделитель
`s.splitlines({keep})`	Разбивает s на строки; если значение keep - `true`, сохраняет переход на новые строки
`s.split({sep {, max}})`	Разделяет s на "слова", используя `sep` (по умолчанию `sep` - это пробельный символ); число повторений действия - `max`

Таблица 6. Строковые методы (продолжение)

Метод	Использование

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Выравнивает строку по левому краю в поле ширины `width`
	Выравнивает строку по правому краю в поле ширины `width`
	Выравнивает строку по центру в поле ширины `width`
	Дополняет символами 0.

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Удаляет пробельные символы в начале
	Удаляет пробельные символы в конце
	Удаляет начальные и конечные пробельные символы

`s.translate(str {,delc})`	Переводит `s` с помощью таблицы, после удаления всех символов `delc`. `str` должна быть строкой длины `==` 256.
`s.replace(old, new {, max})`	Заменяет все или `max` строк `old` на строку `new`

Комментарии

Комментарии (remarks) вводятся в текст кода с помощью символа решетки (#). Весь текст в одной строке после этого знака рассматривается как комментарий и игнорируется. Комментарий может начинаться в любом столбце. В следующем примере показано использование комментариев:

```
#Программа HelloWorld - одна из самых простых программ
print 'Hello World' # печатает строку Hello World
```

Синтаксис операторов

Синтаксис операторов Python очень прост. В общем случае каждая исходная строка - это один оператор. Исключение составляют операторы `expression` и `assignment`, каждый оператор вводится именем ключевого слова, таким как `if` или `for`. Пустые строки или строки комментариев можно вставить в произвольном месте между любыми операторами в коде. Если в строке располагается несколько операторов, их нужно разделить точкой с запятой (`;`).

Очень длинные операторы могут располагаться на нескольких строках. В этом случае оператор, который нужно продолжить на следующую строку должен заканчиваться знаком обратной дробной черты на данной строке (\), например:

```
x = "Длиииииииииииииииииииииииинная строка" + \
     "еще одна длиииииииииииииииииииииииинная строка"
```

Когда некоторая структура кода заключена в скобки (()), квадратные скобки ([]) или фигурные скобки ({}), оператор можно продолжить на новую строку после любой запятой, не используя знака обратной дробной черты, например:

```
x = (1, 2, 3, "привет",
      "пока", 4, 5, 6)
```

Идентификаторы

Идентификаторы используются для именования переменных, функций, классов и ключевых слов. Идентификаторы могут быть любой длины, но должны начинаться или с буквы, или с символа подчеркивания (_). Имена, начинающиеся с подчеркивания, обычно резервируются для

внутренних или частных имен. После первого символа идентификатор может содержать любую комбинацию произвольной длины букв, чисел от 0 до 9 и символов подчеркивания.

В Jython есть несколько зарезервированных слов, которые нельзя использовать в качестве имен переменных, функций или классов. Они попадают в следующие категории:

- **Вводные для операторов:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` и `while`
- **Вводные параметров:** `as`, `import` и `in`
- **Операции:** `and`, `in`, `is`, `lambda`, `not` и `or`

Неправильное использование ключевых слов обычно приводит к ошибке `SyntaxError`.

Блоки кода

Блоки кода - это группы операторов, используемые там, где предполагаются отдельные операторы. Блоки кода могут следовать после любого из следующих операторов: `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` и `class`. Эти операторы вводят блоки кода с помощью символа двоеточия (`:`), например:

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

Отступ используется для разделения блоков кода (в отличие от фигурных скобок, используемых в Java). Все строки в блоке должны иметь одинаковый отступ. Это связано с тем, что изменение отступа обозначает конец блока кода. Обычно производится сдвиг по четыре пробела на каждый уровень. Рекомендуется использовать для отступа строки именно пробелы, а не знаки табуляции. Пробелы и табуляции нельзя смешивать. Строки самого внешнего блока в модуле должны начинаться в первом столбце, иначе возникнет ошибка `SyntaxError`.

Операторы, составляющие блок кода (и следующие за двоеточием), могут располагаться также на одной строке и разделяться точками с запятой, например:

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

Передача аргументов в сценарий

Передача аргументов в сценарий полезна тем, что сценарий можно использовать повторно без изменений. Аргументы, переданные в командной строке, передаются как значения в списке `sys.argv`. Количество передаваемых значений можно получить с использованием команды `len(sys.argv)`. Например:

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

В этом примере команда `import` импортирует весь класс `sys`, поэтому можно использовать методы, существующие для этого класса, такие как `argv`.

Сценарий из этого примера можно вызвать, используя следующую строку:

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

Будет получен следующий вывод:

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

```
don
3
```

Примеры

Ключевое слово `print` выводит непосредственно следующие за ним аргументы на печать. Если после оператора стоит запятая, переход на новую строку в выводе не происходит. Например:

```
print "Показано использование запятой",
print " в конце оператора print."
```

При этом вывод будет следующим:

```
Показано использование запятой в конце оператора print.
```

Оператор `for` используется для итераций по блоку кода. Например:

```
mylist1 = ["один", "два", "три"]
for lv in mylist1:
    print lv
    continue
```

В этом примере списку `mylist1` назначаются три строки. Затем элементы этого списка выводятся на печать по одному на строке. При этом вывод будет следующим:

```
один
два
три
```

В этом примере итератор `lv` принимает значения каждого элемента списка `mylist1` по порядку, как цикл реализует блок кода для каждого элемента. Итератор может быть любым допустимым идентификатором произвольной длины.

Оператор `if` - это условный оператор. Он вычисляет некоторое условие и возвращает значение true или false в зависимости от результата вычислений. Например:

```
mylist1 = ["один", "два", "три"]
for lv in mylist1:
    if lv == "два"
        print "Значение lv - ", lv
    else
        print "Значение lv - не два, а ", lv
    continue
```

В этом примере вычисляется значение итератора `lv`. Если значение `lv` равно два, возвращается не та строка, которая возвращается в случаях, когда значение `lv` не равно два. Это приводит к следующему выводу:

```
Значение lv - не два, а один
Значение lv - два
Значение lv - не два, а три
```

Математические методы

Из модуля `math` можно получить доступ к полезным математическим методам. Некоторые из них приведены в следующей таблице. Если не указано иное, все значения возвращаются в виде чисел с плавающей запятой.

Таблица 7. Математические методы

Метод	**Использование**
`math.ceil(x)`	Возвращает в виде значения с плавающей запятой потолочное значение x, то есть наименьшее целое число, большее или равное x
`math.copysign(x, y)`	Возвращает x со знаком y. `copysign(1, -0.0)` возвращает `-1`
`math.fabs(x)`	Возвращает абсолютное значение x
`math.factorial(x)`	Возвращает факториал x. Если x отрицательное или не целое, возникает ошибка `ValueError`.
`math.floor(x)`	Возвращает в виде значения с плавающей запятой поддерживающее значение x, то есть максимальное целое, меньшее или равное x
`math.frexp(x)`	Возвращает мантиссу (`m`) и экспоненту (`e`) числа x как пару чисел `(m, e)`. `m` - это число с плавающей запятой, а `e` - целое, так что выполняется точное равенство `x == m * 2**e`. Если x равно нулю, возвращается `(0.0, 0)`, в противном случае `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Возвращает в формате числа с плавающей запятой точную сумму значений в `iterable`
`math.isinf(x)`	Проверяет переполнение числа с плавающей запятой x (бесконечно большое отрицательное или положительное)
`math.isnan(x)`	Проверяет, принимает ли x значение `NaN` (not a number, не число)
`math.ldexp(x, i)`	Возвращает `x * (2**i)`. По сути это функция, обратная к `frexp`.
`math.modf(x)`	Возвращает дробную и целую часть числа x. Оба полученных числа - с плавающей запятой, и у обоих значений знак x.
`math.trunc(x)`	Возвращает значение `Real` числа x, усеченного до значения `Integral`.
`math.exp(x)`	Возвращает `e**x`
`math.log(x[, base])`	Возвращает значение логарифма x по данному основанию `base`. Если значение `base` не задано, возвращается значение натурального логарифма x.
`math.log1p(x)`	Возвращает натуральный логарифм `1+x` `(основание e)`
`math.log10(x)`	Возвращает десятичный логарифм x
`math.pow(x, y)`	Возвращает x в степени y. `pow(1.0, x)` и `pow(x, 0.0)` всегда возвращают 1, даже если x равно нулю или NaN.
`math.sqrt(x)`	Возвращает квадратный корень из x

В дополнение к математическим функциям есть несколько полезных тригонометрических методов. Эти методы показаны в следующей таблице.

Таблица 8. Тригонометрические методы

Метод	**Использование**
`math.acos(x)`	Возвращает арккосинус x в радианах
`math.asin(x)`	Возвращает арксинус x в радианах
`math.atan(x)`	Возвращает арктангенс x в радианах
`math.atan2(y, x)`	Возвращает арктангенс `atan(y / x)` в радианах.
`math.cos(x)`	Возвращает косинус x.
`math.hypot(x, y)`	Возвращает евклидову норму `sqrt(x*x + y*y)`. Это длина вектора из начала координат в точку `(x, y)`.
`math.sin(x)`	Возвращает синус x
`math.tan(x)`	Возвращает тангенс x
`math.degrees(x)`	Преобразует угол x из радиан в градусы
`math.radians(x)`	Преобразует угол x из градусов в радианы
`math.acosh(x)`	Возвращает обратный гиперболический косинус x
`math.asinh(x)`	Возвращает обратный гиперболический синус x
`math.atanh(x)`	Возвращает обратный гиперболический тангенс x
`math.cosh(x)`	Возвращает гиперболический косинус x
`math.sinh(x)`	Возвращает гиперболический косинус x
`math.tanh(x)`	Возвращает гиперболический тангенс x

Есть также две математические константы. Значение `math.pi` - это число пи. Значение `math.e` - это основание натуральных логарифмов e.

Использование символов не из кодового набора ASCII

Чтобы использовать символы не из кодового набора ASCII, в Python требуется явное кодирование и декодирование строк в Unicode. В IBM SPSS Modeler сценарии Python считаются написанными в формате UTF-8, представляющем собой стандарт кодировки Unicode с поддержкой символов не из кодового набора ASCII. Следующий сценарий пройдет компиляцию, поскольку компилятор Python был настроен на UTF-8 SPSS Modeler.

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

Однако созданный в результате узел будет иметь неправильную метку.

Рисунок 3. Метка узла, содержащая символы не из кодового набора ASCII, которые выводятся неправильно

Неправильная метка - результат преобразования строкового литерала в строку ASCII, выполненного Python.

Python разрешает задавать строковые литералы Unicode, добавляя перед литералом префикс u:

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

Здесь создается строка Unicode, и метка будет выведена правильно.

Рисунок 4. Метка узла, содержащая символы не из кодового набора ASCII, которые выводятся правильно

Использование Python и Unicode - большая тема, выходящая за рамки этого документа. Подробное обсуждение этой темы имеется во многих книгах и сетевых ресурсах.

Объектно-ориентированное программирование

Объектно-ориентированное программирование основано на идее создания модели основной проблемы в ваших программах. Объектно-ориентированное программирование сокращает количество программных ошибок и способствует повторному использованию кода. Python - это объектно-ориентированный язык. У определенных в Python объектов есть следующие характеристики:

- **Тождество.** Каждый объект должен быть индивидуальным, и должна существовать возможность проверки этого. Для этой цели существуют проверки `is` и `is not`.
- **Состояние.** У каждого объекта должна быть возможность сохранения состояния. Для этой цели существуют атрибуты, такие как поля и переменные экземпляров.
- **Поведение.** Для каждого объекта должна существовать возможность изменения состояния. Для этой цели существуют методы.

Python включает в себя следующие возможности поддержки объектно-ориентированного программирования:

- **Создание объектов на основе классов.** Классы - это шаблоны для создания объектов. Объекты - это структуры данных со связанным поведением.
- **Наследование с полиморфизмом.** Python поддерживает и одиночное, и множественное наследование. Методы всех экземпляров Python полиморфичны и могут быть перезаписаны подклассами.

- **Инкапсуляция с сокрытием данных.** Python допускает сокрытие атрибутов. Получить доступ к скрытым атрибутам извне класса можно только через методы класса. Классы реализуют методы для изменения данных.

Определение класса

В классе Python можно определять и переменные, и методы. В отличие от Java, в Python можно определить любое количество доступных классов на файл источника (или *модуль*). Таким образом, модуль в Python можно рассматривать аналогично пакету в Java.

В Python классы определяются с использованием оператора `class`. Оператор `class` выглядит следующим образом:

```
имя класса (надклассы): оператор
```

или

```
имя класса (надклассы):
    назначение
    .
    .
    функция
    .
    .
```

При определении класса есть возможность задать несколько операторов *assignment* или не задавать их вовсе. Эти операторы создают атрибуты класса, которые совместно используются всеми экземплярами класса. Можно задать также несколько определений *function* или не задавать их вовсе. Эти определения функций создают методы. Список надклассов не обязателен.

Имя класса должно быть уникальным в одной области действия, то есть в модуле, функции или классе. Для ссылки на один класс можно определить несколько переменных.

Создание экземпляра класса

Классы используются для содержания атрибутов класса (или атрибутов совместного использования) или для создания экземпляров классов. Чтобы создать экземпляр класса, класс вызывается так же, как функция. Например, рассмотрим следующий класс:

```
class MyClass:
    pass
```

Здесь используется оператор `pass`, поскольку для завершения определения класса нужен оператор, но никакое действие программно не требуется.

Следующий оператор создает экземпляр класса `MyClass`:

```
x = MyClass()
```

Добавление атрибутов к экземпляру класса

В отличие от Java, в Python клиенты могут добавить атрибуты к экземпляру класса. Изменяется только один экземпляр. Например, чтобы добавить атрибуты к экземпляру `x`, задайте новые значения для этого экземпляра:

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

Определение атрибутов классов и методов

Любая переменная, связанная в класс, - это *атрибут класса*. Любая определенная в классе функция - это *метод*. Методы получают экземпляр класса, условно называемый `self`, в качестве первого аргумента. Например, для определения нескольких атрибутов и методов класса можно использовать следующий код:

```
class MyClass
    attr1 = 10        #атрибуты класса
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #ссылка на атрибут класса

    def method2(self):
        print MyClass.attr2   #ссылка на атрибут класса

    def method3(self, text):
        self.text = text        #атрибут экземпляра
        print text, self.text   #напечатать мой аргумент и мой атрибут

    method4 = method3   #создать алиас для method3
```

Внутри класса все ссылки на атрибуты класса необходимо специфицировать с помощью имени класса; например, `MyClass.attr1`. Все ссылки на атрибуты экземпляра должны специфицироваться переменной `self`; например, `self.text`. Вне класса все ссылки на атрибуты класса должны специфицироваться именем класса (например, `MyClass.attr1`) или экземпляром класса (например, `x.attr1`, где `x` - это экземпляр класса). Вне класса все ссылки на переменные экземпляра должны специфицироваться экземпляром класса; например, `x.text`.

Скрытые переменные

Данные можно скрыть, создав *Приватные* переменные. К приватным переменным может обратиться только сам класс. Если вы объявляете имена в виде `__xxx` или `__xxx_yyy`, то есть с двумя начальными знаками подчеркивания, синтаксический анализатор Python автоматически добавит имя класса к объявленным именам, создавая скрытые переменные, например:

```
class MyClass:
    __attr = 10   #атрибут приватного класса

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #приватный атрибут
```

В отличие от Java, в Python все указания на переменные экземпляра должны быть специфицированы с `self`; использование `this` не применяется.

Наследование

Возможность наследования от классов - это фундаментальное свойство объектно-ориентированного программирования. Python поддерживает и одиночное, и множественное наследование. *Одиночное наследование* означает, что может быть только один надкласс. *Множественное наследование* означает, что может быть несколько надклассов.

Наследование реализуется определением других классов в качестве подклассов. Надклассами может быть любое число классов Python. В Jython, реализации Python, прямо или косвенно можно наследовать только от одного класса Java. Представление надкласса не требуется.

Любой атрибут или метод надкласса содержится также в любом подклассе и может использоваться самим классов или любым клиентом, если атрибут или метод не скрыт. Любой экземпляр подкласса можно использовать там, где допустимо использование и экземпляра

надкласса; это пример *полиморфизма*. Эти возможности допускают повторное использование и облегчают работу с расширением.

Пример

```
class Class1: pass    #нет наследования

class Class2: pass

class Class3(Class1): pass     #одиночное наследование

class Class4(Class3, Class2): pass     #множественное наследование
```

Глава 3. Сценарии в IBM SPSS Modeler

Типы сценариев

В IBM SPSS Modeler есть три типа сценариев:

- *Сценарии потока* используются для управления выполнением одного потока и хранятся в этом потоке.
- *Сценарии надузлов* используются для управления поведением надузлов.
- *Автономные сценарии и сценарии сеансов* можно использовать для координирования выполнения по нескольким разным потокам.

Для использования в сценариях в IBM SPSS Modeler доступны различные методы, с помощью которых можно получить доступ к широкому диапазону функциональных возможностей SPSS Modeler. Эти методы используются в Глава 4, “API сценариев”, на стр. 41 для создания также более расширенных функций.

Потоки, потоки надузлов и диаграммы

В большинстве случаев термин *поток* имеет один и тот же смысл как в отношении потока, загруженного из файла, так и потока, используемого внутри надузла. Обычно он означает собрание соединенных друг с другом узлов, которые можно выполнять. Но в сценариях поддерживаются не все операции и не в любом месте, так что автор сценария должен знать, какую разновидность потока использует.

Потоки

Поток - это основной тип документа IBM SPSS Modeler. Его можно сохранять, загружать, редактировать и выполнять. Кроме того, с потоками может связываться такая информация, как параметры, глобальные переменные, сценарии и другое.

Потоки надузлов

Поток надузла - это тип потока, используемый внутри надузла. Подобно обычному потоку, он состоит из соединенных между собой узлов. У потоков надузла есть ряд отличий от обычного потока:

- Параметры и сценарии связываются не с самим потоком надузла, а с надузлом, которому принадлежит поток надузла.
- Потоки надузла содержат дополнительные узлы входящих и исходящих соединений, зависящие от типа надузла. Эти узлы соединений служат для передачи информации в поток надузла и из потока надузла; такие узлы создаются автоматически при создании надузла.

Диаграммы

Термин *диаграмма* охватывает функции, которые поддерживаются и обычными потоками, и потоками надузла, такими как добавление и удаление узлов и изменение соединений между узлами.

Выполнение потока

В следующем примере запускаются все исполняемые узлы в потоке, и это простейший тип сценария потока:

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

В следующем примере также запускаются все исполняемые узлы в потоке:

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

В этом примере поток хранится в переменной, называемой `stream`. Хранение потока в переменной полезно, так как сценарий обычно используется для изменения или потока, или узлов в потоке. Создание переменной, хранящей поток, приводит к более лаконичному сценарию.

Контекст сценариев

Модуль `modeler.script` предоставляет контекст, в котором выполняется сценарий. Этот модуль автоматически импортируется в сценарий SPSS Modeler во время выполнения. Этот модуль определяет четыре функции, обеспечивающие сценарию доступ в его среду выполнения:

- Функция `session()` возвращает сеанс для сценария. Сеанс определяет такую информацию, как локаль и механизм обработки SPSS Modeler (или локальный процесс, или сетевой процесс SPSS Modeler Server), используемые для запуска любых потоков.
- Функцию `stream()` можно использовать со сценариями потоков и надузлов. Эта функция возвращает поток, который владеет исполняемым сценарием потока или сценарием надузла.
- Функцию `diagram()` можно использовать со сценариями надузлов. Эта функция возвращает диаграмму в надузле. Для остальных типов сценариев возвращаемое значение то же, что у функции `stream()`.
- Функцию `supernode()` можно использовать со сценариями надузлов. Эта функция возвращает надузел, который владеет исполняемым сценарием.

Эти четыре функции и их выводы сведены в следующей таблице.

Таблица 9. Сводка функций `modeler.script`

Тип сценария	**`session()`**	**`stream()`**	**`diagram()`**	**`supernode()`**
Локальный режим	Возвращает сеанс	Возвращает текущий управляемый поток на момент вызова сценария (например, поток, прошедший через опцию `-stream` пакетного режима) или значение None.	То же, что и `stream()`	Неприменимо
Поток	Возвращает сеанс	Возвращает поток	То же, что и `stream()`	Неприменимо
Надузел	Возвращает сеанс	Возвращает поток	Возвращает поток надузла	Возвращает надузел

Модуль `modeler.script` определяет также способ завершения сценария при помощи кода выхода. Функция `exit(`*`код-выхода`*`)` останавливает выполнение сценария и возвращает предоставленный целочисленный код выхода.

Один из определенных для потока методов - это `runAll(List)`. Этот метод запускает все выполняемые узлы. Все модели и выводы, генерируемые при выполнении узлов, добавляются к предоставленному списку.

Обычно при выполнении потока генерируются такие выводы, как модели, графики и другие объекты. Для захвата этих выводов сценарий может использовать переменную, инициализированную для списка, например:

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

Когда выполнение закончено, ко всем сгенерированным объектам можно получить доступ в списке `results`.

Ссылки на существующие узлы

Часто поток строится предварительно с использованием некоторых параметров, которые нужно изменить до выполнения узла. Изменение этих параметров включает в себя следующие задачи:

1. Обнаружение узлов в соответствующем потоке.
2. Изменение параметров узла и/или потока.

Поиск узлов

Потоки предоставляют несколько способов обнаружения существующего узла. Соответствующие методы сведены в следующей таблице.

Таблица 10. Методы для обнаружения существующего узла

Метод	Возвращаемый тип	Описание
`s.findAll(type, label)`	Собрание	Возвращает список всех узлов с заданным типом и меткой. И у типа, и у метки может быть значение None, в этом случае используется другой параметр.
`s.findAll(filter, recursive)`	Собрание	Возвращает собрание всех узлов, которые приняты заданным фильтром. Если для рекурсивного флага задано значение True, поиск проводится и для всех надузлов в заданном потоке.
`s.findByID(id)`	Узел	Возвращает узел с указанным значением ID или значение None, если такого узла не существует. Поиск ограничен текущим потоком.

Таблица 10. Методы для обнаружения существующего узла (продолжение)

Метод	Возвращаемый тип	Описание
`s.findByType(type, label)`	Узел	Возвращает узел с указанным значением типа и/или метки. И у типа, и у имени может быть значение None, в этом случае используется другой параметр. Если совпадение получено для нескольких узлов, выбирается и возвращается произвольный из них. Если критериям поиска не удовлетворяет ни один узел, возвращается значение None.
`s.findDownstream(fromNodes)`	Собрание	Проводит поиск в предоставленном списке узлов и возвращает набор узлов ниже по потоку от предоставленных узлов. Возвращаемый список включает в себя исходные предоставленные узлы.
`s.findUpstream(fromNodes)`	Собрание	Проводит поиск в предоставленном списке узлов и возвращает набор узлов выше по потоку от предоставленных узлов. Возвращаемый список включает в себя исходные предоставленные узлы.
`s.findProcessorForID(String id, boolean recursive)`	Узел	Возвращает узел с указанным значением ID или значение None, если такого узла не существует. Если флаг рекурсии равен `true`, то выполняется поиск всех составных узлов в этой диаграмме.

Например, если в потоке есть один узел фильтра, к которому должен обратиться сценарий, этот узел можно найти с помощью следующего кода в сценарии:

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

Другой вариант - если известен ID узла (показанный на вкладке Аннотации диалогового окна узла), можно использовать для поиска узла этот ID, например:

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # ID узла фильтра
...
```

Задание свойств

У узлов, потоков, моделей и выводов есть свойства, доступные для обращения и в большинстве случаев - для задания. Обычно свойства используются для изменения поведения или условий

создания объектов. Методы, доступные для обращения к свойствам объектов и для задания этих свойств, сведены в следующей таблице.

Таблица 11. Методы для доступа к свойствам объектов и для задания этих свойств

Метод	Возвращаемый тип	Описание
`p.getPropertyValue(propertyName)`	Объект	Возвращает значение именованного свойства или значение None, если такого свойства не существует.
`p.setPropertyValue(propertyName, value)`	Неприменимо	Задает значение именованного свойства.
`p.setPropertyValues(properties)`	Неприменимо	Задает значения именованных свойств. Каждая запись в карте свойств состоит из ключа, представляющего имя свойства, и значения, которое должно быть назначено этому свойству.
`p.getKeyedPropertyValue(propertyName, keyName)`	Объект	Возвращает значение именованного свойства и связанный ключ или значение None, если такого свойства или ключа не существует.
`p.setKeyedPropertyValue(propertyName, keyName, value)`	Неприменимо	Задает значение именованного свойства и ключ.

Например, если вы хотите задать значение узла файла переменных при запуске потока, можно использовать следующий сценарий:

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

Другой вариант - вам может потребоваться отфильтровать поле с узла Фильтр. В этом случае значение также сопровождается ключом для имени поля, например:

```
stream = modeler.script.stream()
# Найти узел Фильтр ...
node = stream.findByType("filter", None)
# ... и отфильтровать поле "Na"
node.setKeyedPropertyValue("include", "Na", False)
```

Создание узлов и изменение потоков

В некоторых ситуациях вам может потребоваться добавить новые узлы к существующим потокам. Добавление узлов к существующим потокам обычно включает в себя следующие задачи:

1. Создание узлов.
2. Соединение узлов с процессами существующего потока.

Создание узлов

Потоки предоставляют несколько способов создания узлов. Соответствующие методы сведены в следующей таблице.

Таблица 12. Методы для создания узлов		
Метод	**Возвращаемый тип**	**Описание**
`s.create(nodeType, name)`	Узел	Создает узел заданного типа и добавляет его в заданный поток.
`s.createAt(nodeType, name, x, y)`	Узел	Создает узел заданного типа и добавляет его в заданный поток в заданном положении. Если x < 0 или y < 0, положение не задается.
`s.createModelApplier(modelOutput, name)`	Узел	Создает узел применения модели, полученный из предоставленного объекта вывода модели.

Например, чтобы создать в потоке новый узел типа, вы можете использовать следующий сценарий:

```
stream = modeler.script.stream()
# Создать новый узел типа
node = stream.create("type", "My Type")
```

Соединение и отсоединение узлов

Когда в потоке создается новый узел, до использования его нужно соединить с последовательностью узлов. Потоки обеспечивают несколько способов соединения и отсоединения узлов. Соответствующие методы сведены в следующей таблице.

Таблица 13. Методы для соединения и отсоединения узлов		
Метод	**Возвращаемый тип**	**Описание**
`s.link(source, target)`	Неприменимо	Создает новое соединение между узлом источника и узлом назначения.
`s.link(source, targets)`	Неприменимо	Создает новые соединения между узлом источника и каждым узлом назначения в предоставленном списке.
`s.linkBetween(inserted, source, target)`	Неприменимо	Соединяет узел между двумя другими экземплярами узлов (узлами источника и назначения) и задает положение для вставки узла между ними. Сначала удаляются все непосредственные соединения между узлами источника и назначения.
`s.linkPath(path)`	Неприменимо	Создает новый путь между экземплярами узла. Первый узел соединяется со вторым, второй с третьим и так далее.

Таблица 13. Методы для соединения и отсоединения узлов (продолжение)

Метод	Возвращаемый тип	Описание
`s.unlink(source, target)`	Неприменимо	Удаляет прямые соединения между узлами источника и назначения.
`s.unlink(source, targets)`	Неприменимо	Удаляет все прямые соединения между узлом источника и каждым объектом в списке назначения.
`s.unlinkPath(path)`	Неприменимо	Удаляет все пути, существующие между экземплярами узла.
`s.disconnect(node)`	Неприменимо	Удаляет все соединения между предоставленным узлом и всеми другими узлами в заданном потоке.
`s.isValidLink(source, target)`	*логическое*	Возвращает значение `True`, если можно создать соединение между заданными узлами источника и назначения. При этом методе проверяется, что оба объекта принадлежат заданному потоку, что узел источника позволяет устанавливать соединение, а узел назначения - принять его, и что при создании такого соединения в потоке не возникнет зацикливания.

Приведенный пример сценария выполняет пять задач:

1. Создает входной узел файла переменных и выходной узел Таблица.
2. Соединяет узлы.
3. Задает имя входного узла файла переменных.
4. Фильтрует поле "Drug" в полученном выводе.
5. Выполняет узел Таблица.

```
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)
```

Импорт, замена и удаление узлов

Кроме создания узлов и соединения с ними, часто необходима замена или удаление узлов из потока. Доступные для импорта, замены и удаления узлов методы сведены в следующей таблице.

Таблица 14. Методы для импорта, замены и удаления узлов

Метод	Возвращаемый тип	Описание
`s.replace(originalNode, replacementNode, discardOriginal)`	Неприменимо	Заменяет заданный узел в указанном потоке. Заданному потоку должны принадлежать и исходный узел, и узел для замены.
`s.insert(source, nodes, newIDs)`	Список	Вставляет копии узлов из предоставленного списка. Предполагается, что все узлы из предоставленного списка содержатся в заданном потоке. Флаг `newIDs` указывает, должны ли генерироваться новые ID для всех узлов, или нужно копировать и использовать только существующий ID. Предполагается, что у всех узлов в потоке уникальные ID, поэтому для этого флага нужно задать значение `True`, если поток источника совпадает с заданным. Этот метод возвращает список вновь вставленных узлов, где порядок узлов не определен (то есть порядок не обязан быть тем же, что во входном списке узлов).
`s.delete(node)`	Неприменимо	Удаляет заданный узел из указанного потока. Владельцем узла должен быть заданный поток.
`s.deleteAll(nodes)`	Неприменимо	Удаляет все заданные узлы из заданного потока. Все узлы в собрании должны принадлежать к заданному потоку.
`s.clear()`	Неприменимо	Удаляет все узлы из заданного потока.

Перемещение по узлам в потоке

Общее требование - это определение узлов, расположенных дальше по потоку или предшествующих в потоке данному узлу. Поток обеспечивает несколько методов, которые можно использовать для обнаружения таких узлов. Соответствующие методы сведены в следующей таблице.

Таблица 15. Методы для обнаружения узлов выше и ниже по потоку

Метод	Возвращаемый тип	Описание
`s.iterator()`	Итератор	Возвращает итератор по объектам узла, содержащимся в заданном потоке. Если поток изменяется между вызовами функции `next()`, поведение итератора не определено.
`s.predecessorAt(node, index)`	Узел	Возвращает заданный непосредственный предшественник предоставленного узла или значение `None`, если значение индекса выходит за границы.
`s.predecessorCount(node)`	*int*	Возвращает количество непосредственных предшественников предоставленного узла.
`s.predecessors(node)`	Список	Возвращает непосредственные предшественники предоставленного узла.
`s.successorAt(node, index)`	Узел	Возвращает заданный непосредственный преемник предоставленного узла или значение `None`, если значение индекса выходит за границы.
`s.successorCount(node)`	*int*	Возвращает количество непосредственных преемников предоставленного узла.
`s.successors(node)`	Список	Возвращает непосредственные преемники предоставленного узла.

Элементы, очистка или удаление

Унаследованные сценарии поддерживают разнообразное использование команды `clear`, например:

- `clear outputs` Удалить все элементы вывода с палитры менеджера.
- `clear generated palette` Очистить все слепки моделей с палитры Модели.
- `clear stream` Удалить содержимое потока.

Сценарии Python поддерживают аналогичный набор функций; команда `removeAll()` используется для очистки менеджеров потоков, выводов и моделей. Например:

- Чтобы очистить менеджер потоков:

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

- Чтобы очистить менеджер выводов:

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

- Чтобы очистить менеджер моделей:

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

Получение информации об узлах

Узлы относятся к нескольким разным категориям, таким как узлы импорта и экспорта данных, узлы построения моделей и узлы других типов. Каждый узел обеспечивает несколько методов, которые можно использовать для поиска информации об этом узле.

Методы, которые можно использовать для получения ID, имени и метки узла, сведены в следующей таблице.

Таблица 16. Методы для получения ID, имени и метки узла

Метод	Возвращаемый тип	Описание
`n.getLabel()`	*строка*	Возвращает метку вывода на экран для заданного узла. Данная метка - это значение свойства `custom_name`, если это свойство - не пустая строка и свойство `use_custom_name` не задано; в противном случае метка - это значение `getName()`.
`n.setLabel(label)`	Неприменимо	Задает метку вывода на экран для заданного узла. Если новая метка - это не пустая строка, она назначается свойству `custom_name`, а свойству `use_custom_name` назначается значение `False`, так что у заданной метки есть преимущество использования; в противном случае свойству `custom_name` назначается пустая строка, а свойству `use_custom_name` назначается значение `True`.
`n.getName()`	*строка*	Возвращает имя заданного узла.
`n.getID()`	*строка*	Возвращает ID заданного узла. Новый ID создается при всяком создании нового узла. Этот ID остается с узлом при его сохранении как части потока, то есть при открытии потока ID узлов сохраняются. Однако если сохраненный узел вставляется в поток, он рассматривается как новый объект, и для него выделяется новый ID.

Методы, которые можно использовать для получения другой информации об узле, сведены в следующей таблице.

Таблица 17. Методы для получения информации об узле

Метод	Возвращаемый тип	Описание
`n.getTypeName()`	*строка*	Возвращает имя в сценарии для этого узла. Это то же самое имя, которое можно использовать для создания нового экземпляра этого узла.
`n.isInitial()`	*Логическое значение*	Возвращает значение `True`, если это *исходный* узел, существовавший при запуске потока.
`n.isInline()`	*Логическое значение*	Возвращает значение `True`, если это *встроенный* узел, появившийся в середине потока.
`n.isTerminal()`	*Логическое значение*	Возвращает значение `True`, если это *конечный* узел, создаваемый при завершении потока.
`n.getXPosition()`	*int*	Возвращает смещение положения x для узла в потоке.
`n.getYPosition()`	*int*	Возвращает смещение положения y для узла в потоке.
`n.setXYPosition(x, y)`	Неприменимо	Задает положение узла в потоке.
`n.setPositionBetween(source, target)`	Неприменимо	Задает положение узла в потоке, чтобы он располагался между двумя указанными узлами.
`n.isCacheEnabled()`	*Логическое значение*	Возвращает значение `True`, если включено кэширование; в противном случае возвращает `False`.
`n.setCacheEnabled(val)`	Неприменимо	Включает или отключает кэш для этого объекта. Если кэш заполнен и возможность кэширования отключается, происходит очистка кэша.
`n.isCacheFull()`	*Логическое значение*	Возвращает значение `True`, если кэш заполнен; в противном случае возвращает `False`.
`n.flushCache()`	Неприменимо	Очищает кэш для этого узла. Не приводит ни к какому действию, если кэш не включен или не заполнен.

Глава 4. API сценариев

Введение в API сценариев

API сценариев предоставляет доступ к широкому диапазону функциональных возможностей SPSS Modeler. Все описанные до сих пор методы - это часть API, к ним можно неявно обратиться в сценарии без дальнейшего импорта. Однако если вы хотите сослаться на классы API, необходимо явным образом импортировать API с помощью следующего оператора:

```
import modeler.api
```

Этот оператор import требуется во многих примерах API сценариев.

Полное руководство по классам, методам и параметрам, доступным через API сценариев, можно найти в документе *Справочное руководство по API сценариев Python IBM SPSS Modeler*.

Пример 1: поиск узлов с помощью пользовательского фильтра

В раздел “Поиск узлов” на стр. 31 включен пример поиска для узла в потоке с помощью имени типа узла в качестве критерия поиска. В некоторых ситуациях требуется более общий поиск, и его можно реализовать с помощью класса `NodeFilter` и метода потока `findAll()`. Поиск такого типа включает в себя следующие два шага:

1. Создание нового класса, который расширяет `NodeFilter`, реализующий пользовательскую версию метода `accept()`.
2. Вызов метода потока `findAll()` с помощью экземпляра этого нового класса. При этом возвращаются все узлы, для которых выполнен критерий, определенный в методе `accept()`.

В следующем примере показано, как искать узлы в потоке, для которых включен кэш узлов. Возвращенный список узлов можно использовать для очищения или отключения их кэша.

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
    """Фильтр узлов с включенным кэшированием"""
    def accept(this, node):
        return node.isCacheEnabled()

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

Пример 2: разрешение пользователям получать информацию о каталоге или файле на основе их привилегий

Чтобы не открывать PSAPI для пользователей, можно использовать метод `session.getServerFileSystem()` через вызов функции PSAPI для создания объекта файловой системы.

В следующем примере показано, как разрешить пользователю получить информацию о каталоге или файле на основании привилегий пользователя, соединенного с 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()
```

Метаданные: Информация о данных

Поскольку узлы соединены, образуя поток, доступна информация о столбцах или полях на каждом узле. Например, в пользовательском интерфейсе Modeler это дает возможность выбрать поля, по которым нужно выполнить сортировку или объединение. Такая информация называется моделью данных.

Кроме того, сценарии могут обращаться к модели данных путем поиска по входным или выходным полям узла. Для некоторых узлов входная и выходная модели данных одинаковы; например, узел сортировки изменяет только порядок записей, не изменяя модель данных. Некоторые узлы, такие как узел вычислений, могут добавлять новые поля. Другие, например, узел фильтра, могут переименовывать и удалять поля.

В приведенном ниже примере сценарий обращается к стандартному потоку IBM SPSS Modeler `druglearn.str` и для каждого поля строит модель, в которой одно входное поле отбрасывается. Для этого выполняется следующие действия:

1. Оценка модели данных на выходе узла типа.
2. Цикл, перебирающий все поля входной модели данных.
3. Изменение узла фильтра для каждого входного поля.
4. Изменение имени создаваемой модели.
5. Выполнение узла построения модели.

Прим.: Перед запуском сценария в потоке `druglean.str` не забудьте задать язык сценариев Python (поскольку поток был создан в прошлой версии IBM SPSS Modeler, для него задан унаследованный язык сценариев).

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Всегда используйте пользовательское имя модели
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # Если поле выходное, оно игнорируется
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Снова активировать последнее удаленное поле
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Удалить поле
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Задать имя новой модели и выполнить построение
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

Объектом DataModel поддерживается ряд методов для оценки информации о полях и столбцах в модели данных. Соответствующие методы сведены в следующей таблице.

Таблица 18. Методы объекта модели данных для доступа к информации о полях или столбцах		
Метод	**Возвращаемый тип**	**Описание**
`d.getColumnCount()`	*int*	Возвращает количество столбцов в модели данных.
`d.columnIterator()`	Итератор	Возвращает итератор, который возвращает каждый столбец в "естественном" порядке вставки. Итератор возвращает экземпляры столбца.
`d.nameIterator()`	Итератор	Возвращает итератор, который возвращает имя каждого столбца в "естественном" порядке вставки.
`d.contains(name)`	*Логическое значение*	Возвращает значение `True`, если столбец с указанным именем существует в этой модели данных, и значение `False` - в противном случае.
`d.getColumn(name)`	Столбец	Возвращает столбец с указанным именем.
`d.getColumnGroup(name)`	ColumnGroup	Возвращает именованную группу столбцов или `None`, если такая группа столбцов не существует.
`d.getColumnGroupCount()`	*int*	Возвращает количество групп столбцов в этой модели данных.
`d.columnGroupIterator()`	Итератор	Возвращает итератор, который возвращает каждый столбец группы по очереди.
`d.toArray()`	Column[]	Возвращает модель данных в виде массива столбцов. Столбцы упорядочены в "естественном" порядке вставки.

Каждый объект поля (столбца) содержит ряд методов для доступа к информации об этом столбце. Некоторые из них представлены в следующей таблице.

Таблица 19. Методы объекта столбца для доступа к информации о столбце		
Метод	**Возвращаемый тип**	**Описание**
`c.getColumnName()`	*строка*	Возвращает имя столбца.
`c.getColumnLabel()`	*строка*	Возвращает метку столбца или пустую строку, если со столбцом не связана никакая метка.
`c.getMeasureType()`	MeasureType	Возвращает тип показателя для столбца.

Таблица 19. Методы объекта столбца для доступа к информации о столбце (продолжение)

Метод	**Возвращаемый тип**	**Описание**
`c.getStorageType()`	StorageType	Возвращает тип хранения для столбца.
`c.isMeasureDiscrete()`	*Логическое значение*	Возвращает значение `True`, если столбец - дискретный. Дискретными считаются столбцы типа набора или флага.
`c.isModelOutputColumn()`	*Логическое значение*	Возвращает значение `True`, если столбец - это выходной столбец модели.
`c.isStorageDatetime()`	*Логическое значение*	Возвращает значение `True`, если тип хранения столбца - время, дата или отметка времени.
`c.isStorageNumeric()`	*Логическое значение*	Возвращает значение `True`, если тип хранения столбца - целое или действительное число.
`c.isValidValue(value)`	*Логическое значение*	Возвращает значение `True`, если заданное значение допустимо для этого типа хранения, и `valid`, если для столбца известны допустимые значения.
`c.getModelingRole()`	ModelingRole	Возвращает роль моделирования для столбца.
`c.getSetValues()`	Object[]	Возвращает массив допустимых значений для столбца или `None`, если нет известных значений или столбец не задан.
`c.getValueLabel(value)`	*строка*	Возвращает метку для значения в столбце или пустую строку, если с этим значением не связана никакая метка.
`c.getFalseFlag()`	Объект	Возвращает индикаторное значение "false" для столбца или `None`, если значение неизвестно или столбец - не флаг.
`c.getTrueFlag()`	Объект	Возвращает индикаторное значение "true" для столбца или `None`, если значение неизвестно или столбец - не флаг.

Таблица 19. Методы объекта столбца для доступа к информации о столбце (продолжение)		
Метод	**Возвращаемый тип**	**Описание**
`c.getLowerBound()`	Объект	Возвращает значение нижней границы для значений в столбце или None, если значение неизвестно или столбец - не непрерывный.
`c.getUpperBound()`	Объект	Возвращает значение верхней границы для значений в столбце или None, если значение неизвестно или столбец - не непрерывный.

Имейте в виду, что для большинства методов доступа к информации о столбце есть эквивалентные методы, определенные в самом объекте модели данных. Например, следующие два оператора эквивалентны:

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

Доступ к сгенерированным объектам

При выполнении потока обычно создаются дополнительные объекты вывода. Этими дополнительными объектами могут быть новая модель или часть вывода, предоставляющая информацию для последующих выполнений.

В приведенном ниже примере поток `druglearn.str` снова используется как стартовая точка для потока. В этом примере выполняются все узлы в потоке и результаты сохраняются в списке. Затем сценарий перебирает результаты в цикле, а все выводы модели, получаемые при выполнении, сохраняются как файл модели IBM SPSS Modeler (`.gm`), и модель экспортируется в формате PMML.

```
import modeler.api

stream = modeler.script.stream()

# Задайте здесь существующую папку в вашей системе.
# Включите в имя завершающий разделитель каталогов
modelFolder = "C:/temp/models/"

# Выполнить поток
models = []
stream.runAll(models)

# Сохранить все созданные модели
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # Если при выполнении потока создаются другие объекты вывода, игнорировать их
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

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

    # сохранить каждую модель...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...и экспортировать PMML каждой модели...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

Класс запуска задач предоставляет удобный способ запуска различных общих задач. Доступные в этом классе методы сведены в следующей таблице.

Таблица 20. Методы класса запуска задач для выполнения общих задач		
Метод	**Возвращаемый тип**	**Описание**
`t.createStream(name, autoConnect, autoManage)`	Поток	Создает и возвращает новый поток. Обратите внимание на то, что в коде, который должен создавать скрытые потоки, невидимые для пользователя, для флага `autoManage` должно быть задано значение `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	Неприменимо	Экспортирует описание потока в файл с использованием заданного формата файла.
`t.exportModelToFile(model Output, filename, fileFormat)`	Неприменимо	Экспортирует модель в файл с использованием заданного формата файла.
`t.exportStreamToFile(stre am, filename, fileFormat)`	Неприменимо	Экспортирует поток в файл с использованием заданного формата файла.
`t.insertNodeFromFile(file name, diagram)`	Узел	Читает и возвращает узел из заданного файла, вставляя его в предоставленную диаграмму. Обратите внимание на то, что это может использоваться для чтения и объектов узлов, и объектов надузлов.
`t.openDocumentFromFile(fi lename, autoManage)`	DocumentOutput	Читает и возвращает документ из заданного файла.
`t.openModelFromFile(filen ame, autoManage)`	ModelOutput	Читает и возвращает модель из заданного файла.
`t.openStreamFromFile(file name, autoManage)`	Поток	Читает и возвращает поток из заданного файла.
`t.saveDocumentToFile( documentOutput, filename)`	Неприменимо	Сохраняет документ в заданное положение файла.
`t.saveModelToFile(modelOu tput, filename)`	Неприменимо	Сохраняет модель в заданное положение файла.
`t.saveStreamToFile(stream , filename)`	Неприменимо	Сохраняет поток в заданное положение файла.

Обработка ошибок

Язык программирования Python предоставляет средства обработки ошибок через блок кода `try...except`. Его можно использовать в сценариях для локализации исключительных ситуаций и исправления ошибок, которые в противном случае привели бы к прерыванию сценария.

В приведенном ниже примере сценария сделана попытка получить модель из IBM SPSS Collaboration and Deployment Services Repository. Эта операция может привести к возникновению исключительной ситуации, например, могут быть неправильно заданы регистрационные данные при входе в репозиторий, или указанный путь к репозиторию может быть неправильным. В данном сценарии из-за этого может возникнуть исключительная ситуация `ModelerException` (все

исключительные ситуации, генерируемые IBM SPSS Modeler, являются производными от `modeler.api.ModelerException`).

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # печать направляется на вкладку Отладка панели сценариев пользовательского интерфейса
моделирования
    print "Все в порядке"
except modeler.api.ModelerException, e:
    print "Произошла ошибка:", e.getMessage()
```

Прим.: Некоторые операции сценариев могут вызвать появление стандартных исключительных ситуаций Java; они не являются производными от `ModelerException`. Чтобы учесть эти исключительные ситуации, можно использовать дополнительный блок отслеживания всех исключительных ситуаций Java, например:

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # печать направляется на вкладку Отладка панели сценариев пользовательского интерфейса
моделирования
    print "Все в порядке"
except modeler.api.ModelerException, e:
    print "Произошла ошибка:", e.getMessage()
except java.lang.Exception, e:
    print "Произошла исключительная ситуация Java:", e.getMessage()
```

Параметры потока, сеанса и надузла

Параметры предоставляют удобный способ передачи значений во время выполнения, чтобы не кодировать их непосредственно в сценарии. Параметры и их значения определяются так же, как для потоков, то есть как записи в таблице потока или надузла или параметрами в командной строке. Классы потока и надузла реализуют набор функций, определенный объектом ParameterProvider, как показано в следующей таблице. Сеанс обеспечивает вызов `getParameters()`, в результате чего возвращается объект, определяющий эти функции.

Таблица 21. Функции, определенные объектом ParameterProvider

Метод	Возвращаемый тип	Описание
`p.parameterIterator()`	Итератор	Возвращает итератор имен параметров для этого объекта.
`p.getParameterDefinition(parameterName)`	ParameterDefinition	Возвращает определение для параметра с заданным именем или значение None, если у провайдера такого параметра нет. Результат может быть снимком определения на момент вызова метода и не должен отображать последовательные изменения, произведенные данным провайдером для параметра.

Таблица 21. Функции, определенные объектом ParameterProvider (продолжение)		
Метод	**Возвращаемый тип**	**Описание**
`p.getParameterLabel(parameterName)`	*строка*	Возвращает метку именованного параметра или значение None, если такого параметра не существует.
`p.setParameterLabel(parameterName, label)`	Неприменимо	Задает метку именованного параметра.
`p.getParameterStorage(parameterName)`	ParameterStorage	Возвращает систему хранения именованного параметра или значение None, если такого параметра не существует.
`p.setParameterStorage(parameterName, storage)`	Неприменимо	Задает систему хранения именованного параметра.
`p.getParameterType(parameterName)`	ParameterType	Возвращает тип именованного параметра или значение None, если такого параметра не существует.
`p.setParameterType(parameterName, type)`	Неприменимо	Задает тип именованного параметра.
`p.getParameterValue(parameterName)`	Объект	Возвращает значение именованного параметра или значение None, если такого параметра не существует.
`p.setParameterValue(parameterName, value)`	Неприменимо	Задает значение именованного параметра.

В следующем примере сценарий агрегирует некоторые данные Telco для определения, в каком регионе самые низкие значения среднего дохода. Затем для этого региона задается параметр потока. После этого данный параметр потока используется на узле Выбор для исключения данных этого региона до построения модели перехода в конкурирующую компанию на основании оставшихся данных.

Это искусственный пример, так как сценарий сам генерирует узел Выбор, то есть может использовать сгенерированные точные значения непосредственно в выражении узла Выбор. Однако потоки обычно строятся заранее, поэтому задание параметров таким образом предоставляет полезный пример.

В первой части сценария для этого примера создается параметр потока, который будет определять регион с наименьшим средним доходом. Сценарий создает также узлы на ветви агрегации и на ветви построения модели и соединяет их.

```
import modeler.api

stream = modeler.script.stream()

# Инициализировать параметр потока
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# Сначала создаем ветвь агрегации для вычисления среднего дохода по регионам
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")
# Указываем параметр потока при выборе
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)
```

В этом примере сценарий создает следующий поток.

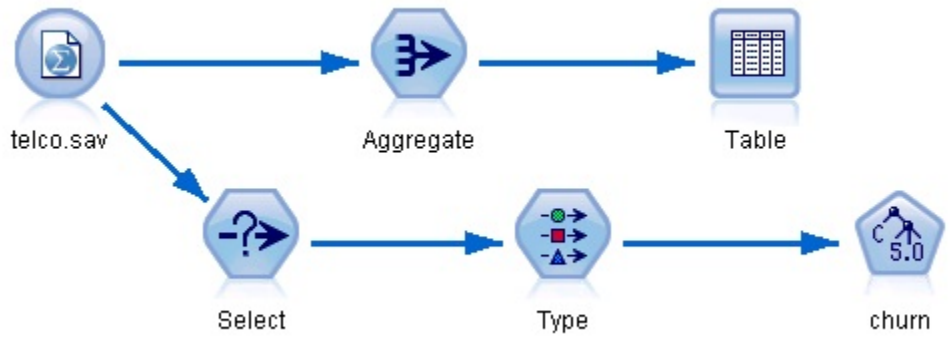

Рисунок 5. Поток, возникающий при выполнении сценария в примере

В следующей части сценария примера выполняется узел Таблица в конце ветви агрегации.

```
# Сначала выполним узел Таблица
results = []
tablenode.run(results)
```

В следующей части примера сценарий обращается к табличному выводу, сгенерированному при выполнении узла Таблица. Затем сценарий проводит итерации по строкам таблицы, находя регион с минимальным средним доходом.

```
# При выполнении узла Таблица в качестве вывода должна быть создана одна таблица
table = results[0]

# Табличный вывод содержит RowSet, поэтому к значениям можно обращаться, как к строкам и
столбцам
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# По тому, как определен узел агрегации, первый столбец
# содержит регион, а второй - средний доход
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
```

В следующей части сценария примера используется регион с наименьшим средним доходом, чтобы задать параметр потока "LowestRegion", созданный ранее. Затем сценарий запускает построитель моделей, в котором заданный регион исключен из данных обучения.

```
# Проверяем, что значение было назначено
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# В завершение запускаем построитель модели с критерием выбора
c50node.run([])
```

Полный пример сценария показан ниже.

```
import modeler.api

stream = modeler.script.stream()

# Создаем параметр потока
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# Сначала создаем ветвь агрегации для вычисления среднего дохода по регионам
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")
# Указываем параметр потока при выборе
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)


# Сначала выполним узел Таблица
results = []
tablenode.run(results)

# При выполнении узла Таблица в качестве вывода должна быть создана одна таблица
table = results[0]

# Табличный вывод содержит RowSet, поэтому к значениям можно обращаться, как к строкам и
столбцам
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# По тому, как определен узел агрегации, первый столбец
# содержит регион, а второй - средний доход
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

# Проверяем, что значение было назначено
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)
```

```
# В завершение запускаем построитель модели с критерием выбора
c50node.run([])
```

Глобальные значения

Глобальные значения используются при вычислении разнообразных сводных статистических показателей для заданных полей. К этим сводным значениям можно получить доступ отовсюду в потоке. Глобальные значения похожи на параметры потока тем, что к ним можно обратиться отовсюду в потоке по имени. Они отличаются от параметров потока тем, что связанные значения изменяются автоматически при запуске узла Задать глобальные значения, а не назначаются сценарием или из командной строки. К глобальным значениям для потока можно обратиться, вызвав метод потока `getGlobalValues()`.

Объект GlobalValues определяет функции, показанные в следующей таблице.

Таблица 22. Функции, определенные объектом GlobalValues

Метод	Возвращаемый тип	Описание
`g.fieldNameIterator()`	Итератор	Возвращает итератор для каждого имени поля по крайней мере с одним глобальным значением.
`g.getValue(type, fieldName)`	Объект	Возвращает глобальное значение для заданного типа и имени поля или None, если значение не удается обнаружить. В общем случае предполагается, что возвращаемое значение - это число, хотя будущие функциональные возможности позволят возвращать различные типы данных.
`g.getValues(fieldName)`	Отобразить	Возвращает карту, содержащую известные записи для заданного имени поля, или значение None, если для этого поля не существует записей.

`GlobalValues.Type` определяет тип доступных сводных статистических показателей. Доступны следующие сводные статистики:

- `MAX`: максимальное значение в поле.
- `MEAN`: среднее значение в поле.
- `MIN`: минимальное значение в поле.
- `STDDEV`: среднеквадратичное отклонение значений в поле.
- `SUM`: сумма значений в поле.

Например, следующий сценарий обращается к среднему значению поля "income", вычисляемому узлом Задать глобальные значения:

```
import modeler.api

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

Работа с несколькими потоками: автономные сценарии

Для работы с несколькими потоками нужно использовать автономный сценарий. Автономный сценарий можно изменить и запустить в пользовательском интерфейсе IBM SPSS Modeler или передать как параметр командной строки в пакетном режиме.

Следующий автономный сценарий открывает два потока. Один из этих потоков строит модель, а второй создает графики распределения предсказанных значений.

```
# Перейти в нужный каталог вашей системы
demosDir = "C:/Program Files/IBM/SPSS/Modeler/18.3.0/DEMOS/streams/"

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

# Открыть поток построения модели, найти узел C5.0 и запустить его
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Теперь откроем поток построения графиков, найдем извлеченные значения Na_to_K и гистограмму
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Создаем узел применения модели, вставляем его между узлами извлечения данных и гистограммы
# и затем запускаем гистограмму
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([])

# Наконец, завершаем потоки
buildstream.close()
plotstream.close()
```

В следующем примере показано, как можно выполнить перебор для открытых потоков (все потоки открываются на вкладке Потоки). Учтите, что это поддерживается только в автономных сценариях.

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

Глава 5. Подсказки для сценариев

В этом разделе представлен обзор подсказок и способов использования сценариев, в том числе изменение выполнения потоков, использование в сценариях закодированных паролей и доступ к объектам в IBM SPSS Collaboration and Deployment Services Repository.

Изменение выполнения потока

Когда поток запущен, его конечные узлы выполняются в порядке, который оптимизирован для ситуации по умолчанию. В некоторых случаях вы можете предпочесть другой порядок выполнения. Чтобы изменить порядок выполнения потока, выполните следующие шаги на вкладке Выполнение диалогового окна свойств потока:

1. Начните с пустого сценария.
2. Нажмите кнопку **Присоединить сценарий по умолчанию** на панели инструментов, чтобы добавить сценарий потока по умолчанию.
3. Измените порядок операторов в сценарии потока по умолчанию на такой, в котором должны выполняться операторы.

Циклы по узлам

Можно использовать цикл `for`, чтобы пройти цикл по всем узлам в потоке. Например, в следующих двух примерах сценариев выполняется цикл по всем узлам, и имена полей изменяются на верхний регистр во всех узлах Фильтр.

Эти сценарии можно использовать в любом потоке, у которого есть узел Фильтр, даже если фактически никакие поля не фильтруются. Просто добавьте узел Фильтр, который передает все поля, чтобы повсюду изменить имена полей на верхний регистр.

```
# Вариант 1: использовании функции nameIterator() модели данных
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() возвращает имена полей
        для поля в узле node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Вариант 2: использовании функции iterator() модели данных
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() возвращает объекты полей, поэтому требуется
        # вызвать getColumnName(), чтобы получить имя
        для поля в node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

Этот сценарий осуществляет цикл по всем узлам в текущем потоке и проверяет, каждый ли узел - это фильтр. Если это так, сценарий выполняет циклы по всем полям в узле и использует функцию `field.upper()` или `field.getColumnName().upper()`, чтобы изменить имя на верхний регистр.

Доступ к объектам IBM SPSS Collaboration and Deployment Services Repository

Если у вас есть лицензия на IBM SPSS Collaboration and Deployment Services Repository, можно сохранять объекты в репозитории и удалять их оттуда с помощью команд сценария. Используйте

репозиторий для управления жизненным циклом моделей исследования данных и связанных объектов предсказания в контексте прикладных программ предприятия, инструментов и решений.

Соединение с IBM SPSS Collaboration and Deployment Services Repository

Для доступа к репозиторию необходимо сначала сконфигурировать допустимое соединение с ним или через меню **Инструменты** в пользовательском интерфейсе SPSS Modeler, или в командной строке. Дополнительную информацию смотрите в разделе “ Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository” на стр. 72.

Получение доступа к репозиторию

Доступ к репозиторию можно получить из сеанса, например:

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

Получение объектов из репозитория

Используйте в сценарии функции `retrieve*`, чтобы обратиться к различным объектам, в том числе к потокам, моделям, выходным данным и узлам. Сводка функций получения представлена в следующей таблице.

Таблица 23. Функции сценария для получения

Тип объекта	**Функция репозитория**
Поток	repo.retrieveStream(String path, String version, String label, Boolean autoManage)
Модель	repo.retrieveModel(String path, String version, String label, Boolean autoManage)
Вывод	repo.retrieveDocument(String path, String version, String label, Boolean autoManage)
Узел	repo.retrieveProcessor(String path, String version, String label, ProcessorDiagram diagram)

Например, можно получить поток из репозитория, используя следующую функцию:

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

В этом примере будет получен поток `risk_score.str` из указанной папки. Метка `production` определяет, какую версию потока получать, а последний параметр указывает, что для управления потоком выбирается SPSS Modeler (из-за этого, например, поток появится на вкладке **Потоки**, если выводится пользовательский интерфейс SPSS Modeler). Другой вариант - использовать конкретную версию без метки:

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

Прим.: Если оба параметра (версии и метки) - `None`, будет возвращена последняя версия.

Хранение объектов в репозитории

Чтобы использовать сценарии для сохранения объектов в репозитории, используйте функции `store*`. Сводка функций хранения представлена в следующей таблице.

Таблица 24. Функции сценария для сохранения	
Тип объекта	**Функция репозитория**
Поток	repo.storeStream(ProcessorStream stream, String path, String label)
Модель	repo.storeModel(ModelOutput modelOutput, String path, String label)
Вывод	repo.storeDocument(DocumentOutput documentOutput, String path, String label)
Узел	repo.storeProcessor(Processor node, String path, String label)

Например, новую версию потока `risk_score.str` можно сохранить при помощи следующей функции:

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

В этом примере сохраняется новая версия потока, с ним связывается метка `"test"` и возвращается маркер для вновь созданной версии.

Прим.: Если вы не хотите связывать метку с новой версией, передайте для метки значение None.

Управление папками репозитория

Используя папки в репозитории, можно организовать объекты по логическим группам и упростить просмотр взаимосвязанных объектов. Создайте папки с помощью функции `createFolder()`, как в следующем примере:

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

В этом примере новая папка с названием `"cross-sell"` создается в папке `"/projects"`. Эта функция возвращает полный путь к новой папке.

Чтобы переименовать папку, используйте функцию `renameFolder()`:

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

Первый параметр - это полный путь к папке, которая будет переименована, а второй параметр - это новое имя, которое будет назначено этой папке.

Чтобы удалить пустую папку, используйте функцию `deleteFolder()`:

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

Блокировка и разблокировка объектов

В сценарии можно заблокировать объект, чтобы другие пользователи не могли изменить никакую из его версий или создавать новые версии. Можно также разблокировать объект, который вы ранее заблокировали.

Синтаксис для блокирования и разблокирования объекта следующий:

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

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

При сохранении и извлечении объектов REPOSITORY_PATH указывает на положение объекта в репозитории. Путь нужно заключать в кавычки, а в качестве разделителей использовать символы дробной черты. Зависимости от регистра нет.

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

Как вариант, можно использовать не путь в репозитории, а Uniform Resource Identifier (URI), чтобы указать положение объекта. URI должен содержать префикс `spsscr:` и целиком заключаться в кавычки. Допускаются только символы прямой дробной черты, а пробелы должны быть закодированы. То есть, вместо пробела в пути надо использовать `%20`. От регистра URI не зависит. Ниже приведено несколько примеров:

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

Обратите внимание на то, что блокирование применяется ко всем версиям объекта, нельзя заблокировать или разблокировать индивидуальные версии.

Генерирование закодированного пароля

В определенных случаях вам может потребоваться включить пароль в сценарий; например, если нужен доступ к защищенному паролем источнику данных. Закодированные пароли можно использовать в следующих ситуациях:

- Свойства узлов Источник базы данных и Выходные данные
- Аргументы командной строки для регистрации на сервере
- Свойства соединения с базой данных, хранящиеся в файле *.par* (файл параметров, сгенерированный на вкладке Опубликовать узла экспорта)

Через пользовательский интерфейс доступен инструмент генерирования закодированных паролей на основе алгоритма Blowfish (дополнительную информацию смотрите на сайте http://www.schneier.com/blowfish.html). После кодирования пароль можно копировать и сохранять в файлы сценария и в аргументы командной строки. Закодированный пароль хранит свойство узла `epassword`, используемое для `databasenode` и `databaseexportnode`.

1. Чтобы сгенерировать закодированный пароль, в меню Инструменты выберите:

   Закодировать пароль...
2. Задайте пароль в текстовом поле Пароль.
3. Нажмите кнопку **Кодировать**, чтобы сгенерировать случайное кодирование вашего пароля.
4. Нажмите кнопку Копировать, чтобы скопировать закодированный пароль в буфер обмена.
5. Вставьте пароль в нужный сценарий или параметр.

Проверка сценария

Синтаксис всех типов сценариев можно быстро проверить, нажав красную кнопку проверки на панели инструментов диалогового окна Автономный сценарий.

Рисунок 6. Значки панели инструментов потокового сценария

Проверка сценариев предупреждает вас об ошибках в коде и предлагает рекомендации по улучшению. Для просмотра строки с ошибками щелкните по ссылке обратной связи внизу диалогового окна. При этом ошибка будет выделена красным цветом.

Работа со сценариями из командной строки

Сценарии позволяют запускать операции, обычно выполняемые в пользовательском интерфейсе. Просто задайте и запустите автономный сценарий в командной строке при запуске IBM SPSS Modeler. Например:

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

При наличии флага `-script` загружается заданный сценарий, а при флаге `-execute` выполняются все команды в файле сценария.

Совместимость с предыдущими выпусками

Обычно сценарии, созданные в предыдущих выпусках IBM SPSS Modeler, должны без изменений работать в текущем выпуске. Однако сейчас слепки моделей можно автоматически вставлять в поток (это параметр по умолчанию), чтобы или заменить существующий слепок такого типа в потоке, или присоединить его. Произойдет ли это на самом деле, зависит от заданных опций **Добавить в модель в поток** и **Заменить предыдущую модель** (**Инструменты > Опции > Пользовательские опции > Уведомления**). Например, вам может потребоваться изменить сценарий предыдущего выпуска, в котором замена производится посредством удаления существующего слепка и вставки нового.

Созданные в текущем выпуске сценарии могут не работать в более старых выпусках.

Если созданный в более старом выпуске сценарий использует команду, которая была заменена (или объявлена устаревшей), старая форма будет по-прежнему поддерживаться, но появится предупреждение. Например, старое ключевое слово `generated` было заменено на `model`, а `clear generated` - на `clear generated palette`. Использующие старые формы сценарии будут по-прежнему работать, но появится предупреждение.

Доступ к результатам выполнения потока

Многие узлы IBM SPSS Modeler генерируют такие объекты вывода, как модели, диаграммы и табличные данные. Многие из этих выходных данных содержат полезные значения, которые могут использоваться сценариями для проведения последующего выполнения. Эти значения группируются в контейнеры содержимого, или просто контейнеры, к которым можно обратиться при помощи тегов или ID, идентифицирующих каждый контейнер. Способ обращения к этим контейнерам зависит от формата или "модели содержимого", используемой конкретным контейнерном.

Например, для многих выходных данных предсказательных моделей используется вариант XML, называемый PMML, представляющий информацию о модели следующего содержания: какие поля использует дерево решений при каждом разбиении или как и с какой силой соединяются нейроны в нейросети. Выходные данные модели, где используется PMML, поддерживают модель содержимого XML, с помощью которой можно обращаться к этой информации. Например:

```
stream = modeler.script.stream()
# Допустим, что поток содержит один узел построителя моделей
# и что источник данных, предикторы и назначения уже
# сконфигурированы
modelbuilder = stream.findByType("c50", None)
results = []
modelbuilder.run(results)
modeloutput = results[0]

# Теперь, когда у нас есть объект вывода модели C5.0, обратимся
# к соответствующей модели содержимого
cm = modeloutput.getContentModel("PMML")

# Модель содержимого PMML - это типовая модель содержимого на основе XML,
```

```
# где используется синтаксис XPath. Она применяется для поиска имен полей
данных.
# Вызов возвращает список строк, соответствующих значениям XPath
dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
```

IBM SPSS Modeler поддерживает в сценариях следующие модели содержимого:

- **Модель табличного содержимого** предоставляет доступ к простым табличным данным в виде строк и столбцов.
- **Модель содержимого XML** предоставляет доступ к содержимому, хранимому в формате XML.
- **Модель содержимого JSON** предоставляет доступ к содержимому, хранимому в формате JSON.
- **Модель содержимого статистики столбцов** предоставляет доступ к сводной статистике конкретного поля.
- **Модель содержимого попарной статистики столбцов** предоставляет доступ к сводной статистике между двумя полями или значениями между двумя отдельными полями.

Обратите внимание, что следующие узлы не содержат следующие модели содержимого:

- Временные ряды
- Дискриминантный
- SLRM
- TCM
- Все узлы Python
- Все узлы Spark
- Все узлы моделирования баз данных
- Модель расширения
- STP

Модель табличного содержимого

Модель табличного содержимого представляет собой простую модель для обращения к простым данным строк и столбцов. У значений в отдельном столбце должен быть один и тот же тип хранения (например, это могут быть строки или целые числа).

API

Таблица 25. API

Возврат	Метод	Описание
`int`	`getRowCount()`	Возвращает количество строк в этой таблице.
`int`	`getColumnCount()`	Возвращает количество столбцов в этой таблице.
`Текстовое`	`getColumnName(int columnIndex)`	Возвращает имя столбца для заданного индекса столбца. Индексы столбцов начинаются с нуля.
`StorageType`	`getStorageType(int columnIndex)`	Возвращает тип хранения столбца для заданного индекса. Индексы столбцов начинаются с нуля.

Таблица 25. API (продолжение)		
Возврат	**Метод**	**Описание**
Объект	getValueAt(int rowIndex, int columnIndex)	Возвращает значение для заданного индекса строки и столбца. Индексы строк и столбцов начинаются с нуля.
void	reset()	Сбрасывает на диск все внутреннее хранение, связанное с этой моделью содержимого.

Узлы и выходные данные

В следующей таблице перечислены узлы, выполняющие построение выходных данных, в которые включается содержимое указанного типа.

Таблица 26. Узлы и выходные данные		
Имя узла	**Имя вывода**	**ID контейнера**
таблица	таблица	"table"

Пример сценария

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

# Конфигурируем узел импорта файлов переменных
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Далее создаем узел агрегации и соединяем его с узлом файлов переменных
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Конфигурируем узел агрегации
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Затем создаем узел вывода таблицы и соединяем его с узлом агрегации
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]

# Обращаемся к модели содержимого вывода таблицы
tablecontent = tableoutput.getContentModel("table")

# Для каждого столбца выводим на печать его имя, тип и первую строку
# значений из табличного содержимого
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
```

```
    tablecontent.getValueAt(0, col)
    col = col + 1
```

Вывод на вкладке Отладка сценариев будет выглядеть примерно так:

```
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
```

Модель содержимого XML

Модель содержимого XML предоставляет доступ к содержимому на основе XML.

Модель содержимого XML поддерживает возможность доступа к компонентам на основе выражений XPath. Выражения XPath - это строки, определяющие, какие элементы или атрибуты требуются вызывающему абоненту. Модель содержимого XML скрывает подробности конструирования различных объектов и компилирования выражений, которые, как правило, требует поддержка XPath. Это упрощает вызовы из сценариев Python.

В модель содержимого XML входит функция, возвращающая документ XML как строку. Это позволяет пользователям сценариев Python использовать для синтаксического анализа XML предпочитаемую ими библиотеку Python.

API

Таблица 27. API

Возврат	**Метод**	**Описание**
Текстовое	`getXMLAsString()`	Возвращает XML как строку.
число	`getNumericValue(String xpath)`	Возвращает результат оценки пути с числовым типом возврата (например, подсчет числа элементов, соответствующих выражению пути).
boolean	`getBooleanValue(String xpath)`	Возвращает логический результат оценки заданного выражения пути.
Текстовое	`getStringValue(String xpath, String attribute)`	Возвращает значение атрибута или значение узла XML, соответствующее заданному пути.
Список строк	`getStringValues(String xpath, String attribute)`	Возвращает список всех значений атрибутов или значений узлов XML, соответствующих заданному пути.
Список списков строк	`getValuesList(String xpath, <Список строк> attributes, boolean includeValue)`	Возвращает список всех значений атрибутов, соответствующих заданному пути, наряду со значением узла XML, если оно требуется.

Таблица 27. API (продолжение)		
Возврат	**Метод**	**Описание**
Хеш-таблица (ключ:строка, значение:список строк)	getValuesMap(String xpath, String keyAttribute, <Список строк> attributes, boolean includeValue)	Возвращает хеш-таблицу (где в качестве ключа используется атрибут ключа или значение узла XML) и список заданных значений атрибутов в качестве значений таблицы.
boolean	isNamespaceAware()	Возвращает указание, должны ли синтаксические анализаторы XML учитывать пространства имен. Значение по умолчанию - False.
void	setNamespaceAware(boolean value)	Задает, должны ли синтаксические анализаторы XML учитывать пространства имен. Этот метод вызывает также метод reset(), чтобы изменения учитывались последующими вызовами.
void	reset()	Сбрасывает на диск все внутреннее хранение, связанное с этой моделью содержимого (например, кэшированный объект DOM).

Узлы и выходные данные

В следующей таблице перечислены узлы, выполняющие построение выходных данных, в которые включается содержимое указанного типа.

Таблица 28. Узлы и выходные данные		
Имя узла	**Имя вывода**	**ID контейнера**
Большинство построителей моделей	Большинство генерируемых моделей	"PMML"
"autodataprep"	(не задается)	"PMML"

Пример сценария

Код сценариев Python для обращения к содержимому может выглядеть примерно так:

```
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")
```

Модель содержимого JSON

Модель содержимого JSON используется для обеспечения поддержки содержимого в формате JSON. Она предоставляет базовый API, разрешающий вызывающим абонентам извлекать значения в предположении, что они знают, к каким значениям должен быть обеспечен доступ.

API

Таблица 29. API

Возврат	Метод	Описание
Текстовое	getJSONAsString()	Возвращает содержимое JSON как строку.
Объект	getObjectAt(<Список объектов> path, JSONArtifact artifact) throws Exception	Возвращает объект по заданному пути. Заданный корневой артефакт может оказаться пуст; тогда будет использоваться корень содержимого. Возвращаемое значение может быть строкой литерала, целым числом, действительным числом или логическим значением либо артефактом JSON (объектом или массивом JSON).
Хэш-таблица (ключ:объект, значение:объект>	getChildValuesAt(<List of object> path, JSONArtifact artifact) throws Exception	Возвращает дочерние значения указанного пути, если путь ведет к объекту JSON, либо, в противном случае, пуст. Ключи в таблице представляют собой строки, а связанное значение может быть строкой литерала, целым числом, действительным числом или логическим значением либо артефактом JSON (объектом или массивом JSON).
Список объектов	getChildrenAt(<Список объектов> path path, JSONArtifact artifact) throws Exception	Возвращает список объектов по заданному пути, если путь ведет к массиву JSON, либо, в противном случае, пуст. Возвращаемые значения могут быть строкой литерала, целым числом, действительным числом или логическим значением либо артефактом JSON (объектом или массивом JSON).
void	reset()	Сбрасывает на диск все внутреннее хранение, связанное с этой моделью содержимого (например, кэшированный объект DOM).

Пример сценария

При наличии узла построителя вывода, создающего вывод на основе формата JSON, можно обратиться к информации о наборе книг с помощью следующего кода:

```
results = []
outputbuilder.run(results)
output = results[0]
cm = output.getContentModel("jsonContent")

bookTitle = cm.getObjectAt(["books", "ISIN123456", "title"], None)

# Другой вариант: получаем объект книги (book) и используем его в качестве
корня
# для последующих записей
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Получаем все дочерние значения для конкретной книги
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Получаем третью запись book. Предполагается, что значение "books" верхнего
уровня
# содержит массив JSON, который может быть проиндексирован
bookInfo = cm.getObjectAt(["books", 2], None)

# Получаем список всех дочерних записей
allBooks = cm.getChildrenAt(["books"], None)
```

Модель содержимого статистики столбцов и модель содержимого попарной статистики

Модель содержимого статистики столбцов предоставляет доступ к статистике, которая может быть вычислена для каждого поля (одномерной статистике). Модель содержимого попарной статистики предоставляет доступ к статистике, которая может быть вычислена между парами полей или значений в поле.

Возможные статистические показатели:

- Количество
- UniqueCount
- ValidCount
- Среднее значение
- Сумма
- Минимум
- Максимум
- Диапазон
- Дисперсия
- StandardDeviation
- StandardErrorOfMean
- Асимметрия
- SkewnessStandardError
- Эксцесс
- KurtosisStandardError
- Медиана

- Мода
- Пирсона
- Ковариация
- TTest
- FTest

Некоторые значения подходят только для одностолбцовой статистики, тогда как другие - только для попарной статистики.

Вот узлы, которые будут их вычислять эти статистики:

- **Узел статистики** вычисляет статистику столбцов и может вычислить попарную статистику, если будут заданы поля корреляции.
- **Узел аудита данных** вычисляет статистику столбцов и может вычислить попарную статистику, если будет задано поле наложения.
- **Узел средних** генерирует попарную статистику при сравнении пар полей или сравнении значений поля со сводками других полей.

Какие модели содержимого и статистики будут доступны, зависит и от возможностей конкретного узла, и от заданных на узле параметров.

API ColumnStatsContentModel

Таблица 30. API ColumnStatsContentModel

Возврат	Метод	Описание
`List<StatisticType>`	`getAvailableStatistics()`	Возвращает доступные статистики в этой модели. Необязательно, что у всех полей будут значения для всех статистик.
`List<String>`	`getAvailableColumns()`	Возвращает имена столбцов, для которых были вычислены статистики.
`Число`	`getStatistic(String column, StatisticType statistic)`	Возвращает связанные со столбцом значения статистики.
`void`	`reset()`	Сбрасывает на диск все внутреннее хранение, связанное с этой моделью содержимого.

API PairwiseStatsContentModel

Таблица 31. API PairwiseStatsContentModel

Возврат	Метод	Описание
`List<StatisticType>`	`getAvailableStatistics()`	Возвращает доступные статистики в этой модели. Необязательно, что у всех полей будут значения для всех статистик.

Таблица 31. API PairwiseStatsContentModel (продолжение)		
Возврат	**Метод**	**Описание**
`List<String>`	`getAvailablePrimaryColumns()`	Возвращает имена первичных столбцов, для которых были вычислены статистики.
`List<Object>`	`getAvailablePrimaryValues()`	Возвращает значения первичного столбца, для которых для которых были вычислены статистики.
`List<String>`	`getAvailableSecondaryColumns()`	Возвращает имена вторичных столбцов, для которых для которых были вычислены статистики.
`Число`	`getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)`	Возвращает связанные со столбцами значения статистики.
`Число`	`getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)`	Возвращает значения, связанные со значением первичного столбца и вторичным столбцом.
`void`	`reset()`	Сбрасывает на диск все внутреннее хранение, связанное с этой моделью содержимого.

Узлы и выходные данные

В следующей таблице перечислены узлы, выполняющие построение выходных данных, в которые включается содержимое указанного типа.

Таблица 32. Узлы и выходные данные			
Имя узла	**Имя вывода**	**ID контейнера**	**Примечания**
`"means"` (Узел Средние)	`"means"`	`"columnStatistics"`	
`"means"` (Узел Средние)	`"means"`	`"pairwiseStatistics"`	
`"dataaudit"` (Узел Аудит данных)	`"means"`	`"columnStatistics"`	
`"statistics"` (Узел Статистика)	`"statistics"`	`"columnStatistics"`	Генерируется только при объяснении конкретных полей.

Таблица 32. Узлы и выходные данные (продолжение)			
Имя узла	**Имя вывода**	**ID контейнера**	**Примечания**
"statistics" (Узел Статистика)	"statistics"	"pairwiseStatistics"	Генерируется только при оценке степени корреляции полей.

Пример сценария

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

# Конфигурируем входные данные
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

# Теперь создаем узел статистики. Он может генерировать
# и статистику столбцов, и попарную статистику
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
```

Глава 6. Аргументы командной строки

Вызов программного обеспечения

Для запуска IBM SPSS Modeler можно использовать командную строку операционной системы, как описано ниже.

1. Откройте окно DOS (окно командной строки) на компьютере с IBM SPSS Modeler.
2. Для запуска интерфейса IBM SPSS Modeler в интерактивном режиме введите команду `modelerclient` с нужными аргументами, например:

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

Доступные аргументы (флаги) позволяют подключаться к серверу, загружать потоки, выполнять сценарии и указывать при необходимости прочие параметры выполнения.

Использование аргументов командной строки

Аргументы командной строки (также известные как *флаги*) можно присоединить к начальной команде `modelerclient` для изменения вызова IBM SPSS Modeler.

Доступно несколько типов аргументов командной строки; они описаны ниже в этом разделе.

Таблица 33. Типы аргументов командной строки

Тип аргумента	**Где описано**
Системные аргументы	Дополнительную информацию смотрите в разделе “Системные аргументы” на стр. 68.
Аргументы параметров	Дополнительную информацию смотрите в разделе “Аргументы параметров” на стр. 70.
Аргументы соединений с сервером	Дополнительную информацию смотрите в разделе “Аргументы соединения с сервером” на стр. 70.
Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository	Дополнительную информацию смотрите в разделе “ Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository” на стр. 72.
Аргументы соединения с IBM SPSS Analytic Server	Дополнительную информацию смотрите в разделе “ Аргументы соединения с IBM SPSS Analytic Server” на стр. 72.

Например, можно использовать флаги `-server`, `-stream` и `-execute` для соединения с сервером, а затем загрузить и запустить поток, как в следующем примере:

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

Обратите внимание на то, что при запуске для локальной установки клиента аргументы соединения с сервером не требуются.

Значения параметров, содержащие пробелы, могут быть заключены в двойные кавычки, например:

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

Можно также выполнять состояния и сценарии IBM SPSS Modeler подобным образом, задавая соответственно флаги `-state` и `-script`.

Прим.: Если в команде используется структурированный параметр, символ кавычек нужно предварять обратной дробной чертой. Она препятствует удалению кавычек при интерпретации строк.

Отладка аргументов командной строки

Для отладки командной строки используйте команду `modelerclient`, чтобы запустить IBM SPSS Modeler с нужными аргументами. Это позволяет вам проверить, что команды будут выполняться, как предполагается. Вы можете подтвердить также значения любых параметров, переданных из командной строки в диалоговом окне Параметры сеанса (меню Инструменты, параметры Задать сеанс).

Системные аргументы

В следующей таблице описаны системные параметры, доступные для вызова из командной строки пользовательского интерфейса.

Таблица 34. Системные аргументы

Аргумент	**Поведение/описание**
`@` <commandFile>	Символ @ с последующим именем файла определяет список команд. Когда `modelerclient` встречает аргумент, начинающийся с @, он работает с командами из этого файла, как будто они используются в командной строке. Дополнительную информацию смотрите в разделе “Объединение нескольких аргументов” на стр. 73.
`-directory` <dir>	Задает рабочий каталог по умолчанию. В локальном режиме этот каталог используется и для данных, и для вывода. Пример: `-directory c:/` или `-directory c:\\`
`-server_directory` <dir>	Задает каталог сервера для данных по умолчанию. Рабочий каталог, обозначенный флагом `-directory`, используется для выходных данных.
`-execute`	После запуска выполняет любой поток, состояние или сценарий, загруженные при запуске. Если сценарий загружается в дополнение к потоку или состоянию, будет выполняться один этот сценарий.
`-stream` <stream>	При запуске загрузить указанный поток. Можно задать несколько потоков, но в качестве текущего потока будет использоваться последний из них.
`-script` <script>	При запуске загрузить указанный автономный сценарий. Он может быть задан в дополнение к потоку или состоянию, как описано ниже, но при запуске можно загрузить только один сценарий.
`-model` <model>	При запуске загрузить заданную сгенерированную модель (файл формата `.gm`).
`-state` <state>	При запуске загрузить указанное сохраненное состояние.
`-project` <project>	Загрузить заданный проект. При запуске можно загрузить только один проект.

Таблица 34. Системные аргументы (продолжение)	
Аргумент	**Поведение/описание**
`-output` <output>	При запуске загрузить сохраненный объект вывода (файл формата `.cou`).
`-help`	Выводит список аргументов командной строки. При указании этой опции все другие аргументы игнорируются и выводится окно Справка.
`-P` <имя>=<значение>	Используется для задания параметра запуска. Может использоваться также для задания свойств узла (параметров слота).

Прим.: Каталоги по умолчанию можно задать также в пользовательском интерфейсе. Для доступа к этим опциям в меню Файл выберите **Задать рабочий каталог** или **Задать каталог сервера**.

Загрузка нескольких файлов

Из командной строки можно загрузить несколько потоков, состояний и файлов выходных данных при запуске, повторив соответствующий аргумент для каждого загружаемого объекта. Например, чтобы загрузить и запустить два потока с названиями `report.str` и `train.str`, можно использовать следующую команду:

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

Загрузка объектов из IBM SPSS Collaboration and Deployment Services Repository

Так как определенные объекты можно загрузить из файла или из IBM SPSS Collaboration and Deployment Services Repository (если есть лицензия), префикс имени файла `spsscr:` и, дополнительно, `file:` (для объектов на диске) указывает IBM SPSS Modeler, где искать данный объект. Префикс работает со следующими флагами:

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

Этот префикс используется для создания URI, указывающего положение объекта, например, `-stream "spsscr:///folder_1/scoring_stream.str"`. Наличие префикса `spsscr:` требует, чтобы в той же команде было задано допустимое соединение с IBM SPSS Collaboration and Deployment Services Repository. Поэтому полная команда может выглядеть, как в следующем примере:

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

Обратите внимание на то, что в командной строке вы *должны* использовать URI. Более простая форма `REPOSITORY_PATH` не поддерживается. (Это работает только в сценариях). Дополнительные подробности о URI для объектов в IBM SPSS Collaboration and Deployment Services Repository смотрите в теме “Доступ к объектам IBM SPSS Collaboration and Deployment Services Repository ” на стр. 53.

Аргументы параметров

Во время выполнения командной строки IBM SPSS Modeler параметры можно использовать как флаги. В аргументах командной строки флаг `-P` используется для обозначения параметра в форме `-P <имя>=<значение>`.

Параметрами могут быть любые из следующих:

- **Простые параметры** (или параметры, прямо используемые в выражениях CLEM).
- **Параметры слота**, также называемые 'Свойства узла'. Эти параметры используются для изменения параметров узлов в потоке. Дополнительную информацию смотрите в разделе “Обзор свойств узлов” на стр. 77.
- **Параметры командной строки**, используемые для изменения вызова IBM SPSS Modeler.

Например, вы можете предоставить имена пользователей источников данных и пароли в виде флага командной строки следующим образом:

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

Формат тот же, что и у параметра `datasource` свойства узла `databasenode`. Дополнительную информацию смотрите в разделе: “Свойства databasenode” на стр. 98.

Если вы передаете зашифрованный пароль, для последнего параметра надо задать значение `true`. Обратите внимание также на то, что в имени пользователя базы данных и в пароле не надо использовать ведущие пробелы (естественно, кроме случая, когда имя пользователя или пароль действительно содержат ведущие пробелы).

Прим.: Если узел именован, надо заключить имя узла в двойные кавычки, предварив их обратными дробными чертами. Например, если узел источника данных в предыдущем примере называется `Source_ABC`, запись выглядела бы так:

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

Обратная дробная черта требуется также перед кавычками, определяющими структурированный параметр, как в следующем примере источника данных 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%\"}
```

Прим.: Если имя базы данных (в свойстве `datasource`) содержит один или несколько пробелов, точек или знаков подчеркивания, чтобы оно было обработано как строка, можно применить формат "обратная дробная черта двойная кавычка". Например: `"{\"db2v9.7.6_linux\"}"` или `"{\"TDATA 131\"}"`. Кроме этого, всегда заключайте значения строки `datasource` в двойные кавычки и в фигурные скобки, как в следующем примере: `"{\"SQL Server \",spssuser,abcd1234,false}"`.

Аргументы соединения с сервером

Флаг `-server` сообщает IBM SPSS Modeler, что нужно соединиться с общедоступным сервером, а флаги `-hostname`, `-use_ssl`, `-port`, `-username`, `-password` и `-domain` используются, чтобы указать IBM SPSS Modeler, как соединиться с этим общедоступным сервером. Если аргумент -server не задан, используется сервер по умолчанию или локальный сервер.

Примеры

Чтобы соединиться с общедоступным сервером:

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

Чтобы соединиться с кластером серверов:

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

Обратите внимание на то, что для соединения с кластером серверов требуется координатор процессов через IBM SPSS Collaboration and Deployment Services, поэтому аргумент `-cluster` нужно использовать в комбинации с опциями соединения с репозиторием (`spsscr_*`). Дополнительную информацию смотрите в разделе “ Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository” на стр. 72.

Таблица 35. Аргументы соединения с сервером

Аргумент	**Поведение/описание**
`-server`	Запускает IBM SPSS Modeler в режиме сервера, соединяясь с публичным сервером при помощи флагов `-hostname`, `-port`, `-username`, `-password` и `-domain`.
`-hostname <имя>`	Имя хоста компьютера сервера. Доступно только в режиме сервера.
`-use_ssl`	Задает, что соединение должно использовать SSL. Этот флаг не обязательный; по умолчанию SSL *не* используется.
`-port <номер>`	Номер порта заданного сервера. Доступно только в режиме сервера.
`-cluster <имя>`	Задает соединение с кластером серверов, а не с указанным по имени сервером; этот аргумент используется как альтернатива для аргументов `hostname`, `port` и `use_ssl`. Имя - это имя кластера или уникальный URI, определяющий кластер в IBM SPSS Collaboration and Deployment Services Repository. Кластер серверов управляется координатором процессов через IBM SPSS Collaboration and Deployment Services. Дополнительную информацию смотрите в разделе “ Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository” на стр. 72.
`-username <имя>`	Имя пользователя для регистрации на сервере. Доступно только в режиме сервера.
`-password <пароль>`	Пароль для регистрации на сервере. Доступно только в режиме сервера. **Прим.:** Если аргумент `-password` не используется, последует предложение ввести пароль.
`-epassword <строка_закодир._пароля>`	Закодированный пароль для регистрации на сервере. Доступно только в режиме сервера. **Прим.:** Закодированный пароль можно сгенерировать в меню Инструменты прикладной программы IBM SPSS Modeler.
`-domain <имя>`	Домен, используемый для регистрации на сервере. Доступно только в режиме сервера.
`-P <имя>=<значение>`	Используется для задания параметра запуска. Может использоваться также для задания свойств узла (параметров слота).

Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository

Если вы хотите сохранять данные в IBM SPSS Collaboration and Deployment Services или извлекать их оттуда с помощью командной строки, необходимо задать допустимое соединение с IBM SPSS Collaboration and Deployment Services Repository. Например:

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

В следующей таблице перечислены аргументы, которые можно использовать для конфигурирования соединения.

Таблица 36. Аргументы соединения с IBM SPSS Collaboration and Deployment Services Repository

Аргумент	**Поведение/описание**
`-spsscr_hostname` <имя хоста или IP-адрес>	Имя хоста или IP-адрес сервера, на котором установлен IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_port` <номер>	Номер порта, через который IBM SPSS Collaboration and Deployment Services Repository получает доступ к соединению (обычно по умолчанию это 8080).
`-spsscr_use_ssl`	Задает, что соединение должно использовать SSL. Этот флаг не обязательный; по умолчанию SSL *не* используется.
`-spsscr_username` <имя>	Имя пользователя для регистрации в IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <пароль>	Пароль для регистрации в IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` <закодированный пароль>	Закодированный пароль для регистрации в IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_providername` <имя>	Провайдер аутентификации, используемый для входа в систему IBM SPSS Collaboration and Deployment Services Repository (Active Directory или LDAP). Этот аргумент не обязателен, если используется собственный провайдер (Локальный репозиторий).

Аргументы соединения с IBM SPSS Analytic Server

Если вы хотите сохранять данные в IBM SPSS Analytic Server или извлекать их оттуда с помощью командной строки, необходимо задать допустимое соединение с IBM SPSS Analytic Server.

Прим.: Положение по умолчанию Analytic Server предоставляет SPSS Modeler Server. Пользователи могут также определять собственные соединения Analytic Server, выбрав **Инструменты** > **Соединения с Analytic Server**.

В следующей таблице перечислены аргументы, которые можно использовать для конфигурирования соединения.

Таблица 37. Аргументы соединения с IBM SPSS Analytic Server

Аргумент	**Поведение/описание**
`-analytic_server_username`	Имя пользователя для регистрации в IBM SPSS Analytic Server.
`-analytic_server_password`	Пароль для регистрации в IBM SPSS Analytic Server.

Таблица 37. Аргументы соединения с IBM SPSS Analytic Server (продолжение)	
Аргумент	**Поведение/описание**
`-analytic_server_epassword`	Зашифрованный пароль для регистрации в IBM SPSS Analytic Server.
`-analytic_server_credential`	Идентификационные данные, используемые для регистрации в IBM SPSS Analytic Server.

Объединение нескольких аргументов

Несколько аргументов можно объединить в один командный файл, задаваемый при вызове, используя символ @ перед именем файла. Это позволяет укоротить вызов командной строки и преодолеть все ограничения операционных систем на длину команды. Например, следующая команда запуска использует аргументы, заданные в файле, указанном `<commandFileName>`.

```
modelerclient @<commandFileName>
```

Если требуются пробелы, заключите имя файла и путь к командному файлу в кавычки следующим образом:

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

При запуске командный файл может содержать все ранее заданные аргументы индивидуально, по одному аргументу на строку. Например:

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

При записи в командные файлы и ссылках на них убедитесь, что выполнены следующие ограничения:

- Использовать только одну команду на строку.
- Не включайте аргумент `@CommandFile` в командный файл.

Глава 7. Справочник по свойствам

Справочный обзор свойств

Для узлов, потоков, проектов и суперузлов можно задать много различных свойств. Некоторые свойства общие для всех узлов, например, имя, аннотация и подсказки, а другие свойства специфичны для определенных типов узлов. Есть и свойства, относящиеся к операциям потоков высокого уровня, таким как кэширование или поведение надузлов. К свойствам можно обратиться через стандартный пользовательский интерфейс (например, когда вы открываете диалоговое окно для изменения опций узла), а также использовать их в самых разных ситуациях.

- Свойства можно изменять через сценарии, как описано в этом разделе. Дополнительную информацию смотрите в разделе “Синтаксис для свойств” на стр. 75.
- Свойства узлов можно использовать в параметрах надузла.
- Свойства узлов можно использовать также как часть опций командной строки (используя флаг -P) при запуске IBM SPSS Modeler.

В контексту сценариев в IBM SPSS Modeler свойства узлов и потоков часто называются **параметрами слота**. В этом руководстве на них указывается как на свойства узлов и потоков.

Синтаксис для свойств

Свойства можно задать с использованием следующего синтаксиса

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

или:

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

Значения свойств можно получить с использованием следующего синтаксиса:

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

или:

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

где OBJECT - это узел или вывод, PROPERTY - имя свойства узла, на которое ссылается ваше выражение, а KEY - значение ключа для ключевых свойств. Например, следующий синтаксис используется, чтобы найти узел фильтра, а затем задать значение по умолчанию для включения всех полей и отфильтровать поле Age из данных нижележащего уровня:

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

Все используемые в IBM SPSS Modeler узлы можно найти, используя функцию потока `findByType(TYPE, LABEL)`. Нужно задать по крайней мере одно значение TYPE или LABEL.

Структурированные свойства

Есть два способа использования сценариями структурированных свойств для упрощения синтаксического анализа:

- Структурировать имена свойств для сложных узлов, таких как Тип, Фильтр или Баланс.

- Определить формат для одновременного задания нескольких свойств.

Структурирование для сложных интерфейсов

Для правильного синтаксического анализа сценарии для узлов с таблицами и другими сложными интерфейсами (например, для узлов Тип, Фильтр и Баланс) должны следовать конкретной структуре. Для этих свойств нужно более сложное имя, чем имя одного идентификатора, и это имя называется ключом. Например, на узле Фильтр каждое доступное поле (на вышележащем уровне) должно быть включено или выключено. Чтобы сослаться на эту информацию, узел Фильтр хранит по одному элементу информации на поле (что для него задано, true или false). У этого свойства может быть значение (или быть задано значение) `True` или `False`. Допустим, что у узла Фильтр с именем `mynode` есть (на вышележащем уровне) поле с именем `Age`. Чтобы выключить это поле, задайте для свойства `include` с ключом `Age` значение `False`, как указано ниже:

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

Структурирование для задания нескольких свойств

Для многих узлов одновременно можно задать несколько свойств узла или потока. Для этого используется **команда multiset** или **set block**.

В некоторых случаях структурированное свойство может быть весьма сложным. Пример:

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

Другое преимущество структурированных свойств - их возможность задать несколько свойств на узле до стабилизации этого узла. По умолчанию команда multiset задает все свойства в блоке, прежде чем выполнить какие-то действия на основании индивидуальных настроек свойств. Например, при определении узла Фиксированный файл использование двух шагов для задания свойств полей приведет к ошибке, так как узел несогласован, пока оба параметра не окажутся допустимыми. Определение свойств через multiset обходит эту проблему, задавая оба свойства до изменения модели данных.

Сокращения

Во всех синтаксических формах для свойств узлов используются стандартные сокращения. Изучение сокращений полезно для конструирования сценариев.

Таблица 38. Стандартные сокращения, используемые во всех синтаксических формах

Сокращение	**Значение**
abs	Абсолютное значение
len	Длина
мин	Минимум
max	Максимум
correl	Корреляция
covar	Ковариация
num	Число или числовой
pct	Процент или процентная доля
transp	Прозрачность
xval	Перекрестная проверка

Таблица 38. Стандартные сокращения, используемые во всех синтаксических формах (продолжение)	
Сокращение	**Значение**
пер	Дисперсия или переменная (в узлах источника)

Примера свойств узла и потока

Свойства узлов и потоков можно использовать с IBM SPSS Modeler разными способами. Чаще всего они применяются как часть сценария, **автономного**, используемого для автоматизации нескольких потоков или операций, или **потокового**, используемого для автоматизации процессов в одном потоке. Можно задать также параметры узла, используя свойства узла в надузле. На самом базовом уровне свойства можно использовать также как опцию в командной строке для запуска IBM SPSS Modeler. Применив аргумент `-p` как часть вызова из командной строки, можно использовать свойство потока для изменения настроек в потоке.

Таблица 39. Примера свойств узла и потока	
Свойство	**Значение**
`s.max_size`	Относится к свойству `max_size` узла с именем `s`.
`s:samplenode.max_size`	Относится к свойству `max_size` узла с именем `s`, который должен быть узлом выборки.
`:samplenode.max_size`	Относится к свойству `max_size` узла выборки в текущем потоке (должен существовать только один узел выборки).
`s:sample.max_size`	Относится к свойству `max_size` узла с именем `s`, который должен быть узлом выборки.
`t.direction.Age`	Относится к роли поля *Age* на узле Тип с именем `t`.
`:.max_size`	*** ЗАПРЕЩЕНО *** Необходимо задать или имя узла, или тип узла.

Пример `s:sample.max_size` иллюстрирует, что не обязательно полностью указывать типы узлов.

Пример `t.direction.Age` иллюстрирует, что некоторые имена слотов могут быть сами структурированы, когда атрибуты узла сложнее, чем простые индивидуальные слоты с индивидуальными значениями. Такие слоты называются **структурированными** или **сложными** свойствами.

Обзор свойств узлов

У узла каждого типа есть свой собственный набор разрешенных свойств, и у каждого свойства есть тип. Это может быть общий тип (число, флаг, строка), и в этом случае параметры для свойства будут принудительно преобразованы к правильному типу. Ошибка возникает, если такое принудительное преобразование невозможно. Как вариант, ссылка на свойство может задавать диапазон допустимых значений, таких как `Discard`, `PairAndDiscard` и `IncludeAsText`, и в этом случае ошибка возникает при использовании другого значения. Свойства флага должны читаться или задаваться с использованием значений `true` и `false`. (Возможные варианты, в том числе `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` или `0`, также распознаются при задании значений, но в некоторых случаях могут вызвать ошибки при чтении значений свойств. Все другие значения рассматриваются как true. Согласованное использование `true` и `false` исключит любые противоречия). В справочных таблицах этого руководства структурированные свойства указаны в столбце **Описание свойств**, приводятся также форматы их использования.

Общие свойства узлов

Многие свойства общие для всех узлов (в том числе надузлов) в IBM SPSS Modeler.

Таблица 40. Общие свойства узлов

Имя свойства	Тип переменной	Описание свойства
use_custom_name	*флаг*	
name	*string*	Предназначенное только для чтения свойство, по которому читается имя для узла (автоматически сгенерированное или пользовательское) на холсте.
custom_name	*string*	Задает пользовательское имя для узла.
tooltip	*string*	
annotation	*string*	
keywords	*string*	Структурированный слот, задающий список ключевых слов, связанных с объектом (например, ["Ключевое_слово_1" "Ключевое_слово_2"]).
cache_enabled	*флаг*	
node_type	source_supernode process_supernode terminal_supernode Все имена узлов, определяющих сценарий	Предназначенное только для чтения свойство, используемое для ссылки на узел по типу. Например, вместо ссылки на узел только по имени, как real_income, можно задать и тип, такой как userinputnode или filternode.

Специфичные для надузла свойства обсуждаются отдельно от всех других узлов. Дополнительную информацию смотрите в разделе Глава 21, “Свойства надузлов”, на стр. 481.

Глава 8. Свойства потока

При написании сценариев можно управлять многими свойствами потоков. Для указания на свойства потока надо задать метод выполнения, использующий сценарии:

```
stream = modeler.script.stream()
stream.setPropertyValue("метод_выполнения", "сценарий")
```

Пример

Свойство узла используется для ссылки на узлы в текущем потоке. Пример представлен в следующем сценарии потока:

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

annotation = annotation + "\n\nЭтот поток называется \"" + stream.getLabel() + "\" и
 содержит следующие узлы:\n"

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

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

Приведенный пример использует свойство узла для создания списка всех узлов в потоке и записывает этот список в аннотации потоков. Создаваемая аннотация выглядит следующим образом:

```
Этот поток называется "druglearn" и содержит следующие узлы:

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"
```

Свойства потока описываются в следующей таблице.

Таблица 41. Свойства потока

Имя свойства	**Тип переменной**	**Описание свойства**
execute_method	Нормальный Сценарий	

Таблица 41. Свойства потока (продолжение)

Имя свойства	Тип переменной	Описание свойства
date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДД" DAY MONTH "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ" "ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	
date_baseline	*number*	
date_2digit_baseline	*number*	
time_format	"ЧЧММСС" "ЧЧММ" "ММСС" "ЧЧ:ММ:СС" "ЧЧ:ММ" "ММ:СС" "(Ч)Ч:(М)М:(С)С" "(Ч)Ч:(М)М" "(М)М:(С)С" "ЧЧ.ММ.СС" "ЧЧ.ММ" "ММ.СС" "(Ч)Ч.(М)М.(С)С" "(Ч)Ч.(М)М" "(М)М.(С)С"	
time_rollover	*флаг*	
import_datetime_as_string	*флаг*	
decimal_places	*number*	
decimal_symbol	Default Точка Comma	
angles_in_radians	*флаг*	
use_max_set_size	*флаг*	
max_set_size	*number*	

Таблица 41. Свойства потока (продолжение)		
Имя свойства	**Тип переменной**	**Описание свойства**
ruleset_evaluation	Голосование FirstHit	
refresh_source_nodes	*флаг*	Используйте для автоматического обновления узлов источника после выполнения потока.
script	*string*	
annotation	*string*	
name	*string*	**Прим.:** Это свойство предназначено только для чтения. Если вы хотите изменить имя потока, его необходимо сохранить с другим именем.
parameters		Используйте это свойство для изменения параметров потока изнутри автономного сценария.
nodes		Смотрите подробную информацию ниже.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*логический*	
stream_rewriting_maximise_sql	*логический*	
stream_rewriting_optimise_clem_execution	*логический*	
stream_rewriting_optimise_syntax_execution	*логический*	
enable_parallelism	*логический*	
sql_generation	*логический*	
database_caching	*логический*	
sql_logging	*логический*	
sql_generation_logging	*логический*	
sql_log_native	*логический*	
sql_log_prettyprint	*логический*	

Таблица 41. Свойства потока (продолжение)

Имя свойства	Тип переменной	Описание свойства
record_count_suppress_input	*логический*	
record_count_feedback_interval	*целое*	
use_stream_auto_create_node_установки	*логическое*	При значении true используются параметры, заданные для потока; в противном случае используются предпочтения пользователя.
create_model_applier_for_new_модели	*логическое*	Если значение - true, при создании в построителе моделей новой модели без активных ссылок обновления добавляется новый применитель модели. **Прим.:** Если используется IBM SPSS Modeler Batch версии 15, вы должны явно добавить применитель модели в сценарии.
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Задает тип создаваемой ссылки при автоматическом добавлении узла применителя модели.
create_source_node_from_builders	*логическое*	Если значение - true, при создании в построителе источников нового вывода источника без активных ссылок обновления добавляется новый узел источника.
create_source_node_update_links	createEnabled createDisabled doNotCreate	Задает тип создаваемой ссылки при автоматическом добавлении узла источника.
has_coordinate_system	*логическое*	Если задано true, система координат применяется ко всему потоку.
coordinate_system	*строка*	Имя выбранной проекционной системы координат.
deployment_area	ModelRefresh Скоринг Нет	Выберите способ внедрения потока. Если в качестве этого значения задать None, никакие другие записи внедрения использоваться не будут.

Таблица 41. Свойства потока (продолжение)

Имя свойства	Тип переменной	Описание свойства
scoring_terminal_node_id	*строка*	Выберите в потоке ветвь скоринга. Это может быть любой конечный узел в потоке.
scoring_node_id	*строка*	Выберите слепок в ветви скоринга.
model_build_node_id	*строка*	Выберите узел моделирования в потоке.

Глава 9. Свойства узла источника

Общие свойства узлов источников

Ниже перечислены все свойства, общие для узлов источников, а информация о конкретных узлах представлена в следующих разделах.

Пример 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")
```

Пример 2

В этом сценарии предполагается, что заданный файл данных содержит поле с именем Region, представляющее многострочный текст.

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

# Создаем узел файлов переменных, который будет читать набор данных,
содержащий
# поле "Region"
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)

# Переопределяем тип хранения как список (list)...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region",
StorageType.LIST)
# ...и задаем тип значений в списке и глубину списка
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region",
StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Теперь изменяем измерение, чтобы идентифицировать поле как
геопространственное значение...
varfilenode.setKeyedPropertyValue("measure_type", "Region",
MeasureType.GEOSPATIAL)
# ...и, наконец, задаем необходимую информацию о конкретном
# типе геопространственного объекта
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
node.setPropertyValue("other", ",")
 "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Таблица 42. Общие свойства узлов источников

Имя свойства	Тип переменной	Описание свойства
direction	Ввод Target Both Нет Подмножества Разбиения Frequency RecordID	Ключевое свойство для ролей полей. Формат использования: NODE.direction.FIELDNAME **Прим.:** Значения In и Out в настоящее время объявлены устаревшими. Их поддержка может быть прекращена в следующем выпуске.
type	Range Флаг Set Без типа Дискретное Порядковое Default	Тип поля. При задании этого свойства *По умолчанию* будут очищены все параметры свойства значения, и если для режим_значения будет установлено *Задать*, эта настройка сбросится до *Читать*. Если режим_значения уже задан как *Передать* или *Читать*, на него не будет влиять параметр тип. Формат использования: NODE.type.FIELDNAME
storage	Нет данных Строка Целое Действительное число Время Дата Отметка времени	Предназначенное только для чтения ключевое свойство для типа хранения поля. Формат использования: NODE.storage.FIELDNAME

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
check	Нет Аннулировать Принуждать Исключение Предупреждение Прервать	Ключевое свойство для проверки типа и диапазона поля. Формат использования: NODE.check.FIELDNAME
values	[*значение значение*]	Для количественного поля (диапазона) первое значение - это минимум, а последнее - максимум. Для номинальных полей (набора) задайте все значения. Для флаговых полей первое значение представляет *false*, а последнее - *true*. Задание этого свойства автоматически устанавливает для свойства режим_значения значение *Задать*. Система хранения определяется на основании первого значения в списке, например, если первое значение - это *string*, задается строковая система хранения. Формат использования: NODE.values.FIELDNAME
value_mode	Чтение Успех Read+ Текущий Задать	Определяет, как при следующем проходе данных устанавливаются значения для поля. Формат использования: NODE.value_mode.FIELDNAME Обратите внимание на то, что вы не можете автоматически установить для этого свойства значение *Задать*; чтобы использовать конкретные значения, задайте свойство значения.

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
default_value_mode	Чтение Успех	Задает способ по умолчанию для задания значений для всех полей. Формат использования: NODE.default_value_mode Этот параметр может быть перезаписан для конкретных полей с помощью свойства режим_значения.
extend_values	*флаг*	Применяется, когда для режим_значения задано *Чтение*. Задайте *T*, чтобы добавить вновь прочитанные значения к любым существующим значениям для этого поля. Задайте *F*, чтобы отбросить существующие значения и заменить их на вновь прочитанные значения. Формат использования: NODE.extend_values.FIELDNAME
value_labels	*string*	Используется, чтобы задать метку значения. Обратите внимание на то, что сначала должны быть заданы значения.
enable_missing	*флаг*	Когда задано *T*, активирует отслеживание пропущенных значений для поля. Формат использования: NODE.enable_missing.FIELDNAME
missing_values	[*значение значение ...*]	Задает значения данных, отмечающие пропущенные данные. Формат использования: NODE.missing_values.FIELDNAME
range_missing	*флаг*	Когда для этого свойства задано *T*, указывает, определен ли для этого поля диапазон пропущенных (пустых) значений. Формат использования: NODE.range_missing.FIELDNAME

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
missing_lower	*string*	Когда для значения диапазон_отсутствия задано true, указывает нижнюю границу диапазона значений отсутствия. Формат использования: NODE.missing_lower.FIELDNAME
missing_upper	*string*	Когда для значения диапазон_отсутствия задано true, указывает верхнюю границу диапазона значений отсутствия. Формат использования: NODE.missing_upper.FIELDNAME
null_missing	*флаг*	Когда для этого свойства задано *T*, значения nulls (не определенные значения, обозначаемые в программах как $null$) рассматриваются как значения отсутствия. Формат использования: NODE.null_missing.FIELDNAME
whitespace_missing	*флаг*	Когда для этого свойства задано *T*, значения, содержащие только пробельные символы (пробелы, знаки табуляции и новой строки) рассматриваются как значения отсутствия. Формат использования: NODE.whitespace_missing.FIELDNAME
description	*string*	Использовано, чтобы задать метку или описание поля.
default_include	*флаг*	Ключевое свойство для указания, каким будет поведение по умолчанию, передать или отфильтровать поля: NODE.default_include Пример: set mynode:filternode.default_include = false

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
include	*флаг*	Ключевое свойство, используемое для определения, включаются или отфильтровываются индивидуальные поля: NODE.include.FIELDNAME.
new_name	*string*	
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	Это ключевое свойство похоже на свойство type тем, что может использоваться для определения связанного с полем измерения. Отличие - в сценариях Python; функции setter может также передаваться одно из значений MeasureType, тогда как getter будет всегда возвращать значения MeasureType.

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Для полей собраний (списков с глубиной 0) это ключевое свойство определяет тип измерения, связанный с базовыми значениями.
geo_type	Точки Несколько точек Ломаная Мультиломаная Многоугольник Мультиполигон	Для геопространственных полей это ключевое свойство определяет тип геопространственного объекта, представляемого этим полем. Он должен быть согласован с глубиной списка значений.
has_coordinate_system	*логический*	Для геопространственных полей это свойство определяет наличие у поля системы координат.
coordinate_system	*string*	Для геопространственных полей это ключевое свойство определяет для данного поля систему координат.

Таблица 42. Общие свойства узлов источников (продолжение)

Имя свойства	Тип переменной	Описание свойства
ELSE IF	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Это ключевое свойство похоже на свойство custom_storage тем, что может использоваться для определения для поля хранения переопределения. Отличие - в сценариях Python; функции setter может также передаваться одно из значений StorageType, тогда как getter будет всегда возвращать значения StorageType.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Для полей списков это ключевое свойство задает тип хранения базовых значений.
custom_list_depth	*целое*	Для полей списков это ключевое свойство задает глубину поля.

Таблица 42. Общие свойства узлов источников (продолжение)		
Имя свойства	**Тип переменной**	**Описание свойства**
max_list_length	*целое*	Доступно только для данных со шкалой измерений *Геопространственная* или *Собрание*. Задайте максимальную длину списка, указав количество элементов, которые может содержать этот список.
max_string_length	*целое*	Доступно только для данных *typeless* и используется, когда вы производите SQL для составления таблицы. Введите значение самой большой строки в ваших данных; тогда сгенерируется столбец в таблице, которая будет достаточно большой для содержания строки.

Свойства asimport

При помощи источника Analytic Server поток можно выполнить в файловой системе HDFS (Hadoop Distributed File System).

Пример

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

Таблица 43. Свойства asimport		
Свойства asimport	**Тип переменной**	**Описание свойства**
data_source	*string*	Имя источника данных.
use_default_as	*логический*	Если задано True, используется соединение Analytic Server по умолчанию, сконфигурированное в файле options.cfg сервера. Если задано False, используется соединение данного узла.

Таблица 43. Свойства asimport (продолжение)		
Свойства asimport	**Тип переменной**	**Описание свойства**
connection	["string","string","string", "string","string","string","string", "string" ,"string","string", "string" ,"string"]	Свойство списка, содержащее подробности соединения Analytic Server. Формат: ["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"] где: is_secure_connect: указывает, используется ли защищенное соединение; возможные значения - true или false. use-kerberos-auth: указывает, используется ли аутентификация kerberos; возможные значения - true или false. enable-kerberos-debug: указывает, используется ли отладочный режим аутентификации kerberos; возможные значения - true или false.

Свойства узла cognosimport

Узел источника IBM Cognos импортирует данные из баз данных Cognos Analytics.

Пример

```
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]"])
```

Таблица 44. Свойства узла cognosimport		
Свойства узла cognosimport	**Тип переменной**	**Описание свойства**
mode	Данные Отчет	Задает, импортировать ли данные Cognos (по умолчанию), или отчеты.

Таблица 44. Свойства узла cognosimport (продолжение)

Свойства узла cognosimport	**Тип переменной**	**Описание свойства**
cognos_connection	*["строка",флаг,"строка", "строка","строка"]*	Свойство списка, содержащего подробности соединения для сервера Cognos. Используется следующий формат: ["URL_сервера_Cognos", режим_регистрации, "пространство имен", "имя пользователя", "пароль"] где: URL_сервера_Cognos - это URL сервера Cognos, содержащего источник данных. режим_регистрации обозначает, используется ли анонимная регистрация, и может принимать значение true или false; если задано true, следующие поля должны быть заданы как "". пространство имен задает провайдера аутентификации защиты, используемого для регистрации на сервере. имя_пользователя и пароль - это значения для регистрации на сервере Cognos. Вместо режима режим_регистрации доступны также следующие режимы: • anonymousMode. Например: ['URL_сервера_Cognos', 'anonymousMode', "пространство имен", "имя пользователя", "пароль"] • credentialMode. Например: ['URL_сервера_Cognos', 'credentialMode', "пространство имен", "имя пользователя", "пароль"] • storedCredentialMode. Например: ['URL_сервера_Cognos', 'storedCredentialMode', "имя_хранимых_идентификац_данных"]

Таблица 44. Свойства узла cognosimport (продолжение)

Свойства узла cognosimport	**Тип переменной**	**Описание свойства**
cognos_package_name	*string*	Путь и имя пакета Cognos, из которого импортируются объекты данных, например: /Public Folders/GOSALES **Прим.:** Допустимы символы только прямой дробной черты.
cognos_items	*["поле","поле", ... ,"поле"]*	Имя одного или нескольких объектов данных, которые будут импортированы. Формат *поле* следующий: [*пространство_имен*]. [*субъект_запроса*]. [*элемент_запроса*]
cognos_filters	*поле*	Имя одного или нескольких фильтров для применения перед импортом данных.
cognos_data_parameters	*список*	Значения для предложения параметров данных. Пары имя-и-значение заключены в квадратные скобки, а несколько пар разделяются запятыми, и вся строка заключается в квадратные скобки. Формат: [["*параметр1*", "*значение*"],..., ["*параметрN*", "*значение*"]]
cognos_report_directory	*поле*	Путь Cognos к папке или пакету, из которой будут импортироваться отчеты, например: /Public Folders/GOSALES **Прим.:** Допустимы символы только прямой дробной черты.
cognos_report_name	*поле*	Путь и имя в положении отчетов того отчета, который будет импортироваться.

Таблица 44. Свойства узла cognosimport (продолжение)		
Свойства узла cognosimport	**Тип переменной**	**Описание свойства**
cognos_report_parameters	*список*	Значения для параметров отчета. Пары имя-и-значение заключены в квадратные скобки, а несколько пар разделяются запятыми, и вся строка заключается в квадратные скобки. Формат: [["*параметр1*", "*значение*"],..., ["*параметрN*", "*значение*"]]

Свойства databasenode

Узел базы данных можно использовать для импорта данных из ряда других пакетов при помощи ODBC (Open Database Connectivity), включая Microsoft SQL Server, Db2, Oracle и другие.

Пример

```
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")
```

Таблица 45. Свойства databasenode		
Свойства databasenode	**Тип переменной**	**Описание свойства**
mode	Таблица Query	Укажите *Таблицу* для соединения с таблицей базы данных, используя элементы управления диалогового окна, или задайте *Запрос* для запроса в выбранной базе данных, используя SQL.
datasource	*строка*	Имя базы данных (смотрите также примечание ниже).
username	*строка*	Подробности соединения с базой данных (смотрите также примечания ниже).
password	*строка*	

Таблица 45. Свойства databasenode (продолжение)

Свойства databasenode	Тип переменной	Описание свойства
credential	*строка*	Имя регистрационных данных, хранимых в IBM SPSS Collaboration and Deployment Services. Его можно использовать вместо свойств username и password. Имя пользователя и пароль из регистрационных данных должны соответствовать имени пользователя и паролю для доступа к базе данных.
use_credential		Задайте True или False.
epassword	*строка*	Задает закодированный пароль как альтернативу жестко закодированному в сценарии паролю. Дополнительную информацию смотрите в разделе “Генерирование закодированного пароля” на стр. 56. При выполнении это свойство предназначено только для чтения.
tablename	*строка*	Имя таблицы, к которой вы хотите обратиться.
strip_spaces	Нет Слева Справа Both	Опции для отброса начальных и завершающих пробелов в строках.
use_quotes	AsNeeded Всегда Никогда	Задайте, заключаются ли имена таблиц и столбцов в кавычки при отправлении запросов в базу данных (например, если они содержат пробелы или знаки пунктуации).
query	*строка*	Задает код SQL для запроса, который вы хотите передать.

Прим.: Если имя базы данных (в свойстве datasource) содержит пробелы, вместо отдельных свойств (для datasource, username и password) можно также использовать одно свойство datasource в следующем формате:

Таблица 46. Свойства узла базы данных (databasenode) - для конкретного источника данных		
Свойства databasenode	**Тип переменной**	**Описание свойства**
datasource	*строка*	Формат: [имя_базы_данных,имя_пользователя, пароль[,true | false]] Последний параметр предназначен для использования с зашифрованными паролями. Если для него задано значение true, перед использованием пароль будет расшифрован.

Используйте этот формат также в том случае, если вы изменяете источник данных; однако если нужно изменить только имя пользователя или пароль, можно использовать свойства имя_пользователя или пароль.

Свойства узла импорта собрания данных (datacollectionimportnode)

Узел импорта данных Data Collection импортирует материалы обследования на основании модели данных Data Collection, используемой продуктами изучения рынка. Для использования этого узла должна быть установлена библиотека данных Data Collection.

Пример

```
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")
```

Таблица 47. Свойства datacollectionimportnode		
Свойства datacollectionimportnode	**Тип переменной**	**Описание свойства**
metadata_name	*string*	Имя MDSC. Специальное значение DimensionsMDD указывает, что нужно использовать стандартный документ метаданных Data Collection. Другие возможные значения включают в себя: mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC Специальное значение none означает, что нет MDSC.
metadata_file	*string*	Имя файла, где хранятся метаданные.

Таблица 47. Свойства datacollectionimportnode (продолжение)		
Свойства datacollectionimportnode	**Тип переменной**	**Описание свойства**
casedata_name	*string*	Имя CDSC. Возможные значения включают в себя: mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc Специальное значение none означает, что нет CDSC.
casedata_source_type	Unknown File Папка UDL DSN	Указывает тип источника CDSC.
casedata_file	*string*	Когда casedata_source_type - это *Файл*, задает файл, содержащий данные наблюдения.
casedata_folder	*строка*	Когда casedata_source_type - это *Папка*, задает папку, содержащую данные наблюдения.
casedata_udl_string	*строка*	Когда casedata_source_type - это *UDL*, задает строку соединения OLD-DB для источника данных, содержащего данные наблюдения.

Таблица 47. Свойства datacollectionimportnode (продолжение)

Свойства datacollectionimportnode	**Тип переменной**	**Описание свойства**
casedata_dsn_string	*string*	Когда casedata_source_type - это *DSN*, задает строку соединения ODBC для источника данных.
casedata_project	*string*	При чтении данных наблюдения из базы данных Data Collection можно ввести имя проекта. Для всех остальных типов данных наблюдений значение этого параметра следует оставить пустым.
version_import_mode	All Последняя Задать	Определяет, как должны обрабатываться версии.
specific_version	*string*	Когда version_import_mode - это *Specify*, определяет версию данных наблюдения для импорта.
use_language	*string*	Определяет, должны ли использоваться метки конкретного языка.
язык	*string*	Если значение use_language - это true, определяет используемый при импорте код языка. Код языка должен быть одним из доступных в данных наблюдения.
use_context	*string*	Определяет, нужно ли импортировать конкретный контекст. Контексты используются для изменения описания, связанного с откликами.
context	*string*	Если значение use_context - это true, определяет контекст для импорта. Контекст должен быть одним из доступных в данных наблюдения.
use_label_type	*string*	Определяет, нужно ли импортировать конкретный тип метки.
label_type	*string*	Если значение use_label_type - это true, определяет тип метки для импорта. Тип метки должен быть одним из доступных в данных наблюдения.
user_id	*string*	Для баз данных, требующих непосредственной регистрации, можно предоставить ID пользователя и пароль для доступа к источнику данных.
password	*string*	

Таблица 47. Свойства datacollectionimportnode (продолжение)		
Свойства datacollectionimportnode	**Тип переменной**	**Описание свойства**
import_system_variables	Общий Нет All	Задает, какие системные переменные импортируются.
import_codes_variables	*флаг*	
import_sourcefile_variables	*флаг*	
import_multi_response	MultipleFlags Single	

Свойства узла импорта Excel (excelimportnode)

Узел импорта Excel импортирует данные из Microsoft Excel в формате файлов .xlsx. Источник данных ODBC не требуется.

Примеры

```
#Чтобы использовать названный диапазон:
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)

#Чтобы использовать явный диапазон:
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")
```

Таблица 48. Свойства excelimportnode		
Свойства excelimportnode	**Тип переменной**	**Описание свойства**
excel_file_type	Excel2007	
full_filename	*string*	Полное имя файла, включающее путь.
use_named_range	*Логический*	Использовать ли названный диапазон. При значении true свойство named_range используется для задания диапазона чтения, а другие параметры данных и рабочего листа игнорируются.

Таблица 48. Свойства excelimportnode (продолжение)		
Свойства excelimportnode	**Тип переменной**	**Описание свойства**
named_range	*string*	
worksheet_mode	Индекс Имя	Задает, чем определяется рабочий лист, индексом или именем.
worksheet_index	*целое*	Индекс рабочего листа для чтения, начинается с 0 для первого листа, 1 - для второго и так далее.
worksheet_name	*string*	Имя рабочего листа для чтения.
data_range_mode	FirstNonBlank ExplicitRange	Задает, как нужно определить диапазон.
blank_rows	StopReading ReturnBlankRows	Когда значение data_range_mode - это *FirstNonBlank*, определяет, как должны обрабатываться пустые строки.
explicit_range_start	*string*	Когда значение data_range_mode - это *ExplicitRange*, задает начальную точку диапазона для чтения.
explicit_range_end	*string*	
read_field_names	*Логический*	Задает, должна ли первая строка в заданном диапазоне использоваться для имен полей (столбцов).
scanLineCount	*целое*	Задает число просматриваемых строк для определения типа столбца и хранения. Значение по умолчанию - 200.

свойства extensionimportnode

При помощи узла Extension Import можно запускать сценарии R или Python for Spark для импорта данных.

Пример Python for Spark

```
#### Пример сценария для 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)
```

Пример R

```
#### Пример сценария для R
node.setPropertyValue("syntax_type", "R")

R_script = """# 'JSON Import' Node v1.0 for IBM SPSS Modeler
# Пакет 'RJSONIO' создан Duncan Temple Lang - http://cran.r-project.org/web/packages/RJSONIO
# Пакет 'plyr' создан Hadley Wickham http://cran.r-project.org/web/packages/plyr
# Разработчик узла: Danil Savine - IBM Extreme Blue 2014
# Описание: этот узел позволяет импортировать данные из JSON в таблицу SPSS.
# Установить функцию для пакетов
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)
  }
}
#  пакеты
packages(RJSONIO)
packages(plyr)
### Эта функция используется для автоматического генерирования модели данных 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
      #если x - это фактор, typeof вернет целое значение, и мы рассмотрим этот случай отдельно
      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)
}
# Из JSON в список
txt <- readLines('C:/test.json')
formatedtxt <- paste(txt, collapse = '')
json.list <- fromJSON(formatedtxt)
  # Применить путь к 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
      }
    }
  }
# Из списке в dataframe через json вне списка
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
}
# Экспорт в данные SPSS Modeler
modelerData <- filled
print(modelerData)
modelerDataModel <- getMetaData(modelerData)
print(modelerDataModel)

"""

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

Таблица 49. свойства extensionimportnode

Свойства extensionimportnode	**Тип переменной**	**Описание свойства**
syntax_type	*R* *Python*	Указывает, какой сценарий работает – R или Python (R - значение по умолчанию).
r_syntax	*строка*	Синтаксис R для запускаемого сценария.
python_syntax	*строка*	Синтаксис Python для запускаемого сценария.

Свойства fixedfilenode

Узел фиксированного файла импортирует данные из текстовых файлов с фиксированными полями, то есть, файлов, поля которых не разделяются, но начинаются с одного положения и у них фиксированная длина. В формате с полями фиксированной ширины хранятся данные, генерируемые компьютером или представленные в устаревшем формате.

Пример

```
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)
```

Таблица 50. Свойства fixedfilenode

Свойства fixedfilenode	**Тип переменной**	**Описание свойства**
record_len	*number*	Задает количество символов в каждой записи.
line_oriented	*флаг*	Пропускает символ новой строки в конце каждой записи.
decimal_symbol	Default Comma Period	Тип десятичного разделителя, используемого в вашем источнике данных.
skip_header	*number*	Задает число строк, которые будут игнорироваться в начале первой записи. Полезно, чтобы игнорировать заголовки столбцов.
auto_recognize_datetime	*флаг*	Задает, идентифицируются ли автоматически в исходных данных значения даты или времени.
lines_to_scan	*number*	
fields	*список*	Структурированное свойство.
full_filename	*string*	Полное имя файла для чтения, включая каталог.
strip_spaces	Нет Слева Справа Both	Отбрасывает начальные и завершающие пробелы в строках при импорте.
invalid_char_mode	Исключение Replace	Удаляет недопустимые символы (null, 0 или любой символ, которого нет в текущей кодировке) из входных данных или замещает недопустимые символы на заданный однозначный символ.
invalid_char_replacement	*string*	
use_custom_values	*флаг*	

Таблица 50. Свойства fixedfilenode (продолжение)

Свойства `fixedfilenode`	Тип переменной	Описание свойства

`custom_storage`	`Нет данных`
	
	`Строка`
	
	`Целое`
	
	`Действительное число`
	
	`Время`
	
	`Дата`
	
	`Отметка времени`	

Таблица 50. Свойства fixedfilenode (продолжение)

Свойства fixedfilenode	Тип переменной	Описание свойства
custom_date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДДД" ДЕНЬ МЕСЯЦ "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ"	Это свойство применимо только в том случае, если была задана пользовательская система хранения.

Таблица 50. Свойства fixedfilenode (продолжение)

Свойства fixedfilenode	Тип переменной	Описание свойства
	"ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	
custom_time_format	"ЧЧММСС" "ЧЧММ" "ММСС" "ЧЧ:ММ:СС" "ЧЧ:ММ" "ММ:СС" "(Ч)Ч:(М)М:(С)С" "(Ч)Ч:(М)М" "(М)М:(С)С" "ЧЧ.ММ.СС" "ЧЧ.ММ" "ММ.СС" "(Ч)Ч.(М)М.(С)С" "(Ч)Ч.(М)М" "(М)М.(С)С"	Это свойство применимо только в том случае, если была задана пользовательская система хранения.

Таблица 50. Свойства fixedfilenode (продолжение)		
Свойства fixedfilenode	**Тип переменной**	**Описание свойства**
custom_decimal_symbol	*поле*	Это свойство применимо только в том случае, если была задана пользовательская система хранения.
encoding	StreamDefault SystemDefault "UTF-8"	Задает способ текстового кодирования.

Свойства узла gsdata_import

Узел Геопространственный источник используется для переноса карты или пространственных данных в сеанс исследования данных.

Таблица 51. свойства узла gsdata_import		
Свойства узла gsdata_import	**Тип переменной**	**Описание свойства**
full_filename	*строка*	Введите путь к файлу .shp, который вы хотите загрузить.
map_service_URL	*строка*	Введите URL службы карт, с которым устанавливается соединение.
map_name	*строка*	Только если используется map_service_URL; это свойство содержит верхний уровень структуры папок службы карт.

Свойства jsonimportnode

Исходный узел JSON импортирует данные из файла JSON.

Таблица 52. Свойства jsonimportnode		
Свойства jsonimportnode	**Тип переменной**	**Описание свойства**
full_filename	*строка*	Полное имя файла, включающее путь.
string_format	*records* *значения*	Задайте формат строки JSON. Значение по умолчанию - records.
auto_label		Добавлено в версию 18.2.1.1.

Свойства узла импорта SAS (sasimportnode)

Узел импорта SAS импортирует данные SAS в IBM SPSS Modeler.

Пример

```
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)
```

Таблица 53. Свойства sasimportnode

Свойства sasimportnode	**Тип переменной**	**Описание свойства**
формат	Windows UNIX Transport SAS7 SAS8 SAS9	Формат файла для импорта.
full_filename	*string*	Полное вводимое имя файла, в том числе его путь.
member_name	*string*	Задать участника для импорта из заданного транспортного файла SAS.
read_formats	*флаг*	Читает форматы данных (такие как метки переменных) из заданного файла форматов.
full_format_filename	*string*	
import_names	NamesAndLabels LabelsasNames	Задает способ для отображения имен переменных и меток при импорте.

Свойства simgennode

Узел генерирования имитации обеспечивает удобный путь сгенерировать имитационные данные - либо с нуля, используя указанные пользователем статистические распределения, либо автоматически, используя распределения, полученные при выполнении узла подгонки имитации для существующих данных хронологии. Это полезно, когда нужно оценить вывод прогнозной модели при наличии неопределенности во входных данных модели.

Таблица 54. Свойства simgennode		
Свойства simgennode	**Тип переменной**	**Описание свойства**
поля	Структурированное свойство	Смотрите пример
корреляции	Структурированное свойство	Смотрите пример
keep_min_max_setting	*логическое*	
refit_correlations	*логическое*	
max_cases	*целое*	Минимальное значение - 1000, максимальное значение - 2 147 483 647
create_iteration_field	*логическое*	
iteration_field_name	*строка*	
replicate_results	*логическое*	
random_seed	*целое*	
parameter_xml	*строка*	Возвращает параметр Xml как строку

Пример fields

Это структурированный параметр слота со следующим синтаксисом:

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

distribution - это объявление имени распределения, после которого следует список с парами имен и значений атрибутов. Каждое распределение определяется следующим способом:

```
[имя_распределения, [[пар1], [пар2], [пар3]]]

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

Например, чтобы создать узел, генерирующий одно поле с распределением Биномиальное, можно использовать следующий сценарий:

```
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]]], "", ""]])
```

Биномиальное распределение использует два параметра: n и prob. Так как биномиальное распределение не поддерживает минимальных и максимальных значений, они представлены как пустая строка.

Прим.: Невозможно непосредственно задать distribution; оно используется в связи со свойством fields.

В следующих примерах показаны все возможные типа распределений. Обратите внимание на то, что порог вводится как thresh и в NegativeBinomialFailures, и в 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
])
```

Пример correlations

Это структурированный параметр слота со следующим синтаксисом:

```
simgennode.setPropertyValue("correlations", [
    [поле1, поле2, корреляция],
    [поле1, поле3, корреляция],
    [поле2, поле3, корреляция]
])
```

Корреляция может быть любым числом от +1 до -1. Можно задать любое число корреляций. Незаданной корреляции присваивается значение ноль. Если какие-то из полей неизвестны, значение корреляции должно быть задано в корреляционной матрице (или таблице) и текст показан красным. Если есть неизвестные поля, выполнить узел нельзя.

Свойства узла импорта статистики (statisticsimportnode)

Узел Файл IBM SPSS Statistics читает данные из файла формата *.sav*, используемого IBM SPSS Statistics, а также файлы кэша, сохраненные в IBM SPSS Modeler, которые также используют тот же формат.

Свойства этого узла описаны в разделе “Свойства узла импорта статистики (statisticsimportnode)” на стр. 453.

Свойства узла tm1odataimport

Узел источника IBM Cognos TM1 импортирует данные из баз данных Cognos TM1.

Таблица 55. Свойства узла tm1odataimport		
Свойства узла tm1odataimport	**Тип переменной**	**Описание свойства**
credential_type	*inputCredential* или *storedCredential*	Используется для указания типа идентификационных данных.
input_credential	*список*	Когда credential_type - inputCredential; задайте домен, имя пользователя и пароль.
stored_credential_name	*строка*	Когда credential_type - storedCredential; задайте имя регистрационных данных на сервере C&DS.
selected_view	*["поле" "поле"]*	Свойство списка, содержащее подробности выбранного куба TM1 и имя представления кубов, откуда данные будут импортированы в SPSS. Например: TM1_import.setPropertyValue(" выбранное_ представление", ['план_BudgetPlan', 'Goal Input'])
is_private_view	*флаг*	Указывает, является ли selected_view собственным представлением. Значение по умолчанию - false.
selected_columns	*["поле"]*	Укажите выбранный столбец; можно указать только один элемент. Например: setPropertyValue("selected_columns ", ["Measures"])
selected_rows	*["поле" "поле"]*	Укажите выбранные строки. Например: setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])
connection_type	AdminServer TM1Server	Указывает тип соединения. По умолчанию AdminServer.
admin_host	*string*	URL для имени хоста API REST. Требуется, если для connection_type задано AdminServer.
server name	*string*	Имя сервера TM1, выбранного на хост_admin. Требуется, если для connection_type задано AdminServer.
server_url	*string*	URL для API REST сервера TM1. Требуется, если для connection_type задано TM1Server.

Свойства узла tm1import (объявлено устаревшим)

Узел источника IBM Cognos TM1 импортирует данные из баз данных Cognos TM1.

Прим.: Этот узел устарел в Modeler 18.0. Имя заменяющего сценария узла - *tm1odataimport*.

Таблица 56. Свойства узла tm1import

Свойства узла tm1import	Тип переменной	Описание свойства
pm_host	*строка*	**Прим.:** Только для версий 16.0 и 17.0 Имя хоста. Например: `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*["поле","поле", ... ," поле"]*	**Прим.:** Только для версий 16.0 и 17.0 Свойство списка, содержащего подробности соединения для сервера TM1. Формат: `"Имя_сервера_TM1", "имя_пользователя_tm1","пароль_tm1"]` Например: `TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])`
selected_view	*["поле" "поле"]*	Свойство списка, содержащее подробности выбранного куба TM1 и имя представления кубов, откуда данные будут импортированы в SPSS. Например: `TM1_import.setPropertyValue("выбранное_ представление", ['план_BudgetPlan', 'Goal Input'])`
selected_column	*["поле"]*	Укажите выбранный столбец; можно указать только один элемент. Например: `setPropertyValue("selected_columns", ["Measures"])`
selected_rows	*["поле" "поле"]*	Укажите выбранные строки. Например: `setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])`

Свойства узла twcimport

Исходный узел TWC импортирует метеоданные из компании The Weather Company, дочерней компании IBM. Можно использовать его для получения хронологических или прогностических метеоданных для положения. Это может помочь в разработке бизнес-решений, связанных с погодой для оптимизации принятия решений благодаря использованию доступных самых точных метеоданных.

Таблица 57. свойства узла twcimport		
Свойства узла twcimport	**Тип переменной**	**Описание свойства**
TWCDataImport.latitude	*Действительное число*	Указывает значение широты в формате [-90.0~90.0]
TWCDataImport.longitude	*Действительное число*	Указывает значение долготы в формате [-180.0~180.0].
TWCDataImport.licenseKey	*строка*	Задает лицензионный ключ, полученный от The Weather Company.
TWCDataImport.measurmentUnit	English Метрическая Гибридная	Задает систему единиц измерения. Возможные значения - English, Metric или Hybrid. Metric используется по умолчанию.
TWCDataImport.dataType	Historical Прогноз	Задает тип данных погоды для ввода. Возможные значения - Historical или Forecast. Historical используется по умолчанию.
TWCDataImport.startDate	*Целое*	Если для TWCDataImport.dataType задано Historical, укажите начальную дату в формате ггггММдд.
TWCDataImport.endDate	*Целое*	Если для TWCDataImport.dataType задано Historical, укажите конечную дату в формате ггггММдд.
TWCDataImport.forecastHour	6 12 24 48	Если для TWCDataImport.dataType задано Forecast, укажите срок в часах - 6, 12, 24 или 48.

Свойства userinputnode

Узел пользовательского ввода представляет простой способ создания синтетических данных, или от нуля, или изменением существующих данных. Например, это полезно, если вы хотите создать испытательный набор данных для моделирования.

Пример

```
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")
```

Таблица 58. Свойства userinputnode

Свойства userinputnode	Тип переменной	Описание свойства
data		
names		Структурированный слот, устанавливающий или возвращающий список имен полей, сгенерированных узлом.
custom_storage	Нет данных Строка Целое Действительное число Время Дата Отметка времени	Ключевой слот, задающий или возвращающий систему хранения для поля.
data_mode	Объединенный Ordered	Если задана опция Объединенная, записи генерируются для каждого объединения заданных значений и минимального/максимального значений. Количество сгенерированных записей равно произведению числа значений в каждом поле. Если задана опция Упорядоченная, из каждого столбца берется одно значение для каждой записи по порядку, чтобы сгенерировать строку данных. Количество сгенерированных записей равно максимальному числу значений, связанных с полем. Любые поля с меньшим числом данных будут дополнены значениями null.
values		**Прим.:** Это свойство было объявлено устаревшим с заменой на userinputnode.data и больше не должно использоваться.

Свойства узла файла переменных (variablefilenode)

Узел файла переменных читает данные из текстовых файлов со свободными полями, то есть, такие файлы, записи которых содержат фиксированное количество полей, но переменное число символов. Этот узел полезен также для файлов с текстовыми заголовками фиксированной длины и с некоторыми типами аннотаций.

Пример

```
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])
```

Таблица 59. Свойства variablefilenode

Свойства variablefilenode	**Тип переменной**	**Описание свойства**
skip_header	*number*	Задает число символов, которые будут игнорироваться в начале первой записи.
num_fields_auto	*флаг*	Автоматически определяет количество полей в каждой записи. Записи должны быть ограничены символом новой строки.
num_fields	*number*	Вручную задает количество полей в каждой записи.
delimit_space	*флаг*	Задает символ, который будет использоваться для разделения границ полей в файле.
delimit_tab	*флаг*	
delimit_new_line	*флаг*	
delimit_non_printing	*флаг*	
delimit_comma	*флаг*	В случаях, когда запятая - это и разделитель полей, и десятичный разграничитель для потоков, задайте для delimit_other значение *true* и укажите запятую в качестве разделителя, используя свойство other.
delimit_other	*флаг*	Позволяет задать пользовательский разделитель, используя свойство other.
other	*string*	Задает разделитель, используемый при установленном для delimit_other значении *true*.
decimal_symbol	Default Запятая Period	Задает десятичный разделитель, используемый в источнике данных.

Таблица 59. Свойства variablefilenode (продолжение)

Свойства variablefilenode	**Тип переменной**	**Описание свойства**
multi_blank	*флаг*	Рассматривает несколько смежных пустых символов разделителя как один разделитель.
read_field_names	*флаг*	Рассматривает первую строку в файле данных как метки для столбца.
strip_spaces	Нет Слева Справа Both	Отбрасывает начальные и завершающие пробелы в строках при импорте.
invalid_char_mode	Исключение Replace	Удаляет недопустимые символы (null, 0 или любой символ, которого нет в текущей кодировке) из входных данных или замещает недопустимые символы на заданный однозначный символ.
invalid_char_replacement	*string*	
break_case_by_newline	*флаг*	Задает, что разделитель строк - это символ новой строки.
lines_to_scan	*number*	Задает, сколько строк просматривать для заданных типов данных.
auto_recognize_datetime	*флаг*	Задает, идентифицируются ли автоматически в исходных данных значения даты или времени.
quotes_1	Исключение PairAndDiscard IncludeAsText	Задает, как рассматриваются при импорте одинарные кавычки.
quotes_2	Исключение PairAndDiscard IncludeAsText	Задает, как рассматриваются при импорте двойные кавычки.
full_filename	*строка*	Полное имя файла для чтения, включая каталог.
use_custom_values	*флаг*	

Таблица 59. Свойства variablefilenode (продолжение)

Свойства variablefilenode	Тип переменной	Описание свойства
custom_storage	Нет данных Строка Целое Действительное число Время Дата Отметка времени	

Таблица 59. Свойства variablefilenode (продолжение)

Свойства variablefilenode	**Тип переменной**	**Описание свойства**
custom_date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДДД" ДЕНЬ МЕСЯЦ "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ"	Это свойство применимо только в том случае, если была задана пользовательская система хранения.

Таблица 59. Свойства variablefilenode (продолжение)

Свойства variablefilenode	Тип переменной	Описание свойства
	"ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	
custom_time_format	"ЧЧММСС" "ЧЧММ" "ММСС" "ЧЧ:ММ:СС" "ЧЧ:ММ" "ММ:СС" "(Ч)Ч:(М)М:(С)С" "(Ч)Ч:(М)М" "(М)М:(С)С" "ЧЧ.ММ.СС" "ЧЧ.ММ" "ММ.СС" "(Ч)Ч.(М)М.(С)С" "(Ч)Ч.(М)М" "(М)М.(С)С"	Это свойство применимо только в том случае, если была задана пользовательская система хранения.

Таблица 59. Свойства variablefilenode (продолжение)		
Свойства variablefilenode	**Тип переменной**	**Описание свойства**
custom_decimal_symbol	*поле*	Это свойство применимо только в том случае, если была задана пользовательская система хранения.
encoding	StreamDefault SystemDefault "UTF-8"	Задает способ текстового кодирования.

Свойства узла импорта XML (xmlimportnode)

Узел источника XML импортирует данные в формате XML в поток. Вы можете импортировать в каталог один файл или все файлы. Дополнительно вы можете задать файл схемы, из которой можно прочесть структуру XML.

Пример

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

Таблица 60. Свойства xmlimportnode		
Свойства xmlimportnode	**Тип переменной**	**Описание свойства**
read	single directory	Читает один файл данных (по умолчанию) или все файлы XML в каталоге.
recurse	*флаг*	Задает, читать ли дополнительно файлы XML из всех подкаталогов заданного каталога.
full_filename	*string*	(обязательно) Полный путь и имя файла импорта XML (если read = single).
directory_name	*string*	(обязательно) Полный путь и имя каталога, из которого будут импортироваться файлы XML (если read = directory).
full_schema_filename	*string*	Полный путь и имя файла XSD или DTD, из которого будет читаться структура XML. Если опустить этот параметр, структура будет читаться из файла источника XML.
records	*string*	Выражение XPath (например, /author/name) для определения границы записи. При всякой встрече этого элемента в файле источника будет создаваться новая запись.

Таблица 60. Свойства xmlimportnode (продолжение)

Свойства xmlimportnode	**Тип переменной**	**Описание свойства**
mode	read specify	Читать все записи (по умолчанию) или задать, какие элементы читать.
fields		Список объектов (элементов и атрибутов) для импорта. Каждый элемент в списке - это выражение XPath.

Глава 10. Запись свойств узла операций

Свойства appendnode

Узел присоединения проводит конкатенацию наборов записей. Это полезно для объединения наборов данных с похожими структурами, но различными данными.

Пример

```
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")
```

Таблица 61. Свойства appendnode

Свойства appendnode	Тип переменной	Описание свойства
match_by	Положению Имя	Вы можете присоединить наборы данных на основе положения полей в главном источнике полей или имени полей во входных наборах полей.
match_case	*флаг*	Включает учет регистра при сравнении имен полей.
include_fields_from	Главные All	
create_tag_field	*флаг*	
tag_field_name	*string*	

Свойства aggregatenode

Узел Агрегат замещает последовательность входных записей на итоговые, агрегированные выходные записи.

Пример

```
node = stream.create("aggregate", "My node")
# dbnode - сконфигурированный узел импорта базы данных
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")
```

Таблица 62. Свойства aggregatenode

Свойства aggregatenode	Тип переменной	Описание свойства
ключи	*список*	Перечисляет поля, которые можно использовать как ключи для агрегации. Например, если Пол и Область будут вашими полями ключа, то у каждой уникальной комбинации М и Ж с областями С и Ю (четыре уникальных комбинации) будет агрегированная запись.
contiguous	*флаг*	Выберите эту опцию, если вам известно, что все записи с одинаковыми значениями ключа группируются совместно во входных данных (например, если входные данные отсортированы по полям ключей). Выбрав эту опцию, можно увеличить производительность.
aggregates		Структурированное свойство, перечисляющее числовые поля, значения которых будут агрегированы, а также выбранные режимы агрегации.
aggregate_exprs		Ключевое свойство, включающее для имени полученного поля выражение агрегации, используемое для его вычисления. Например: `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*строка*	Задайте префикс или суффикс для дубликатов агрегированных полей (пример ниже).
add_as	Суффикс Префикс	
inc_record_count	*флаг*	Создает дополнительное поле, в котором указывается, сколько входных записей было агрегировано для образования каждой записи агрегата.
count_field	*строка*	Задает имя поля количества записей.
allow_approximation	*Логический*	Разрешает аппроксимацию порядковых статистик при выполнении агрегирования в Analytic Server
bin_count	*целое*	Задает число интервалов для использования в аппроксимации

Свойства balancenode

Узел Баланс исправляет дисбаланс в наборе данных, чтобы он соответствовал заданному условию. Директива балансировки корректирует часть записей, где выполнено условие по заданному фактору.

Пример

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

Таблица 63. Свойства balancenode

Свойства balancenode	Тип переменной	Описание свойства
directives		Структурированное свойство для балансировки доли значений поля на основании заданного числа (см. пример ниже).
training_data_only	*флаг*	Задает, что балансировка должна выполняться только для данных обучения. Если поле раздела отсутствует в потоке, эта опция игнорируется.

Это свойство узла использует следующий формат:

[[*число, строка*] \ [*число, строка*] \ ... [*число, строка*]].

Прим.: Если строки заключены в выражении (с использованием знаков двойных кавычек), перед ними должен быть символ обратной дробной черты " \ ". Символ " \ " является также символом продолжения строки, с помощью которого можно выравнивать для ясности аргументы.

Свойства cplexoptnode

Узел Оптимизация CPLEX обеспечивает возможность использования сложной математической оптимизации (CPLEX) при помощи файла модели Optimization Programming Language (OPL). Эта функциональность была доступна в продукте IBM Analytical Decision Management, который больше не поддерживается. Но можно также использовать узел CPLEX в SPSS Modeler без необходимости IBM Analytical Decision Management.

Таблица 64. Свойства cplexoptnode

Свойства cplexoptnode	Тип переменной	Описание свойства
opl_model_text	*строка*	Программа сценария OPL (Optimization Programming Language - язык программирования оптимизации), которую узел оптимизации выполнит, а затем сгенерирует результат оптимизации.

Таблица 64. Свойства cplexoptnode (продолжение)

Свойства cplexoptnode	**Тип переменной**	**Описание свойства**
opl_tuple_set_name	*строка*	Имя набора кортежа в модели OPL, соответствующей поступающим данным. Это не обязательно, обычно это свойство не задается с помощью сценария. Это требуется использовать только для редактирования отображений полей выбранного источника данных.
data_input_map	*Список структурированных свойств*	Отображения поля ввода для источника данных. Это не обязательно, обычно это свойство не задается с помощью сценария. Это требуется использовать только для редактирования отображений полей выбранного источника данных.

Таблица 64. Свойства cplexoptnode (продолжение)

Свойства cplexoptnode	Тип переменной	Описание свойства
md_data_input_map	*Список структурированных свойств*	Отображения полей между каждым кортежем, определенным в OPL, и каждым соответствующим полем источника данных (входящие данные). Пользователи могут редактировать их по отдельности для каждого источника данных. В этом сценарии можно задать свойство непосредственно, чтобы определить все отображения сразу. Этот параметр не выводится в пользовательском интерфейсе. Каждый объект в списке - структурированные данные: **Тег источника данных.** Тег источника данных, который может быть найден в выпадающем списке источников данных. Например, для 0_Products_Type этот тег - 0. **Индекс источника данных.** Физическая последовательность (индекс) источника данных. Это определяется порядком соединения. **Исходный узел.** Исходный узел (аннотация) источника данных. Это может быть найдено в выпадающем списке источников данных. Например, для 0_Products_Type исходный узел - Products. **Соединенный узел.** Предшествующий узел (аннотация), соединяющая текущий узел оптимизации CPLEX. Это может быть найдено в выпадающем списке источников данных. Например, для 0_Products_Type соединенный узел - Type. **Имя набора кортежей.** Имя набора кортежей источника данных. Оно должно соответствовать определенному в OPL. **Имя поля кортежа.** Имя поля набора кортежей источника данных. Оно должно соответствовать заданному в определении набора кортежей OPL. **Тип хранения.** Тип хранения поля. Возможные значения - int, float или string.

Таблица 64. Свойства cplexoptnode (продолжение)

Свойства cplexoptnode	Тип переменной	Описание свойства
		Имя поля данных. Имя поля источника данных. **Пример:** `[[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	*строка*	Определение некоторых переменных или данных, используемых в OPL.
output_value_mode	*строка*	Возможные значения - raw или dvar. Если задано dvar, пользователь на вкладке Вывод должен указать имя переменной функции объектов в OPL для вывода. Если задано raw, целевая функция будет выводиться непосредственно независимо от имени.
decision_variable_name	*строка*	Имя переменной целевой функции в определении OPL. Это поле активно, только когда для свойства output_value_mode задано значение DVAR.
objective_function_value_fieldname	*строка*	Имя поля для значения целевой функции для использования в выводе. Значение по умолчанию - _OBJECTIVE.
output_tuple_set_names	*строка*	Имя предопределенных кортежей из входящих данных. Они действуют как индексы переменной решения и ожидаются в качестве вывода с Выводом переменных. Кортеж вывода должен быть совместим с определением переменной решения в OPL. Если индексов несколько, именам кортежей надо разделять запятыми (,). Пример отдельного кортежа - Products, соответствующее определение OPL будет `dvar float+ Production[Products];` Пример нескольких кортежей - Products,Components, соответствующее определение OPL будет `dvar float+ Production[Products] [Components];`

Таблица 64. Свойства cplexoptnode (продолжение)		
Свойства cplexoptnode	**Тип переменной**	**Описание свойства**
decision_output_map	*Список структурированных свойств*	Отображение полей между переменными, определенными в OPL, которые будут выводом, и полями вывода. Каждый объект в списке - структурированные данные: **Имя переменной.** Имя переменной в OPL для вывода. **Тип хранения.** Возможные значения - `int`, `float` или `string`. **Имя поля вывода.** Ожидаемое имя поля в результатах (вывода или экспорта). **Пример:** `[['Production','int','res'], ['Remark','string','res_1']['Cost', 'float','res_2']]`

Свойства derive_stbnode

Узел Пространственно-временные диапазоны выводит пространственно-временные диапазоны из полей широты, долготы и отметок времени. Вы также можете указать часто встречающиеся пространственно-временные диапазоны как аттракторы.

Пример

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Режим отдельных записей
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_")

# Режим аттракторов
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)
```

Таблица 65. Свойства узла Space-Time-Boxes		
Свойства derive_stbnode	**Тип переменной**	**Описание свойства**
mode	`IndividualRecords` `Hangouts`	
latitude_field	*поле*	

Таблица 65. Свойства узла Space-Time-Boxes (продолжение)

Свойства derive_stbnode	**Тип переменной**	**Описание свойства**
longitude_field	*поле*	
timestamp_field	*поле*	
hangout_density	*density*	Отдельная плотность. Допустимые значения плотности смотрите в описании плотности.
densities	[*density*,*density*,..., *density*]	Каждая плотность представляет собой строку, например, STB_GH8_1DAY. **Прим.:** Существуют пределы допустимой плотности. Для геохеша можно использовать значения от GH1 до GH15. Для временной части можно использовать следующие значения: ВСЕГДА 1ГОД 1МЕСЯЦ 1ДЕНЬ 12ЧАСОВ 8ЧАСОВ 6ЧАСОВ 4ЧАСА 3ЧАСА 2ЧАСА 1ЧАС 30МИН 15МИН 10МИН 5МИН 2МИН 1МИН 30СЕК 15СЕК 10СЕК 5СЕК 2СЕК 1СЕК
id_field	*поле*	
qualifying_duration	1ДЕНЬ 12ЧАСОВ 8ЧАСОВ 6ЧАСОВ 4ЧАСА 3ЧАСА 2ЧАСА 1ЧАС 30МИН 15МИН 10МИН 5МИН 2МИН 1МИН 30СЕК 15СЕК 10СЕК 5СЕК 2СЕК 1СЕК	Должно быть строковым.
min_events	*целое*	Минимальное целое значение - 2.
qualifying_pct	*целое*	Должно быть в диапазоне от 1 до 100.

Таблица 65. Свойства узла Space-Time-Boxes (продолжение)		
Свойства derive_stbnode	**Тип переменной**	**Описание свойства**
add_extension_as	Префикс Суффикс	
name_extension	*строка*	

Свойства distinctnode

Отдельный узел удаляет дублированные записи, или передавая первую отдельную запись в поток данных, или отбрасывая первую запись и вместо этого передавая в поток данных любые дубликаты.

Пример

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

Таблица 66. Свойства distinctnode		
Свойства distinctnode	**Тип переменной**	**Описание свойства**
mode	Включать Исключение	Вы можете включить первую отдельную запись в потоке данных или отбросить первую отдельную запись и передать вместо этого в поток любые дублированные записи.
grouping_fields	*список*	Перечисляет поля, используемые для определения, идентичны ли записи. **Прим.:** Это свойство устарело, начиная с IBM SPSS Modeler 16.
composite_value	Структурированный слот	Смотрите пример ниже.
composite_values	Структурированный слот	Смотрите пример ниже.
inc_record_count	*флаг*	Создает дополнительное поле, в котором указывается, сколько входных записей было агрегировано для образования каждой записи агрегата.
count_field	*строка*	Задает имя поля количества записей.
sort_keys	Структурированный слот.	**Прим.:** Это свойство устарело, начиная с IBM SPSS Modeler 16.
default_ascending	*флаг*	
low_distinct_key_count	*флаг*	Задает, что у вас только небольшое число записей и/или небольшое количество уникальных значений ключевых полей.

Таблица 66. Свойства distinctnode (продолжение)		
Свойства distinctnode	**Тип переменной**	**Описание свойства**
keys_pre_sorted	*флаг*	Задает, что все записи с одинаковыми значениями ключа совместно группируются во входных данных.
disable_sql_generation	*флаг*	

Пример для свойства composite_value

У свойства composite_value следующая общая форма:

```
node.setKeyedPropertyValue("composite_value", ПОЛЕ, ОПЦИЯ_ЗАПОЛНЕНИЯ)
```

ОПЦИЯ_ЗАПОЛНЕНИЯ имеет вид: [Тип_заполнения, Опция1, Опция2, ...].

Примеры:

```
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"])
```

Пользовательским опциям требуется несколько аргументов; они добавляются списком, например:

```
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"])
```

Пример для свойства composite_values

У свойства composite_values следующая общая форма:

```
node.setPropertyValue("composite_values", [
                        [ПОЛЕ1, [ОПЦИЯ_ЗАПОЛНЕНИЯ1]],
                        [ПОЛЕ2, [ОПЦИЯ_ЗАПОЛНЕНИЯ2]],
.
.
])
```

Пример:

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

Свойства extensionprocessnode

При помощи узла преобразования расширений можно брать данные из потока и применять к ним преобразования, используя сценарии R или Python for Spark.

Пример Python for Spark

```
#### пример сценария для 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)
```

R example

```
#### 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)""")
```

Таблица 67. свойства extensionprocessnode

Свойства extensionprocessnode	**Тип переменной**	**Описание свойства**
syntax_type	*R* *Python*	Указывает, какой сценарий работает – R или Python (R - значение по умолчанию).
r_syntax	*строка*	Синтаксис R для запускаемого сценария.
python_syntax	*строка*	Синтаксис Python для запускаемого сценария.
use_batch_size	*флаг*	Включить использование пакетной обработки.

Таблица 67. свойства extensionprocessnode (продолжение)		
Свойства extensionprocessnode	**Тип переменной**	**Описание свойства**
batch_size	*целое*	Укажите количество записей данных, которые будут включаться в каждый пакет.
convert_flags	`StringsAndDoubles` `LogicalValues`	Опция для преобразования флаговых полей.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	`POSIXct` `POSIXlt`	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.

Свойства mergenode

Узел слияния берет несколько входных записей и создает одну выходную запись, содержащую некоторые или все из входных полей. Он полезен для слияния данных из разных источников, например, из внутренних данных о клиентах и приобретенных демографических данных.

Пример

```
node = stream.create("merge", "My node")
# предполагается, что customerdata и salesdata - это сконфигурированные узлы
импорта баз данных
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"]])
```

Таблица 68. Свойства mergenode

Свойства mergenode	**Тип переменной**	**Описание свойства**
method	Порядок Ключи Условие Rankedcondition	Укажите, выполнять ли слияние записей в порядке их перечисления в файлах данных, если одно или несколько полей ключей будут использоваться для слияния записей с одинаковым в полях ключей значением, если слияние записей будет выполняться при выполнении заданного условия или если у каждой строки есть пара в подлежащих слиянию первичном и всех вторичных наборах данных (при помощи выражения ранжирования для сортировки всех множественных соответствий в порядке от наименьшего к наибольшему).
condition	*string*	Если значение method - это Condition, задает условие для включения или отбрасывания записей.
key_fields	*список*	
common_keys	*флаг*	
join	Внутреннее FullOuter PartialOuter Anti	
outer_join_tag.n	*флаг*	В этом свойстве *n* - это имя тега, как оно выведено на экран в диалоговом окне Выбрать набор данных. Обратите внимание на то, что можно задать несколько имен тегов, так как вклад может внести любое число наборов данных.
single_large_input	*флаг*	Задает, будет ли использоваться оптимизация, чтобы одно входное множество было относительно большим в сравнении с другими вводами.
single_large_input_tag	*string*	Задает имя тега, как оно показывается в диалоговом окне Выбрать большой набор данных. Обратите внимание на то, что использование этого свойства немного отличается от свойства outer_join_tag (флаг по сравнению со строкой), так как можно задать только один входной набор данных.
use_existing_sort_keys	*флаг*	Определяет, отсортированы ли уже входные поля по одному или нескольким полям ключей.

Таблица 68. Свойства mergenode (продолжение)		
Свойства mergenode	**Тип переменной**	**Описание свойства**
existing_sort_keys	[['*string*', 'Ascending'] \ ['*string*', 'Descending']]	Задает поля, которые уже отсортированы, и направление, в котором они отсортированы.
primary_dataset	*string*	Если method - Rankedcondition, выберите первичный набор данных в выражении слияния. Он может рассматриваться как левая часть выражения слияния внешнего объединения.
rename_duplicate_fields	*Логический*	Если method - Rankedcondition и для него задано значение Y, если итоговый набор данных слияния содержит несколько полей с одним и тем же именем из различных источников данных, в начало заголовков столбцов полей добавляются соответствующие теги из этих источников данных.
merge_condition	*string*	
ranking_expression	*string*	
Num_matches	*целое*	Число подлежащих возврату соответствий на основе merge_condition и ranking_expression. Минимум 1, максимум 100.

Свойства rfmaggregatenode

Узел агрегата Новизна, частота, деньги (Recency, Frequency, Monetary - RFM) позволяет рассмотреть хронологические данные транзакций клиента, исключить любые неиспользуемые данные и объединить все оставшиеся данные транзакций в одну строку, где будет представлено, когда в последний раз обращался клиент, сколько транзакций он произвел и какова общая денежная сумма этих транзакций.

Пример

```
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")
```

Таблица 69. Свойства rfmaggregatenode

Свойства rfmaggregatenode	**Тип переменной**	**Описание свойства**
relative_to	Фиксированная Сегодня	Задать дату, по сравнению с которой будет вычисляться новизна транзакций.
reference_date	*дата*	Доступно только в том случае, если для relative_to выбрано Fixed.
contiguous	*флаг*	Если ваши данные предварительно отсортированы, так что все записи с одинаковым ID сгруппированы совместно в потоке данных, выбор этой опции ускоряет обработку.
id_field	*поле*	Задать поле, используемое для идентификации клиентов и их транзакций.
date_field	*поле*	Задать поле даты, которое будет использоваться для вычисления новизны.
value_field	*поле*	Задать поле, которое будет использоваться для вычисления денежного значения.
extension	*string*	Задать префикс или суффикс для дублированных агрегированных полей.
add_as	Суффикс Prefix	Указать использование для расширение - как суффикс или как префикс.
discard_low_value_records	*флаг*	Включить использование параметра discard_records_below.
discard_records_below	*number*	Задать минимальное значение, ниже которого любые подробности транзакций не используются при вычислении суммарных значений RFM. Единицы измерения относятся к выбранному полю значение.
only_recent_transactions	*флаг*	Включить использование одного из параметров, specify_transaction_date или transaction_within_last.
specify_transaction_date	*флаг*	
transaction_date_after	*дата*	Доступно только в том случае, если выбрано specify_transaction_date. Определить дату транзакции, после которой записи будут включены в ваш анализ.

Таблица 69. Свойства rfmaggregatenode (продолжение)		
Свойства rfmaggregatenode	**Тип переменной**	**Описание свойства**
transaction_within_last	*number*	Доступно только в том случае, если выбрано transaction_within_last. Задать количество и тип периодов времени (дни, недели, месяцы и годы) для отсчета назад от даты для вычисления недавних значений и включения этих записей в ваш анализ.
transaction_scale	Days Weeks Months Years	Доступно только в том случае, если выбрано transaction_within_last. Задать количество и тип периодов времени (дни, недели, месяцы и годы) для отсчета назад от даты для вычисления недавних значений и включения этих записей в ваш анализ.
save_r2	*флаг*	Задает дату предпоследней из недавних транзакций для каждого клиента.
save_r3	*флаг*	Доступно только в том случае, если выбрано save_r2. Задает дату третьей по счету из самых недавних транзакций для каждого клиента.

Свойства Rprocessnode

Узел преобразования R дает возможность брать данные из потока IBM(r) SPSS(r) Modeler и модифицировать их при помощи пользовательского сценария R. После модификации данные возвращаются в поток.

Пример

```
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")
```

Таблица 70. Свойства Rprocessnode		
Свойства Rprocessnode	**Тип переменной**	**Описание свойства**
синтаксис	*строка*	
convert_flags	StringsAndDoubles LogicalValues	

Таблица 70. Свойства Rprocessnode (продолжение)		
Свойства Rprocessnode	**Тип переменной**	**Описание свойства**
convert_datetime	*флаг*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*флаг*	
use_batch_size	*флаг*	Включить использование пакетной обработки
batch_size	*целое*	Укажите количество записей данных, которые будут включаться в каждый пакет

Свойства samplenode

Узел Выборка отбирает подмножество записей. Поддерживается несколько типов выборки, в том числе стратифицированные, кластеризованные и неслучайные (структурированные) выборки. Выборки могут быть полезны для повышения производительности и для выбора групп связанных записей или транзакций для анализа.

Пример

```
/* Создать два узла выборки для извлечения
   различных выборок из одинаковых данных */

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]])
```

Таблица 71. Свойства samplenode		
Свойства samplenode	**Тип переменной**	**Описание свойства**
method	Простой Сложный	
mode	Включать Исключение	Включить или отбросить записи, для которых выполнено заданное условие.

Таблица 71. Свойства samplenode (продолжение)

Свойства samplenode	**Тип переменной**	**Описание свойства**
sample_type	Первое OneInN RandomPct	Задает способ выборки.
first_n	*целое*	Записи вплоть до заданной точки отсечения будут включены или отброшены.
one_in_n	*number*	Включить или отбросить каждую *n*-ную запись.
rand_pct	*number*	Задать процентную долю записей для включения или отбрасывания.
use_max_size	*флаг*	Включить использование параметра maximum_size.
maximum_size	*целое*	Задать самую большую выборку, которая будет включена в поток данных или отброшена из него. Эта опция избыточна и поэтому будет отключена, когда заданы First и Include.
set_random_seed	*флаг*	Включение использования параметра начального значения генератора псевдослучайных чисел.
random_seed	*целое*	Задать значение, используемое как начальное значение генератора псевдослучайных чисел.
complex_sample_type	Переменный Систематическая	
sample_units	Доли Количества	
sample_size_proportions	Фиксированная Пользовательские Переменная	
sample_size_counts	Фиксированная Пользовательские Переменная	
fixed_proportions	*number*	
fixed_counts	*целое*	

Таблица 71. Свойства samplenode (продолжение)		
Свойства samplenode	**Тип переменной**	**Описание свойства**
variable_proportions	*поле*	
variable_counts	*поле*	
use_min_stratum_size	*флаг*	
minimum_stratum_size	*целое*	Эта опция применяется только в том случае, когда берется сложная выборка с опцией Sample units=Proportions.
use_max_stratum_size	*флаг*	
maximum_stratum_size	*целое*	Эта опция применяется только в том случае, когда берется сложная выборка с опцией Sample units=Proportions.
кластеры	*поле*	
stratify_by	*[поле1 ... полеN]*	
specify_input_weight	*флаг*	
input_weight	*поле*	
new_output_weight	*string*	
sizes_proportions	[[string *строковое_значение*] [string *строковое_значение*]...]	Если задано sample_units=proportions и sample_size_proportions=Custom, указывает значение для каждой возможной комбинации значений полей стратификации.
default_proportion	*number*	
sizes_counts	[[string *строковое_значение*] [string *строковое_значение*]...]	Указывает значение для каждой возможной комбинации значений полей стратификации. Использование аналогично sizes_proportions, но указывается целое число, а не доля.
default_count	*number*	

Свойства selectnode

Узел Выбор отбирает или отбрасывает подмножество записей из потока данных на основе конкретного условия. Например, вы можете выбрать записи, принадлежащие определенному району продаж.

Пример

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

Таблица 72. Свойства selectnode		
Свойства selectnode	**Тип переменной**	**Описание свойства**
mode	Включать Исключение	Задает, включать или отбрасывать выбранные записи.
condition	*string*	Условие для включения или отбрасывания записей.

Свойства sortnode

Узел Сортировка сортирует записи в восходящем или убывающем порядке на основании значений одного или нескольких полей.

Пример

```
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"]])
```

Таблица 73. Свойства sortnode		
Свойства sortnode	**Тип переменной**	**Описание свойства**
ключи	*список*	Задает поля, которые вам нужно отсортировать. Если направление сортировки не задано, используются значения по умолчанию.
default_ascending	*флаг*	Задает порядок сортировки по умолчанию.
use_existing_keys	*флаг*	Задает, оптимизирована ли сортировка посредством использования предыдущего порядка сортировки для полей, которые уже сортировались.
existing_keys		Задает поля, которые уже сортировались, и направление, в котором они сортировались. Использует тот же формат, что и свойство ключи.

Свойства spacetimeboxes

Пространственно-временные диапазоны (Space-Time-Boxes, STB) - это расширение пространственных положений с географической привязкой. Более конкретно, STB - это алфавитно-цифровая строка, представляющая область в пространстве и времени правильной формы.

Таблица 74. Свойства spacetimeboxes

Свойства spacetimeboxes	**Тип переменной**	**Описание свойства**
mode	*IndividualRecords* *Hangouts*	
latitude_field	*поле*	
longitude_field	*поле*	
timestamp_field	*поле*	
densities	*[плотность, плотность, плотность...]*	Каждая плотность - строка. Например: STB_GH8_1DAY Обратите внимание, что существуют пределы допустимой плотности. Для geohash могут использоваться значения от GH1-GH15. Для временной части можно использовать следующие значения: КАЖДЫЙ 1ГОД 1МЕСЯЦ 1ДЕНЬ 12ЧАСОВ 8ЧАСОВ 6ЧАСОВ 4ЧАСА 3ЧАСА 2ЧАСА 1ЧАС 30МИН 15МИН 10МИН 5МИН 2 MINS (2 минуты) 1 MIN (1 минута) 30СЕК 15СЕК 10СЕК 5 SECS (5 секунд) 2 SECS (2 секунды) 1СЕК
field_name_extension	*строка*	
add_extension_as	*Префикс* *Суффикс*	
hangout_density	*density*	Отдельная плотность (см. выше)
id_field	*поле*	

Таблица 74. Свойства spacetimeboxes (продолжение)		
Свойства spacetimeboxes	**Тип переменной**	**Описание свойства**
qualifying_duration	1ДЕНЬ 12ЧАСОВ 8ЧАСОВ 6ЧАСОВ 4ЧАСА 2ЧАСА 1ЧАС 30МИН 15МИН 10МИН 5МИН 2МИН 1МИН 30СЕК 15СЕК 10СЕК 5СЕК 2СЕК 1СЕК	Должно быть строкой.
min_events	*целое*	Минимальное значение: 2
qualifying_pct	*целое*	Должно быть в диапазоне 1-100

Свойства streamingtimeseries

Узел потоковых временных рядов выполняет построение и скоринг моделей временных рядов за один шаг.

Прим.: Этот узел потоковых временных рядов подобен предыдущему узлу потоковых временных рядов, объявленному устаревшим в SPSS Modeler версии 18.

Таблица 75. Свойства streamingtimeseries		
Свойства streamingtimeseries	**Значения**	**Описание свойства**
targets	*поле*	Узел потоковых временных рядов прогнозирует значения одного или нескольких полей назначения, используя в качестве предикторов одно или несколько входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
candidate_inputs	[*поле1 ... полеN*]	Входные (или предикторные) поля, используемые в модели.

Таблица 75. Свойства streamingtimeseries (продолжение)		
Свойства streamingtimeseries	**Значения**	**Описание свойства**
use_period	*флаг*	
date_time_field	*поле*	
input_interval	Нет Нет данных Год Квартал Месяц Неделя День Час Hour_nonperiod Минута Minute_nonperiod Секунда Second_nonperiod	
period_field	*поле*	
period_start_value	*целое*	
num_days_per_week	*целое*	
start_day_of_week	Воскресенье Понедельник Вторник Среда Четверг Пятница Суббота	
num_hours_per_day	*целое*	

Таблица 75. Свойства streamingtimeseries (продолжение)		
Свойства streamingtimeseries	**Значения**	**Описание свойства**
start_hour_of_day	*целое*	
timestamp_increments	*целое*	
cyclic_increments	*целое*	
cyclic_periods	*список*	
output_interval	Нет Год Квартал Месяц Неделя День Час Минута Секунда	
is_same_interval	*флаг*	
cross_hour	*флаг*	
aggregate_and_distribute	*список*	
aggregate_default	Mean Сумма Мода Минимум Максимум	
distribute_default	Mean Сумма	

Таблица 75. Свойства streamingtimeseries (продолжение)		
Свойства streamingtimeseries	**Значения**	**Описание свойства**
group_default	Mean Сумма Мода Минимум Максимум	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*целое*	
use_estimation_period	*флаг*	
estimation_period	Observations Times	
date_estimation	*список*	Доступно, только если используется date_time_field
period_estimation	*список*	Доступно, только если используется use_period
observations_type	Последняя Самые ранние	
observations_num	*целое*	
observations_exclude	*целое*	
method	ExpertModeler Exsmooth Arima	

Таблица 75. Свойства streamingtimeseries (продолжение)

Свойства streamingtimeseries	**Значения**	**Описание свойства**
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*флаг*	
detect_outliers	*флаг*	
expert_outlier_additive	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_innovational	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_transient	*флаг*	
expert_outlier_seasonal_additive	*флаг*	
expert_outlier_local_trend	*флаг*	
expert_outlier_additive_patch	*флаг*	
consider_newesmodels	*флаг*	

Таблица 75. Свойства streamingtimeseries (продолжение)

Свойства streamingtimeseries	**Значения**	**Описание свойства**
exsmooth_model_type	Простые HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrend-Multiplicative MultiplicativeTrend	
futureValue_type_method	Compute specify	
exsmooth_transformation_type	Нет SquareRoot NaturalLog	
arima.p	*целое*	
arima.d	*целое*	
arima.q	*целое*	
arima.sp	*целое*	
arima.sd	*целое*	
arima.sq	*целое*	

Таблица 75. Свойства streamingtimeseries (продолжение)		
Свойства streamingtimeseries	**Значения**	**Описание свойства**
arima_transformation_type	Нет SquareRoot NaturalLog	
arima_include_constant	*флаг*	
tf_arima.p. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.d. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.q. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sp. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sd. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sq. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.delay. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.transformation_type. *имя_поля*	Нет SquareRoot NaturalLog	Для передаточных функций.
arima_detect_outliers	*флаг*	
arima_outlier_additive	*флаг*	
arima_outlier_level_shift	*флаг*	
arima_outlier_innovational	*флаг*	
arima_outlier_transient	*флаг*	
arima_outlier_seasonal_additive	*флаг*	
arima_outlier_local_trend	*флаг*	
arima_outlier_additive_patch	*флаг*	
conf_limit_pct	*real*	
события	*fields*	
forecastperiods	*целое*	
extend_records_into_future	*флаг*	
conf_limits	*флаг*	
noise_res	*флаг*	

Свойства streamingts (объявлено устаревшим)

Прим.: Этот исходный узел Потоковые временные ряды в версии 18 SPSS Modeler был объявлен устаревшим и заменен новым узлом Потоковые временные ряды, который спроектирован с учетом мощности IBM SPSS Analytic Server и обработки больших объемов данных.

Узел Потоковые временные ряды строит и оценивает модели временных рядов за один шаг, без необходимости использования узла Временные интервалы.

Пример

```
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")
```

Таблица 76. Свойства streamingts

Свойства streamingts	**Тип переменной**	**Описание свойства**
custom_fields	*флаг*	Если custom_fields=false, используются параметры с вышележащего узла типа. Если custom_fields=true, надо задать поля назначения и входные поля.
targets	[*поле_1...поле_N*]	
inputs	[*поле_1...поле_N*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*флаг*	
conf_limit_pct	*действительное число*	
use_time_intervals_node	*флаг*	Если use_time_intervals_node=true, используются параметры с вышележащего узла временных интервалов. Если use_time_intervals_node=false, должны быть заданы interval_offset_position, interval_offset, и interval_type.
interval_offset_position	LastObservation LastRecord	LastObservation - это **Последнее допустимое наблюдение**. LastRecord - это **Обратный отсчет с последней записи**.
interval_offset	*число*	

Таблица 76. Свойства streamingts (продолжение)

Свойства streamingts	**Тип переменной**	**Описание свойства**
interval_type	Периоды Годы Кварталы Месяцы WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*поля*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*флаг*	
detect_outliers	*флаг*	
expert_outlier_additive	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_innovational	*флаг*	
expert_outlier_transient	*флаг*	
expert_outlier_seasonal_additive	*флаг*	
expert_outlier_local_trend	*флаг*	
expert_outlier_additive_patch	*флаг*	
exsmooth_model_type	Простой HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	Нет SquareRoot NaturalLog	
arima_p	*целое*	То же самое свойство, что для узла моделирования временных рядов
arima_d	*целое*	То же самое свойство, что для узла моделирования временных рядов
arima_q	*целое*	То же самое свойство, что для узла моделирования временных рядов

Таблица 76. Свойства streamingts (продолжение)

Свойства streamingts	**Тип переменной**	**Описание свойства**
arima_sp	*целое*	То же самое свойство, что для узла моделирования временных рядов
arima_sd	*целое*	То же самое свойство, что для узла моделирования временных рядов
arima_sq	*целое*	То же самое свойство, что для узла моделирования временных рядов

arima_transformation_type	`Нет`
	`SquareRoot`
	`NaturalLog`
		То же самое свойство, что для узла моделирования временных рядов

arima_include_constant	*флаг*	То же самое свойство, что для узла моделирования временных рядов
tf_arima_p.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_d.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_q.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_sp.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_sd.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_sq.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.
tf_arima_delay.*fieldname*	*целое*	То же самое свойство, что для узла моделирования временных рядов. Для передаточных функций.

`tf_arima_transformation_type.`
имя_поля
	`Нет`
	`SquareRoot`
	`NaturalLog`	

arima_detect_outlier_mode	`Нет` `Автоматически`	
arima_outlier_additive	*флаг*	
arima_outlier_level_shift	*флаг*	
arima_outlier_innovational	*флаг*	

Таблица 76. Свойства streamingts (продолжение)		
Свойства streamingts	**Тип переменной**	**Описание свойства**
arima_outlier_transient	*флаг*	
arima_outlier_seasonal_additive	*флаг*	
arima_outlier_local_trend	*флаг*	
arima_outlier_additive_patch	*флаг*	
deployment_force_rebuild	*флаг*	
deployment_rebuild_mode	`Количество` `Percent`	
deployment_rebuild_count	*число*	
deployment_rebuild_pct	*число*	
deployment_rebuild_field	*<field>*	

Глава 11. Поле Свойства узла операций

Свойства anonymizenode

Узел анонимизации преобразует способ представления имен и значений полей уровнем ниже, маскируя таким образом исходные данные. Это может быть полезно, если вы хотите разрешить другим пользователям построить модели, используя чувствительные данные, такие как имена клиентов или другие подробности.

Пример

```
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)
# Узлу анонимизации требуются входные поля при задании значений
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")
```

Таблица 77. Свойства anonymizenode

Свойства anonymizenode	**Тип переменной**	**Описание свойства**
enable_anonymize	*флаг*	Если задано значение True, активирует анонимизацию значений полей (эквивалентно выбору варианта **Да** для этого поля в столбце Анонимизировать значения).
use_prefix	*флаг*	Если задано значение True, будет использоваться пользовательский префикс, если он задан. Применяется к полям, которые будут анонимизированы хеш-методом, что эквивалентно включению радиокнопки **Настроить** в диалоговом окне Заменить значения для этого поля.
prefix	*string*	Эквивалентно вводу префикса в текстовом поле диалогового окна Заменить значения. Если ничего дополнительного не задано, префикс по умолчанию - это значение по умолчанию.
преобразование	Переменный Фиксированная	Определяет, какими будут параметры преобразования для анонимизируемого методом Преобразование поля - случайными или фиксированными.
set_random_seed	*флаг*	Если задано значение True, будет использоваться указанное начальное значение генератора псевдослучайных чисел (если для transformation также задано значение Random).

Таблица 77. Свойства anonymizenode (продолжение)		
Свойства anonymizenode	**Тип переменной**	**Описание свойства**
random_seed	*целое*	Если для set_random_seed задано значение True, это начальное значение генератора псевдослучайных чисел.
scale	*number*	Если для transformation задано значение Fixed, это значение используется для функции "умножить на". Максимальное значение масштабирования обычно равно 10, но его можно уменьшить для предотвращения переполнения.
translate	*number*	Если для transformation задано значение Fixed, это значение используется для функции "умножить на". Максимальное значение умножения обычно равно 1000, но его можно уменьшить для предотвращения переполнения.

Свойства autodataprepnode

Узел автоматической подготовки данных (Automated Data Preparation, ADP) может анализировать ваши данные и находит исправления, выявляет проблемные и малополезные поля, создает при необходимости производные атрибуты и повышает производительность, применяя интеллектуальные способы анализа и выборки. Этот узел можно использовать в полностью автоматическом режиме, позволив ему выбирать и применять исправления или предварительно просматривать изменения перед тем, как они сделаны и приняты, а при желании применять, отклонять или исправлять их.

Пример

```
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")
```

Таблица 78. Свойства autodataprepnode		
Свойства autodataprepnode	**Тип переменной**	**Описание свойства**
objective	Balanced Speed Accuracy Пользовательские	

Таблица 78. Свойства autodataprepnode (продолжение)		
Свойства autodataprepnode	**Тип переменной**	**Описание свойства**
custom_fields	*флаг*	Если значение флага true, это позволяет вам задать поле назначения, входные и другие поля для текущего узла. При значении false используются текущие параметры с вышележащего узла Тип.
target	*поле*	Задает одно поле назначения.
inputs	[*поле1 ... полеN*]	Входные (или предикторные) поля, используемые в модели.
use_frequency	*флаг*	
frequency_field	*поле*	
use_weight	*флаг*	
weight_field	*поле*	
excluded_fields	Фильтр Нет	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*флаг*	Управлять доступом ко всем полям даты и времени
compute_time_until_date	*флаг*	
reference_date	Сегодня Фиксированная	
fixed_date	*дата*	
units_for_date_durations	Автоматически Фиксированная	
fixed_date_units	Years Months Days	
compute_time_until_time	*флаг*	
reference_time	CurrentTime Фиксированная	
fixed_time	*время*	

Таблица 78. Свойства autodataprepnode (продолжение)		
Свойства autodataprepnode	**Тип переменной**	**Описание свойства**
units_for_time_durations	Автоматически Фиксированная	
fixed_date_units	Hours Minutes Секунды	
extract_year_from_date	*флаг*	
extract_month_from_date	*флаг*	
extract_day_from_date	*флаг*	
extract_hour_from_time	*флаг*	
extract_minute_from_time	*флаг*	
extract_second_from_time	*флаг*	
exclude_low_quality_inputs	*флаг*	
exclude_too_many_missing	*флаг*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*флаг*	
maximum_number_categories	*number*	
exclude_if_large_category	*флаг*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*флаг*	
adjust_type_inputs	*флаг*	
adjust_type_target	*флаг*	
reorder_nominal_inputs	*флаг*	
reorder_nominal_target	*флаг*	
replace_outliers_inputs	*флаг*	
replace_outliers_target	*флаг*	
replace_missing_continuous_inputs	*флаг*	
replace_missing_continuous_target	*флаг*	
replace_missing_nominal_inputs	*флаг*	

Таблица 78. Свойства autodataprepnode (продолжение)		
Свойства autodataprepnode	**Тип переменной**	**Описание свойства**
replace_missing_nominal_target	*флаг*	
replace_missing_ordinal_inputs	*флаг*	
replace_missing_ordinal_target	*флаг*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Удалить	
rescale_continuous_inputs	*флаг*	
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	*флаг*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*флаг*	
maximize_association_with_target	*флаг*	
p_value_for_merging	*number*	
merge_ordinal_features	*флаг*	
merge_nominal_features	*флаг*	
minimum_cases_in_category	*число*	
bin_continuous_fields	*флаг*	
p_value_for_binning	*number*	
perform_feature_selection	*флаг*	
p_value_for_selection	*number*	

Таблица 78. Свойства autodataprepnode (продолжение)		
Свойства autodataprepnode	**Тип переменной**	**Описание свойства**
perform_feature_construction	*флаг*	
transformed_target_name_extension	*string*	
transformed_inputs_name_extension	*string*	
constructed_features_root_name	*string*	
years_duration_name_extension	*string*	
months_duration_name_extension	*string*	
days_duration_name_extension	*string*	
hours_duration_name_extension	*string*	
minutes_duration_name_extension	*string*	
seconds_duration_name_extension	*string*	
year_cyclical_name_extension	*string*	
month_cyclical_name_extension	*string*	
day_cyclical_name_extension	*string*	
hour_cyclical_name_extension	*строка*	
minute_cyclical_name_extension	*строка*	
second_cyclical_name_extension	*строка*	

Свойства astimeintervalsnode

Узел Интервалы времени используется для задания интервалов и получения нового поля времени для операций оценки или прогноза. Поддерживается весь диапазон интервалов времени от секунд до лет.

Таблица 79. свойства astimeintervalsnode		
Свойства astimeintervalsnode	**Тип переменной**	**Описание свойства**
time_field	*поле*	Может принимать только одно количественное поле. Это поле используется узлом в качестве ключа агрегирования для преобразования интервала. Если здесь используется целочисленное поле, считается, что у него есть индекс времени.
измерения	*[поле1 поле2 ... полеn]*	Эти поля используются для создания отдельных временных рядов, основанных на значениях этих полей.
fields_to_aggregate	*[поле1 поле2 ... полеn]*	Агрегирование этих полей входит в состав операции изменения периода поля времени. Из любых полей, не включенных в этот инструмент выбора, отфильтровываются данные, исключаемые из узла.

Свойства binningnode

Узел разделения на интервалы автоматически создает новые номинальные поля на основе значений одного или нескольких существующих количественных полей (числового диапазона). Например, можно преобразовать количественное входное поле в новое категориальное поле, содержащее группы входных данных, как отклонения от среднего. После создания интервалов для нового поля вы можете сгенерировать узел извлечения на основе точек деления.

Пример

```
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)
```

Таблица 80. Свойства binningnode		
Свойства binningnode	**Тип переменной**	**Описание свойства**
fields	*[поле1 поле2 ... fieldn]*	Отложенное преобразование количественных полей (числовой диапазон). Можно разделить на интервалы несколько полей одновременно.

Таблица 80. Свойства binningnode (продолжение)		
Свойства binningnode	**Тип переменной**	**Описание свойства**
method	FixedWidth EqualCount Ранг SDev Оптимальная	Способ, используемый для определения точек разделения для новых интервалов поля (категорий).
rcalculate_bins	Always IfNecessary	Задает, вычисляются ли повторно интервалы для размещения данных в соответствующем интервале при каждом выполнении узла, или же данные добавляются только в существующие и любые новые добавленные интервалы.
fixed_width_name_extension	*string*	Расширение по умолчанию - это *_BIN*.
fixed_width_add_as	Суффикс Prefix	Задает, куда добавляется расширение имени поля, в конец имени (суффикс) или в начало (префикс). Расширение по умолчанию - это *income_BIN*.
fixed_bin_method	Width Количество	
fixed_bin_count	*целое*	Задает целое число, используемое для определения количества интервалов фиксированной ширины (категорий) для нового поля или полей.
fixed_bin_width	*real*	Значение (целое или действительное) для вычисления ширины интервала.
equal_count_name_ extension	*string*	Расширение по умолчанию - это *_TILE*.
equal_count_add_as	Суффикс Prefix	Задает расширение, суффикс или префикс, используемое для имени поля, сгенерированного при помощи стандартных процентилей. Расширение по умолчанию - это *_TILE* плюс *N*, где *N* - это порядок процентили.
tile4	*флаг*	Задает четыре интервала квантили, каждая из которых содержит 25% наблюдений.
tile5	*флаг*	Генерирует пять интервалов квинтили.
tile10	*флаг*	Генерирует десять интервалов децили.

Таблица 80. Свойства binningnode (продолжение)		
Свойства binningnode	**Тип переменной**	**Описание свойства**
tile20	*флаг*	Генерирует 20 интервалов вингтили.
tile100	*флаг*	Генерирует 100 интервалов процентили.
use_custom_tile	*флаг*	
custom_tile_name_extension	*string*	Расширение по умолчанию - это *_TILEN*.
custom_tile_add_as	Суффикс Prefix	
custom_tile	*целое*	
equal_count_method	RecordCount ValueSum	Способ RecordCount направлен на назначение одинакового числа записей в каждый интервал, в то время как ValueSum назначает записи так, чтобы сумма значений в каждом интервале была одинакова.
tied_values_method	Далее Current Переменный	Задает, в какой интервал будут помещены связанные со значением данные.
rank_order	По возрастанию По убыванию	Это свойство включает в себя опции По возрастанию (минимальное значение помечается как первое) или По убыванию (максимальное значение помечается как первое).
rank_add_as	Суффикс Префикс	Эта опция применяется к рангу, дробному рангу и процентному рангу.
rank	*флаг*	
rank_name_extension	*string*	Расширение по умолчанию - это *_RANK*.
rank_fractional	*флаг*	Ранжирует наблюдения, причем значение в новом поле равно рангу, деленному на сумму весов непропущенных наблюдений. Дробные ранги попадают в диапазон 0 – 1.
rank_fractional_name_ extension	*string*	Расширение по умолчанию - это *_F_RANK*.
rank_pct	*флаг*	Каждый ранг делится на число записей с непропущенными значениями и умножается на 100. Процент дробных рангов попадают в диапазон 0 – 100.

Таблица 80. Свойства binningnode (продолжение)

Свойства binningnode	Тип переменной	Описание свойства
rank_pct_name_extension	*string*	Расширение по умолчанию - это *_P_RANK*.
sdev_name_extension	*string*	
sdev_add_as	Суффикс Prefix	
sdev_count	One Two Три	
optimal_name_extension	*string*	Расширение по умолчанию - это *_OPTIMAL*.
optimal_add_as	Суффикс Prefix	
optimal_supervisor_field	*поле*	Поле, выбранное как контрольное, с которым связаны поля, выбранные для разделения на интервалы.
optimal_merge_bins	*флаг*	Задает, что любые интервалы с малым количеством наблюдений будут добавлены к большему соседнему интервалу.
optimal_small_bin_threshold	*целое*	
optimal_pre_bin	*флаг*	Обозначает, что должно выполняться предварительное разделение набора данных на интервалы.
optimal_max_bins	*целое*	Задает верхний предел, чтобы исключить создания неограниченно большого числа интервалов.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Неограниченный Bounded	
optimal_last_bin	Неограниченный Bounded	

Свойства derivenode

Узел извлечения изменяет значения данных или создает новые поля из одного или нескольких существующих полей. Он создает поля формулы типа, флага, номинала, состояния, количества и условного выражения.

Пример 1

```
# Создать и сконфигурировать флаг Извлечь узел поля
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\"")

# Создать и сконфигурировать условное извлечение узла поля
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\"")
```

Пример 2

В этом сценарии предполагается наличие двух столбцов с именами XPos и YPos, представляющих координаты X и Y точки (например, точки, где произошло событие). Этот сценарий создает узел извлечения, вычисляющий столбец геопространственного типа на основе координат X и Y, представляя полученную точку в конкретной системе координат.

```
stream = modeler.script.stream()
# Остальной код конфигурирования потока
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Теперь мы задаем общий тип измерения, определяем
# особенности геопространственного объекта
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Таблица 81. Свойства derivenode

Свойства derivenode	Тип переменной	Описание свойства
new_name	*string*	Имя нового поля.
mode	Одиночный Несколько	Создает одно или несколько полей.
fields	*список*	Используется только в режиме Несколько для выбора несколько полей.
name_extension	*string*	Задает расширение для имен новых полей.

Таблица 81. Свойства derivenode (продолжение)		
Свойства derivenode	**Тип переменной**	**Описание свойства**
add_as	Суффикс Prefix	Добавляет расширение как префикс (в начале) или как суффикс (в конце) имени поля.
result_type	Формула Флаг Set Состояние Количество Условное	Шесть типов новых полей, которые можно создать.
formula_expr	*строка*	Выражение для вычисления значения в новом поле на узле Извлечение.
flag_expr	*строка*	
flag_true	*строка*	
flag_false	*string*	
set_default	*string*	
set_value_cond	*string*	Структурировано для предоставления условия, связанного с данным значением.
state_on_val	*string*	Задает значение для нового поля, когда выполнено условие On.
state_off_val	*string*	Задает значение для нового поля, когда выполнено условие Off.
state_on_expression	*string*	
state_off_expression	*string*	
state_initial	Вкл Off	Назначает каждой записи нового поля начальное значение On или Off. Это значение можно изменить при выполнении любого условия.
count_initial_val	*string*	
count_inc_condition	*string*	
count_inc_expression	*string*	
count_reset_condition	*string*	
cond_if_cond	*string*	

Таблица 81. Свойства derivenode (продолжение)

Свойства derivenode	Тип переменной	Описание свойства
cond_then_expr	*string*	
cond_else_expr	*string*	
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	Это свойство может использоваться для определения измерения, связанного с производным полем. Функции setter можно передавать строку или одно из значений MeasureType. getter будет всегда возвращать значения MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Для полей собраний (списков с глубиной 0) это свойство определяет тип измерения, связанный с базовыми значениями.
geo_type	Точки Несколько точек Ломаная Мультиломаная Многоугольник Мультиполигон	Для геопространственных полей это свойство определяет тип геопространственного объекта, представляемого этим полем. Он должен быть согласован с глубиной списка значений

Таблица 81. Свойства derivenode (продолжение)		
Свойства derivenode	**Тип переменной**	**Описание свойства**
has_coordinate_system	*логическое*	Для геопространственных полей это свойство определяет наличие у поля системы координат.
coordinate_system	*строка*	Для геопространственных полей это свойство определяет для данного поля систему координат.

Свойства ensemblenode

Узел Ансамбль объединяет два или более слепков моделей для получения более точных предсказаний, чем можно получить от любой модели.

Пример

```
# Создать и сконфигурировать узел ансамбля
# Использовать этот узел с моделями в 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")
```

Таблица 82. Свойства ensemblenode		
Свойства ensemblenode	**Тип переменной**	**Описание свойства**
ensemble_target_field	*поле*	Задает поле назначения для всех моделей, используемых в ансамбле.
filter_individual_model_output	*флаг*	Задает, нужно ли отключать результаты скоринга из индивидуальных моделей.
flag_ensemble_method	Голосование ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Задает способ, используемый для определения оценки ансамбля. Этот параметр применяется только в том случае, если выбранное поле назначение - это флаговое поле.

Таблица 82. Свойства ensemblenode (продолжение)		
Свойства ensemblenode	**Тип переменной**	**Описание свойства**
set_ensemble_method	Голосование ConfidenceWeightedVoting HighestConfidence	Задает способ, используемый для определения оценки ансамбля. Этот параметр применим только в том случае, если выбранное поле назначения номинальное.
flag_voting_tie_selection	Переменный HighestConfidence RawPropensity AdjustedPropensity	Если выбран способ голосования, задает, как разрешаются связи. Этот параметр применим только в том случае, если выбранное поле назначения флаговое.
set_voting_tie_selection	Переменный HighestConfidence	Если выбран способ голосования, задает, как разрешаются связи. Этот параметр применим только в том случае, если выбранное поле назначения номинальное.
calculate_standard_error	*флаг*	Если поле назначения количественное, по умолчанию запускается вычисление среднеквадратичной ошибки для определения различий между измеренными или оцененными значениями и действительными значениями, а также для демонстрации, насколько эти оценки совпали.

Свойства fillernode

Узел заполнителя замещает значения полей и заменяет систему хранения. Вы можете заменить значения на основе условия CLEM, такого как @BLANK(@FIELD). Можно также выбрать замену всех пробелов или всех пробельных значений на заданное значение. Узел заполнителя часто используется вместе с узлом Тип для замены пропущенных значений.

Пример

```
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\"")
```

Таблица 83. Свойства fillernode		
Свойства fillernode	**Тип переменной**	**Описание свойства**
fields	*список*	Поля из набора данных, значения которого будут проверены и заменены.
replace_mode	Всегда Условное Пробел Ноль BlankAndNull	Вы можете заменить все значения, пустые значения или значения null, а также выполнить замену на основе заданного условия.
condition	*string*	
replace_with	*string*	

Свойства filternode

Узел Фильтр фильтрует (отбрасывает) поля, переименовывает поля и отображает поля с одного узла источника на другой.

Пример:

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

Использование свойства default_include. Обратите внимание на то, что задание значения для свойства default_include автоматически не включает и не исключает все поля; оно просто определяет значения по умолчанию для текущего выбора. Это функциональный эквивалент нажатия кнопки **Включить поля по умолчанию** в диалоговом окне узла Фильтр. Допустим, например, что вы запускаете следующий сценарий:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Включить эти два поля в список
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

В результате узел передаст только поля Age и Sex и отбросит все другие поля. После запуска предыдущего сценария теперь предположим, что вы добавили к сценарию следующие строки, чтобы указать еще два поля:

```
node.setPropertyValue("default_include", False)
# Включить эти два поля в список
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

При этом к фильтру добавится еще два поля, так что всего будет передано четыре поля (Age, Sex, BP, Na). Другими словами, сброс значения default_include до False автоматически не сбрасывает все поля.

Как вариант, если теперь изменить `default_include` на `True`, используя сценарий или в диалоговом окне Фильтр, это изменит поведение, так что четыре перечисленные поля будут отброшены, а не включены. При сомнениях можно опробовать управляющие элементы диалогового окна узла Фильтр, что будет полезно для понимания таких взаимодействий.

Таблица 84. Свойства filternode

Свойства filternode	**Тип переменной**	**Описание свойства**
`default_include`	*флаг*	Ключевое свойство для указания, каким будет поведение по умолчанию, передать или отфильтровать поля: Обратите внимание на то, что задание значения для этого свойства автоматически не включает и не исключает все поля; оно просто определяет, включаются или исключаются выбранные поля по умолчанию. Дополнительные комментарии смотрите в примере ниже.
`include`	*флаг*	Ключевое свойство для включения или удаления поля.
`new_name`	*string*	

Свойства historynode

Узел Хронология создает новые поля, содержащие данные из полей в предыдущих записях. Хронологические узлы чаще всего используются для последовательных данных, таких как данные временных рядов. Перед использованием узла Хронология может потребоваться отсортировать данные с использованием узла Сортировка.

Пример

```
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")
```

Таблица 85. Свойства historynode

Свойства historynode	**Тип переменной**	**Описание свойства**
`fields`	*список*	Поля, для которых требуется хронология.
`смещение`	*number*	Задает последнюю запись (предшествующую текущей записи), из которой нужно извлекать значения хронологического поля.
`span`	*number*	Задает количество предшествующих записей, из которых нужно извлекать значения.

Таблица 85. Свойства historynode (продолжение)		
Свойства historynode	**Тип переменной**	**Описание свойства**
unavailable	Исключение Leave Заполнить	Обсуждая обработку записей без хронологических значений, обычно имеют в виду первые несколько записей (наверху набора данных), у которых еще нет предыдущих записей, которые можно бы было использовать как хронологические.
fill_with	String Число	Задает значение или строку, которые будут использоваться для записей без доступных хронологических значений.

Свойства partitionnode

Узел Разделы генерирует поле раздела, которое разбивает данные на отдельные подмножества для стадий обучения, испытания и проверки при построении моделей.

Пример

```
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")
```

Таблица 86. Свойства partitionnode		
Свойства partitionnode	**Тип переменной**	**Описание свойства**
new_name	*string*	Имя поля раздела, сгенерированного узлом.
create_validation	*флаг*	Задает, должен ли создаваться раздел проверки.
training_size	*целое*	Процентная доля записей (0–100) для выделения разделу обучения.
testing_size	*целое*	Процентная доля записей (0–100) для выделения разделу испытания.
validation_size	*целое*	Процентная доля записей (0–100) для выделения разделу проверки. Игнорируется, если раздел проверки не создан.
training_label	*string*	Метка для раздела обучения.
testing_label	*string*	Метка для раздела испытания.
validation_label	*string*	Метка для раздела проверки. Игнорируется, если раздел проверки не создан.

Таблица 86. Свойства partitionnode (продолжение)		
Свойства partitionnode	**Тип переменной**	**Описание свойства**
value_mode	Системная SystemAndLabel Метка	Задает значения, используемые для представления каждого раздела в данных. Например, обучающая выборка может быть представлена системным целым числом 1, меткой Обучение или комбинацией обоих значений 1_Обучение.
set_random_seed	*Логический*	Задает, нужно ли использовать определенное пользователем начальное значение генератора псевдослучайных чисел.
random_seed	*целое*	Заданное пользователем случайное начальное значение генератора псевдослучайных чисел. Чтобы использовать это значение, для set_random_seed должно быть задано True.
enable_sql_generation	*Логический*	Задает, использовать ли SQL pushback для назначения записей разделам.
unique_field		Задает входное поле, используемое для обеспечения назначения записей разделам случайным, но повторяемым образом. Чтобы использовать это значение, для enable_sql_generation должно быть задано значение True.

Свойства reclassifynode

Узел переклассификации преобразует один набор категориальных значений в другой. Переклассификация полезна для сворачивания категорий или для перегруппировки данных для анализа.

Пример

```
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"])
```

Таблица 87. Свойства reclassifynode		
Свойства reclassifynode	**Тип переменной**	**Описание свойства**
mode	Single Несколько	Режим Единичный переклассифицирует категории для одного поля. Несколько активирует опции, включающие одновременное преобразование нескольких полей.
replace_field	*флаг*	
поле	*string*	Используется только в режиме Единичный.
new_name	*string*	Используется только в режиме Единичный.
fields	*[поле1 поле2 ... fieldn]*	Используется только в режиме Несколько.
name_extension	*string*	Используется только в режиме Несколько.
add_as	Суффикс Prefix	Используется только в режиме Несколько.
reclassify	*строка*	Структурированное свойство для значений полей.
use_default	*флаг*	Использовать значение по умолчанию.
по умолчанию	*string*	Задать значение по умолчанию.
pick_list	*[строка строка ... строка]*	Позволяет пользователю импортировать список известных новых значений для заполнения раскрывающегося списка в таблице.

Свойства reordernode

Узел переупорядочения полей определяет естественный порядок, используемый для вывода полей нижележащего уровня. Этот порядок влияет на показ полей во многих положениях, таких как таблицы, списки и средство выбора полей. Эта операция полезна при работе с обширными наборами данных, чтобы сделать нужные поля более наглядными.

Пример

```
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"])
```

Таблица 88. Свойства reordernode		
Свойства reordernode	**Тип переменной**	**Описание свойства**
mode	Пользовательские Авто	Значения можно сортировать автоматически или задавать пользовательский порядок.
sort_by	Имя Тип Хранение	
ascending	*флаг*	
start_fields	*[поле1 поле2 ... полеn]*	Новые поля вставляются после этих полей.
end_fields	*[поле1 поле2 ... полеn]*	Новые поля вставляются перед этими полями.

Свойства reprojectnode

В SPSS Modeler элементы, такие как пространственные функции построителя выражений, узел Пространственное предсказание (Spatio-Temporal Prediction, STP) и узел Визуализация карт используют систему координат проекции. При помощи узла Репроецирование можно изменить систему координат для любых импортируемых данных, где используется географическая система координат.

Таблица 89. свойства reprojectnode		
Свойства reprojectnode	**Тип переменной**	**Описание свойства**
reproject_fields	*[поле1 поле2 ... полеn]*	Список всех полей, которые надо репроецировать.
reproject_type	Streamdefault Задать	Выберите способ репроецирования полей.
coordinate_system	*строка*	Имя применяемой к полям системы координат. Пример: set reprojectnode.coordinate_system = “WGS_1984_World_Mercator”

Свойства restructurenode

Узел реструктуризации преобразует номинальное или флаговое поле в группу полей, которые можно заполнить значениями еще одного поля. Например, если задано поле с именем *тип_платежа*, у которого могут быть значения *кредит*, *наличные* и *дебет*, могут быть заданы три новые поля (*кредит*, *наличные*, *дебет*), каждое из которых может содержать значение фактического выполненного платежа.

Пример

```
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"])
```

Таблица 90. Свойства restructurenode

Свойства restructurenode	**Тип переменной**	**Описание свойства**
fields_from	[*категория категория категория*] все	
include_field_name	*флаг*	Указывает, использовать ли имя поля в реструктурированном имени поля.
value_mode	OtherFields Флаги	Указывает на режим для того, чтобы определить значения для реструктурированных полей. Используя OtherFields, нужно указать, какие поля использовать (см. ниже). При использовании Flags значения - это числовые флаги.
value_fields	*список*	Требуется, если для value_mode задано OtherFields. Задает, какие поля использовать как поля значений.

Свойства rfmanalysisnode

Узел анализа Новизна, частота, деньги (Recency, Frequency, Monetary - RFM) позволяет вам количественно определить, какие клиенты вероятно будут лучшими, исследуя, насколько недавно они сделали свои последние покупки (новизна), как часто они покупали (частота) и сколько денег потратили на все транзакции (деньги).

Пример

```
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])
```

Таблица 91. Свойства rfmanalysisnode

Свойства rfmanalysisnode	**Тип переменной**	**Описание свойства**
recency	*поле*	Задать поле новизны. Это может быть дата, отметка времени или просто число.
frequency	*поле*	Задать поле частоты.

Таблица 91. Свойства rfmanalysisnode (продолжение)

Свойства rfmanalysisnode	**Тип переменной**	**Описание свойства**
monetary	*поле*	Задать поле денег.
recency_bins	*целое*	Задать количество интервалов новизны, которые будут сгенерированы.
recency_weight	*number*	Задать веса для применения к недавним данным. По умолчанию используется значение 100.
frequency_bins	*целое*	Задать количество интервалов частоты, которые будут сгенерированы.
frequency_weight	*number*	Задать веса для применения к данным частоты. Значение по умолчанию - 10.
monetary_bins	*целое*	Задать количество интервалов денег, которые будут сгенерированы.
monetary_weight	*number*	Задать веса для применения к денежным данным. Значение по умолчанию - 1.
tied_values_method	Далее Текущий	Задает, в какой интервал будут помещены связанные со значением данные.
recalculate_bins	Всегда IfNecessary	
add_outliers	*флаг*	Доступно только в том случае, если для recalculate_bins задано значение IfNecessary. Если задана эта опция, записи, лежащие ниже самого нижнего интервала, будут добавлены к нему, а записи, лежащие выше самого верхнего интервала, будут добавлены к этому интервалу.
binned_field	Недавность Frequency Monetary	
recency_thresholds	*значение значение*	Доступно только в том случае, если для recalculate_bins задано значение Always. Задать верхний и нижний порог для интервалов новизны. Верхний порог одного интервала используется как нижний порог следующего, например, [10 30 60] будет определять два интервала, первый интервал с верхним и нижним порогами 10 и 30, а второй интервал с порогами 30 и 60.

Таблица 91. Свойства rfmanalysisnode (продолжение)		
Свойства rfmanalysisnode	**Тип переменной**	**Описание свойства**
frequency_thresholds	*значение значение*	Доступно только в том случае, если для recalculate_bins задано значение Always.
monetary_thresholds	*значение значение*	Доступно только в том случае, если для recalculate_bins задано значение Always.

Свойства settoflagnode

Узел Задать как флаг извлекает несколько полей флагов на основании категориальных значений, определенных для одного или нескольких номинальных полей.

Пример

```
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"])
```

Таблица 92. Свойства settoflagnode		
Свойства settoflagnode	**Тип переменной**	**Описание свойства**
fields_from	[*категория категория категория*] все	
true_value	*строка*	Задает значение true, используемое узлом при установке флага. Значение по умолчанию - T.
false_value	*string*	Задает значение false, используемое узлом при установке флага. Значение по умолчанию - F.
use_extension	*флаг*	Использовать расширение как префикс или суффикс для нового поля флага.
extension	*string*	
add_as	Суффикс Prefix	Задает, как добавляется расширение, в виде префикса или суффикса.

Таблица 92. Свойства settoflagnode (продолжение)		
Свойства settoflagnode	**Тип переменной**	**Описание свойства**
aggregate	*флаг*	Группирует совместно записи на основании ключевых полей. Если для какой-то записи задается значение true, включаются все флаговые поля в группе.
ключи	*список*	Ключевые поля.

Свойства statisticstransformnode

Узел Преобразование статистики запускает разнообразные команды синтаксиса IBM SPSS Statistics для источников данных в IBM SPSS Modeler. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Свойства этого узла описаны в разделе “Свойства statisticstransformnode” на стр. 453.

Свойства timeintervalsnode (объявлено устаревшим)

Прим.: Этот узел в версии 18 SPSS Modeler был объявлен устаревшим и заменен новым узлом Временной ряд.

Узел Интервалы времени задает интервалы и создает метки (при необходимости) для моделирования временных рядов. Если значения явно не разделены пробелами, этот узел может заполнить или агрегировать значения, как требуется для генерирования однородных интервалов между записями.

Пример

```
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")
```

Таблица 93. Свойства timeintervalsnode

Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
interval_type	Нет Периоды CyclicPeriods Годы Кварталы Месяцы DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Метка Create	Определяет, хотите ли вы маркировать записи последовательно или построить ряд на основе указанного поля даты, метки времени или времени.
поле	*поле*	При построении ряда из данных задает поле, обозначающее дату или время для каждой записи.
period_start	*целое*	Задает начальный интервал для периодов или периодов циклов
cycle_start	*целое*	Начальный цикл для периодов циклов.
year_start	*целое*	Для применимых типов интервалов - год, в который попадает первый интервал.
quarter_start	*целое*	Для применимых типов интервалов - квартал, в который попадает первый интервал.

Таблица 93. Свойства timeintervalsnode (продолжение)

Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
month_start	Январь Февраль Март Апрель Май Июнь Июль Август Сентябрь Октябрь Ноябрь Декабрь	
day_start	*целое*	
hour_start	*целое*	
minute_start	*целое*	
second_start	*целое*	
periods_per_cycle	*целое*	Для периодов циклов - число в каждом цикле.
fiscal_year_begins	Январь Февраль Март Апрель Май Июнь Июль Август Сентябрь Октябрь Ноябрь Декабрь	Для квартальных интервалов задает месяц начала финансового года.
week_begins_on	Воскресенье Понедельник Tuesday Wednesday Четверг Friday Суббота Воскресенье	Для периодических интервалов (дней в неделю, часов в день, минут в день и секунд в день) задает день, с которого начинается неделя.

Таблица 93. Свойства timeintervalsnode (продолжение)

Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
day_begins_hour	*целое*	Для периодических интервалов (часов в день, минут в день и секунд в день) задает час, с которого начинается день. Можно использовать вместе с day_begins_minute и day_begins_second, чтобы задать точное время, такое как *8:05:01*. Смотрите пример использования ниже.
day_begins_minute	*целое*	Для периодических интервалов (часов в день, минут в день и секунд в день) задает минуту, в которую начинается день (например, *5* в *8:05*).
day_begins_second	*целое*	Для периодических интервалов (часов в день, минут в день и секунд в день) задает секунду, в которую начинается день (например, *17* в *8:05:17*).
days_per_week	*целое*	Для периодических интервалов (дней в неделю, часов в день, минут в день и секунд в день) задает количество дней в неделю.
hours_per_day	*целое*	Для периодических интервалов (часов в день, минут в день и секунд в день) задает количество часов в день.
interval_increment	1 2 3 4 5 6 10 15 20 30	Для минут в день и секунд в день задает количество минут или секунд для инкремента при переходе к следующей записи.
field_name_extension	*string*	

Таблица 93. Свойства timeintervalsnode (продолжение)		
Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
field_name_extension_as_prefix	*флаг*	
date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДД" DAY MONTH "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ" "ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	
time_format	"ЧЧММСС" "ЧЧММ" "ММСС" "ЧЧ:ММ:СС" "ЧЧ:ММ" "ММ:СС" "(Ч)Ч:(М)М:(С)С" "(Ч)Ч:(М)М" "(М)М:(С)С" "ЧЧ.ММ.СС" "ЧЧ.ММ" "ММ.СС" "(Ч)Ч.(М)М.(С)С" "(Ч)Ч.(М)М" "(М)М.(С)С"	
aggregate	Mean Sum Мода Min Max Первое Последнее TrueIfAnyTrue	Задает метод агрегации для поля.

Таблица 93. Свойства timeintervalsnode (продолжение)

Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
pad	Пробел MeanOfRecentPoints True False	Задает метод дополнения значений переменной длины для поля.
agg_mode	All Задать	Задает, как агрегировать или заполнять поля - используя нужные функции по умолчанию или задавая поля и функции для использования.
agg_range_default	Mean Sum Мода Min Max	Задает функцию по умолчанию для агрегирования количественных полей.
agg_set_default	Мода Первое Последнее	Задает функцию по умолчанию для агрегирования номинальных полей.
agg_flag_default	TrueIfAnyTrue Мода Первое Последнее	
pad_range_default	Пробел MeanOfRecentPoints	Задает функцию по умолчанию для заполнения количественных полей.
pad_set_default	Пробел MostRecentValue	
pad_flag_default	Пробел True False	

Таблица 93. Свойства timeintervalsnode (продолжение)		
Свойства timeintervalsnode	**Тип переменной**	**Описание свойства**
max_records_to_create	*целое*	Задает максимальное количество записей для создания при заполнении ряда.
estimation_from_beginning	*флаг*	
estimation_to_end	*флаг*	
estimation_start_offset	*целое*	
estimation_num_holdouts	*целое*	
create_future_records	*флаг*	
num_future_records	*целое*	
create_future_field	*флаг*	
future_field_name	*string*	

Свойства transposenode

Узел Транспонирование меняет данные в строках и столбцах, чтобы записи становились полями, а поля записями.

Пример

```
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")
```

Таблица 94. Свойства transposenode		
Свойства transposenode	**Тип переменной**	**Описание свойства**
transpose_method	*enum*	Задает метод транспонирования: Нормальный (normal), CASE в VAR (casetovar) или VAR в CASE (vartocase).
transposed_names	Prefix Чтение	Свойство для нормального метода транспонирования. Новые имена полей могут быть сгенерированы автоматически на основе указанного префикса, или они могут быть считаны из уже существующего поля в данных.
prefix	*string*	Свойство для нормального метода транспонирования.
num_new_fields	*целое*	Свойство для нормального метода транспонирования. При использовании префикса задает максимальное количество новых полей для создания.

Таблица 94. Свойства transposenode (продолжение)

Свойства transposenode	Тип переменной	Описание свойства
read_from_field	*поле*	Свойство для нормального метода транспонирования. Поле, из которого читаются имена. Это должно быть полностью определенное поле, или при выполнении узла произойдет ошибка.
max_num_fields	*целое*	Свойство для нормального метода транспонирования. При чтении имен из поля задает верхний предел для исключения создания неограниченно большого числа полей.
transpose_type	Numeric String Пользовательские	Свойство для нормального метода транспонирования. По умолчанию транспонируются только количественные поля (числового диапазона), но можно выбрать пользовательское подмножество числовых полей или вместо этого транспонировать все поля строки.
transpose_fields	*список*	Свойство для нормального метода транспонирования. Задает поля, которые будут транспонироваться, если используется опция Настроить.
id_field_name	*поле*	Свойство для нормального метода транспонирования.
transpose_casetovar_idfields	*поле*	Свойство для метода транспонирования CASE to VAR (casetovar). Принимает несколько полей для использования в качестве полей индексов. поле1 ... полеN
transpose_casetovar_columnfields	*поле*	Свойство для метода транспонирования CASE to VAR (casetovar). Принимает несколько полей для использования в качестве полей столбцов. поле1 ... полеN
transpose_casetovar_valuefields	*поле*	Свойство для метода транспонирования CASE to VAR (casetovar). Принимает несколько полей для использования в качестве полей значений. поле1 ... полеN
transpose_vartocase_idfields	*поле*	Свойство для метода транспонирования VAR to CASE (vartocase). Принимает несколько полей для использования в качестве полей переменных ID. поле1 ... полеN

Таблица 94. Свойства transposenode (продолжение)		
Свойства transposenode	**Тип переменной**	**Описание свойства**
transpose_vartocase_valfields	*поле*	Свойство для метода транспонирования VAR to CASE (vartocase). Принимает несколько полей для использования в качестве полей переменных значений. поле1 ... полеN

Свойства typenode

Узел Тип задает метаданные и свойства полей. Например, можно задать уровень измерений (количественный, номинальный, порядковый или флаговый) для каждого поля, задать опции для обработки отсутствующих значений и системных null, задавать роль поля для целей моделирования, задавать метки полей и значений и задавать значения для поля.

Пример

```
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"]])
```

Обратите внимание на то, что в некоторых случаях вам может потребоваться полностью определить узел Тип, чтобы правильно работали другие узлы, например, свойство поля из узла Задать как флаг. Вы можете просто соединиться с узлом Таблица и выполнить его для полного определения полей:

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

Таблица 95. Свойства typenode

Свойства typenode	**Тип переменной**	**Описание свойства**
direction	Input Target Both Нет Partition Split Frequency RecordID	Ключевое свойство для ролей полей. **Прим.:** Значения In и Out в настоящее время объявлены устаревшими. Их поддержка может быть прекращена в следующем выпуске.
type	Range Flag Set Typeless Discrete OrderedSet По умолчанию	Тип измерений поля (прежнее название - тип поля). При задании для type значения Default будут очищены все настройки параметров значения, и если для режим_значения будет установлено Задать, эта настройка сбросится до Читать. Если для value_mode задано Pass или Read, значение type не повлияет на value_mode. **Прим.:** Типы данных, используемые внутренне, отличаются от типов, видимых в узле типа. Соответствие следующее: Range -> Continuous Set - > Nominal OrderedSet -> Ordinal Discrete- > Categorical

Таблица 95. Свойства typenode (продолжение)		
Свойства typenode	**Тип переменной**	**Описание свойства**
storage	Нет данных Строка Целое Действительное число Время Дата Отметка времени	Предназначенное только для чтения ключевое свойство для типа хранения поля.
check	Нет Аннулировать Принуждать Исключение Предупреждение Прервать	Ключевое свойство для проверки типа и диапазона поля.
values	[*значение значение*]	Для количественных полей первое значение - это минимум, а последнее - максимум. Для номинальных полей задайте все значения. Для флаговых полей первое значение представляет *false*, а последнее - *true*. Задание этого свойства автоматически устанавливает для свойства режим_значения значение Задать.
value_mode	Чтение Успех Read+ Текущий Задать	Определяет, как установлены значения. Обратите внимание на то, что вы не можете непосредственно установить для этого свойства значение Задать; чтобы использовать конкретные значения, задайте свойство значения.

Таблица 95. Свойства typenode (продолжение)

Свойства typenode	Тип переменной	Описание свойства
extend_values	*флаг*	Применяется, когда для режим_значения задано Чтение. Задайте T, чтобы добавить вновь прочитанные значения к любым существующим значениям для этого поля. Задайте F, чтобы отбросить существующие значения и заменить их на вновь прочитанные значения.
enable_missing	*флаг*	Когда задано T, активирует отслеживание пропущенных значений для поля.
missing_values	[*значение значение ...*]	Задает значения данных, отмечающие пропущенные данные.
range_missing	*флаг*	Указывает, определен ли для этого поля диапазон пропущенных (пустых) значений.
missing_lower	*string*	Когда для значения диапазон_отсутствия задано true, указывает нижнюю границу диапазона значений отсутствия.
missing_upper	*string*	Когда для значения диапазон_отсутствия задано true, указывает верхнюю границу диапазона значений отсутствия.
null_missing	*флаг*	Когда для этого свойства задано T, значения *nulls* (не определенные значения, обозначаемые в программах как $null$) рассматриваются как значения отсутствия.
whitespace_missing	*флаг*	Когда для этого свойства задано T, значения, содержащие только пробельные символы (пробелы, знаки табуляции и новой строки) рассматриваются как значения отсутствия.
description	*string*	Задает описание для поля.
value_labels	*[[Значение LabelString] [Значение LabelString] ...]*	Используется для задания меток парам значений.
display_places	*целое*	Задает количество десятичных разрядов при выводе поля (применимо только к полям с системой хранения REAL). При значении –1 будут использоваться значения потока по умолчанию.

Таблица 95. Свойства typenode (продолжение)

Свойства typenode	Тип переменной	Описание свойства
export_places	*целое*	Задает количество десятичных разрядов при экспорте поля (применимо только к полям с системой хранения REAL). При значении –1 будут использоваться значения потока по умолчанию.
decimal_separator	DEFAULT PERIOD COMMA	Задает десятичный разделитель для поля (применимо только к полям с системой хранения REAL).
date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДД" DAY MONTH "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ" "ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	Задает формат даты для поля (применимо только к полям с системой хранения DATE или TIMESTAMP).
time_format	"ЧЧММСС" "ЧЧММ" "ММСС" "ЧЧ:ММ:СС" "ЧЧ:ММ" "ММ:СС" "(Ч)Ч:(М)М:(С)С" "(Ч)Ч:(М)М" "(М)М:(С)С" "ЧЧ.ММ.СС" "ЧЧ.ММ" "ММ.СС" "(Ч)Ч.(М)М.(С)С" "(Ч)Ч.(М)М" "(М)М.(С)С"	Задает формат времени для поля (применимо только к полям с системой хранения TIME или TIMESTAMP).

Таблица 95. Свойства typenode (продолжение)

Свойства typenode	**Тип переменной**	**Описание свойства**
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Задает формат вывода чисел для поля.
standard_places	*целое*	Задает количество десятичных разрядов при выводе поля в стандартном формате. При значении –1 будут использоваться значения потока по умолчанию. Обратите внимание на то, что существующий слот разряды_вывода может также использоваться для изменения, но сейчас он объявлен устаревшим.
scientific_places	*целое*	Задает количество десятичных разрядов при выводе поля в экспоненциальном представлении. При значении –1 будут использоваться значения потока по умолчанию.
currency_places	*целое*	Задает количество десятичных разрядов при выводе поля в формате валюты. При значении –1 будут использоваться значения потока по умолчанию.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Задает знак группировки для поля.
column_width	*целое*	Задает ширину столбца для поля. При значении –1 для ширины столбца будет задано Auto.

Таблица 95. Свойства typenode (продолжение)

Свойства typenode	Тип переменной	Описание свойства
justify	AUTO CENTER LEFT RIGHT	Задает выравнивание столбцов для поля.
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	Это ключевое свойство похоже на свойство type тем, что может использоваться для определения связанного с полем измерения. Отличие - в сценариях Python; функции setter может также передаваться одно из значений MeasureType, тогда как getter будет всегда возвращать значения MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Для полей собраний (списков с глубиной 0) это ключевое свойство определяет тип измерения, связанный с базовыми значениями.

Таблица 95. Свойства typenode (продолжение)

Свойства typenode	**Тип переменной**	**Описание свойства**
geo_type	Точки Несколько точек Ломаная Мультиломаная Многоугольник Мультиполигон	Для геопространственных полей это ключевое свойство определяет тип геопространственного объекта, представляемого этим полем. Он должен быть согласован с глубиной списка значений.
has_coordinate_ system	*логический*	Для геопространственных полей это свойство определяет наличие у поля координаты система
coordinate_ system	*string*	Для геопространственных полей это ключевое свойство определяет для данного поля систему координат.
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	Это ключевое свойство похоже на свойство custom_storage тем, что может использоваться для определения для поля хранения переопределения. Отличие - в сценариях Python; функции setter может также передаваться одно из значений StorageType, тогда как getter будет всегда возвращать значения StorageType.
custom_list_ storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Для полей списков это ключевое свойство задает тип хранения базовых значений.

Таблица 95. Свойства typenode (продолжение)		
Свойства typenode	**Тип переменной**	**Описание свойства**
`custom_list_depth`	*целое*	Для полей списков это ключевое свойство задает глубину поля.
`max_list_length`	*целое*	Доступно только для данных со шкалой измерений *Геопространственная* или *Собрание*. Задайте максимальную длину списка, указав количество элементов, которые может содержать этот список.
`max_string_length`	*целое*	Доступно только для данных *typeless* и используется, когда вы производите SQL для составления таблицы. Введите значение самой большой строки в ваших данных; тогда сгенерируется столбец в таблице, которая будет достаточно большой для содержания строки.

Глава 12. Свойства узла графика

Общие свойства узлов графиков

В этом разделе описаны свойства, доступные для узлов графиков, в том числе общие свойства и свойства, специфичные для каждого типа узла.

Таблица 96. Свойства Common graph node

Свойства Common graph node	**Тип переменной**	**Описание свойства**
`title`	*string*	Задает заголовок. Пример: "Это заголовок".
`caption`	*string*	Задает подпись. Пример: "Это подпись".

`output_mode`	`Screen`
	
	`File`
		Задает способ обработки выходных данных с узла графика - будут ли они выводиться на экран, или записываться в файл.

`output_format`	`BMP`
	
	`JPEG`
	
	`PNG`
	
	`HTML`
	
	`output` (*.cou*)
		Задает тип выходных данных. Точный тип выходных данных, разрешенный для каждого из узлов, варьируется.

`full_filename`	*string*	Задает путь назначения и имя файла для выходных данных, сгенерированных на узле графика.
`use_graph_size`	*флаг*	Управляет точностью определения размера графика, используя свойства ширины и высоты ниже. Влияет только на графики, которые выводятся на экран. Недоступно для узла Распределение.
`graph_width`	*number*	Когда значение `use_graph_size` - это `True`, задает ширину графика в пикселях.
`graph_height`	*число*	Когда значение `use_graph_size` - это `True`, задает высоту графика в пикселях.

Выключение дополнительных полей

Дополнительные поля, такие как поля наложения для диаграмм, можно выключить при задании для свойства значения `" "` (пустая строка), как показано в следующем примере:

```
plotnode.setPropertyValue("color_field", "")
```

Указание цветов

Цвета для заголовков, подписей, фона и меток можно задать, используя шестнадцатиричные строки, начинающиеся с символа решетки (#). Например, чтобы задать для фона лазурный цвет, используется следующий оператор:

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

Здесь первые две цифры 87 задают долю красного; средние две цифры CE - зеленого; последние две цифры EB - голубого. У каждой цифры диапазон изменения 0–9 или A–F. Совместно эти значения задают цвет в RGB (красный, зеленый, голубой).

Прим.: При задании цвета в RGB можно использовать средство выбора полей в пользовательском интерфейсе, чтобы определить правильный код цвета. Просто наведите указатель мыши на цвет, чтобы активировать подсказку с нужной информацией.

Свойства узла Собрание (collectionnode)

Узел Собрание показывает распределение значений для одного числового поля относительно значений другого. (При этом создаются диаграммы, похожие на гистограммы). Это полезно для иллюстрации переменной или поля, значения которых изменяется во времени. Используя 3D-представление, вы можете включить также символическую ось, показывающую распределения по категориям.

Пример

```
node = stream.create("collection", "My node")
# Вкладка "График"
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# Раздел "Наложение"
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# Вкладка "Опции"
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)
```

Таблица 97. Свойства collectionnode

Свойства collectionnode	Тип переменной	Описание свойства
over_field	*поле*	
over_label_auto	*флаг*	
over_label	*строка*	
collect_field	*поле*	
collect_label_auto	*флаг*	
collect_label	*string*	
three_D	*флаг*	

Таблица 97. Свойства collectionnode (продолжение)		
Свойства collectionnode	**Тип переменной**	**Описание свойства**
by_field	*поле*	
by_label_auto	*флаг*	
by_label	*string*	
operation	Sum Mean Min Max SDev	
color_field	*string*	
panel_field	*string*	
animation_field	*string*	
range_mode	Автоматически UserDefined	
range_min	*number*	
range_max	*number*	
интервалы	ByNumber ByWidth	
num_bins	*число*	
bin_width	*number*	
use_grid	*флаг*	
graph_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
page_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.

Свойства узла распределения (distributionnode)

Узел распределения показывает появление символических (категориальных) значений, таких как тип закладных или пол. Обычно узел распределения используется для показа разбалансировки данных, которую можно выправить при помощи узла балансировки до создания модели.

Пример

```
node = stream.create("distribution", "My node")
# Вкладка "График"
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)
```

Таблица 98. Свойства distributionnode

Свойства distributionnode	Тип переменной	Описание свойства
plot	SelectedFields Флаги	
x_field	*поле*	
color_field	*поле*	Поле наложения.
normalize	*флаг*	
sort_mode	ByOccurence По алфавиту	
use_proportional_scale	*флаг*	

Свойства узла оценки (evaluationnode)

Узел Оценка помогает оценить и сравнить прогнозирующие модели. Диаграмма оценки показывает, насколько хорошо модели предсказывают конкретные выходные данные. Он сортирует записи на основе предсказанного значения и доверительного интервала предсказания. Он разбивает записи на группы равного размера (**квантили**) и затем выводит значение бизнес-критерия для каждой квантили от самой высокой до самой низкой. Несколько моделей представляются разными линиями на графике.

Пример

```
node = stream.create("evaluation", "My node")
# Вкладка "График"
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")
```

Таблица 99. Свойства evaluationnode		
Свойства evaluationnode	**Тип переменной**	**Описание свойства**
chart_type	Рост Отклик Рост Доход ROI ROC	
inc_baseline	*флаг*	
field_detection_method	Метаданные Имя	
use_fixed_cost	*флаг*	
cost_value	*number*	
cost_field	*string*	
use_fixed_revenue	*флаг*	
revenue_value	*number*	
revenue_field	*string*	
use_fixed_weight	*флаг*	
weight_value	*number*	
weight_field	*поле*	
n_tile	Квартили Quintles Децили Вингтили Процентили 1000-тили	
cumulative	*флаг*	
style	Line Точка	

Таблица 99. Свойства evaluationnode (продолжение)		
Свойства evaluationnode	**Тип переменной**	**Описание свойства**
point_type	Прямоугольник Точки Треугольник Шестиугольник Плюс Пятиугольник Звезда BowTie HorizontalDash VerticalDash IronCross Фабрика Дом Собор OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Beep	
export_data	*флаг*	
data_filename	*string*	
delimiter	*string*	
new_line	*флаг*	
inc_field_names	*флаг*	
inc_best_line	*флаг*	
inc_business_rule	*флаг*	
business_rule_condition	*string*	
plot_score_fields	*флаг*	
score_fields	*[поле1 ... полеN]*	
target_field	*поле*	
use_hit_condition	*флаг*	
hit_condition	*string*	
use_score_expression	*флаг*	
score_expression	*string*	
caption_auto	*флаг*	

Свойства узла панели выбора диаграмм (graphboardnode)

Узел Панель выбора диаграмм предлагает много разных типов диаграмм на одном узле. Используя этот узел, можно выбрать поля данных, которые вы хотите изучать, а затем выбрать диаграмму из доступных для выбранных данных. Узел автоматически отфильтровывает все типы диаграмм, которые нельзя использовать для работы с выбранными полями.

Прим.: Если задается свойство, недопустимое для типа диаграммы (например, для гистограммы указывается поле_y), это свойство игнорируется.

Прим.: В пользовательском интерфейсе на вкладке Подробности для многих разных типов диаграмм есть поле **Сводка**; в настоящее время это поле не поддерживается сценариями.

Пример

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

Таблица 100. Свойства graphboardnode

Свойства graphboard	Тип переменной	Описание свойства
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Область ArrowMap Столбцы BarCounts BarCountsMap BarMap BinnedScatter Коробчатая диаграмма Пузырьковая ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues	Определяет тип диаграммы.

Таблица 100. Свойства graphboardnode (продолжение)		
Свойства graphboard	**Тип переменной**	**Описание свойства**
	ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Точечная диаграмма Тепловая карта HexBinScatter Гистограмма Line LineChartMap LineOverlayMap Параллельная Путь Круг PieCountMap PieCounts PieMap	

Таблица 100. Свойства graphboardnode (продолжение)		
Свойства graphboard	**Тип переменной**	**Описание свойства**
	PointOverlayMap PolygonOverlayMap Лента Диаграмма рассеяния SPLOM Поверхность	
x_field	*поле*	Задает пользовательскую метку для оси *x*. Доступно только для меток.
y_field	*поле*	Задает пользовательскую метку для оси *y*. Доступно только для меток.
z_field	*поле*	Используется в некоторых трехмерных диаграммах.
color_field	*поле*	Используется на картах интенсивности.
size_field	*поле*	Используется на диаграммах с пузырями.
categories_field	*поле*	
values_field	*поле*	
rows_field	*поле*	
columns_field	*поле*	
fields	*поле*	
start_longitude_field	*поле*	Используется со стрелками на опорной карте.
end_longitude_field	*поле*	
start_latitude_field	*поле*	
end_latitude_field	*поле*	
data_key_field	*поле*	Используется на различных картах.
panelrow_field	*string*	
panelcol_field	*string*	

Таблица 100. Свойства graphboardnode (продолжение)		
Свойства graphboard	**Тип переменной**	**Описание свойства**
animation_field	*string*	
longitude_field	*поле*	Используется с координатами на картах.
latitude_field	*поле*	
map_color_field	*поле*	

Свойства узла гистограммы (histogramnode)

Узел Гистограмма показывает существующие значения для числовых полей. Он часто используется для изучения данных перед работой с ними и построением моделей. Аналогично узлу Распределение узел Гистограмма часто выявляет несбалансированность данных.

Пример

```
node = stream.create("histogram", "My node")
# Вкладка "График"
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# Вкладка "Опции"
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)
```

Таблица 101. Свойства histogramnode		
Свойства histogramnode	**Тип переменной**	**Описание свойства**
поле	*поле*	
color_field	*поле*	
panel_field	*поле*	
animation_field	*поле*	
range_mode	Автоматически UserDefined	
range_min	*number*	
range_max	*number*	
интервалы	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	

Таблица 101. Свойства histogramnode (продолжение)		
Свойства histogramnode	**Тип переменной**	**Описание свойства**
normalize	*флаг*	
separate_bands	*флаг*	
x_label_auto	*флаг*	
x_label	*string*	
y_label_auto	*флаг*	
y_label	*string*	
use_grid	*флаг*	
graph_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
page_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
normal_curve	*флаг*	Указывает, должна ли кривая нормального распределения показываться при выводе.

Свойства mapvisualization

Узел Визуализация карты может принять несколько входных соединений и вывести геопространственные данные на карту как несколько слоев. Каждый слой - это одно геопространственное поле; например, базовым слоем может быть карта страны, выше может накладываться один слой дорог, один слой рек и один слой городов.

Таблица 102. Свойства mapvisualization		
Свойства mapvisualization	**Тип переменной**	**Описание свойства**
tag	*строка*	Задает имя тега для ввода. Тег по умолчанию - число на основе порядка подсоединения входных данных к узлу (первый тег соединения - 1, второй тег соединения - 2 и так далее).

Таблица 102. Свойства mapvisualization (продолжение)

Свойства mapvisualization	**Тип переменной**	**Описание свойства**
layer_field	*поле*	Выбирает, какое геопространственное поле из набора данных будет выводиться в качестве слоя на карте. Вариант выбора по умолчанию основан на следующем порядке сортировки: • Первый - Точка • Ломаная • Многоугольник • Несколько точек • Мультиломаная • Последний - Мультиполигон Если существует два поля с одинаковым типом измерений, по умолчанию выбирается первое поле в алфавитном порядке (по имени).
color_type	*логическое*	Указывает, что ко всем возможностям геопространственного поля применяться стандартный цвет (standard), либо задает поле наложения (overlay), где к возможностям будет применяться цвет на основе значений из другого поля в наборе данных. Возможные значения: standard или overlay. По умолчанию: standard.
цвет	*строка*	Если для свойства color_type выбрано значение standard, выпадающее меню будет содержать ту же палитру цветов, что и для порядка цветов категории диаграмм на вкладке пользовательских опций Вывод. Значение по умолчанию - цвет категории диаграмм 1.
color_field	*поле*	Если для свойства color_type выбрано значение overlay, выпадающее меню будет содержать все поля из того же набора данных, что и в геопространственном поле, выбранном в качестве слоя.

Таблица 102. Свойства mapvisualization (продолжение)

Свойства mapvisualization	**Тип переменной**	**Описание свойства**
symbol_type	*логическое*	Указывает, что ко всем записям геопространственного поля применяться стандартное обозначение (standard), либо задает обозначение наложения (overlay), изменяющее значок обозначения для точек на основе значений из другого поля в наборе данных. Возможные значения: standard или overlay. По умолчанию: standard.
symbol	*строка*	Если для свойства symbol_type выбрано значение standard, выпадающее меню будет содержать варианты обозначений, которые можно использовать для вывода точек на карте.
symbol_field	*поле*	Если для свойства symbol_type выбрано значение overlay, выпадающее меню будет содержать все номинальные, порядковые или категориальные поля из того же набора данных, что и в геопространственном поле, выбранном в качестве слоя.
size_type	*логическое*	Указывает, что ко всем записям геопространственного поля применяться стандартный размер (standard), либо задает размер наложения (overlay), изменяющий размер значка обозначения или толщину линий на основе значений из другого поля в наборе данных. Возможные значения: standard или overlay. По умолчанию: standard.
размер	*строка*	Если для свойства size_type выбрано значение standard, для point или multipoint выпадающее меню будет содержать варианты размеров для выбранного обозначения. Для linestring или multilinestring выпадающее меню будет содержать варианты толщины линий.
size_field	*поле*	Если для свойства size_type выбрано значение overlay, выпадающее меню будет содержать все поля из того же набора данных, что и в геопространственном поле, выбранном в качестве слоя.

Таблица 102. Свойства mapvisualization (продолжение)

Свойства mapvisualization	**Тип переменной**	**Описание свойства**
transp_type	*логическое*	Указывает, что ко всем записям геопространственного поля применяться стандартная прозрачность (standard), либо задает прозрачность наложения (overlay), изменяющая уровень прозрачности для обозначения, линии или многоугольника на основе значений из другого поля в наборе данных. Возможные значения: standard или overlay. По умолчанию: standard.
transp	*целое*	Если для свойства transp_type выбрано значение standard, выпадающее меню будет содержать варианты уровней прозрачности от 0% (непроницаемый) до 100% (прозрачный) с шагом 10%. Задает прозрачность точек, линий или многоугольников на карте. Если для свойства size_type выбрано значение overlay, выпадающее меню будет содержать все поля из того же набора данных, что и в геопространственном поле, выбранном в качестве слоя. Для points, multipoints, linestrings и multilinestrings, polygons и multipolygons значение по умолчанию - 0%. Для polygons и multipolygons, находящихся не в нижнем слое, значение по умолчанию - 50% (во избежание затенения слоев под этими многоугольниками).
transp_field	*поле*	Если для свойства transp_type выбрано значение overlay, выпадающее меню будет содержать все поля из того же набора данных, что и в геопространственном поле, выбранном в качестве слоя.
data_label_field	*поле*	Задает поле для использования в качестве меток данных на карте. Например, слой, к которому применяется этот параметр, представляет собой слой многоугольников, метка данных может быть полем имен (name), содержащим название для каждого многоугольника. Поэтому выбор здесь поля имен (name) приводит к выводу соответствующих названий на карте.

Таблица 102. Свойства mapvisualization (продолжение)

Свойства mapvisualization	**Тип переменной**	**Описание свойства**
use_hex_binning	*логическое*	Включает группировку шестиугольниками и поддержку всех выпадающих меню агрегации. Этот параметр выключен по умолчанию.
color_aggregation и transp_aggregation	*строка*	Если выбрано поле наложения для слоя точек, где используется группировка шестиугольниками, все значения для этого поля должны агрегироваться для всех точек в составе шестиугольника. Поэтому нужно задать функцию агрегирования для любых полей наложения, которые вы хотите применить к карте. Доступные функции агрегации: Количественные (система хранения Действительное или Целое число): • Сумма • Среднее значение • Минимум • Максимум • Медиана • 1-я квартиль • 3-я квартиль Количественные (система хранения Время, Дата или Отметка времени): • Среднее значение • Минимум • Максимум Nominal/Categorical: • Мода • Минимум • Максимум Флаг: • True, если любое - true • False, если любое - false
custom_storage	*строка*	Задает общий тип хранения поля. Значение по умолчанию - List (список). Если задано значение List, выводятся следующие элементы управления custom_value_storage и list_depth.

Таблица 102. Свойства mapvisualization (продолжение)		
Свойства mapvisualization	**Тип переменной**	**Описание свойства**
custom_value_storage	*строка*	Задает типы хранения элементов в списке вместо задания типа для поля в целом. Значение по умолчанию - Real.
list_depth	*целое*	Задает глубину поля списка. Требуемая глубина зависит от типа геопространственного поля с учетом следующих критериев: • Точка - 0 • Ломаная – 1 • Многоугольник - 2 • Несколько точек – 1 • Мультиломаная – 2 • Мультиполигон - 3 Вы должны знать тип геопространственного поля, преобразуемого обратно в список, и нужную глубину поля такого сорта. Если задать его неправильно, использовать поле будет нельзя. Значение по умолчанию: 0, минимум 0 и максимум 10.

Свойства узла Несколько графиков (multiplotnode)

Узел нескольких графиков (Multiplot) создает график, выводящий несколько полей *Y* по отношению к одному полю *X*. Значения полей *Y* изображаются на графике как цветные линии, каждая из которых эквивалентна графику на узле График при заданном значении стиля **Линия** и режиме X **Сортировка**. Узел Несколько графиков полезен, когда нужно исследовать флуктуации нескольких переменных во времени.

Пример

```
node = stream.create("multiplot", "My node")
# Вкладка "График"
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# Раздел "Наложение"
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")
```

Таблица 103. Свойства multiplotnode		
Свойства multiplotnode	**Тип переменной**	**Описание свойства**
x_field	*поле*	
y_fields	*список*	
panel_field	*поле*	
animation_field	*поле*	
normalize	*флаг*	
use_overlay_expr	*флаг*	
overlay_expression	*string*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*флаг*	
x_label	*string*	
y_label_auto	*флаг*	
y_label	*string*	
use_grid	*флаг*	
graph_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
page_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.

Свойства plotnode

Узел График показывает взаимосвязь между численными полями. Графики можно создавать, используя точки (диаграммы рассеяния) или линии.

Пример

```
node = stream.create("plot", "My node")
# Вкладка "График"
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# Раздел "Наложение"
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")
```

```
# Вкладка "Выходные данные"
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/
plot_output.jpeg")
```

Таблица 104. Свойства plotnode

Свойства plotnode	**Тип переменной**	**Описание свойства**
x_field	*поле*	Задает пользовательскую метку для оси *x*. Доступно только для меток.
y_field	*поле*	Задает пользовательскую метку для оси *y*. Доступно только для меток.
three_D	*флаг*	Задает пользовательскую метку для оси *y*. Доступно только для меток трехмерных графиков.
z_field	*поле*	
color_field	*поле*	Поле наложения.
size_field	*поле*	
shape_field	*поле*	
panel_field	*поле*	Задает номинальное или флаговое поле для использования при построении отдельной диаграммы для каждой категории. Диаграммы располагаются совместно в одном выходном окне.
animation_field	*поле*	Задает номинальное или флаговое поле для иллюстрации категорий значений данных посредством создания набора диаграмм, показываемых последовательно с использованием анимации.
transp_field	*поле*	Задает номинальное или флаговое поле для иллюстрации категорий значений данных посредством различных уровней прозрачности для каждой категории. Недоступно для линейных графиков.
overlay_type	Нет Сглаживатель Function	Задает, что будет выводиться, функция наложения или сглаживатель LOESS.
overlay_expression	*string*	Задает выражение, которое используется, когда для overlay_type задано значение Function.
style	Точка Line	

Таблица 104. Свойства plotnode (продолжение)		
Свойства plotnode	**Тип переменной**	**Описание свойства**
point_type	Прямоугольник Точки Треугольник Шестиугольник Плюс Пятиугольник Звезда BowTie HorizontalDash VerticalDash IronCross Фабрика Дом Собор OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Beep	
x_mode	Сортировка Наложение AsRead	
x_range_mode	Автоматически UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Автоматически UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Автоматически UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*флаг*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	

Таблица 104. Свойства plotnode (продолжение)		
Свойства plotnode	**Тип переменной**	**Описание свойства**
x_label_auto	*флаг*	
x_label	*string*	
y_label_auto	*флаг*	
y_label	*string*	
z_label_auto	*флаг*	
z_label	*string*	
use_grid	*флаг*	
graph_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
page_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
use_overlay_expr	*флаг*	Объявлено устаревшим с заменой на overlay_type.

Свойства узла График зависимости от времени (timeplotnode)

Узел Временной график выводит один или несколько наборов данных временных рядов. Обычно вы сначала используете узел Временные интервалы для создания поля *TimeLabel*, которое будет использовано для отметок по оси *x*.

Пример

```
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)
# Параметры представления
node.setPropertyValue("symbol_size", 2.0)
```

Таблица 105. Свойства timeplotnode		
Свойства timeplotnode	**Тип переменной**	**Описание свойства**
plot_series	Ряды Модели	
use_custom_x_field	*флаг*	
x_field	*поле*	
y_fields	*список*	
panel	*флаг*	
normalize	*флаг*	

Таблица 105. Свойства timeplotnode (продолжение)		
Свойства timeplotnode	**Тип переменной**	**Описание свойства**
line	*флаг*	
points	*флаг*	
point_type	Прямоугольник Точки Треугольник Шестиугольник Плюс Пятиугольник Звезда BowTie HorizontalDash VerticalDash IronCross Фабрика Дом Собор OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Beep	
smoother	*флаг*	Добавить к графику сглаживатели можно только в том случае, если для panel вы задали значение True.
use_records_limit	*флаг*	
records_limit	*целое*	
symbol_size	*number*	Задает размер знака.
panel_layout	Горизонтально Вертикальный	

Свойства eplotnode

Узел Электронный график (бета-версия) показывает взаимосвязь между числовыми полями. Он подобен узлу График, но его опции отличаются, а вывод использует новый графический интерфейс, предназначенный для этого узла. Используйте бета-версию узла, чтобы попробовать новые возможности построения графиков.

Таблица 106. Свойства eplotnode		
Свойства eplotnode	**Тип переменной**	**Описание свойства**
x_field	*строка*	Выберите поле для вывода по горизонтальной оси X.
y_field	*строка*	Выберите поле для вывода по вертикальной оси Y.
color_field	*строка*	Задайте поле, чтобы при желании использовать в наложении карты цветов.

Таблица 106. Свойства eplotnode (продолжение)		
Свойства eplotnode	**Тип переменной**	**Описание свойства**
size_field	*строка*	Задайте поле, чтобы при желании использовать в наложении карты размера.
shape_field	*строка*	Задайте поле, чтобы при желании использовать в наложении карты форм.
interested_fields	*строка*	Укажите поля, которые вы хотите включить в вывод.
records_limit	*целое*	Задайте число - максимальное количество записей для графика в выводе. 2000 - значение по умолчанию.
if_over_limit	*Логический*	Задайте, использовать ли опцию Выборка или Использовать все данные, если число записей превосходит records_limit. Выборка предполагается по умолчанию, при этом данные выбираются случайным образом, пока не будет достигнут records_limit. Если вы задаете Использовать все данные, чтобы проигнорировать records_limit и нанести на график все точки данных, учтите, что это может существенно снизить производительность.

Свойства tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) - инструмент для визуализации многомерных данных. Он преобразует аффинитеты точек данных в вероятности. Этот узел t-SNE SPSS Modeler реализован на языке Python, и для него требуется библиотека Python scikit-learn©.

Таблица 107. Свойства tsnenode		
Свойства tsnenode	**Тип переменной**	**Описание свойства**
mode_type	*строка*	Задайте режим - simple или expert.
n_components	*строка*	Размерность встроенного пространства (2D или 3D). Задайте 2 или 3. Значение по умолчанию - 2.
method	*строка*	Задайте barnes_hut или exact. Значение по умолчанию - barnes_hut.
init	*строка*	Инициализация встраивания. Задайте random или pca. Значение по умолчанию - random.

Таблица 107. Свойства tsnenode (продолжение)

Свойства tsnenode	**Тип переменной**	**Описание свойства**
target_field Переименовано в target, начиная с версии 18.2.1.1	*строка*	Имя поля назначения. Это будет цветовая карта выходного изображения. Если поле назначения не указано, в изображении будет использоваться один цвет.
perplexity	*плавающее*	Перплексия связана с числом ближайших соседей, используемых в других алгоритмах множественного обучения. Для больших наборов данных обычно требуется большая перплексия. Обдумайте выбор значения от 5 до 50. Значение по умолчанию - 30.
early_exaggeration	*плавающее*	Управляет тем, как тесно естественные кластеры в исходном пространстве находятся во вложенном пространстве, и сколько места будет между ними. Значение по умолчанию - 12,0.
learning_rate	*плавающее*	Значение по умолчанию - 200.
n_iter	*целое*	Максимальное число итераций для оптимизации. Задайте не меньше 250. Значение по умолчанию - 1000.
угол	*плавающее*	Угловой размер удаленного узла, измеренный с точки. Задайте значение в диапазоне от 0 до 1. Значение по умолчанию - 0,5.
enable_random_seed	*Логический*	Задайте true, чтобы включить параметр random_seed. Значение по умолчанию - false.
random_seed	*целое*	Используемое начальное значение для генератора псевдослучайных чисел. Значение по умолчанию - None.
n_iter_without_progress	*целое*	Максимальное число итераций, не дающих прогресса. Значение по умолчанию - 300.

Таблица 107. Свойства tsnenode (продолжение)		
Свойства tsnenode	**Тип переменной**	**Описание свойства**
min_grad_norm	*строка*	Если норма градиента ниже этого порога, оптимизация будет остановлена. Значение по умолчанию - 1.0E-7. Возможные значения: • 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	*Логический*	Задайте true, чтобы выполнить t-SNE с несколькими различными значениями перплексии. Значение по умолчанию - false.
output_Rename	*Логический*	Задайте true, если вы хотите указать пользовательское имя, или false, чтобы назвать вывод автоматически. Значение по умолчанию - false.
output_to	*строка*	Задайте Screen или Output. Значение по умолчанию - Screen.
full_filename	*строка*	Задайте имя файла вывода.
output_file_type	*строка*	Формат файл вывода. Задайте HTML или Output object. Значение по умолчанию - HTML.

Свойства узла Web (webnode)

Узел Web иллюстрирует силу взаимосвязи между значениями двух или более символических (категориальных) полей. На графике используются линии разной ширины для обозначения силы соединения. Например, вы можете использовать узел Web для изучения взаимосвязи между покупкой набора товаров на сайте интернет-магазина.

Пример

```
node = stream.create("web", "My node")
# Вкладка "График"
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)
# Вкладка "Опции"
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")
```

Таблица 108. Свойства webnode

Свойства webnode	**Тип переменной**	**Описание свойства**
use_directed_web	*флаг*	
fields	*список*	
to_field	*поле*	
from_fields	*список*	
true_flags_only	*флаг*	
line_values	Абсолютная OverallPct PctLarger PctSmaller	
strong_links_heavier	*флаг*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*number*	
links_above	*number*	
discard_links_min	*флаг*	
links_min_records	*число*	
discard_links_max	*флаг*	
links_max_records	*число*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*флаг*	

Таблица 108. Свойства webnode (продолжение)

Свойства webnode	**Тип переменной**	**Описание свойства**
web_display	Circular Network Directed Сетка	
graph_background	*цвет*	Стандартные цвета графиков описаны в начале этого раздела.
symbol_size	*number*	Задает размер знака.

Глава 13. Свойства узла моделирования

Общие свойства узлов моделирования

Следующие свойства общие для всех или некоторых узлов моделирования. Любые исключительные ситуации соответствующим образом отмечаются в документации для индивидуальных узлов моделирования.

Таблица 109. Общие свойства узлов моделирования

Свойство	Значения	Описание свойства
custom_fields	*флаг*	Если значение флага true, это позволяет вам задать поле назначения, входные и другие поля для текущего узла. При значении false используются текущие параметры с вышележащего узла Тип.
target или назначения	*поле* или [*поле1 ... полеN*]	Задает одно поле назначения или несколько полей назначения в зависимости от типа модели.
inputs	[*поле1 ... полеN*]	Входные (или предикторные) поля, используемые в модели.
partition	*поле*	
use_partitioned_data	*флаг*	Если определено поле раздела, эта опция обеспечивает, что для построения модели используются только данные из раздела обучения.
use_split_data	*флаг*	
splits	*[поле1 ... полеN]*	Задает поле или поля, которые будут использоваться для моделирования разбиения. Действует только в том случае, если для use_split_data задано значение True.
use_frequency	*флаг*	Поля веса и частоты используются конкретными моделями, как отмечено для каждого типа модели.
frequency_field	*поле*	
use_weight	*флаг*	
weight_field	*поле*	
use_model_name	*флаг*	
model_name	*string*	Пользовательское имя для новой модели.

Таблица 109. Общие свойства узлов моделирования (продолжение)		
Свойство	**Значения**	**Описание свойства**
mode	Простые Эксперт	

Свойства anomalydetectionnode

Узел выявления аномалий определяет необычные наблюдения, или выбросы, которые не соответствуют структуре “нормальных” данных. При помощи этого узла можно находить выбросы даже в том случае, если они не подходят ни под какие ранее известные шаблоны или вы точно не уверены, что именно ищете.

Пример

```
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)
```

Таблица 110. Свойства anomalydetectionnode		
Свойства anomalydetectionnode	**Значения**	**Описание свойства**
inputs	*[поле1 ... полеN]*	Модели обнаружения аномалий изучают данные на основе заданных входных полей. Они не используют поле назначения. Поля веса и частоты также не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
mode	Эксперт Простые	
anomaly_method	IndexLevel PerRecords NumRecords	Задает способ определения значения отсечения для пометки записей как аномальных.
index_level	*число*	Задает минимальное значение отсечения для пометки аномалий.
percent_records	*число*	Задает порог для отметки записей на основании процентной доли записей в данных обучения.

Таблица 110. Свойства anomalydetectionnode (продолжение)

Свойства anomalydetectionnode	**Значения**	**Описание свойства**
num_records	*number*	Задает порог для отметки записей на основании количества записей в данных обучения.
num_fields	*целое*	Количество полей для отчета о каждой аномальной записи.
impute_missing_values	*флаг*	
adjustment_coeff	*number*	Значение для балансировки относительного веса, предназначенное для количественных и категориальных полей при вычислении расстояния.
peer_group_num_auto	*флаг*	Автоматически вычисляет количество равноправных групп.
min_num_peer_groups	*целое*	Задает минимальное количество равноправных групп, используемых, если для поля peer_group_num_auto задано значение True.
max_num_per_groups	*целое*	Задает максимальное количество равноправных групп.
num_peer_groups	*целое*	Задает минимальное количество равноправных групп, используемых, если для поля peer_group_num_auto задано значение False.
noise_level	*number*	Определяет, как при кластеризации обрабатываются выбросы. Укажите значение от 0 до 0,5.
noise_ratio	*number*	Определяет часть памяти, выделенной для компонента, который должен использоваться для буферизации шума. Укажите значение от 0 до 0,5.

Свойства apriorinode

Узел Априори извлекает набор правил из данных, выделяя правила с наибольшим информационным содержимым. Узел Априори предлагает пять различных способов выбора правил и использует сложные схемы индексирования для эффективной обработки больших наборов данных. Для больших задач узел Априори обычно быстрее при обучении; у него нет произвольного ограничения количества правил, которые можно сохранить, и он может обрабатывать правила с количеством предварительных условий до 32. Для узла Априори требуются категориальные входные и выходные поля, он был оптимизирован для полей такого типа и показывает с ними высокую производительность.

Пример

```
node = stream.create("apriori", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# Для нетранзакционных
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# Для транзакционных
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# Вкладка "Модель"
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")
# Вкладка "Эксперт"
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Таблица 111. Свойства apriorinode

Свойства apriorinode	Значения	Описание свойства
consequents	*поле*	Априорные модели используют консеквенты и антецеденты вместо стандартных полей назначения и входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
antecedents	[*поле1 ... полеN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	

Таблица 111. Свойства apriorinode (продолжение)		
Свойства apriorinode	**Значения**	**Описание свойства**
true_flags	*флаг*	
optimize	Скорость Память	
use_transactional_data	*флаг*	Если значение равно true, оценка для каждого ID транзакции не зависит от других ID транзакций. Когда данные для скоринга слишком велики, чтобы получить приемлемую производительность, мы рекомендуем разделить данные.
contiguous	*флаг*	
id_field	*string*	
content_field	*string*	
mode	Простые Эксперт	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Скорость Память	Используется для определения, нужно ли при построении модели оптимизировать скорость или использование памяти.

Свойства associationrulesnode

Узел Правила связывания аналогичен узлу Априори, однако в отличие от Априори узел Правила связывания может обрабатывать данные списков. Кроме того, узел Правила связывания можно использовать с IBM SPSS Analytic Server для обработки данных большого объема, получая преимущества более быстрой параллельной обработкой.

Таблица 112. свойства associationrulesnode		
Свойства associationrulesnode	**Тип переменной**	**Описание свойства**
предсказания	*поле*	Поля в этом списке могут появляться только в качестве предиктора правила.
условия	*[поле_1...поле_N]*	Поля в этом списке могут появляться только в качестве условия правила.
max_rule_conditions	*целое*	Максимальное количество условий, которые можно включить в одно правило. Минимум 1, максимум 9.
max_rule_predictions	*целое*	Максимальное количество предсказаний, которые можно включить в одно правило. Минимум 1, максимум 5.
max_num_rules	*целое*	Максимальное число правил, которые могут рассматриваться в составе операции построения правил. Минимум 1, максимум 10000.
rule_criterion_top_n	Показатель доверия Rulesupport Рост Conditionsupport Deployability	Критерий правила, определяющий значение, по которому выбираются первые "N" правил в модели.
true_flags	*Логический*	Задание значения *Y* определяет, что при построении правил для полей флагов будут учитываться только значения true, рассматриваются.
rule_criterion	*Логический*	Задание значения *Y* определяет, что при построении модели для правил исключения будут учитываться только значения критерия правил.
min_confidence	*number*	От 0,1 до 100 - процентное значение достоверности в качестве минимально необходимого доверительного уровня для правила, генерируемого моделью. Если модель сгенерирует правило с уровнем конфиденциальности меньше заданного здесь значения, правило будет отброшено.
min_rule_support	*number*	От 0,1 до 100 - процентное значение достоверности в качестве минимально необходимой поддержки правила для правила, генерируемого моделью. Если модель сгенерирует правило с уровнем поддержки правила меньше заданного значения, правило будет отброшено.

Таблица 112. свойства associationrulesnode (продолжение)

Свойства associationrulesnode	**Тип переменной**	**Описание свойства**
min_condition_support	*number*	От 0,1 до 100 - процентное значение достоверности в качестве минимально необходимой поддержки условия для правила, генерируемого моделью. Если модель сгенерирует правило с уровнем поддержки условия меньше заданного значения, правило будет отброшено.
min_lift	*целое*	Значение от 1 до 10 представляет минимально необходимое значение роста для правила, генерируемого моделью. Если модель сгенерирует правило с уровнем роста меньше заданного значения, правило будет отброшено.
exclude_rules	*Логический*	Используется для выбора списка связанных полей, из которых вы не хотите создания моделью правил. Например, set :gsarsnode.exclude_rules = [[[поле1,поле2, поле3]],[[поле4, поле5]]] - здесь каждый список выделенных скобками [] полей - это строка в таблице.
num_bins	*целое*	Задайте число интервалов автоматического разделения непрерывных полей. Минимум 2, максимум 10.
max_list_length	*целое*	Применяется ко всем полям, максимальная длина которых неизвестна. Элементы в списке до указанного здесь числа включаются в сборку модели; все последующие элементы отбрасываются. Минимум 1, максимум 100.
output_confidence	*Логический*	
output_rule_support	*Логический*	
output_lift	*Логический*	
output_condition_support	*Логический*	
output_deployability	*Логический*	
rules_to_display	upto все	Максимальное число правил, выводимых в выходных таблицах.
display_upto	*целое*	Если в качестве upto задано rules_to_display, задайте число правил, выводимых в выходных таблицах. Минимум - 1.

Таблица 112. свойства associationrulesnode (продолжение)		
Свойства associationrulesnode	**Тип переменной**	**Описание свойства**
field_transformations	*Логический*	
records_summary	*Логический*	
rule_statistics	*Логический*	
most_frequent_values	*Логический*	
most_frequent_fields	*Логический*	
word_cloud	*Логический*	
word_cloud_sort	Показатель доверия Rulesupport Рост Conditionsupport Deployability	
word_cloud_display	*целое*	Минимум 1, максимум 20
max_predictions	*целое*	Максимальное число правил, которые могут быть применены к каждому элементу ввода в скоринг.
criterion	Показатель доверия Rulesupport Рост Conditionsupport Deployability	Выберите показатель для определения силы правил.
allow_repeats	*Логический*	Определите, включать ли в скоринг правила с одинаковым предсказанием.
check_input	NoPredictions Predictions NoCheck	

Свойства autoclassifiernode

Узел автоклассификации создает и сравнивает несколько различных моделей для двоичных выходных данных (да или нет, уйдет клиент или останется и так далее), что позволяет выбрать лучший подход для данного анализа. Поддерживается несколько алгоритмов моделирования, что делает возможным выбор желательных для использования способов, конкретных опций для каждого из них и критериев сравнения результатов. Этот узел генерирует набор моделей на основе заданных опций и ранжирует лучших кандидатов в соответствии с заданными вами критериями.

Пример

```
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)
```

Таблица 113. Свойства autoclassifiernode

Свойства autoclassifiernode	**Значения**	**Описание свойства**
target	*поле*	Для флаговых полей назначения узлу автоклассификации требуется одно поле назначения и одно или несколько входных полей. Можно задать также поля веса и частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
ranking_measure	Точность Area_under_curve Доход Рост Num_variables	
ranking_dataset	Обучающее Критерий	
number_of_models	*целое*	Количество моделей для включения в слепок моделей. Указать целое число от 1 до 100.
calculate_variable_importance	*флаг*	
enable_accuracy_limit	*флаг*	

Таблица 113. Свойства autoclassifiernode (продолжение)

Свойства autoclassifiernode	**Значения**	**Описание свойства**
accuracy_limit	*целое*	Целое число от 0 до 100.
enable_ area_under_curve _limit	*флаг*	
area_under_curve_limit	*number*	Действительное число от 0,0 до 1,0.
enable_profit_limit	*флаг*	
profit_limit	*number*	Целое число больше 0.
enable_lift_limit	*флаг*	
lift_limit	*number*	Действительное число, большее 1,0.
enable_number_of_variables_lim it	*флаг*	
number_of_variables_limit	*number*	Целое число больше 0.
use_fixed_cost	*флаг*	
fixed_cost	*число*	Действительное число, большее 0,0.
variable_cost	*поле*	
use_fixed_revenue	*флаг*	
fixed_revenue	*число*	Действительное число, большее 0,0.
variable_revenue	*поле*	
use_fixed_weight	*флаг*	
fixed_weight	*number*	Действительное число, большее 0,0
variable_weight	*поле*	
lift_percentile	*number*	Целое число от 0 до 100.
enable_model_build_time_limit	*флаг*	
model_build_time_limit	*number*	Целое число, заданное для числа минут, чтобы ограничить время на построение каждой конкретной модели.
enable_stop_after_time_limit	*флаг*	
stop_after_time_limit	*number*	Действительное число, заданное для количества часов для ограничения общего используемого времени на выполнение автоклассификации.
enable_stop_after_valid_model_ produced	*флаг*	

Таблица 113. Свойства autoclassifiernode (продолжение)		
Свойства autoclassifiernode	**Значения**	**Описание свойства**
use_costs	*флаг*	
<алгоритм>	*флаг*	Включает или отключает использование конкретного алгоритма.
<алгоритм>.<свойство>	*string*	Задает значение свойства для конкретного алгоритма. Дополнительную информацию смотрите в разделе “Задание свойств алгоритмов” на стр. 239.

Задание свойств алгоритмов

Для узлов автоклассификации, автонумерации и автокластеризации свойства конкретных используемых узлом алгоритмов можно задать с использованием общей формы:

```
autonode.setKeyedPropertyValue(<алгоритм>, <свойство>, <значение>)
```

Например:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Имена алгоритмов для узла автоклассификации - это cart, chaid, quest, c50, logreg, decisionlist, bayesnet, discriminant, svm и knn.

Имена алгоритмов для узла автонумерации - это cart, chaid, neuralnetwork, genlin, svm, regression, linear и knn.

Имена алгоритмов для узла автокластеризации - это twostep, k-means и kohonen.

Имена свойств стандартные, как документировано для каждого узла алгоритмов.

Свойства алгоритмов, содержащие точки или другие знаки пунктуации, должны заключаться в одинарные кавычки, например:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

Для свойства можно назначить также несколько значений, например:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up",
"Down"])
```

Чтобы включить или отключить использование конкретного алгоритма:

```
node.setPropertyValue("chaid", True)
```

Прим.: В тех случаях, когда некоторые опции алгоритмов недоступны на узле автоклассификации, или можно задать только одно значение, а не диапазон значений, такие же ограничения применяются к сценариям и при обращении к узлу обычным образом.

Свойства autoclusternode

Узел автоматической кластеризации оценивает и сравнивает модели кластеризации, идентифицирующие группы записей со сходными характеристиками. Этот узел работает аналогично другим узлам автоматического моделирования, допуская экспериментирование с несколькими комбинациями опций при одном проходе моделирования. Модели можно сравнивать при помощи базовых показателей, пытаясь фильтровать и ранжировать с их использованием полезность моделей кластеризации и предоставить показатель на основе важности конкретных полей.

Пример

```
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)
```

Таблица 114. Свойства autoclusternode

Свойства autoclusternode	Значения	Описание свойства
evaluation	*поле*	**Прим.:** Только для узла автокластеризации. Определяет поле, для которого будет вычислено значение важности. Как вариант, может использоваться для определения, насколько хорошо кластер различает значение этого поля, и как следствие - насколько хорошо модель предскажет значение в этом поле.
ranking_measure	Силуэтная мера Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Важность	
ranking_dataset	Обучающее Критерий	
summary_limit	*целое*	Количество моделей для перечисления в отчете. Указать целое число от 1 до 100.
enable_silhouette_limit	*флаг*	

Таблица 114. Свойства autoclusternode (продолжение)

Свойства autoclusternode	Значения	Описание свойства
silhouette_limit	*целое*	Целое число от 0 до 100.
enable_number_less_limit	*флаг*	
number_less_limit	*число*	Действительное число от 0,0 до 1,0.
enable_number_greater_limit	*флаг*	
number_greater_limit	*число*	Целое число больше 0.
enable_smallest_cluster_limit	*флаг*	
smallest_cluster_units	Процент Количества	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*целое*	Целое число больше 0.
enable_largest_cluster_limit	*флаг*	
largest_cluster_units	Процент Количества	
largest_cluster_limit_percentage	*number*	
largest_cluster_limit_count	*целое*	
enable_smallest_largest_limit	*флаг*	
smallest_largest_limit	*number*	
enable_importance_limit	*флаг*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	Целое число от 0 до 100.
importance_limit_less_than	*number*	Целое число от 0 до 100.
<алгоритм>	*флаг*	Включает или отключает использование конкретного алгоритма.

Таблица 114. Свойства autoclusternode (продолжение)		
Свойства autoclusternode	**Значения**	**Описание свойства**
<алгоритм>.<свойство>	*string*	Задает значение свойства для конкретного алгоритма. Дополнительную информацию смотрите в разделе “Задание свойств алгоритмов” на стр. 239.

Свойства autonumericnode

Узел автонумерации оценивает и сравнивает модели для выходных данных в количественном числовом диапазоне при помощи нескольких разных способов. Этот узел работает аналогично другим узлам автоклассификации, допуская выбор алгоритмов для использования и экспериментирование с несколькими комбинациями опций при одном проходе моделирования. Поддерживаемые алгоритмы включают в себя нейросети, дерево C&R, CHAID, линейную регрессию, обобщенную линейную регрессию и механизмы опорных векторов (support vector machines, SVM). Модели можно сравнивать на основе корреляции, относительной ошибки или числа используемых переменных.

Пример

```
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)
```

Таблица 115. Свойства autonumericnode		
Свойства autonumericnode	**Значения**	**Описание свойства**
custom_fields	*флаг*	При значении True вместо параметров узла Тип будут использоваться пользовательские параметры полей.
target	*поле*	Для узла автонумерации требуется одно поле назначения и одно или несколько входных полей. Можно задать также поля веса и частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
inputs	*[поле1 ... поле2]*	
partition	*поле*	
use_frequency	*флаг*	
frequency_field	*поле*	
use_weight	*флаг*	

Таблица 115. Свойства autonumericnode (продолжение)

Свойства autonumericnode	Значения	Описание свойства
weight_field	*поле*	
use_partitioned_data	*флаг*	Если определено поле раздела, для построения модели используются только данные из раздела обучения.
ranking_measure	Корреляция NumberOfFields	
ranking_dataset	Критерий Обучающее	
number_of_models	*целое*	Количество моделей для включения в слепок моделей. Указать целое число от 1 до 100.
calculate_variable_importance	*флаг*	
enable_correlation_limit	*флаг*	
correlation_limit	*целое*	
enable_number_of_fields_limit	*флаг*	
number_of_fields_limit	*целое*	
enable_relative_error_limit	*флаг*	
relative_error_limit	*целое*	
enable_model_build_time_limit	*флаг*	
model_build_time_limit	*целое*	
enable_stop_after_time_limit	*флаг*	
stop_after_time_limit	*целое*	
stop_if_valid_model	*флаг*	
<алгоритм>	*флаг*	Включает или отключает использование конкретного алгоритма.
<алгоритм>.<свойство>	*string*	Задает значение свойства для конкретного алгоритма. Дополнительную информацию смотрите в разделе “Задание свойств алгоритмов” на стр. 239.

Свойства bayesnetnode

Узел Байесовская сеть позволяет построить вероятностную модель, комбинируя наблюдаемые и записанные сведения с очевидными с точки зрения здравого смысла данными, чтобы установить правдоподобие возникновения событий. Этот узел в основном работает с усиленными деревом наивными байесовскими сетями (Tree Augmented Naïve Bayes, TAN) и полными марковскими сетями, которые изначально используются для классификации.

Пример

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Вкладка Эксперт
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Таблица 116. Свойства bayesnetnode

Свойства bayesnetnode	Значения	Описание свойства
inputs	*[поле1 ... полеN]*	Модели Байесовской сети используют одно поле назначения и одно или несколько входных полей. Количественные поля автоматически разбиваются на интервалы. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue_training_existing_model	*флаг*	
structure_type	TAN MarkovBlanket	Выберите структуру, которая будет использоваться при построении Байесовской сети.
use_feature_selection	*флаг*	
parameter_learning_method	Вероятность Bayes	Задает способ, используемый для оценки таблиц условной вероятности между узлами, когда родительские элементы известны.
mode	Эксперт Простые	
missing_values	*флаг*	
all_probabilities	*флаг*	
independence	Вероятность Пирсона	Задает способ, используемый для определения, независимы ли друг от друга парные наблюдения двух переменных.

Таблица 116. Свойства bayesnetnode (продолжение)		
Свойства bayesnetnode	**Значения**	**Описание свойства**
significance_level	*number*	Задает значение отсечения для определения независимости.
maximal_conditioning_set	*number*	Задает максимальное количество переменных настройки, которые будут использоваться для проверки независимости.
inputs_always_selected	*[поле1 ... полеN]*	Задает, какие поля из набора данных всегда будут использоваться при построении Байесовской сети. **Прим.:** Поле назначения всегда выбрано.
maximum_number_inputs	*number*	Задает максимальное количество входных полей, которые будут использоваться при построении Байесовской сети.
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

свойства buildr

Узел построения R позволяет ввести пользовательский сценарий R для выполнения построения и скоринга модели, внедренной в IBM SPSS Modeler.

Пример

```
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)""")
```

Таблица 117. свойства buildr		
Свойства buildr	**Значения**	**Описание свойства**
build_syntax	*string*	Синтаксис сценария R для построения моделей.

Таблица 117. свойства buildr (продолжение)		
Свойства buildr	**Значения**	**Описание свойства**
score_syntax	*строка*	Синтаксис сценария R для скоринга моделей.
convert_flags	StringsAndDoubles LogicalValues	Опция для преобразования флаговых полей.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	POSIXct POSIXlt	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
output_html	*флаг*	Опция для вывода графиков на вкладке в слепке модели R.
output_text	*флаг*	Опция для записи текстового вывода консоли R на вкладку в слепке модели R.

Свойства c50node

Узел C5.0 строит или дерево решений, или набор правил. Эта модель работает, разделяя выборку на основании значения в поле, дающего максимальный информационный выигрыш на каждом уровне. Поле назначения должно быть категориальным. Разрешено несколько разделений на подгруппы, и таких подгрупп может быть больше двух.

Пример

```
node = stream.create("c50", "My node")
# Вкладка "Модель"
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)
# Вкладка "Стоимости"
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Таблица 118. Свойства c50node

Свойства c50node	Значения	Описание свойства
target	*поле*	Модели C50 используют одно поле назначения и одно или несколько входных полей. Можно задать также поле веса. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
output_type	DecisionTree RuleSet	
group_symbolics	*флаг*	
use_boost	*флаг*	
boost_num_trials	*число*	
use_xval	*флаг*	
xval_num_folds	*number*	
mode	Простые Эксперт	
favor	Точность Общность	Предпочтение точности или общности.
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*флаг*	
costs	*структурированный*	Это - структурированное свойство.
use_winnowing	*флаг*	
use_global_pruning	*флаг*	По умолчанию значение (True).
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства carmanode

Модель CARMA извлекает из данных набор правил, не требуя, чтобы вы задавали входные или выходные поля. В отличие от узла Априори, узел CARMA предлагает параметры построения для поддержки правил (поддержка относится и к антецедентам, и к консеквентам), а не только для поддержки антецедентов. Это означает, что сгенерированные правила можно использовать в более широком наборе прикладных программ, например, чтобы найти список продуктов или услуг (антецедентов), консеквент которых - это товар, который вы хотите продвигать в этом летнем сезоне.

Пример

```
node = stream.create("carma", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# Вкладка "Модель"
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)
# Опции эксперта
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)
```

Таблица 119. Свойства carmanode

Свойства carmanode	**Значения**	**Описание свойства**
inputs	*[поле1 ... fieldn]*	Модели CARMA используют список входных полей, но не поля назначения. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
id_field	*поле*	Поле, используемое в качестве поля ID для построения модели.
contiguous	*флаг*	Используется для указания, последовательные ли ID содержатся в поле ID.
use_transactional_data	*флаг*	
content_field	*поле*	
min_supp	*число(процент)*	Относится к поддержке правила, а не антецедента. Значение по умолчанию - 20%.
min_conf	*число(процент)*	Значение по умолчанию - 20%.

Таблица 119. Свойства carmanode (продолжение)		
Свойства carmanode	**Значения**	**Описание свойства**
max_size	*number*	Значение по умолчанию - 10.
mode	Простые Эксперт	Значение по умолчанию - Simple.
exclude_multiple	*флаг*	Исключает правила с несколькими консеквентами. Значение по умолчанию: Ложь .
use_pruning	*флаг*	Значение по умолчанию: Ложь .
pruning_value	*number*	Значение по умолчанию - 500.
vary_support	*флаг*	
estimated_transactions	*целое*	
rules_without_antecedents	*флаг*	

Свойства cartnode

Узел дерева классификации и регрессии (Classification and Regression, C&R) генерирует дерево решений, позволяющее предсказывать или классифицировать будущие наблюдения. Этот метод использует рекурсивное разделение, чтобы расщепить обучающие записи на сегменты, на каждом шаге минимизируя неоднородность, причем узел дерева считается “чистым”, если все 100% наблюдений в узле попадают в конкретную категорию поля назначения. Входные поля и поля назначения могут быть из числового диапазона или категориальными (номинальными, порядковыми или флагами); все расщепления бинарны (только две подгруппы).

Пример

```
node = stream.createAt("cart", "My node", 200, 100)
# Вкладка "Поля"
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""")
# Вкладка "Опции сборки", панель "Основные параметры"
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# Вкладка "Опции сборки", панель "Правила остановки"
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# Вкладка "Опции сборки", панель "Дополнительно"
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# Вкладка "Опции модели"
```

```
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Таблица 120. Свойства cartnode

Свойства cartnode	**Значения**	**Описание свойства**
target	*поле*	Моделям дерева C&R требуется одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue_training_existing_model	*флаг*	
objective	Стандартные Бустинг Бэггинг psm	psm используется для очень больших наборов данных и требует соединения с сервером.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*флаг*	
tree_directives	*string*	Задайте директивы роста дерева. Чтобы не использовать символов перехода на новую строку или двойных кавычек, директивы можно заключить в тройные знаки кавычек. Обратите внимание на то, что директивы могут быть очень чувствительны к небольшим изменениям данных или опций моделирования, и их нельзя обобщать на другие наборы данных.
use_max_depth	Default Пользовательские	
max_depth	*целое*	Максимальное количество уровней в дереве, от 0 до 1000. Используется только в том случае, если use_max_depth = Custom.
prune_tree	*флаг*	Отсекать ветви, чтобы избежать переобучения.

Таблица 120. Свойства cartnode (продолжение)

Свойства cartnode	**Значения**	**Описание свойства**
use_std_err	*флаг*	Использовать максимальную разницу в риске (в стандартных ошибках).
std_err_multiplier	*число*	Максимальная разность.
max_surrogates	*число*	Максимум суррогатов.
use_percentage	*флаг*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*флаг*	
costs	*структурированный*	Структурированное свойство.
priors	Данные Равенство Пользовательские	
custom_priors	*структурированный*	Структурированное свойство.
adjust_priors	*флаг*	
trails	*number*	Число моделей компонентов для бустинга или бэггинга.
set_ensemble_method	Голосование HighestProbability HighestMeanProbability	Принятое по умолчанию правило объединения для категориальных целевых полей.
range_ensemble_method	Mean Медиана	Принятое по умолчанию правило объединения для непрерывных целевых полей.
large_boost	*флаг*	Применить бустинг к очень большим наборам данных.
min_impurity	*number*	
impurity_measure	Gini Бинаризация Ordered	
train_pct	*number*	Множество предотвращения сверхобучения.

Таблица 120. Свойства cartnode (продолжение)		
Свойства cartnode	**Значения**	**Описание свойства**
set_random_seed	*флаг*	Опция репликации результатов.
seed	*number*	
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства chaidnode

Узел CHAID генерирует деревья решений, используя статистику хи-квадрат для определения оптимальных расщеплений. В отличие от узлов дерева C&R и QUEST, CHAID может генерировать не только бинарные деревья, то есть у некоторых расщеплений может быть больше двух ветвей. Входные поля и поле назначения могут быть количественными (числовой диапазон) или категориальными. Исчерпывающий CHAID - это модификация метода CHAID, при котором проделывается более тщательная работа по изучению всех возможных расщеплений для каждого предиктора, но это требует больше времени для вычислений.

Пример

```
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)
```

Таблица 121. Свойства chaidnode

Свойства chaidnode	Значения	Описание свойства
target	*поле*	Моделям CHAID требуется одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue_training_existing_model	*флаг*	
objective	Стандартные Бустинг Бэггинг psm	psm используется для очень больших наборов данных и требует соединения с сервером.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*флаг*	
tree_directives	*string*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Пользовательские	
max_depth	*целое*	Максимальное количество уровней в дереве, от 0 до 1000. Используется только в том случае, если use_max_depth = Custom.
use_percentage	*флаг*	
min_parent_records_pc	*число*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*флаг*	
costs	*структурированный*	Структурированное свойство.
trails	*number*	Число моделей компонентов для бустинга или бэггинга.

Таблица 121. Свойства chaidnode (продолжение)		
Свойства chaidnode	**Значения**	**Описание свойства**
set_ensemble_method	Голосование HighestProbability HighestMeanProbability	Принятое по умолчанию правило объединения для категориальных целевых полей.
range_ensemble_method	Mean Медиана	Принятое по умолчанию правило объединения для непрерывных целевых полей.
large_boost	*флаг*	Применить бустинг к очень большим наборам данных.
split_alpha	*number*	Уровень значимости для разбиения.
merge_alpha	*number*	Уровень значимости для слияния.
bonferroni_adjustment	*флаг*	Скорректировать уровни значимости, используя метод Бонферрони.
split_merged_categories	*флаг*	Допускать разбиение объединенных категорий.
chi_square	Пирсона LR	Используемый для вычисления статистики хи-квадрат метод: Пирсона или отношения правдоподобия
epsilon	*number*	Минимальное изменение ожидаемых частот в ячейках.
max_iterations	*число*	Максимум итераций до сходимости.
set_random_seed	*целое*	
seed	*number*	
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	
maximum_number_of_models	*целое*	

Свойства coxregnode

Узел регрессии Кокса позволяет построить модель дожития для данных времени-до-события в присутствии цензурируемых записей. Эта модель создает функцию дожития, которая предсказывает вероятность, что изучаемое событие произойдет в данное время (*t*) для данных значений входных переменных.

Пример

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Вкладка Эксперт
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Таблица 122. Свойства coxregnode

Свойства coxregnode	**Значения**	**Описание свойства**
survival_time	*поле*	Модели регрессии Кокса требуют одного поля, содержащего времена выживания.
target	*поле*	Моделям Кокса требуется одно поле назначения и одно или несколько входных полей. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	Ввод Пошаговый BackwardsStepwise	
groups	*поле*	
model_type	MainEffects Пользовательские	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Эксперт Простые	
max_iterations	*number*	

Таблица 122. Свойства coxregnode (продолжение)		
Свойства coxregnode	**Значения**	**Описание свойства**
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	
removal_criterion	LR Вальда Условное	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*флаг*	

Таблица 122. Свойства coxregnode (продолжение)		
Свойства coxregnode	**Значения**	**Описание свойства**
ci_value	90 95 99	
корреляция	*флаг*	
display_baseline	*флаг*	
survival	*флаг*	
hazard	*флаг*	
log_minus_log	*флаг*	
one_minus_survival	*флаг*	
separate_line	*поле*	
value	*число* или *строка*	Если никакое значение для поля не задано, для него будет использоваться опция по умолчанию "Среднее".

Свойства decisionlistnode

Узел списка решений определяет подгруппы или сегменты, которые показывают более высокое или более низкое правдоподобие для данного бинарного результата по сравнению с полной совокупностью. Например, вы могли бы искать клиентов с низкой вероятностью оттока или с высокой вероятностью отклика на кампанию. Вы можете включить свои знания о бизнесе в модель, добавляя свои собственные пользовательские сегменты и параллельно просматривая альтернативные модели, чтобы сравнить результаты. Модели списка решений состоят из списка правил, в котором каждое правило имеет условие и следствие. Правила применяются по очереди, и первое подходящее правило определяет результат.

Пример

```
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)
```

Таблица 123. Свойства decisionlistnode		
Свойства decisionlistnode	**Значения**	**Описание свойства**
target	*поле*	Модели списка решений используют одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
model_output_type	Модель InteractiveBuilder	
search_direction	Вверх Вниз	Относится к нахождению сегментов; где Вверх - это эквивалент высокой вероятности, а Вниз - низкой.
target_value	*string*	Если не задано, для флагов предполагается значение true.
max_rules	*целое*	Максимальное количество сегментов за исключением остатка.
min_group_size	*целое*	Минимальный размер сегмента.
min_group_size_pct	*number*	Минимальный размер сегмента, выраженный как процентная доля.
confidence_level	*number*	Минимальный порог, насколько входное поле должно повысить правдоподобие отклика (обеспечить подъем), чтобы был смысл добавить определение сегмента.
max_segments_per_rule	*целое*	
mode	Простые Эксперт	
bin_method	EqualWidth EqualCount	
bin_count	*число*	
max_models_per_cycle	*целое*	Ширина поиска для списков.
max_rules_per_cycle	*целое*	Ширина поиска для правил сегментов.
segment_growth	*число*	
include_missing	*флаг*	
final_results_only	*флаг*	
reuse_fields	*флаг*	Позволяет повторное использование атрибутов (входных полей, появляющихся в правилах).

Таблица 123. Свойства decisionlistnode (продолжение)		
Свойства decisionlistnode	**Значения**	**Описание свойства**
max_alternatives	*целое*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства discriminantnode

Дискриминантный анализ делает более строгие предположения, чем логистическая регрессия, но он может быть ценной альтернативой или дополнением к анализу логистической регрессии, когда эти предположения оказываются правильными.

Пример

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Таблица 124. Свойства discriminantnode		
Свойства discriminantnode	**Значения**	**Описание свойства**
target	*поле*	Моделям дискриминанта требуется одно поле назначения и одно или несколько входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	Ввод Пошаговый	
mode	Простые Эксперт	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	

Таблица 124. Свойства discriminantnode (продолжение)		
Свойства discriminantnode	**Значения**	**Описание свойства**
средние значения	*флаг*	Опции статистики в диалоговом окне расширенного вывода.
univariate_anovas	*флаг*	
box_m	*флаг*	
within_group_covariance	*флаг*	
within_groups_correlation	*флаг*	
separate_groups_covariance	*флаг*	
total_covariance	*флаг*	
fishers	*флаг*	
unstandardized	*флаг*	
casewise_results	*флаг*	Опции классификации в диалоговом окне расширенного вывода.
limit_to_first	*number*	Значение по умолчанию - 10.
summary_table	*флаг*	
leave_one_classification	*флаг*	
combined_groups	*флаг*	
separate_groups_covariance	*флаг*	Опция матриц **Ковариации отдельно по группам**.
territorial_map	*флаг*	
combined_groups	*флаг*	Опция графика **Объединенные группы**.
separate_groups	*флаг*	Опция графика **Отдельные группы**.
summary_of_steps	*флаг*	
F_pairwise	*флаг*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
критерии	UseValue UseProbability	

Таблица 124. Свойства discriminantnode (продолжение)		
Свойства discriminantnode	**Значения**	**Описание свойства**
F_value_entry	*число*	Значение по умолчанию - 3,84.
F_value_removal	*number*	Значение по умолчанию - 2,71.
probability_entry	*number*	Значение по умолчанию - 0,05.
probability_removal	*number*	Значение по умолчанию - 0,10.
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства extensionmodelnode

При помощи узла Extension Model можно запускать сценарии R или Python for spark для построения и скоринга результатов.

Пример Python for Spark

```
#### пример сценария для 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))

# число классов назначения по счетчику
predictorClassCount = len(metadata[lookup[target]])

# определить функцию для извлечения информации категориальных предикторов из модели данных
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

# преобразование кадра данных в RDD, содержащий 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)
```

R example

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <- lm(modelerData$Na~modelerData
$K,modelerData)
modelerDataModel
```

```
modelerModel
""")
```

Таблица 125. свойства extensionmodelnode

Свойства extensionmodelnode	**Значения**	**Описание свойства**
syntax_type	*R* *Python*	Указывает, какой сценарий работает – R или Python (R - значение по умолчанию).
r_build_syntax	*строка*	Синтаксис сценария R для построения моделей.
r_score_syntax	*строка*	Синтаксис сценария R для скоринга моделей.
python_build_syntax	*строка*	Синтаксис сценария Python для построения моделей.
python_score_syntax	*строка*	Синтаксис сценария Python для скоринга моделей.
convert_flags	`StringsAndDoubles` `LogicalValues`	Опция для преобразования флаговых полей.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	`POSIXct` `POSIXlt`	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.
output_html	*флаг*	Опция для вывода графиков на вкладке в слепке модели R.
output_text	*флаг*	Опция для записи текстового вывода консоли R на вкладку в слепке модели R.

Свойства factornode

Узел PCA/фактора предоставляет мощные средства сокращения числа данных для уменьшения сложности ваших данных. Анализ главных компонентов (principal components analysis, PCA) находит линейные комбинации входных полей, которыми главным образом определяются изменения в целом наборе полей, где компоненты ортогональны друг другу. Факторный анализ направлен на выявление скрытых факторов, объясняющих структуру корреляций в наборе наблюдаемых полей. Цель обоих подходов - найти небольшое количество производных полей, которые эффективно суммируют информацию исходного набора входных полей.

Пример

```
node = stream.create("factor", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# Вкладка "Модель"
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Опции эксперта
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)
# Раздел "Вращение"
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Таблица 126. Свойства factornode

Свойства factornode	**Значения**	**Описание свойства**
inputs	[*поле1 ... полеN*]	Модели PCA/факторов используют список входных полей, но не поле назначения. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	PC ULS GLS ML PAF Альфа Рисунок	
mode	Простые Эксперт	
max_iterations	*number*	

Таблица 126. Свойства factornode (продолжение)		
Свойства factornode	**Значения**	**Описание свойства**
complete_records	*флаг*	
матрица	Корреляция Ковариация	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*число*	
rotation	Нет Варимакс DirectOblimin Эквимакс Quartimax Промакс	
delta	*number*	При выборе DirectOblimin в качестве типа данных вращения, можно задать значение для delta. Если значение не задавать, будет использоваться значение delta по умолчанию.
каппа	*number*	Если в качестве типа данных вращения выбрать Promax, можно задать значение для kappa. Если значение не задавать, будет использоваться значение kappa по умолчанию.
sort_values	*флаг*	
hide_values	*флаг*	
hide_below	*number*	

Свойства featureselectionnode

Узел выбора возможностей изучает входные поля на возможность удаления, основываясь на наборе критериев (таких как процентная доля пропущенных значений); затем этот узел ранжирует важность оставшихся полей по отношению к заданному полю назначения. Например, если у набора данных сотни потенциальных входных полей, какие из них потенциально наиболее полезны при моделировании исхода лечения пациента?

Пример

```
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)
```

Более подробный пример создания и применения модели выбора возможностей смотрите в разделе “Пример автономного сценария: генерирование модели выбора возможностей” на стр. 5.

Таблица 127. Свойства featureselectionnode

Свойства featureselectionnode	**Значения**	**Описание свойства**
target	*поле*	Модели выбора возможностей ранжируют предикторы относительно заданного назначения. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
screen_single_category	*флаг*	При значении True экранирует поля, у которых слишком много записей относительно общего количества записей попадает в одну категорию.
max_single_category	*число*	Задает порог, используемый при значении screen_single_category, равном True.
screen_missing_values	*флаг*	При значении True экранирует поля, у которых слишком много пропущенных значений, выражается процентной долей от общего числа записей.
max_missing_values	*number*	

Таблица 127. Свойства featureselectionnode (продолжение)

Свойства featureselectionnode	**Значения**	**Описание свойства**
screen_num_categories	*флаг*	При значении True экранирует поля, у которых слишком много категорий относительно общего числа записей.
max_num_categories	*число*	
screen_std_dev	*флаг*	При значении True экранирует поля, для которых среднеквадратичные отклонения меньше или равны заданному минимуму.
min_std_dev	*число*	
screen_coeff_of_var	*флаг*	При значении True экранирует поля, для которых коэффициент изменчивости меньше или равен заданному минимуму.
min_coeff_of_var	*number*	
критерии	Пирсона Вероятность CramersV Лямбда	При ранжировании категориальных предикторов по категориальным полям назначения задает показатель, на котором основывается значение важности.
unimportant_below	*number*	Задает пороговые значения *p*, используемые для ранжирования переменных как важных, пограничных или не важных. Принимает значения от 0,0 до 1,0.
important_above	*number*	Принимает значения от 0,0 до 1,0.
unimportant_label	*string*	Задает метку для ранжирования как не важного.
marginal_label	*string*	
important_label	*string*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли важные поля.

Таблица 127. Свойства featureselectionnode (продолжение)		
Свойства featureselectionnode	**Значения**	**Описание свойства**
select_marginal	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли пограничные поля.
select_unimportant	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли не важные поля.
importance_value	*number*	Когда для selection_mode задано значение ImportanceValue, задает значение отсечения для использования. Принимает значения от 0 до 100.
top_n	*целое*	Когда для selection_mode задано значение TopN, задает значение отсечения для использования. Принимает значения от 0 до 1000.

Свойства genlinnode

Обобщенная линейная модель расширяет общую линейную модель, так что зависимая переменная считается линейно связанной с факторами и ковариатами через заданную функцию связи. Более того, модель допускает наличие у зависимой переменной распределения, отличающегося от нормального. Она включает в себя функциональные возможности большого количества статистических моделей, в том числе линейной регрессии, логистической регрессии, логлинейных моделей для количества данных и интервал-цензурированных моделей выживания.

Пример

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Таблица 128. Свойства genlinnode

Свойства genlinnode	**Значения**	**Описание свойства**
target	*поле*	Обобщенным линейным моделям требуется одно поле назначения, которое должно быть номинальным или флаговым, и одно или несколько входных полей. Можно задать также поле веса. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
use_weight	*флаг*	
weight_field	*поле*	Тип поля - только количественный.
target_represents_trials	*флаг*	
trials_type	Переменная FixedValue	
trials_field	*поле*	Тип поля - количественный, флаговый или порядковый.
trials_number	*number*	Значение по умолчанию - 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Переменная FixedValue	
offset_field	*поле*	Тип поля - только количественный.
offset_value	*number*	Должно быть действительным числом.
base_category	Последнее Первое	
include_intercept	*флаг*	
mode	Простые Эксперт	

Таблица 128. Свойства genlinnode (продолжение)		
Свойства genlinnode	**Значения**	**Описание свойства**
распределение	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Обратный гауссиан. NEGBIN: Отрицательное биномиальное.
negbin_para_type	Задать Оценка	
negbin_parameter	*number*	Значение по умолчанию 1. Должно содержать неотрицательное действительное число.
tweedie_parameter	*number*	

Таблица 128. Свойства genlinnode (продолжение)

Свойства genlinnode	**Значения**	**Описание свойства**
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: Дополнительное лог-лог. LOGC: Дополняющее лог. NEGBIN: Отрицательное биномиальное. NLOGLOG: Отрицательное лог-лог. CUMCAUCHIT: Кумулятивное Коши. CUMCLOGLOG: Кумулятивное дополняющее лог-лог. CUMLOGIT: Кумулятивное логит. CUMNLOGLOG: Кумулятивное отрицательное лог-лог. CUMPROBIT: Кумулятивное пробит.
Степень	*number*	Значение должно быть действительным ненулевым числом.
method	Гибридный Фишера NewtonRaphson	
max_fisher_iterations	*number*	Значение по умолчанию - 1; допустимы только положительные целые.

Таблица 128. Свойства genlinnode (продолжение)		
Свойства genlinnode	**Значения**	**Описание свойства**
scale_method	MaxLikelihoodEstimate Отклонение PearsonChiSquare FixedValue	
scale_value	*number*	Значение по умолчанию - 1; должно быть больше 0.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*number*	Значение по умолчанию - 100; допустимы только неотрицательные целые.
max_step_halving	*number*	Значение по умолчанию - 5; допустимы только положительные целые.
check_separation	*флаг*	
start_iteration	*number*	Значение по умолчанию - 20; допустимы только положительные целые.
estimates_change	*флаг*	
estimates_change_min	*число*	Значение по умолчанию - 1E-006; допустимы только положительные числа.
estimates_change_type	Абсолютная Относительный	
loglikelihood_change	*флаг*	
loglikelihood_change_min	*число*	Допускаются только положительные числа.
loglikelihood_change_type	Абсолютная Относительный	
hessian_convergence	*флаг*	
hessian_convergence_min	*number*	Допускаются только положительные числа.
hessian_convergence_type	Абсолютная Относительный	
case_summary	*флаг*	

Таблица 128. Свойства genlinnode (продолжение)

Свойства genlinnode	**Значения**	**Описание свойства**
contrast_matrices	*флаг*	
descriptive_statistics	*флаг*	
estimable_functions	*флаг*	
model_info	*флаг*	
iteration_history	*флаг*	
goodness_of_fit	*флаг*	
print_interval	*number*	Значение по умолчанию - 1; допустимы только положительные целые.
model_summary	*флаг*	
lagrange_multiplier	*флаг*	
parameter_estimates	*флаг*	
include_exponential	*флаг*	
covariance_estimates	*флаг*	
correlation_estimates	*флаг*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Вальда LR	
citype	Вальда Профиль	
tolerancelevel	*number*	Значение по умолчанию - 0,0001.
confidence_interval	*number*	Значение по умолчанию - 95.
loglikelihood_function	Заполнено Ядро	

Таблица 128. Свойства genlinnode (продолжение)		
Свойства genlinnode	**Значения**	**Описание свойства**
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	По возрастанию По убыванию DataOrder	
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства glmmnode

Обобщенная линейная смешанная модель (generalized linear mixed model, GLMM) обобщает линейную модель таким образом, что у значений назначения может быть отличное от нормального распределение и оно будет линейно связано с факторами и ковариатами через задаваемую функцию связи, так что наблюдения могут быть скоррелированными. Обобщенные линейные смешанные модели включают широкий набор моделей, начиная от простой линейной регрессии и кончая сложными многоуровневыми моделями для не нормально распределенных данных с повторными измерениями.

Таблица 129. Свойства glmmnode		
Свойства glmmnode	**Значения**	**Описание свойства**
residual_subject_spec	*структурированный*	Сочетание значений заданных категориальных полей, уникальным образом определяющее субъекты в наборе данных

Таблица 129. Свойства glmmnode (продолжение)

Свойства glmmnode	Значения	Описание свойства
repeated_measures	*структурированный*	Поля, используемые для идентификации повторных наблюдений.
residual_group_spec	[*поле1 ... полеN*]	Поля, определяющие независимые наборы параметров ковариации повторяющихся эффектов.
residual_covariance_type	Диагональная AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Задает ковариационную структуру для остатков.
custom_target	*флаг*	Обозначает, использовать ли поле назначения, определенное на вышележащем узле (false), или пользовательское поле назначения, заданное в поле target_field (true).
target_field	*поле*	Поле для использования в качестве поля назначения, если значение custom_target - это true.
use_trials	*флаг*	Обозначает, что будет использоваться, когда отклик назначения - это число произошедших в наборе испытаний событий, дополнительное поле или значение, задающее число испытаний. Значение по умолчанию - false.
use_field_or_value	Поле Значение	Обозначает, что используется для задания числа испытаний, поле (по умолчанию) или значение.
trials_field	*поле*	Поле, используемое для задания числа испытаний.

Таблица 129. Свойства glmmnode (продолжение)

Свойства glmmnode	**Значения**	**Описание свойства**
trials_value	*целое*	Значение, используемое для задания числа испытаний. Если задано, минимальное значение равно 1.
use_custom_target_reference	*флаг*	Обозначает, будет ли использоваться для категориального поля пользовательская опорная категория. Значение по умолчанию - false.
target_reference_value	*string*	Опорная категория для использования, если значение use_custom_target_reference равно true.
dist_link_combination	Номинальная Логит ГаммаLog БиномиальнЛогит ПуассонLog БиномиальнПробит ОтрицБиномLog БиномиальнLogC Пользовательские	Общие модели для распределения значений поля назначения. Выберите Custom, чтобы задать распределение из списка в поле target_distribution.
target_distribution	Нормальный Биномиальное Multinomial Гамма Обратная NegativeBinomial Poisson	Распределение значений для поля назначения, когда dist_link_combination равно Custom.

Таблица 129. Свойства glmmnode (продолжение)

Свойства glmmnode	Значения	Описание свойства
link_function_type	Тождество LogC Логарифмическое CLOGLOG Логит NLOGLOG PROBIT POWER CAUCHIT	Функция связи для установления соотношений между значениями назначения и предикторами. Если target_distribution - Binomial, можно использовать любую из перечисленных функций связи. Если target_distribution - Multinomial, можно использовать CLOGLOG, CAUCHIT, LOGIT, NLOGLOG или PROBIT. Если target_distribution - не Binomial и не Multinomial, можно использовать IDENTITY, LOG или POWER.
link_function_param	*number*	Значение параметра функции связи для использования. Применимо только в том случае, если normal_link_function или link_function_type - это POWER.
use_predefined_inputs	*флаг*	Обозначает, что представляют собой фиксированные поля эффектов - определяются ли они вышележащими полями как входные (true), или берутся из списка fixed_effects_list (false). Значение по умолчанию - false.
fixed_effects_list	*структурированный*	Если use_predefined_inputs - это false, задается, что входные поля будут использоваться как фиксированные поля эффектов.
use_intercept	*флаг*	При значении true (по умолчанию) в модель включается свободный член.
random_effects_list	*структурированный*	Список полей для задания в качестве случайных эффектов.
regression_weight_field	*поле*	Поле для использования в качестве поля веса анализа.
use_offset	Нет offset_value offset_field	Обозначает, как задается смещение. Значение Нет означает, что смещение не используется.

Таблица 129. Свойства glmmnode (продолжение)

Свойства glmmnode	Значения	Описание свойства
offset_value	*number*	Значение для использования в качестве смещения, если для use_offset задано offset_value.
offset_field	*поле*	Поле, используемое для значения смещения, если для use_offset задано offset_field.
target_category_order	По возрастанию По убыванию Data	Порядок сортировки для категориальных целевых переменных. Значение Данные задает использование порядка сортировки, найденного в данных. Значение по умолчанию - Ascending (по возрастанию).
inputs_category_order	По возрастанию По убыванию Data	Порядок сортировки для категориальных предикторов. Значение Данные задает использование порядка сортировки, найденного в данных. Значение по умолчанию - Ascending (по возрастанию).
max_iterations	*целое*	Максимальное количество итераций, которые могут быть выполнены алгоритмом. Неотрицательное целое число; значение по умолчанию - 100.
confidence_level	*целое*	Доверительный уровень, используемый для вычисления оценок интервалов коэффициентов модели. Неотрицательное целое число; максимальное значение 100, значение по умолчанию 95.
degrees_of_freedom_method	Фиксированная Varied	Задает, как вычисляется число степеней свободы для критерия значимости.
test_fixed_effects_coefficients	Модель Робастная	Способ вычисления матрицы ковариации оценок параметров.
use_p_converge	*флаг*	Опция для сходимости параметра.
p_converge	*число*	Пробел или любое положительное значение.
p_converge_type	Абсолютная Относительная	
use_l_converge	*флаг*	Опция для сходимости логарифмического правдоподобия.
l_converge	*число*	Пробел или любое положительное значение.

Таблица 129. Свойства glmmnode (продолжение)

Свойства glmmnode	Значения	Описание свойства
l_converge_type	Абсолютная Относительная	
use_h_converge	*флаг*	Опция для сходимости гессиана.
h_converge	*число*	Пробел или любое положительное значение.
h_converge_type	Абсолютная Относительная	
max_fisher_steps	*целое*	
singularity_tolerance	*число*	
use_model_name	*флаг*	Обозначает, как определять имя модели, задавать пользовательское имя (true) или использовать сгенерированное системой имя (false). Значение по умолчанию - false.
model_name	*string*	Если значение поля use_model_name равно true, задает имя модели для использования.
confidence	onProbability onIncrease	Основание для вычисления доверительного значения оценки: максимальная предсказанная вероятность или разность между максимальной и второй по значению предсказанной вероятностью.
score_category_probabilities	*флаг*	При значении true создает предсказанные вероятности для категориальных полей назначения. Значение по умолчанию - false.
max_categories	*целое*	Если значение score_category_probabilities - это true, задает максимальное число категорий для сохранения.
score_propensity	*флаг*	При значении true создает оценки склонности для флаговых полей назначения, определяющие правдоподобие выходного значения "true" для поля.
emeans	*structure*	Для каждого категориального поля из списка фиксированных эффектов задает, создавать ли оценочные маргинальные средние.

Таблица 129. Свойства glmmnode (продолжение)		
Свойства glmmnode	**Значения**	**Описание свойства**
covariance_list	*structure*	Для каждого количественного поля из списка фиксированных эффектов задает, что использовать при вычислении оценочных маргинальных средних, среднее или пользовательское значение.
mean_scale	Исходное Преобразование	Задает, как вычислять оценочные маргинальные средние, на основе исходной шкалы назначения (по умолчанию) или преобразования функции связи.
comparison_adjustment_method	НЗР SEQBONFERRONI SEQSIDAK	Способ корректировки для использования при выполнении испытаний гипотез с несколькими контрастами.

Свойства gle

Модель GLE (generalized linear engine - обобщенный линейный механизм) обобщает линейную модель таким образом, что у значений назначения может быть отличное от нормального распределение и оно будет линейно связано с факторами и ковариатами через задаваемую функцию связи, так что наблюдения могут быть скоррелированными. Обобщенные линейные смешанные модели включают широкий набор моделей, начиная от простой линейной регрессии и кончая сложными многоуровневыми моделями для не нормально распределенных данных с повторными измерениями.

Таблица 130. свойства gle		
Свойства gle	**Значения**	**Описание свойства**
custom_target	*флаг*	Обозначает, использовать ли поле назначения, определенное на вышележащем узле (false), или пользовательское поле назначения, заданное в поле target_field (true).
target_field	*поле*	Поле для использования в качестве поля назначения, если значение custom_target - это true.
use_trials	*флаг*	Обозначает, что будет использоваться, когда отклик назначения - это число произошедших в наборе испытаний событий, дополнительное поле или значение, задающее число испытаний. Значение по умолчанию - false.
use_trials_field_or_value	Поле Значение	Обозначает, что используется для задания числа испытаний, поле (по умолчанию) или значение.

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
trials_field	*поле*	Поле, используемое для задания числа испытаний.
trials_value	*целое*	Значение, используемое для задания числа испытаний. Если задано, минимальное значение равно 1.
use_custom_target_reference	*флаг*	Обозначает, будет ли использоваться для категориального поля пользовательская опорная категория. Значение по умолчанию - false.
target_reference_value	*строка*	Опорная категория для использования, если значение use_custom_target_reference равно true.
dist_link_combination	NormalIdentity ГаммаLog PoissonLog NegbinLog TweedieIdentity NominalLogit БиномиальнЛогит БиномиальнПробит БиномиальнLogC CUSTOM	Общие модели для распределения значений поля назначения. Выберите CUSTOM, чтобы задать распределение из списка в поле target_distribution.

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
target_distribution	Нормальный Биномиальное Multinomial Гамма INVERSE_GAUSS (Обратное Гаусса) NEG_BINOMIAL (Отрицательное биномиальное) Пуассона TWEEDIE (Твиди) Неизвестно	Распределение значений для поля назначения, когда dist_link_combination равно Custom.

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
link_function_type	Неизвестно 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	Функция связи для установления соотношений между значениями назначения и предикторами. Если target_distribution - Binomial, можно использовать: Неизвестно IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER Если target_distribution - NEG_BINOMIAL, можно использовать: NEG_BINOMIAL. Если значение target_distribution - UNKNOWN, можно использовать: GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
link_function_param	*number*	Значение параметра Tweedie для использования. Применимо только в том случае, если normal_link_function или link_function_type - это POWER.
tweedie_param	*число*	Значение параметра функции связи для использования. Применимо только в том случае, когда для dist_link_combination задано значение TweedieIdentity или link_function_type - это TWEEDIE.
use_predefined_inputs	*флаг*	Обозначает, что представляют собой поля эффектов модели - определяются ли они вышележащими полями как входные (true), или берутся из списка fixed_effects_list (false).
model_effects_list	*структурированный*	Если use_predefined_inputs - это false, задается, что входные поля будут использоваться как поля эффектов модели.
use_intercept	*флаг*	При значении true (по умолчанию) в модель включается свободный член.
regression_weight_field	*поле*	Поле для использования в качестве поля веса анализа.
use_offset	Нет Значение Переменная	Обозначает, как задается смещение. Значение Нет означает, что смещение не используется.
offset_value	*число*	Значение для использования в качестве смещения, если для use_offset задано offset_value.
offset_field	*поле*	Поле, используемое для значения смещения, если для use_offset задано offset_field.
target_category_order	По возрастанию По убыванию	Порядок сортировки для категориальных целевых переменных. Значение по умолчанию - Ascending (по возрастанию).
inputs_category_order	По возрастанию По убыванию	Порядок сортировки для категориальных предикторов. Значение по умолчанию - Ascending (по возрастанию).
max_iterations	*целое*	Максимальное количество итераций, которые могут быть выполнены алгоритмом. Неотрицательное целое число; значение по умолчанию - 100.

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
confidence_level	*число*	Доверительный уровень, используемый для вычисления оценок интервалов коэффициентов модели. Неотрицательное целое число; максимальное значение 100, значение по умолчанию 95.
test_fixed_effects_coefficients	Модель Робастная	Способ вычисления матрицы ковариации оценок параметров.
detect_outliers	*флаг*	При значении true алгоритм находит влияющие выбросы для всех распределений, кроме полиномиального.
conduct_trend_analysis	*флаг*	При значении true алгоритм проводит анализ тенденций для диаграммы рассеяния.
estimation_method	FISHER_SCORING NEWTON_RAPHSON HYBRID	Задает алгоритм оценки правдоподобия.
max_fisher_iterations	*целое*	Максимальное число итераций при использовании метода estimation_method для FISHER_SCORING. Минимум 0, максимум 20.
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	Укажите метод, который будет использоваться для оценки масштабного коэффициента.
scale_value	*число*	Доступно только в том случае, если для scale_parameter_method задано значение Fixed.
negative_binomial_method	MLE FIXED	Укажите метод, который будет использоваться для оценки вспомогательного параметра отрицательного биномиального распределения.
negative_binomial_value	*число*	Доступно только в том случае, если для negative_binomial_method задано значение Fixed.
non_neg_least_squares	*флаг*	Применять ли метод неотрицательных наименьших квадратов. Значение по умолчанию - false.
use_p_converge	*флаг*	Опция для сходимости параметра.
p_converge	*число*	Пробел или любое положительное значение.

Таблица 130. свойства gle (продолжение)		
Свойства gle	**Значения**	**Описание свойства**
p_converge_type	*флаг*	True = абсолютный, False = относительный
use_l_converge	*флаг*	Опция для сходимости логарифмического правдоподобия.
l_converge	*число*	Пробел или любое положительное значение.
l_converge_type	*флаг*	True = абсолютный, False = относительный
use_h_converge	*флаг*	Опция для сходимости гессиана.
h_converge	*число*	Пробел или любое положительное значение.
h_converge_type	*флаг*	True = абсолютный, False = относительный
max_iterations	*целое*	Максимальное количество итераций, которые могут быть выполнены алгоритмом. Неотрицательное целое число; значение по умолчанию - 100.
sing_tolerance	*целое*	
use_model_selection	*флаг*	Включает элементы управления пороговыми параметрами и выбором модели.

method	`LASSO`
	`ELASTIC_NET`
	`FORWARD_STEPWISE`
	`RIDGE`
		Определяет используемый метод выбора модели или, при использовании Ridge, метод регуляризации.

detect_two_way_interactions	*флаг*	При значении True модель будет автоматически обнаруживать двусторонние взаимодействия между входными полями. Этот элемент управления должен включаться исключительно в том случае, когда модель содержит только главные эффекты (то есть пользователь не создал эффектов более высокого порядка), а выбранный метод (method) - Прямой пошаговый, Лассо или Эластичная сеть.

Таблица 130. свойства gle (продолжение)

Свойства gle	Значения	Описание свойства
automatic_penalty_params	*флаг*	Доступно только в том случае, если метод выбора модели (method) - это Лассо или Эластичная сеть. Используйте эту функцию для ввода штрафных параметров, связанных с методами выбора переменных Лассо или Эластичной сети. При значении True используются значения по умолчанию. При значении False для штрафных параметров можно вводить пользовательские значения.
lasso_penalty_param	*число*	Доступно только в том случае, если метод выбора модели (method) - это Лассо или Эластичная сеть, а значение automatic_penalty_params - это False. Задайте значение штрафного параметра для метода Лассо.
elastic_net_penalty_param1	*число*	Доступно только в том случае, если метод выбора модели (method) - это Лассо или Эластичная сеть, а значение automatic_penalty_params - это False. Задайте значение штрафного параметра для параметра 1 метода Эластичная сеть.
elastic_net_penalty_param2	*число*	Доступно только в том случае, если метод выбора модели (method) - это Лассо или Эластичная сеть, а значение automatic_penalty_params - это False. Задайте значение штрафного параметра для параметра 2 метода Эластичная сеть.
probability_entry	*число*	Доступно только в том случае, если выбранный метод (method) - Прямой шаговый. Укажите уровень значимости критерия F-статистики для включения эффекта.
probability_removal	*число*	Доступно только в том случае, если выбранный метод (method) - Прямой шаговый. Укажите уровень значимости критерия F-статистики для удаления эффекта.

Таблица 130. свойства gle (продолжение)		
Свойства gle	**Значения**	**Описание свойства**
use_max_effects	*флаг*	Доступно только в том случае, если выбранный метод (method) - Прямой шаговый. Включает элемент управления max_effects. При значении False количество включенных эффектов по умолчанию должно равняться общему числу предоставленных для модели эффектов за исключением свободного члена.
max_effects	*целое*	Укажите максимальное число эффектов при использовании прямого пошагового метода построения.
use_max_steps	*флаг*	Включает элемент управления max_steps. При значении False количество шагов по умолчанию должно равняться утроенному числу предоставленных для модели эффектов за исключением свободного члена.
max_steps	*целое*	Укажите максимальное число шагов, которые будут выполняться при использовании прямого пошагового метода (method) построения.
use_model_name	*флаг*	Обозначает, как определять имя модели, задавать пользовательское имя (true) или использовать сгенерированное системой имя (false). Значение по умолчанию - false.
model_name	*строка*	Если значение поля use_model_name равно true, задает имя модели для использования.
usePI	*флаг*	При значении true вычисляется важность предикторов.

Свойства kmeansnode

Узел K-средних кластеризует набор данных в отдельные группы (или кластеры). Этот метод определяет фиксированное количество кластеров, итерационно распределяет записи по кластерам и настраивает центры кластеров, пока дальнейшие уточнения более не улучшают модель. Вместо попытки предсказать выходное значение *k*-средние используют процесс, называемый неконтролируемым обучением, чтобы обнаружить структуры в наборе входных полей.

Пример

```
node = stream.create("kmeans", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K",
"Age"])
# Вкладка "Модель"
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")
# Вкладка "Эксперт"
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)
```

Таблица 131. Свойства kmeansnode

Свойства kmeansnode	**Значения**	**Описание свойства**
inputs	[*поле1 ... полеN*]	Модели k-средних выполняют кластерный анализ для набора входных полей, но не используют поле назначения. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
num_clusters	*number*	
gen_distance	*флаг*	
cluster_label	Строка Число	
label_prefix	*string*	
mode	Простые Эксперт	
stop_on	Default Пользовательские	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	

Таблица 131. Свойства kmeansnode (продолжение)		
Свойства kmeansnode	**Значения**	**Описание свойства**
optimize	Скорость Память	Используется для определения, нужно ли при построении модели оптимизировать скорость или использование памяти.

Свойства kmeansasnode

K-Means - это один из наиболее часто используемых алгоритмов кластеризации. Он распределяет точки данных по заданному числу кластеров. Узел K-Means-AS в SPSS Modeler реализован на языке Spark. Подробности об алгоритмах K-Means смотрите в разделе https://spark.apache.org/docs/2.2.0/ml-clustering.html. Обратите внимание на то, что узел K-Means-AS автоматически применяет для категорийных переменных унитарное кодирование.

Таблица 132. свойства kmeansasnode		
Свойства kmeansasnode	**Значения**	**Описание свойства**
roleUse	*строка*	Задайте значение predefined для использования заранее заданных ролей или значение custom для использования пользовательских назначений полей. Значение по умолчанию - predefined.
autoModel	*Логический*	Задайте true, чтобы использовать имя по умолчанию ($S-prediction) для нового поля сгенерированной оценки, или false, чтобы использовать пользовательское имя. Значение по умолчанию - true.
элементы	*поле*	Список имён полей для ввода, когда для roleUse задано значение custom.
name	*строка*	Название нового поля сгенерированной оценки, когда для свойства autoModel задано значение false.
clustersNum	*целое*	Число создающихся кластеров. Значение по умолчанию - 5.
initMode	*строка*	Алгоритм инициализации. Возможные значения: k-means|| или random. Значение по умолчанию - k-means||.
initSteps	*целое*	Число шагов инициализации, когда для initMode задано значение k-means||. Значение по умолчанию - 2.

Таблица 132. свойства kmeansasnode (продолжение)		
Свойства kmeansasnode	**Значения**	**Описание свойства**
advancedSettings	*Логический*	Задайте true, чтобы сделать доступными следующие четыре свойства. Значение по умолчанию - false.
maxIteration	*целое*	Максимальное число итераций для кластеризации. Значение по умолчанию - 20.
tolerance	*строка*	Допуск для прекращения итераций. Возможные значения 1,0E-1, 1,0E-2, ..., 1,0E-6. Значение по умолчанию - 1,0E-4.
setSeed	*Логический*	Задайте true, чтобы использовать пользовательское начальное значение случайных чисел. Значение по умолчанию - false.
randomSeed	*целое*	Пользовательское начальное значение случайных чисел, когда для свойства setSeed задано значение true.

Свойства knnnode

Узел *k* ближайших соседей (k-Nearest Neighbor, KNN) связывает новое наблюдение с категорией или значением *k* объектов, ближайших к нему в пространстве предикторов, где *k* - это целое число. Подобные наблюдения близки друг к другу, а непохожие наблюдения, наоборот, удалены друг от друга.

Пример

```
node = stream.create("knn", "My node")
# Вкладка Цели
node.setPropertyValue("objective", "Custom")
# Вкладка Параметры - панель Соседи
node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Вкладка Параметры - панель Анализ
node.setPropertyValue("save_distances", True)
```

Таблица 133. Свойства knnnode		
Свойства knnnode	**Значения**	**Описание свойства**
analysis	PredictTarget IdentifyNeighbors	

Таблица 133. Свойства knnnode (продолжение)		
Свойства knnnode	**Значения**	**Описание свойства**
objective	Баланс Скорость Точность Пользовательские	
normalize_ranges	*флаг*	
use_case_labels	*флаг*	Переключатель для включения следующей опции.
case_labels_field	*поле*	
identify_focal_cases	*флаг*	Переключатель для включения следующей опции.
focal_cases_field	*поле*	
automatic_k_selection	*флаг*	
fixed_k	*целое*	Включается только в том случае, если значение automatic_k_selectio - это False.
minimum_k	*целое*	Включается только в том случае, если значение automatic_k_selectio - это True.
maximum_k	*целое*	
distance_computation	Евклидова CityBlock	
weight_by_importance	*флаг*	
range_predictions	Mean Медиана	
perform_feature_selection	*флаг*	
forced_entry_inputs	[*поле1 ... полеN*]	
stop_on_error_ratio	*флаг*	
number_to_select	*целое*	
minimum_change	*number*	
validation_fold_assign_by_field	*флаг*	

Таблица 133. Свойства knnnode (продолжение)		
Свойства knnnode	**Значения**	**Описание свойства**
number_of_folds	*целое*	Включается только в том случае, если значение validation_fold_assign_by_field - это False
set_random_seed	*флаг*	
random_seed	*number*	
folds_field	*поле*	Включается только в том случае, если значение validation_fold_assign_by_field - это True
all_probabilities	*флаг*	
save_distances	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства kohonennode

Узел Коонена генерирует тип нейросети, которую можно использовать для кластеризации набора данных в отдельные группы. Когда сеть полностью обучена, похожие записи должны быть близко друг от друга на выходной карте, а отличающиеся записи должны быть сильно разделены. По количеству наблюдений, захваченных каждым нейроном в слепке модели, можно определить сильные нейроны. Это может дать представление об оправданном количестве кластеров.

Пример

```
node = stream.create("kohonen", "My node")
# Вкладка "Модель"
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")
# Вкладка "Эксперт"
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)
```

Таблица 134. Свойства kohonennode

Свойства kohonennode	Значения	Описание свойства
inputs	[*поле1 ... полеN*]	Модели Коонена используют список входных полей, но не поля назначения. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue	*флаг*	
show_feedback	*флаг*	
stop_on	Default Время	
время	*number*	
optimize	Скорость Память	Используется для определения, нужно ли при построении модели оптимизировать скорость или использование памяти.
cluster_label	*флаг*	
mode	Простые Эксперт	
ширина	*число*	
length	*число*	
decay_style	Линейная Экспоненциальное	
phase1_neighborhood	*число*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

Свойства linearnode

Модели линейной регрессии предсказывают значения непрерывного целевого поля на основе линейных взаимосвязей между целевым полем и одним или несколькими предикторами.

Пример

```
node = stream.create("linear", "My node")
# Вкладка Опции сборки - панель Цели
node.setPropertyValue("objective", "Standard")
# Вкладка Опции сборки - панель Выбор модели
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Вкладка Опции сборки - панель Ансамбли
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Таблица 135. Свойства linearnode

Свойства linearnode	**Значения**	**Описание свойства**
target	*поле*	Задает одно поле назначения.
inputs	[*поле1 ... полеN*]	Предикторные поля, используемые моделью.
continue_training_existing_model	*флаг*	
objective	Стандартные Бэггинг Бустинг psm	psm используется для очень больших наборов данных и требует соединения с сервером.
use_auto_data_preparation	*флаг*	
confidence_level	*number*	
model_selection	ForwardStepwise BestSubsets Нет	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*флаг*	
max_effects	*number*	
use_max_steps	*флаг*	

Таблица 135. Свойства linearnode (продолжение)		
Свойства linearnode	**Значения**	**Описание свойства**
max_steps	*число*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Медиана	
component_models_n	*number*	
use_random_seed	*флаг*	
random_seed	*number*	
use_custom_model_name	*флаг*	
custom_model_name	*string*	
use_custom_name	*флаг*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

Свойства linearasnode

Модели линейной регрессии предсказывают значения непрерывного целевого поля на основе линейных взаимосвязей между целевым полем и одним или несколькими предикторами.

Таблица 136. Свойства linearasnode		
Свойства linearasnode	**Значения**	**Описание свойства**
target	*поле*	Задает одно поле назначения.
inputs	[*поле1 ... полеN*]	Предикторные поля, используемые моделью.
weight_field	*поле*	Поле анализа, используемое моделью.
custom_fields	*флаг*	Значение по умолчанию - TRUE.
свободный член	*флаг*	Значение по умолчанию - TRUE.
detect_2way_interaction	*флаг*	Указывает, рассматривать или нет двустороннее взаимодействие. Значение по умолчанию - TRUE.

Таблица 136. Свойства linearasnode (продолжение)

Свойства linearasnode	Значения	Описание свойства
cin	*number*	Это доверительный уровень, используемый при вычислении оценок коэффициентов модели. Задайте значение больше 0 и меньше 100. Значение по умолчанию - 95.
factor_order	ascending descending	Порядок сортировки для категориальных предикторов. Значение по умолчанию - ascending.
var_select_method	ForwardStepwise BestSubsets none	Используемый метод выбора модели. Значение по умолчанию - ForwardStepwise.
criteria_for_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	Статистика, используемая для определения того, следует ли эффект добавить в модель или исключить из нее. Значение по умолчанию - AdjustedRSquare.
pin	*number*	Эффект, у которого наименьшее p-значение меньше указанного порога pin, добавляется в модель. Значение по умолчанию - 0.05.
pout	*number*	Все эффекты в модели с p-значением больше этого указанного порога pout удаляются. Значение по умолчанию - 0.10.
use_custom_max_effects	*флаг*	Использовать ли максимальное количество эффектов в окончательной модели. Значение по умолчанию - FALSE.
max_effects	*number*	Максимальное количество эффектов для использования в окончательной модели. Значение по умолчанию - 1.
use_custom_max_steps	*флаг*	Использовать ли максимальное число шагов. Значение по умолчанию - FALSE.
max_steps	*number*	Максимальное число шагов до остановки пошагового алгоритма. Значение по умолчанию - 1.

Таблица 136. Свойства linearasnode (продолжение)		
Свойства linearasnode	**Значения**	**Описание свойства**
criteria_for_best_subsets	AICC AdjustedRSquare ASE	Используемый режим критериев. Значение по умолчанию - AdjustedRSquare.

Свойства logregnode

Логистическая регрессия - это статистический метод для классификации записей на основании значений входных полей. Она аналогична линейной регрессии, но логистическая регрессия использует категориальные поля назначения вместо численных.

Полиномиальный пример

```
node = stream.create("logreg", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# Вкладка "Модель"
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)
# Вкладка "Эксперт"
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")
# Раздел "Сходимость..."
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)
# Раздел "Выход..."
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)
# Опции "Пошаговые модели"
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")
```

Биномиальный пример

```
node = stream.create("logreg", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# Вкладка "Модель"
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)
# Вкладка "Эксперт"
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")
# Раздел "Сходимость..."
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# Раздел "Выход..."
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)
# Опции "Пошаговые модели"
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Таблица 137. Свойства logregnode

Свойства logregnode	**Значения**	**Описание свойства**
target	*поле*	Моделям логистической регрессии требуется одно поле назначения и одно или несколько входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
logistic_procedure	Биномиальное Multinomial	
include_constant	*флаг*	

Таблица 137. Свойства logregnode (продолжение)		
Свойства logregnode	**Значения**	**Описание свойства**
mode	Простые Эксперт	
method	Ввод Пошаговый Вперед Backwards BackwardsStepwise	
binomial_method	Ввод Вперед Backwards	
model_type	MainEffects FullFactorial Пользовательские	Когда в качестве типа модели задано FullFactorial, пошаговые способы не будут запущены, даже если они заданы. Вместо этого будет использоваться способ Ввод. Если для типа модели задано Пользовательский, но сами пользовательские поля не заданы, будет построена модель главных эффектов.
custom_terms	[*[BP Sex][BP][Age]*]	
multinomial_base_category	*string*	Задает, как определяется опорная категория.
binomial_categorical_input	*string*	

Таблица 137. Свойства logregnode (продолжение)

Свойства logregnode	Значения	Описание свойства
binomial_input_contrast	Индикатор Простые Разность Хелмерт Повторяемый Полиномиальный Отклонение	Ключевое свойство для категориального входного поля, которое задает, как определяется контраст.
binomial_input_category	Первое Последнее	Ключевое свойство для категориального входного поля, которое задает, как определяется контраст.
scale	Нет UserDefined Пирсона Отклонение	
scale_value	*number*	
all_probabilities	*флаг*	
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	*флаг*	
max_terms	*number*	

Таблица 137. Свойства logregnode (продолжение)		
Свойства logregnode	**Значения**	**Описание свойства**
entry_criterion	Оценка LR	
removal_criterion	LR Вальда	
probability_entry	*number*	
probability_removal	*число*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment Нет	
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*	

Таблица 137. Свойства logregnode (продолжение)

Свойства logregnode	Значения	Описание свойства
iteration_history	*флаг*	
history_steps	*number*	
summary	*флаг*	
likelihood_ratio	*флаг*	
asymptotic_correlation	*флаг*	
goodness_fit	*флаг*	
parameters	*флаг*	
confidence_interval	*number*	
asymptotic_covariance	*флаг*	
classification_table	*флаг*	
stepwise_summary	*флаг*	
info_criteria	*флаг*	
monotonicity_measures	*флаг*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*флаг*	
binomial_parameters	*флаг*	
binomial_iteration_history	*флаг*	
binomial_classification_plots	*флаг*	
binomial_ci_enable	*флаг*	
binomial_ci	*number*	
binomial_residual	выбросы все	
binomial_residual_enable	*флаг*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Вальда Условное	

Таблица 137. Свойства logregnode (продолжение)		
Свойства logregnode	**Значения**	**Описание свойства**
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	

Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.

Узел линейного механизма опорных векторов (Linear Support Vector Machine, LSVM) позволяет классифицировать данные по одной или двум группам без переобучения. Модель LSVM линейная и хорошо работает с широкими наборами данных, в частности, в случае очень большого числа записей.

Таблица 138. Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.		
Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.	**Значения**	**Описание свойства**
intercept	*флаг*	Включает в модель свободный член. Значение по умолчанию - True.
target_order	По возрастанию По убыванию	Задает порядок сортировки для категориальных полей назначения. Для непрерывных полей назначения игнорируется. Значение по умолчанию - Ascending (по возрастанию).
precision	*number*	Используется только в том случае, если уровень измерения поля назначения - Количественный. Задает параметр, относящийся к чувствительности потерь для регрессии. Минимальное значение 0, а максимум не определен. Значение по умолчанию - 0.1.
exclude_missing_values	*флаг*	При значении True запись исключается в случае отсутствия любого из значений. Значение по умолчанию - False.

Таблица 138. Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM. (продолжение)

Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.	**Значения**	**Описание свойства**
penalty_function	L1 L2	Задает тип используемой функции штрафа. Значение по умолчанию - L2.
lambda	*число*	Параметр штрафа (регуляризации).
calculate_variable_importance	*флаг*	Для моделей, создающих соответствующую меру важности, при задании этой опции выводится диаграмма, указывающая относительную важность каждого предиктора при оценке модели. Обратите внимание на то, что для некоторых моделей вычисление важности переменных - это длительный процесс, особенно при работе с большими наборами данных, и в результате по умолчанию для некоторых моделей эта опция будет выключена. Важность переменных недоступна для моделей списка решений.

Свойства neuralnetnode

Важное замечание: В этом выпуске доступна более новая версия узла моделирования нейронных сетей с расширенными возможностями, которая обсуждается в следующем разделе (*neuralnetwork*). Хотя вы по-прежнему можете построить и оценить модель в предыдущей версии, рекомендуется изменить сценарии для использования новой версии. Подробности предыдущей версии приведены здесь для справки.

Пример

```
node = stream.create("neuralnet", "My node")
# Вкладка "Поля"
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# Вкладка "Модель"
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")
# Раздел "Опции эксперта для нескольких методов"
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Таблица 139. Свойства neuralnetnode

Свойства neuralnetnode	**Значения**	**Описание свойства**
назначения	[*поле1 ... полеN*]	Для узла нейронных сетей предполагаются одно или несколько полей назначения и одно или несколько входных полей. Поля веса и частоты игнорируются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	Быстрый Динамический Несколько Сокращение ExhaustivePrune RBFN	
prevent_overtrain	*флаг*	
train_pct	*number*	
set_random_seed	*флаг*	
random_seed	*number*	
mode	Простые Эксперт	
stop_on	Default Точность Циклы Время	Режим остановки.
accuracy	*number*	Точность остановки.
cycles	*number*	Циклов для обучения.
время	*number*	Время для обучения (минуты).

Таблица 139. Свойства neuralnetnode (продолжение)		
Свойства neuralnetnode	**Значения**	**Описание свойства**
continue	*флаг*	
show_feedback	*флаг*	
binary_encode	*флаг*	
use_last_model	*флаг*	
gen_logfile	*флаг*	
logfile_name	*string*	
alpha	*число*	
initial_eta	*число*	
high_eta	*число*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	Один Два Три	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*число*	
persistence	*number*	
m_topologies	*string*	
m_non_pyramids	*флаг*	
m_persistence	*number*	
p_hid_layers	Один Два Три	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*число*	
p_persistence	*число*	
p_hid_rate	*число*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	

Таблица 139. Свойства neuralnetnode (продолжение)		
Свойства neuralnetnode	**Значения**	**Описание свойства**
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*флаг*	
r_alpha	*number*	
r_eta	*число*	
optimize	Скорость Память	Используется для определения, нужно ли при построении модели оптимизировать скорость или использование памяти.
calculate_variable_importance	*флаг*	Комментарий: Свойство sensitivity_analysis, использованное в предыдущих выпусках, объявлено устаревшим и заменяется этим свойством. Старое свойство еще поддерживается, но рекомендуется использовать calculate_variable_importance.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства neuralnetworknode

Узел нейросетей использует упрощенную модель обработки информации человеческим мозгом. Нейросети работают, обсчитывая большое количество связанных между собой обрабатываемых элементов, которые представляют абстрактную версию нейронов. Нейросети - это мощные средства оценки общих функциональных зависимостей, требующие минимальных знаний статистики и математики для их обучения и применения.

Пример

```
node = stream.create("neuralnetwork", "My node")
# Вкладка Опции сборки - панель Цели
node.setPropertyValue("objective", "Standard")
# Вкладка Опции сборки - панель Ансамбли
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Таблица 140. Свойства neuralnetworknode

Свойства neuralnetworknode	**Значения**	**Описание свойства**
targets	[*поле1 ... полеN*]	Задает поля назначения.
inputs	[*поле1 ... полеN*]	Предикторные поля, используемые моделью.
splits	*[поле1 ... полеN*	Задает поле или поля для использования для моделирования разбиения.
use_partition	*флаг*	Если определено поле раздела, эта опция обеспечивает, что для построения модели используются только данные из раздела обучения.
continue	*флаг*	Продолжить обучение существующей модели.
objective	Стандартные Бэггинг Бустинг psm	psm используется для очень больших наборов данных и требует соединения с сервером.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*флаг*	
first_layer_units	*число*	
second_layer_units	*number*	
use_max_time	*флаг*	
max_time	*number*	
use_max_cycles	*флаг*	
max_cycles	*number*	
use_min_accuracy	*флаг*	
min_accuracy	*number*	
combining_rule_categorical	Голосование HighestProbability HighestMeanProbability	

Таблица 140. Свойства neuralnetworknode (продолжение)		
Свойства neuralnetworknode	**Значения**	**Описание свойства**
combining_rule_continuous	Mean Медиана	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*флаг*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*логический*	
model_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*флаг*	
max_categories	*number*	
score_propensity	*флаг*	
use_custom_name	*флаг*	
custom_name	*string*	
tooltip	*string*	
keywords	*строка*	
annotation	*строка*	

Свойства questnode

Узел QUEST предоставляет метод бинарной классификации для построения деревьев решений, разработанный для уменьшения времени обработки, требуемого для анализа больших деревьев C&R, при одновременном подавлении обнаруженного в способах деревьев классификации предпочтения входных полей, допускающих больше расщеплений. Входные поля могут быть в числовом диапазоне (количественными), но поле назначения должно быть категориальным. Все расщепления бинарные.

Пример

```
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)
```

Таблица 141. Свойства questnode

Свойства questnode	**Значения**	**Описание свойства**
target	*поле*	Моделям QUEST требуется одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue_training_existing_model	*флаг*	
objective	Стандартные Бустинг Бэггинг psm	psm используется для очень больших наборов данных и требует соединения с сервером.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*флаг*	
tree_directives	*string*	
use_max_depth	Default Пользовательские	
max_depth	*целое*	Максимальное количество уровней в дереве, от 0 до 1000. Используется только в том случае, если use_max_depth = Custom.
prune_tree	*флаг*	Отсекать ветви, чтобы избежать переобучения.
use_std_err	*флаг*	Использовать максимальную разницу в риске (в стандартных ошибках).
std_err_multiplier	*number*	Максимальная разность.

Таблица 141. Свойства questnode (продолжение)		
Свойства questnode	**Значения**	**Описание свойства**
max_surrogates	*number*	Максимум суррогатов.
use_percentage	*флаг*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*флаг*	
costs	*структурированный*	Структурированное свойство.
priors	Данные Равенство Пользовательские	
custom_priors	*структурированный*	Структурированное свойство.
adjust_priors	*флаг*	
trails	*number*	Число моделей компонентов для бустинга или бэггинга.
set_ensemble_method	Голосование HighestProbability HighestMeanProbability	Принятое по умолчанию правило объединения для категориальных целевых полей.
range_ensemble_method	Mean Медиана	Принятое по умолчанию правило объединения для непрерывных целевых полей.
large_boost	*флаг*	Применить бустинг к очень большим наборам данных.
split_alpha	*number*	Уровень значимости для разбиения.
train_pct	*number*	Множество предотвращения сверхобучения.
set_random_seed	*флаг*	Опция репликации результатов.
seed	*number*	
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Таблица 141. Свойства questnode (продолжение)		
Свойства questnode	**Значения**	**Описание свойства**
adjusted_propensity_partition	Критерий Проверка	

Свойства randomtrees

Узел случайных деревьев аналогичен существующему узлу C&RT; однако узел случайных деревьев сконструирован для обработки большого объема данных с целью создания одного дерева и выводит конечную модель в средстве просмотра вывода, добавленном в SPSS Modeler версии 17. Узел случайных деревьев генерирует дерево решений, которое используется для предсказания или классификации будущих наблюдений. Этот метод использует рекурсивное разделение, чтобы расщепить обучающие записи на сегменты, на каждом шаге минимизируя неоднородность, причем узел дерева считается *чистым*, если все 100% наблюдений в узле попадают в конкретную категорию поля назначения. Входные поля и поля назначения могут быть из числового диапазона или категориальными (номинальными, порядковыми или флагами); все расщепления бинарны (только две подгруппы).

Таблица 142. свойства randomtrees		
Свойства randomtrees	**Значения**	**Описание свойства**
target	*поле*	Моделям узла случайных деревьев требуется одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
number_of_models	*целое*	Определяет количество моделей для построения как часть моделирования ансамблей.
use_number_of_predictors	*флаг*	Определяет, будет ли использоваться number_of_predictors.
number_of_predictors	*целое*	Задает число предикторов, используемых при построении моделей расщепления.
use_stop_rule_for_accuracy	*флаг*	Определяет, останавливается ли построение модели, если точность нельзя улучшить.
sample_size	*number*	Уменьшите это значение для повышения производительности при обработке очень больших наборов данных.

Таблица 142. свойства randomtrees (продолжение)

Свойства randomtrees	**Значения**	**Описание свойства**
handle_imbalanced_data	*флаг*	Если поле назначения модели - это отдельный выходной флаг, отношение числа предпочтительных результатов к числу нежелательных очень мало, данные не сбалансированы, а выполняемая моделью бутстреп-выборка может повлиять на точность модели. Включите обработку несбалансированных данных, чтобы генерируемая модель была более надежной и захватывала большую часть предпочтительных выходных данных.
use_weighted_sampling	*флаг*	При значении *False* переменные для каждого узла выбираются случайно с равной вероятностью. При значении *True* переменным назначаются веса, определяющие вероятность выбора.
max_node_number	*целое*	Максимальное количество узлов, допустимых в отдельных деревьях. Если при следующем расщеплении это число может быть превышено, рост дерева остановится.
max_depth	*целое*	Максимальная глубина дерева до остановки роста.
min_child_node_size	*целое*	Определяет минимальное число записей, разрешенных для дочернего узла после расщепления родительского узла. Если дочерний узел содержал бы при расщеплении меньше указанного здесь значения, родительский узел не расщепляется
use_costs	*флаг*	
costs	*структурированный*	Структурированное свойство. Формат - это список из трех значений: фактическое значение, предсказанное значение и стоимость ошибочного предсказания. Например: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])

Таблица 142. свойства randomtrees (продолжение)		
Свойства randomtrees	**Значения**	**Описание свойства**
default_cost_increase	none Линейная square custom	**Прим.:** разрешено только для порядковых полей назначения. Задайте значения по умолчанию в матрице стоимостей.
max_pct_missing	*целое*	Если процентная доля отсутствующих значений в любом входном поле превышает заданное здесь значение, это входное поле исключается. Минимум 0, максимум 100.
exclude_single_cat_pct	*целое*	Если одно значение категории представляет большую процентную долю записей, чем указано здесь, все поле будет исключено из построения модели. Минимум 1, максимум 99.
max_category_number	*целое*	Если число категорий в поле превосходит данное значение, это поле будет исключено из построения модели. Минимум - 2.
min_field_variation	*number*	Если коэффициент вариации непрерывного поля меньше данного значения, это поле исключается из построения модели.
num_bins	*целое*	Используется только в том случае, когда данные представлены непрерывными входными полями. Задайте число равных интервалов частоты, которые будут использоваться для входных полей; возможны варианты 2, 4, 5, 10, 20, 25, 50 или 100.
topN	*целое*	Задает число правил для отчета. Значение по умолчанию - 50, минимальное значение - 1, максимальное - 1000.

Свойства regressionnode

Линейная регрессия - это общепринятый статистический метод обработки данных и вычисления предсказаний при подгонке прямой линии или плоскости, минимизирующих разности между предсказанными и фактическими выходными значениями.

Прим.: В следующем выпуске узел регрессии будет заменен узлом линейных моделей. Рекомендуется для линейной регрессии с этого момента перейти на использование узла Линейные модели.

Пример

```
node = stream.create("regression", "My node")
# Вкладка "Поля"
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")
# Вкладка "Модель"
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)
# Вкладка "Эксперт"
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# Раздел "По шагам..."
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)
# Раздел "Выход..."
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)
```

Таблица 143. Свойства regressionnode

Свойства regressionnode	**Значения**	**Описание свойства**
target	*поле*	Моделям регрессии требуется одно поле назначения и одно или несколько входных полей. Можно задать также поле веса. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	Ввод Пошаговый Backwards Вперед	

Таблица 143. Свойства regressionnode (продолжение)		
Свойства regressionnode	**Значения**	**Описание свойства**
include_constant	*флаг*	
use_weight	*флаг*	
weight_field	*поле*	
mode	Простые Эксперт	
complete_records	*флаг*	
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	Используйте двойные кавычки.
stepping_method	useP useF	useP : использовать вероятность F useF: использовать значение F
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*флаг*	
confidence_interval	*флаг*	
covariance_matrix	*флаг*	
collinearity_diagnostics	*флаг*	

Таблица 143. Свойства regressionnode (продолжение)		
Свойства regressionnode	**Значения**	**Описание свойства**
regression_coefficients	*флаг*	
exclude_fields	*флаг*	
durbin_watson	*флаг*	
model_fit	*флаг*	
r_squared_change	*флаг*	
p_correlations	*флаг*	
описательные статистики	*флаг*	
calculate_variable_importance	*флаг*	

Свойства sequencenode

Узел последовательности обнаруживает правила связывания для последовательных или зависящих от времени данных. Последовательность - это список наборов элементов с тенденцией появления в предсказуемом порядке. Например, покупатель, который приобрел лезвия и лосьон после бритья, с большой вероятностью в следующий раз купит крем для бритья. Узел последовательности основан на алгоритме правил связывания CARMA, использующем эффективный двухпроходный способ обнаружения последовательностей.

Пример

```
node = stream.create("sequence", "My node")
# Вкладка "Поля"
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")
# Вкладка "Модель"
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)
# Вкладка "Эксперт"
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)
```

Таблица 144. Свойства sequencenode		
Свойства sequencenode	**Значения**	**Описание свойства**
id_field	*поле*	Для создания модели Последовательности необходимо определить поле ID, дополнительное поле времени и одно или несколько полей содержимого. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
time_field	*поле*	
use_time_field	*флаг*	
content_fields	[*поле1 ... полеn*]	
contiguous	*флаг*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Простые Эксперт	
use_max_duration	*флаг*	
max_duration	*number*	
use_gaps	*флаг*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*флаг*	
pruning_value	*number*	
set_mem_sequences	*флаг*	
mem_sequences	*целое*	

Свойства slrmnode

Узел Самообучаемая модель откликов (Self-Learning Response Model, SLRM) позволяет построить модель, в которой одно новое наблюдение или всего несколько наблюдений могут быть использованы для повторной оценки модели без необходимости повторного обучения модели с использованием всех данных.

Пример

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
```

```
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Таблица 145. Свойства slrmnode

Свойства slrmnode	Значения	Описание свойства
target	*поле*	Поле назначения должно быть номинальным или флаговым. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
target_response	*поле*	Тип должен быть флаговым.
continue_training_existing_model	*флаг*	
target_field_values	*флаг*	Использовать все: использовать все значения источника. Задать: выбрать нужные значения.
target_field_values_specify	*[поле1 ... полеN]*	
include_model_assessment	*флаг*	
model_assessment_random_seed	*number*	Должно быть действительным числом.
model_assessment_sample_size	*number*	Должно быть действительным числом.
model_assessment_iterations	*number*	Число итераций.
display_model_evaluation	*флаг*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*число*	
sort	По возрастанию По убыванию	Задает, в каком порядке будут показываться предложения, начиная с максимальной или минимальной оценки.
model_reliability	*флаг*	
calculate_variable_importance	*флаг*	

Свойства statisticsmodelnode

Узел Statistics Model позволяет анализировать данные и работать с ними, вызывая процедуру IBM SPSS Statistics, генерирующую PMML. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Свойства этого узла описаны в разделе “Свойства statisticsmodelnode” на стр. 454.

Свойства stpnode

Узел пространственно-временного предсказания (Spatio-Temporal Prediction, STP) использует данные, содержащие информацию о положении, входные поля для прогноза (предикторы), поле времени и целевое поле. В этих данных для каждого положения в каждом времени измерения по каждому предиктора есть значительный ряд значений. После анализа данных их можно использовать для предсказания целевых значений в любом положении в области данных форм, используемых при анализе.

Таблица 146. свойства stpnode

Свойства stpnode	**Тип переменной**	**Описание свойства**
Вкладка **Поля**		
назначение	*поле*	Это поле назначения.
местоположение	*поле*	Поле положения для модели. Разрешены только геопространственные поля.
location_label	*поле*	Категориальное поле, которое будет использоваться в выводе в качестве метки положений, выбранных в свойстве location
time_field	*поле*	Поле времени для модели. Разрешены только поля с непрерывным типом измерения, а тип хранения должен быть время, дата, отметка времени или integer.
inputs	*[поле1 ... полеN]*	Список входных полей.
Вкладка **Интервалы времени**		
interval_type_timestamp	Years Quarters Months Weeks Days Hours Minutes Секунды	

Таблица 146. свойства stpnode (продолжение)

Свойства stpnode	**Тип переменной**	**Описание свойства**
interval_type_date	Years Quarters Months Weeks Дни	
interval_type_time	Hours Minutes Секунды	Ограничивает число дней в неделе, учитываемых при создании индекса времени, который используется STP для вычисления
interval_type_integer	Периоды (Только поля индексов времени; хранение типа Integer)	Интервал, в который должен быть преобразован набор данных. Переменная выбора зависит от типа хранения поля, выбираемого в качестве time_field для модели.
period_start	*целое*	
start_month	Январь Февраль Март Апрель Май Июнь Июль Август Сентябрь Октябрь Ноябрь Декабрь	Месяц, с которого модель начнет индексирование (например, если задано Март, но для первой записи в наборе данных задано Январь, модель пропустит первые две записи и начнет индексирования с марта).

Таблица 146. свойства stpnode (продолжение)

Свойства stpnode	Тип переменной	Описание свойства
week_begins_on	Воскресенье Понедельник Tuesday Wednesday Четверг Friday Суббота	Начальная точка для индекса времени, создаваемого STP (Spatio-Temporal Prediction - пространственно-временное предсказание) из данных
days_per_week	*целое*	Минимум 1, максимум 7 с шагом 1.
hours_per_day	*целое*	Сколько часов в день учитывает модель. Если задано значение 10, модель начнет индексирование в момент времени day_begins_at и продолжит его в течение 10 часов, затем пропустит все записи до следующего значения, подходящего для значения day_begins_at, и т.д.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Задает значение для часа, с которого модель начинает индексирование.

Таблица 146. свойства stpnode (продолжение)

Свойства stpnode	**Тип переменной**	**Описание свойства**
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Это параметр инкремента для минут или секунд. Он определяет, где модель создает индексы из данных. Поэтому при инкременте 30 и типе интервала seconds модель будет создавать индекс из данных каждые 30 секунд.
data_matches_interval	*Логическое значение*	Если задано N, перед построением модели выполняется преобразование данных в регулярный interval_type. Если данные уже в верном формате и interval_type и все связанные значения параметров соответствуют вашим данным, задайте для этого свойства значение Y, чтобы предотвратить преобразование или агрегирование данных. Задание значения Y отключает все элементы управления агрегированием.

Таблица 146. свойства stpnode (продолжение)		
Свойства stpnode	**Тип переменной**	**Описание свойства**
agg_range_default	Sum Mean Минимум Максимум Медиана 1stQuartile 3rdQuartile	Это определяет способ агрегирования по умолчанию, используемый для количественных полей. Любые количественные поля, не включенные специально в пользовательское агрегирование, будут агрегироваться с использованием указанного здесь способа.
custom_agg	[[field, aggregation method],[]..] Демо: [['x5' 'FirstQuartile']['x4' 'Sum']]	Структурированное свойство: Параметр сценария: custom_agg Например: set :stpnode.custom_agg = [[field1 function] [field2 function]] Где функция - это функция агрегирования, которую следует использовать с этим полем.
Вкладка **основные параметры**		
include_intercept	*флаг*	

Таблица 146. свойства stpnode (продолжение)		
Свойства stpnode	**Тип переменной**	**Описание свойства**
max_autoregressive_lag	*целое*	Минимум 1, максимум 5 с инкрементом 1. Это количество предыдущих записей, нужных для предсказания. Поэтому, например, при задании значения 5 для создания нового прогноза будут использоваться пять предыдущих записей. Заданное здесь число записей из данных сборки встроено в модель, и поэтому пользователю не нужно снова предоставлять эти данные при скоринге модели.
estimation_method	Parametric Nonparametric	Метод для моделирования матрицы пространственной ковариации.
parametric_model	Gaussian Экспоненциальное PoweredExponential	Параметр порядка для модели пространственной ковариации Parametric
exponential_power	*число*	Показатель степени для модели PoweredExponential. Минимум 1, максимум 2.
Вкладка **Дополнительно**		
max_missing_values	*целое*	Максимальный процент записей с разрешенным в модели числом пропущенных значений.
значимость	*число*	Уровень значимости для проверки гипотез при построении модели. Задает значение значимости для всех критериев в оценке моделей STP, в том числе для двух критериев согласия, F-критериев эффектов и t-критериев коэффициентов.
Вкладка **Вывод**		
model_specifications	*флаг*	
temporal_summary	*флаг*	
location_summary	*флаг*	Определяет, включена ли таблица Сводка положений в вывод модели.
model_quality	*флаг*	
test_mean_structure	*флаг*	

Таблица 146. свойства stpnode (продолжение)		
Свойства stpnode	**Тип переменной**	**Описание свойства**
mean_structure_coefficients	*флаг*	
autoregressive_coefficients	*флаг*	
test_decay_space	*флаг*	
parametric_spatial_covariance	*флаг*	
correlations_heat_map	*флаг*	
correlations_map	*флаг*	
location_clusters	*флаг*	
similarity_threshold	*число*	Порог, при котором кластеры вывода будут считаться достаточно подобными для слияния в один кластер.
max_number_clusters	*целое*	Верхний предел для числа кластеров, которые могут быть включены в вывод модели.
Вкладка **Опции модели**		
use_model_name	*флаг*	
model_name	*строка*	
uncertainty_factor	*число*	Минимум 0, максимум 100. Определяет увеличение неопределенности (ошибки), применяемой к предсказаниям в будущем. Это верхняя и нижняя граница для предсказаний.

Свойства svmnode

Узел механизма опорных векторов (Support Vector Machine, SVM) позволяет классифицировать данные по одной или двум группам без переобучения. SVM хорошо работает с широкими наборами данных, в частности, в случае очень большого числа входных полей.

Пример

```
node = stream.create("svm", "My node")
# Вкладка Эксперт
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Таблица 147. Свойства svmnode		
Свойства svmnode	**Значения**	**Описание свойства**
all_probabilities	*флаг*	

Таблица 147. Свойства svmnode (продолжение)		
Свойства svmnode	**Значения**	**Описание свойства**
stopping_criteria	1,0E-1 1,0E-2 1,0E-3 (по умолчанию) 1,0E-4 1,0E-5 1,0E-6	Определяет, когда остановить алгоритм оптимизации.
regularization	*number*	Известно также как C-параметр.
precision	*number*	Используется только в том случае, если уровень измерения поля назначения - Количественный.
kernel	RBF(default) Полиномиальный Сигмоид Линейная	Тип функции ядра, используемой для преобразования.
rbf_gamma	*number*	Используется только в том случае, если kernel - это RBF.
gamma	*number*	Используется только в том случае, если kernel - это Polynomial или Sigmoid.
bias	*number*	
degree	*number*	Используется только в том случае, если kernel - это Polynomial.
calculate_variable_importance	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	
adjusted_propensity_partition	Критерий Проверка	

Свойства tcmnode

При создании причинных моделей времени делается попытка обнаружить ключевую причинную взаимосвязь в данных ряда. При создании причинной модели времени вы задаете набор рядов назначения и набор входных рядов-кандидатов для этих назначений. Затем процедура строит авторегрессивную модель временного ряда для каждого назначения и включает только те входные ряды, у которых наиболее существенная причинная взаимосвязь с назначением.

Таблица 148. свойства tcmnode

Свойства tcmnode	**Значения**	**Описание свойства**
custom_fields	*Логический*	
dimensionlist	[*измерение1 ... измерениеN*]	
data_struct	Несколько Single	
metric_fields	*fields*	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Отметка времени Period	

Таблица 148. свойства tcmnode (продолжение)		
Свойства tcmnode	**Значения**	**Описание свойства**
input_interval	Нет Нет данных Год Квартал Месяц Неделя День Час Hour_nonperiod Минута Minute_nonperiod Секунда Second_nonperiod	
period_field	*строка*	
period_start_value	*целое*	
num_days_per_week	*целое*	
start_day_of_week	Воскресенье Понедельник Вторник Среда Четверг Пятница Суббота	
num_hours_per_day	*целое*	
start_hour_of_day	*целое*	
timestamp_increments	*целое*	

Таблица 148. свойства tcmnode (продолжение)		
Свойства tcmnode	**Значения**	**Описание свойства**
cyclic_increments	*целое*	
cyclic_periods	*список*	
output_interval	Нет Год Квартал Месяц Неделя День Час Минута Секунда	
is_same_interval	То же Notsame	
cross_hour	*Логический*	
aggregate_and_distribute	*список*	
aggregate_default	Mean Сумма Мода Минимум Максимум	
distribute_default	Mean Сумма	

Таблица 148. свойства tcmnode (продолжение)		
Свойства tcmnode	**Значения**	**Описание свойства**
group_default	Mean Сумма Мода Минимум Максимум	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend Нет	
k_mean_param	*целое*	
k_median_param	*целое*	
missing_value_threshold	*целое*	
conf_level	*целое*	
max_num_predictor	*целое*	
max_lag	*целое*	
epsilon	*number*	
threshold	*целое*	
is_re_est	*Логический*	
num_targets	*целое*	
percent_targets	*целое*	
fields_display	*список*	
series_display	*список*	
network_graph_for_target	*Логический*	
sign_level_for_target	*number*	
fit_and_outlier_for_target	*Логический*	
sum_and_para_for_target	*Логический*	
impact_diag_for_target	*Логический*	

Таблица 148. свойства tcmnode (продолжение)		
Свойства tcmnode	**Значения**	**Описание свойства**
impact_diag_type_for_target	Эффект Cause Both	
impact_diag_level_for_target	*целое*	
series_plot_for_target	*Логический*	
res_plot_for_target	*Логический*	
top_input_for_target	*Логический*	
forecast_table_for_target	*Логический*	
same_as_for_target	*Логический*	
network_graph_for_series	*Логический*	
sign_level_for_series	*number*	
fit_and_outlier_for_series	*Логический*	
sum_and_para_for_series	*Логический*	
impact_diagram_for_series	*Логический*	
impact_diagram_type_for_series	Эффект Cause Both	
impact_diagram_level_for_series	*целое*	
series_plot_for_series	*Логический*	
residual_plot_for_series	*Логический*	
forecast_table_for_series	*Логический*	
outlier_root_cause_analysis	*Логический*	
causal_levels	*целое*	
outlier_table	Интерактивно Сводная таблица Both	
rmsp_error	*Логический*	

Таблица 148. свойства tcmnode (продолжение)		
Свойства tcmnode	**Значения**	**Описание свойства**
bic	*Логический*	
r_square	*Логический*	
outliers_over_time	*Логический*	
series_transormation	*Логический*	
use_estimation_period	*Логический*	
estimation_period	Times Наблюдение	
observations	*список*	
observations_type	Последняя Самые ранние	
observations_num	*целое*	
observations_exclude	*целое*	
extend_records_into_future	*Логический*	
forecastperiods	*целое*	
max_num_distinct_values	*целое*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*Логический*	
aic	*Логический*	
rmse	*Логический*	

Свойства ts

Узел временных рядов оценивает экспоненциальное сглаживание, а также одномерные и многомерные модели авторегрессии и проинтегрированного скользящего среднего (Autoregressive Integrated Moving Average, ARIMA) для временных рядов и создает прогнозы будущего выполнения. Этот узел временных рядов подобен предыдущему узлу временных рядов, объявленному устаревшим в SPSS Modeler версии 18. Однако этот более новый узел временных рядов спроектирован так, чтобы использовать мощность IBM SPSS Analytic Server для обработки объемных данных и вывода полученной модели в программе просмотра вывода, которая была добавлена в SPSS Modeler версии 17.

Таблица 149. свойства ts		
Свойства ts	**Значения**	**Описание свойства**
targets	*поле*	Узел временных рядов прогнозирует значения одного или нескольких полей назначения, используя в качестве предикторов одно или несколько входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
candidate_inputs	[*поле1 ... полеN*]	Входные (или предикторные) поля, используемые в модели.
use_period	*флаг*	
date_time_field	*поле*	
input_interval	Нет Unknown Год Квартал Месяц Неделя День Час Hour_nonperiod Минута Minute_nonperiod Секунда Second_nonperiod	

Таблица 149. свойства ts (продолжение)		
Свойства ts	**Значения**	**Описание свойства**
period_field	*поле*	
period_start_value	*целое*	
num_days_per_week	*целое*	
start_day_of_week	Sunday Monday Вторник Среда Четверг Пятница Суббота	
num_hours_per_day	*целое*	
start_hour_of_day	*целое*	
timestamp_increments	*целое*	
cyclic_increments	*целое*	
cyclic_periods	*список*	
output_interval	Нет Год Квартал Месяц Неделя День Час Минута Секунда	
is_same_interval	*флаг*	
cross_hour	*флаг*	
aggregate_and_distribute	*список*	

Таблица 149. свойства ts (продолжение)

Свойства ts	**Значения**	**Описание свойства**
aggregate_default	Mean Сумма Мода Минимум Максимум	
distribute_default	Mean Сумма	
group_default	Mean Сумма Мода Минимум Максимум	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*целое*	
use_estimation_period	*флаг*	
estimation_period	Observations Times	
date_estimation	*список*	Доступно, только если используется date_time_field
period_estimation	*список*	Доступно, только если используется use_period
observations_type	Последняя Самые ранние	

Таблица 149. свойства ts (продолжение)

Свойства ts	**Значения**	**Описание свойства**
observations_num	*целое*	
observations_exclude	*целое*	
method	ExpertModeler Exsmooth Arima	
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*флаг*	
detect_outliers	*флаг*	
expert_outlier_additive	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_innovational	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_transient	*флаг*	
expert_outlier_seasonal_additive	*флаг*	
expert_outlier_local_trend	*флаг*	
expert_outlier_additive_patch	*флаг*	
consider_newesmodels	*флаг*	

Таблица 149. свойства ts (продолжение)

Свойства ts	Значения	Описание свойства
exsmooth_model_type	Простые HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	Задает метод экспоненциального сглаживания. Значение по умолчанию - Simple.

Таблица 149. свойства ts (продолжение)

Свойства ts	Значения	Описание свойства
futureValue_type_method	Compute specify	Если используется Compute, система вычисляет будущие значения для периода прогноза для каждого предиктора. Для каждого предиктора можно выбрать функцию в списке функций или выбрать specify, чтобы ввести значения вручную. Чтобы задать отдельные поля и свойства, используйте свойство extend_metric_values. Например: `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	Нет SquareRoot NaturalLog	
arima.p	*целое*	
arima.d	*целое*	
arima.q	*целое*	
arima.sp	*целое*	
arima.sd	*целое*	
arima.sq	*целое*	
arima_transformation_type	Нет SquareRoot NaturalLog	
arima_include_constant	*флаг*	

Таблица 149. свойства ts (продолжение)

Свойства ts	**Значения**	**Описание свойства**
tf_arima.p. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.d. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.q. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sp. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sd. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.sq. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.delay. *имя_поля*	*целое*	Для передаточных функций.
tf_arima.transformation_type. *имя_поля*	Нет SquareRoot NaturalLog	Для передаточных функций.
arima_detect_outliers	*флаг*	
arima_outlier_additive	*флаг*	
arima_outlier_level_shift	*флаг*	
arima_outlier_innovational	*флаг*	
arima_outlier_transient	*флаг*	
arima_outlier_seasonal_additive	*флаг*	
arima_outlier_local_trend	*флаг*	
arima_outlier_additive_patch	*флаг*	
max_lags	*целое*	
cal_PI	*флаг*	
conf_limit_pct	*real*	
события	*fields*	
continue	*флаг*	
scoring_model_only	*флаг*	Использовать для моделей с очень большим числом временных рядов (десятки тысяч).
forecastperiods	*целое*	
extend_records_into_future	*флаг*	

Таблица 149. свойства ts (продолжение)		
Свойства ts	**Значения**	**Описание свойства**
extend_metric_values	*fields*	Позволяет задать будущие значения для предикторов.
conf_limits	*флаг*	
noise_res	*флаг*	
max_models_output	*целое*	Управляет тем, сколько моделей показывается в выводе. Значение по умолчанию - 10. Модели не показываются в выводе, если полное число построенных моделей превосходит это значение. При этом модели остаются доступными для скоринга.

Свойства timeseriesnode (объявлено устаревшим)

Прим.: Этот исходный узел Временные ряды в версии 18 SPSS Modeler был объявлен устаревшим и заменен новым узлом Временные ряды, который спроектирован с учетом мощности IBM SPSS Analytic Server и обработки больших объемов данных.

Узел временных рядов оценивает экспоненциальное сглаживание, а также одномерные и многомерные модели авторегрессии и проинтегрированного скользящего среднего (Autoregressive Integrated Moving Average, ARIMA) для временных рядов и создает прогнозы будущего выполнения. Предшественником узла временных рядов всегда должен быть узел Интервалы времени.

Пример

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Таблица 150. Свойства timeseriesnode

Свойства timeseriesnode	**Значения**	**Описание свойства**
назначения	*поле*	Узел временных рядов прогнозирует значения одного или нескольких полей назначения, используя в качестве предикторов одно или несколько входных полей. Поля веса и частоты не используются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
continue	*флаг*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*флаг*	
consider_seasonal	*флаг*	
detect_outliers	*флаг*	
expert_outlier_additive	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_innovational	*флаг*	
expert_outlier_level_shift	*флаг*	
expert_outlier_transient	*флаг*	
expert_outlier_seasonal_additive	*флаг*	
expert_outlier_local_trend	*флаг*	
expert_outlier_additive_patch	*флаг*	

Таблица 150. Свойства timeseriesnode (продолжение)

Свойства timeseriesnode	**Значения**	**Описание свойства**
exsmooth_model_type	Простые HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	Нет SquareRoot NaturalLog	
arima_p	*целое*	
arima_d	*целое*	
arima_q	*целое*	
arima_sp	*целое*	
arima_sd	*целое*	
arima_sq	*целое*	
arima_transformation_type	Нет SquareRoot NaturalLog	
arima_include_constant	*флаг*	
tf_arima_p. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_d. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_q. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_sp. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_sd. *имя_поля*	*целое*	Для передаточных функций.

Таблица 150. Свойства timeseriesnode (продолжение)		
Свойства timeseriesnode	**Значения**	**Описание свойства**
tf_arima_sq. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_delay. *имя_поля*	*целое*	Для передаточных функций.
tf_arima_transformation_type. *имя_поля*	Нет SquareRoot NaturalLog	Для передаточных функций.
arima_detect_outlier_mode	Нет Автоматически	
arima_outlier_additive	*флаг*	
arima_outlier_level_shift	*флаг*	
arima_outlier_innovational	*флаг*	
arima_outlier_transient	*флаг*	
arima_outlier_seasonal_additive	*флаг*	
arima_outlier_local_trend	*флаг*	
arima_outlier_additive_patch	*флаг*	
conf_limit_pct	*real*	
max_lags	*целое*	
события	*fields*	
scoring_model_only	*флаг*	Использовать для моделей с очень большим числом временных рядов (десятки тысяч).

Свойства treeas

Узел деревьев-AS аналогичен существующему узлу CHAID, однако узел деревьев-AS сконструирован для обработки большого объема данных с целью создания одного дерева и выводит конечную модель в программе просмотра вывода, добавленной в SPSS Modeler версии 17. Этот узел генерирует дерево решений с использованием статистики хи-квадрат (CHAID) для определения оптимальных расщеплений. Такое использование CHAID может сгенерировать небинарные деревья, то есть у некоторых расщеплений может быть больше двух ветвей. Входные поля и поле назначения могут быть количественными (числовой диапазон) или категориальными. Исчерпывающий CHAID - это модификация метода CHAID, при котором проделывается более тщательная работа по изучению всех возможных расщеплений для каждого предиктора, но это требует больше времени для вычислений.

Таблица 151. свойства treeas		
Свойства treeas	**Значения**	**Описание свойства**
target	*поле*	На узле деревьев-AS моделям CHAID требуется одно поле назначения и одно или несколько входных полей. Может быть задано также поле частоты. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
method	chaid exhaustive_chaid	
max_depth	*целое*	Максимальное количество уровней в дереве, от 0 до 20. Значение по умолчанию - 5.
num_bins	*целое*	Используется только в том случае, когда данные представлены непрерывными входными полями. Задайте число равных интервалов частоты, которые будут использоваться для входных полей; возможны варианты 2, 4, 5, 10, 20, 25, 50 или 100.
record_threshold	*целое*	Количество записей, по которым при построении дерева модель будет переключаться с использования p-значений на размер эффекта. Значение по умолчанию - 1000000; увеличьте или уменьшите это значение с инкрементом 10000.
split_alpha	*number*	Уровень значимости для разбиения. Это значение должно находиться в диапазоне от 0,01 до 0,99.
merge_alpha	*number*	Уровень значимости для слияния. Это значение должно находиться в диапазоне от 0,01 до 0,99.
bonferroni_adjustment	*флаг*	Скорректировать уровни значимости, используя метод Бонферрони.
effect_size_threshold_cont	*number*	Задайте порог размера эффекта для расщепления полей и слияния категорий при использовании непрерывного поля назначения. Это значение должно находиться в диапазоне от 0,01 до 0,99.

Таблица 151. свойства treeas (продолжение)

Свойства treeas	**Значения**	**Описание свойства**
effect_size_threshold_cat	*number*	Задайте порог размера эффекта для расщепления полей и слияния категорий при использовании категориального поля назначения. Это значение должно находиться в диапазоне от 0,01 до 0,99.
split_merged_categories	*флаг*	Допускать разбиение объединенных категорий.
grouping_sig_level	*number*	Используется для определения, как формируются группы узлов и как идентифицируются необычные узлы.
chi_square	Пирсона likelihood_ratio	Используемый для вычисления статистики хи-квадрат метод: Пирсона или отношения правдоподобия
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*number*	Значение по умолчанию - 2. Минимальное значение - 1, максимальное - 100, инкремент 1. Значение для родительской ветви должно быть больше, чем для дочерней.
min_child_records_pc	*number*	Значение по умолчанию - 1. Минимальное значение - 1, максимальное - 100, инкремент 1.
min_parent_records_abs	*number*	Значение по умолчанию - 100. Минимальное значение 1, максимальное - 100, инкремент 1. Значение для родительской ветви должно быть больше, чем для дочерней.
min_child_records_abs	*number*	Значение по умолчанию - 50. Минимум 1, максимум 100 с шагом 1.
epsilon	*number*	Минимальное изменение ожидаемых частот в ячейках.
max_iterations	*number*	Максимум итераций до сходимости.
use_costs	*флаг*	

Таблица 151. свойства treeas (продолжение)		
Свойства treeas	**Значения**	**Описание свойства**
costs	*структурированный*	Структурированное свойство. Формат - это список из трех значений: фактическое значение, предсказанное значение и стоимость ошибочного предсказания. Например: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none Линейная square custom	**Прим.:** разрешено только для порядковых полей назначения. Задайте значения по умолчанию в матрице стоимостей.
calculate_conf	*флаг*	
display_rule_id	*флаг*	Добавляет поле в выводе данных скоринга, обозначающее ID для конечного узла, которому назначена каждая запись.

Свойства узла двухшаговых моделей (twostepnode)

Узел Двухшаговый использует метод двухшаговой кластеризации. На первом шаге проводится первый проход по данным, при котором необработанные входные данные сжимаются в управляемый набор подкластеров. На втором шаге используется способ иерархической кластеризации для все большего слияния подкластеров в крупные и еще более крупные кластеры. У двухшагового метода есть преимущество автоматической оценки оптимального числа кластеров для обучающих данных. Он может эффективно обрабатывать поля смешанных типов и большие наборы данных.

Пример

```
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)
```

Таблица 152. Свойства twostepnode		
Свойства twostepnode	**Значения**	**Описание свойства**
inputs	[*поле1 ... полеN*]	Двухшаговые модели используют список входных полей, но не поле назначения. Поля веса и частоты не распознаются. Дополнительную информацию смотрите в разделе “Общие свойства узлов моделирования” на стр. 229.
стандартизация	*флаг*	
exclude_outliers	*флаг*	
процент	*number*	
cluster_num_auto	*флаг*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	Строка Число	
label_prefix	*string*	
distance_measure	Евклидова Loglikelihood	
clustering_criterion	AIC BIC	

Свойства twostepAS

Кластерный анализ TwoStep - это инструмент разведочного анализа, предназначенный для выявления естественного разбиения набора данных на группы (или кластеры), которое без его применения трудно обнаружить. У используемого в этой процедуре алгоритма есть несколько ценных возможностей, отличающих его от традиционных способов кластеризации, таких как обработка категориальных и количественных переменных, автоматический выбор количества кластеров и масштабируемость.

Таблица 153. свойства twostepAS		
Свойства twostepAS	**Значения**	**Описание свойства**
inputs	[*f1 ... fN*]	Модели TwoStepAS используют список входных полей, но не полей назначения. Поля веса и частоты не распознаются.

Таблица 153. свойства twostepAS (продолжение)

Свойства twostepAS	**Значения**	**Описание свойства**
use_predefined_roles	Логическое значение	Значение по умолчанию=True
use_custom_field_assignments	Логическое значение	Значение по умолчанию=False
cluster_num_auto	Логическое значение	Значение по умолчанию=True
min_num_clusters	целое_число	Значение по умолчанию=2
max_num_clusters	целое_число	Значение по умолчанию=15
num_clusters	целое_число	Значение по умолчанию=5
clustering_criterion	AIC BIC	
automatic_clustering_method	использовать_параметр_критерия_кластеризации Скачок_расстояния Минимум Максимум	
feature_importance_method	использовать_параметр_критерия_кластеризации размер_эффекта	
use_random_seed	Логическое значение	
random_seed	целое_число	
distance_measure	Евклидова Loglikelihood	
include_outlier_clusters	Логическое значение	Значение по умолчанию=True
num_cases_in_feature_tree_leaf_is_less_than	целое_число	Значение по умолчанию=10
top_perc_outliers	целое_число	Значение по умолчанию=5
initial_dist_change_threshold	целое_число	Значение по умолчанию=0
leaf_node_maximum_branches	целое_число	Значение по умолчанию=8

Таблица 153. свойства twostepAS (продолжение)		
Свойства twostepAS	**Значения**	**Описание свойства**
non_leaf_node_maximum_branches	целое_число	Значение по умолчанию=8
max_tree_depth	целое_число	Значение по умолчанию=3
adjustment_weight_on_measurement_level	целое_число	Значение по умолчанию=6
memory_allocation_mb	число	Значение по умолчанию=512
delayed_split	Логическое значение	Значение по умолчанию=True
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	Логическое значение	Значение по умолчанию=True
featureMisPercent	целое_число	Значение по умолчанию=70
coefRange	число	Значение по умолчанию=0,05
percCasesSingleCategory	целое_число	Значение по умолчанию=95
numCases	целое_число	Значение по умолчанию=24
include_model_specifications	Логическое значение	Значение по умолчанию=True
include_record_summary	Логическое значение	Значение по умолчанию=True
include_field_transformations	Логическое значение	Значение по умолчанию=True
excluded_inputs	Логическое значение	Значение по умолчанию=True
evaluate_model_quality	Логическое значение	Значение по умолчанию=True
show_feature_importance bar chart	Логическое значение	Значение по умолчанию=True
show_feature_importance_ word_cloud	Логическое значение	Значение по умолчанию=True
show_outlier_clusters interactive_table_and_chart	Логическое значение	Значение по умолчанию=True
show_outlier_clusters_pivot_table	Логическое значение	Значение по умолчанию=True
across_cluster_feature_importance	Логическое значение	Значение по умолчанию=True

Таблица 153. свойства twostepAS (продолжение)		
Свойства twostepAS	**Значения**	**Описание свойства**
across_cluster_profiles_pivot_table	Логическое значение	Значение по умолчанию=True
withinprofiles	Логическое значение	Значение по умолчанию=True
cluster_distances	Логическое значение	Значение по умолчанию=True
cluster_label	Строка Число	
label_prefix	Строка	

Глава 14. Свойства узлов слепков моделей

У узлов слепков моделей те же общие свойства, что и у других узлов. Дополнительную информацию смотрите в разделе “Общие свойства узлов” на стр. 78.

Свойства применения узла обнаружения аномалий (applyanomalydetectionnode)

Узлы моделирования обнаружения аномалий можно использовать для генерирования слепка модели обнаружения аномалий. Имя сценария этого слепка модели - *applyanomalydetectionnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства anomalydetectionnode” на стр. 230

Таблица 154. Свойства applyanomalydetectionnode

Свойства applyanomalydetectionnode	**Значения**	**Описание свойства**
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Определяет, какие выходные значения создаются для скоринга.
num_fields	*целое*	Поля для отчета.
discard_records	*флаг*	Указывает, отбрасываются или нет записи из выходных данных.
discard_anomalous_records	*флаг*	Индикатор, какие записи отбрасывать - аномальные или *не* аномальные. Значение по умолчанию off соответствует отбрасыванию *не* аномальных полей. В противном случае, если задано on, будут отброшены аномальные записи. Это свойство включается только в том случае, если включено свойство discard_records.

Свойства применения узла Априори (applyapriorinode)

Узлы моделирования Априори можно использовать для генерирования слепка априорной модели. Имя сценария этого слепка модели - *applyapriorinode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства apriorinode” на стр. 232

Таблица 155. Свойства applyapriorinode

Свойства applyapriorinode	**Значения**	**Описание свойства**
max_predictions	*количество (целое число)*	
ignore_unmatached	*флаг*	
allow_repeats	*флаг*	

Таблица 155. Свойства applyapriorinode (продолжение)		
Свойства applyapriorinode	**Значения**	**Описание свойства**
check_basket	NoPredictions Predictions NoCheck	
criterion	Показатель доверия Поддержка RuleSupport Рост Deployability	

Свойства applyassociationrulesnode

При помощи узла моделирования Правила связывания можно сгенерировать слепок модели правил связывания. Имя сценария этого слепка модели - *applyassociationrulesnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства associationrulesnode” на стр. 233.

Таблица 156. свойства applyassociationrulesnode		
Свойства applyassociationrulesnode	**Тип переменной**	**Описание свойства**
max_predictions	*целое*	Максимальное число правил, которые могут быть применены к каждому элементу ввода для скоринга.
criterion	Показатель доверия Rulesupport Рост Conditionsupport Deployability	Выберите показатель для определения силы правил.
allow_repeats	*Логический*	Определите, включать ли в скоринг правила с одинаковым предсказанием.

Таблица 156. свойства applyassociationrulesnode (продолжение)		
Свойства applyassociationrulesnode	**Тип переменной**	**Описание свойства**
check_input	NoPredictions Predictions NoCheck	

Свойство применения узла автоклассификации (applyautoclassifiernode)

Узлы моделирования автоклассификации можно использовать для генерирования слепка модели автоклассификации. Имя сценария этого слепка модели - *applyautoclassifiernode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства autoclassifiernode” на стр. 237

Таблица 157. Свойства applyautoclassifiernode		
Свойства applyautoclassifiernode	**Значения**	**Описание свойства**
flag_ensemble_method	Голосование EvaluationWeightedVoting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Задает способ, используемый для определения оценки ансамбля. Этот параметр применяется только в том случае, если выбранное поле назначение - это флаговое поле.
flag_evaluation_selection	Accuracy AUC_ROC	Эта опция предназначена только для флаговых назначений и задает, какая мера оценки выбрана для голосования со взвешенной оценкой.
filter_individual_model_output	*флаг*	Задает, нужно ли отключать результаты скоринга из индивидуальных моделей.

Таблица 157. Свойства applyautoclassifiernode (продолжение)

Свойства applyautoclassifiernode	**Значения**	**Описание свойства**
is_ensemble_update	*флаг*	Включает режим непрерывного автоматического машинного обучения, который добавляет новые компонентные модели в существующий автоматический набор моделей вместо того, чтобы заменять существующую автоматическую модель, и переоценивает показатели существующих компонентных моделей с использованием новых доступных данных.
is_auto_ensemble_weights_reevaluation	*флаг*	Включает автоматическую переоценку весов моделей.
use_accumulated_factor	*флаг*	Накопленный фактор используется для вычисления накопленных показателей.
accumulated_factor	*число (двойной точности)*	Максимальное значение - 0.99, минимальное значение - 0.85.
use_accumulated_reducing	*флаг*	Выполняет редуцирование модели на основе накопленного предела при обновлении модели
accumulated_reducing_limit	*число (двойной точности)*	Максимальное значение - 0.7, минимальное значение - 0.1.
use_accumulated_weighted_evaluation	*флаг*	Накопленный показатель оценки используется для голосованияЮ когда в качестве метода ансамбля выбран метод голосования со взвешенными оценками.
flag_voting_tie_selection	Переменный HighestConfidence RawPropensity	Если выбран способ голосования, задает, как разрешаются связи. Этот параметр применим только в том случае, если выбранное поле назначения флаговое.
set_ensemble_method	Голосование EvaluationWeightedVoting ConfidenceWeightedVoting HighestConfidence	Задает способ, используемый для определения оценки ансамбля. Этот параметр применим только в том случае, если выбранное поле назначения - это поле набора.

Таблица 157. Свойства applyautoclassifiernode (продолжение)		
Свойства applyautoclassifiernode	**Значения**	**Описание свойства**
set_voting_tie_selection	Переменный HighestConfidence	Если выбран способ голосования, задает, как разрешаются связи. Этот параметр применим только в том случае, если выбранное поле назначения номинальное.

Свойства применения узла автокластеризации (applyautoclusternode)

Узлы моделирования автокластеризации можно использовать для генерирования слепка модели автокластеризации. Имя сценария этого слепка модели - *applyautoclusternode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства autoclusternode” на стр. 240

Свойства узла автонумерации (applyautonumericnode)

Узлы моделирования автонумерации можно использовать для генерирования слепка модели автонумерации. Имя сценария этого слепка модели - *applyautonumericnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства autonumericnode” на стр. 242

Таблица 158. Свойства applyautonumericnode		
Свойства applyautonumericnode	**Значения**	**Описание свойства**
calculate_standard_error	*флаг*	

Свойства применения узла Байесовской сети (applybayesnetnode)

Узлы моделирования Байесовской сети можно использовать для генерирования слепка модели Байесовской сети. Имя сценария этого слепка модели - *applybayesnetnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства bayesnetnode” на стр. 244.

Таблица 159. Свойства applybayesnetnode		
Свойства applybayesnetnode	**Значения**	**Описание свойства**
all_probabilities	*флаг*	
raw_propensity	*флаг*	
adjusted_propensity	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойство применения узла C5.0 (applyc50node)

Узлы моделирования C5.0 можно использовать для генерирования слепка модели C5.0. Имя сценария этого слепка модели - *applyc50node*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства c50node” на стр. 246.

Таблица 160. Свойства applyc50node

Свойства applyc50node	Значения	Описание свойства
sql_generate	udf Никогда NoMissingValues	Используется для задания опций генерирования SQL во время выполнения набора правил. Значение по умолчанию - udf.
calculate_conf	*флаг*	Доступно, когда включено генерирование SQL; это свойство включает в себя вычисления доверительных показателей в сгенерированном дереве.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства применения узла CARMA (applycarmanode)

Узлы моделирования CARMA можно использовать для генерирования слепка модели CARMA. Имя сценария этого слепка модели - *applycarmanode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства carmanode” на стр. 248.

Свойства применения узла CART (applycartnode)

Узлы моделирования дерева C&R можно использовать для генерирования слепка модели дерева C&R. Имя сценария этого слепка модели - *applycartnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства cartnode” на стр. 249.

Таблица 161. Свойства applycartnode

Свойства applycartnode	Значения	Описание свойства
enable_sql_generation	Никогда MissingValues NoMissingValues	Используется для задания опций генерирования SQL во время выполнения набора правил.
calculate_conf	*флаг*	Доступно, когда включено генерирование SQL; это свойство включает в себя вычисления доверительных показателей в сгенерированном дереве.

Таблица 161. Свойства applycartnode (продолжение)		
Свойства applycartnode	**Значения**	**Описание свойства**
display_rule_id	*флаг*	Добавляет поле в выводе данных скоринга, обозначающее ID для конечного узла, которому назначена каждая запись.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства применения узла CHAID (applychaidnode)

Узлы моделирования CHAID можно использовать для генерирования слепка модели CHAID. Имя сценария этого слепка модели - *applychaidnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства chaidnode” на стр. 252.

Таблица 162. Свойства applychaidnode		
свойства applychaidnode	**Значения**	**Описание свойства**
enable_sql_generation	Никогда MissingValues	Используется для задания опций генерирования SQL во время выполнения набора правил.
calculate_conf	*флаг*	
display_rule_id	*флаг*	Добавляет поле в выводе данных скоринга, обозначающее ID для конечного узла, которому назначена каждая запись.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства узла применения регрессии Кокса (applycoxregnode)

Узлы моделирования Кокса можно использовать для генерации слепка модели Кокса. Имя сценария этого слепка модели - *applycoxregnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства coxregnode” на стр. 255.

Таблица 163. Свойства applycoxregnode		
Свойства applycoxregnode	**Значения**	**Описание свойства**
future_time_as	Интервалы Поля	
time_interval	*number*	
num_future_times	*целое*	
time_field	*поле*	

Таблица 163. Свойства applycoxregnode (продолжение)		
Свойства applycoxregnode	**Значения**	**Описание свойства**
past_survival_time	*поле*	
all_probabilities	*флаг*	
cumulative_hazard	*флаг*	

Свойства применения узла списка решений (applydecisionlistnode)

Узлы моделирования списка решений можно использовать для генерирования слепка модели списка решений. Имя сценария этого слепка модели - *applydecisionlistnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства decisionlistnode” на стр. 257.

Таблица 164. Свойства applydecisionlistnode		
Свойства applydecisionlistnode	**Значения**	**Описание свойства**
enable_sql_generation	*флаг*	При значении true IBM SPSS Modeler будет стараться продвинуть модель списка решений обратно в SQL.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства применения узла дискриминанта (applydiscriminantnode)

Узлы моделирования дискриминанта можно использовать для генерирования слепка модели дискриминанта. Имя сценария этого слепка модели - *applydiscriminantnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства discriminantnode” на стр. 259.

Таблица 165. Свойства applydiscriminantnode		
Свойства applydiscriminantnode	**Значения**	**Описание свойства**
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства applyextension

Узлы Extension Model можно использовать для генерирования слепка модели Extension. Имя сценария этого слепка модели - *applyextension*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства extensionmodelnode” на стр. 261.

Пример Python for Spark

```
#### пример сценария для 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))

    # преобразование рамки данных в RDD, содержащий LabeledPoint
    lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
    treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
    # скоринг модели, генерирует RDD, содержащий только значения-дубликаты
    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)
```

R example

```
#### 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)""")
```

Таблица 166. свойства applyextension

Свойства applyextension	Значения	Описание свойства
r_syntax	*строка*	Синтаксис сценария R для скоринга моделей.
python_syntax	*строка*	Синтаксис сценария Python для скоринга моделей.
use_batch_size	*флаг*	Включить использование пакетной обработки.
batch_size	*целое*	Укажите количество записей данных, которые будут включаться в каждый пакет.
convert_flags	`StringsAndDoubles` `LogicalValues`	Опция для преобразования флаговых полей.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	`POSIXct` `POSIXlt`	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.

Свойства применения узла факторов (applyfactornode)

Узлы моделирования PCA/факторов можно использовать для генерирования слепка модели PCA/факторов. Имя сценария этого слепка модели - *applyfactornode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства factornode” на стр. 263.

Свойства применения узла выбора возможностей (applyfeatureselectionnode)

Узлы моделирования выбора возможностей можно использовать для генерирования слепка модели выбора возможностей. Имя сценария этого слепка модели - *applyfeatureselectionnode*.

Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства featureselectionnode” на стр. 266.

Таблица 167. Свойства applyfeatureselectionnode

Свойства applyfeatureselectionnode	**Значения**	**Описание свойства**
selected_ranked_fields		Задает, какие ранжированные поля отмечаются для выбора в браузере моделей.
selected_screened_fields		Задает, какие экранированные поля отмечены для выбора в браузере моделей.

Свойства применения узла обобщенной линейной регрессии (applygeneralizedlinearnode)

Узлы моделирования обобщенной линейной регрессии (genlin) можно использовать для генерирования слепка обобщенной линейной модели. Имя сценария этого слепка модели - *applygeneralizedlinearnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства genlinnode” на стр. 268.

Таблица 168. Свойства applygeneralizedlinearnode

Свойства applygeneralizedlinearnode	**Значения**	**Описание свойства**
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства применения узла GLMM (applyglmmnode)

Узлы моделирования GLMM можно использовать для генерирования слепка модели GLMM. Имя сценария этого слепка модели - *applyglmmnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства glmmnode” на стр. 274.

Таблица 169. Свойства applyglmmnode

Свойства applyglmmnode	**Значения**	**Описание свойства**
confidence	onProbability onIncrease	Основание для вычисления доверительного значения оценки: максимальная предсказанная вероятность или разность между максимальной и второй по значению предсказанной вероятностью.
score_category_probabilities	*флаг*	При значении True создает предсказанные вероятности для категориальных полей назначения. Для каждой категории создается поле. Значение по умолчанию - False.

Таблица 169. Свойства applyglmmnode (продолжение)		
Свойства applyglmmnode	**Значения**	**Описание свойства**
max_categories	*целое*	Максимальное количество категорий, для которых будут предсказываться вероятности. Используется только в том случае, когда значение score_category_probabilities - это True.
score_propensity	*флаг*	Если задано значение True, создает простые оценки склонности (правдоподобие выхода "True") для моделей с флаговыми полями назначения. Если используются разделы, создаются также скорректированные оценки склонности на основании обучающего раздела. Значение по умолчанию - False.
enable_sql_generation	udf native	Используется для задания опций генерирования SQL во время выполнения потока. Возможные варианты - обратный перенос в базу данных и скоринг при помощи адаптера скоринга SPSS® Modeler Server (при соединении с базой данных с установленным адаптером скоринга) либо скоринг в SPSS Modeler. Значение по умолчанию - udf.

Свойства applygle

Узел моделирования GLE можно использовать для генерирования слепка модели GLE. Имя сценария этого слепка модели - *applygle*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства gle” на стр. 280.

Таблица 170. Свойства applygle		
Свойства applygle	**Значения**	**Описание свойства**
enable_sql_generation	udf native	Используется для задания опций генерирования SQL во время выполнения потока. Choose either to pushback to the database and score using a SPSS Modeler Server scoring adapter (if connected to a database with a scoring adapter installed), or score within SPSS Modeler.

Свойства applygmm

Смешанный Гауссов узел можно использовать для генерирования слепка смешанной Гауссовой модели. Имя сценария этого слепка модели - *applygmm*. Свойства в следующей таблице доступны в

версии 18.2.1.1 и новее. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства gmm” на стр. 457.

Таблица 171. Свойства applygmm

Свойства applygmm	Тип переменной	Описание свойства
центры		
item_count		
итог		
измерение		
компоненты		
partition		

Свойства применения узла k-средних (applykmeansnode)

Узлы моделирования K-средних можно использовать для генерирования слепка модели K-средних. Имя сценария этого слепка модели - *applykmeansnode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства kmeansnode” на стр. 288.

Свойства применения узла KNN (applyknnnode)

Узлы моделирования KNN можно использовать для генерирования слепка модели KNN. Имя сценария этого слепка модели - *applyknnnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства knnnode” на стр. 291.

Таблица 172. Свойства applyknnnode

Свойства applyknnnode	Значения	Описание свойства
all_probabilities	*флаг*	
save_distances	*флаг*	

Свойства применения узла Коонена (applykohonennode)

Узлы моделирования Коонена можно использовать для генерирования слепка модели Коонена. Имя сценария этого слепка модели - *applykohonennode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства c50node” на стр. 246.

Свойства применения узла линейных моделей (applylinearnode)

Узлы линейного моделирования можно использовать для генерирования слепка линейной модели. Имя сценария этого слепка модели - *applylinearnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства linearnode” на стр. 294.

Таблица 173. Свойства applylinearnode

Свойства линейных моделей	Значения	Описание свойства
use_custom_name	*флаг*	
custom_name	*string*	

Таблица 173. Свойства applylinearnode (продолжение)		
Свойства линейных моделей	**Значения**	**Описание свойства**
enable_sql_generation	udf native puresql	Используется для задания опций генерирования SQL во время выполнения потока. К этим опциям относится обратный перенос в базу данных и скоринг при помощи адаптера скоринга сервера SPSS® Modeler (при соединении с базой данных с установленным адаптером скоринга), скоринг в SPSS Modeler или обратный перенос в базу данных и скоринг с использованием SQL. Значение по умолчанию - udf.

Свойства applylinearasnode

Узлы линейных-AS моделей можно использовать для генерирования слепка линейной-AS модели. Имя сценария этого слепка модели - *applylinearasnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства linearasnode” на стр. 296.

Таблица 174. Свойства applylinearasnode		
Свойство applylinearasnode	**Значения**	**Описание свойства**
enable_sql_generation	udf native	Значение по умолчанию - udf.

Свойства применения узла логистической регрессии (applylogregnode)

Узлы моделирования логистической регрессии можно использовать для генерирования слепка модели логистической регрессии. Имя сценария этого слепка модели - *applylogregnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства logregnode” на стр. 298.

Таблица 175. Свойства applylogregnode		
Свойства applylogregnode	**Значения**	**Описание свойства**
calculate_raw_propensities	*флаг*	
calculate_conf	*флаг*	
enable_sql_generation	*флаг*	

Свойства applylsvmnode

Узлы моделирования LSVM можно использовать для генерирования слепка модели LSVM. Имя сценария этого слепка модели - *applylsvmnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Матрица ошибок, иногда называемая таблицей сводки,

содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.” на стр. 304.

Таблица 176. Свойства applylsvmnode		
Свойства applylsvmnode	**Значения**	**Описание свойства**
calculate_raw_propensities	*флаг*	Указывает, вычислять ли простые оценки склонности.
enable_sql_generation	udf native	Указывает, как выполнять скоринг, - при помощи адаптера скоринга (если он установлен), в процессе или вне базы данных.

Свойства applyneuralnetnode

Узлы моделирования нейронной сети можно использовать для генерирования слепка модели нейронной сети. Имя сценария этого слепка модели - *applyneuralnetnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства neuralnetnode” на стр. 305.

Внимание: В этом выпуске доступна более новая версия узла моделирования нейронных сетей с расширенными возможностями, которая обсуждается в следующем разделе (*applyneuralnetwork*). Несмотря на то, что предыдущая версия все еще доступна, мы рекомендуем обновить ваши сценарии для использования новой версии. Подробности предыдущей версии приведены здесь для справки, но ее поддержка будет прекращена в следующем выпуске.

Таблица 177. Свойства applyneuralnetnode		
Свойства applyneuralnetnode	**Значения**	**Описание свойства**
calculate_conf	*флаг*	Доступно, когда включено генерирование SQL; это свойство включает в себя вычисления доверительных показателей в сгенерированном дереве.
enable_sql_generation	*флаг*	
nn_score_method	Разность SoftMax	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства applyneuralnetworknode

Узлы моделирования нейронной сети можно использовать для генерирования слепка модели нейронной сети. Имя сценария этого слепка модели - *applyneuralnetworknode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе Свойства neuralnetworknode.

Таблица 178. Свойства applyneuralnetworknode		
Свойства applyneuralnetworknode	**Значения**	**Описание свойства**
use_custom_name	*флаг*	
custom_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*флаг*	
max_categories	*number*	
score_propensity	*флаг*	
enable_sql_generation	udf native puresql	Используется для задания опций генерирования SQL во время выполнения потока. К этим опциям относится обратный перенос в базу данных и скоринг при помощи адаптера скоринга сервера SPSS® Modeler (при соединении с базой данных с установленным адаптером скоринга), скоринг в SPSS Modeler или обратный перенос в базу данных и скоринг с использованием SQL. Значение по умолчанию - udf.

Свойства applyocsvmnode

Узлы One-Class SVM можно использовать для генерирования слепка модели One-Class SVM. Имя сценария этого слепка модели - *applyocsvmnode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “свойства ocsvmnode” на стр. 463.

Свойства применения узла QUEST (applyquestnode)

Узлы моделирования QUEST можно использовать для генерирования слепка модели QUEST. Имя сценария этого слепка модели - *applyquestnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства questnode” на стр. 310.

Таблица 179. Свойства applyquestnode		
Свойства applyquestnode	**Значения**	**Описание свойства**
enable_sql_generation	Никогда MissingValues NoMissingValues	Используется для задания опций генерирования SQL во время выполнения набора правил.
calculate_conf	*флаг*	

Таблица 179. Свойства applyquestnode (продолжение)		
Свойства applyquestnode	**Значения**	**Описание свойства**
display_rule_id	*флаг*	Добавляет поле в выводе данных скоринга, обозначающее ID для конечного узла, которому назначена каждая запись.
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства applyr

Узлы моделирования R можно использовать для генерирования слепка модели R. Имя сценария этого слепка модели - *applyr*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “свойства buildr” на стр. 245.

Таблица 180. свойства applyr		
Свойства applyr	**Значения**	**Описание свойства**
score_syntax	*строка*	Синтаксис сценария R для скоринга моделей.
convert_flags	StringsAndDoubles LogicalValues	Опция для преобразования флаговых полей.
convert_datetime	*флаг*	Опции для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	POSIXct POSIXlt	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.
convert_missing	*флаг*	Опция для преобразования значений отсутствия в значение R NA.
use_batch_size	*флаг*	Включить использование пакетной обработки
batch_size	*целое*	Укажите количество записей данных, которые будут включаться в каждый пакет

Свойства applyrandomtrees

Узел моделирования случайных деревьев можно использовать для генерирования слепка модели случайных деревьев. Имя сценария этого слепка модели - *applyrandomtrees*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства randomtrees” на стр. 313.

Таблица 181. Свойства applyrandomtrees		
Свойства applyrandomtrees	**Значения**	**Описание свойства**
calculate_conf	*флаг*	Это свойство включает в себя вычисления доверительных показателей в сгенерированном дереве.
enable_sql_generation	udf native	Используется для задания опций генерирования SQL во время выполнения потока. Choose either to pushback to the database and score using a SPSS Modeler Server scoring adapter (if connected to a database with a scoring adapter installed), or score within SPSS Modeler.

Свойства применения узла регрессии (applyregressionnode)

Узлы моделирования линейной регрессии можно использовать для генерирования слепка модели линейной регрессии. Имя сценария этого слепка модели - *applyregressionnode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства regressionnode” на стр. 315.

Свойства applyselflearningnode

Узлы моделирования откликов самообучения (Self-Learning Response Model, SLRM) можно использовать для генерирования слепка модели SLRM. Имя сценария этого слепка модели - *applyselflearningnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства slrmnode” на стр. 319.

Таблица 182. Свойства applyselflearningnode		
Свойства applyselflearningnode	**Значения**	**Описание свойства**
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	ascending descending	Задает, в каком порядке будут показываться предложения, начиная с максимальной или минимальной оценки.
model_reliability	*флаг*	Принимает во внимание опцию надежности модели на вкладке Параметры.

Свойства применения узла последовательности (applysequencenode)

Узлы моделирования последовательности можно использовать для генерации слепка модели последовательности. Имя сценария этого слепка модели - *applysequencenode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства sequencenode” на стр. 318.

Свойства применения узла SVM (applysvmnode)

Узлы моделирования SVM можно использовать для генерирования слепка модели SVM. Имя сценария этого слепка модели - *applysvmnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства svmnode” на стр. 327.

Таблица 183. Свойства applysvmnode

Свойства applysvmnode	Значения	Описание свойства
all_probabilities	*флаг*	
calculate_raw_propensities	*флаг*	
calculate_adjusted_propensities	*флаг*	

Свойства applystpnode

При помощи узла STP (Spatio-Temporal Prediction - пространственно-временное предсказание) можно сгенерировать связанный слепок модели, представляющий вывод модели в средстве просмотра вывода. Имя сценария этого слепка модели - *applystpnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства stpnode” на стр. 321.

Таблица 184. Свойства applystpnode

Свойства applystpnode	Тип переменной	Описание свойства
uncertainty_factor	*Логический*	Минимум 0, максимум 100.

Свойства applytcmnode

Узлы моделирования TCM (Temporal Causal Modeling) можно использовать для генерирования слепка модели TCM. Имя сценария этого слепка модели - *applytcmnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства tcmnode” на стр. 329.

Таблица 185. Свойства applytcmnode

Свойства applytcmnode	Значения	Описание свойства
ext_future	*логический*	
ext_future_num	*целое*	
noise_res	*логический*	
conf_limits	*логический*	
target_fields	*список*	
target_series	*список*	

Свойства applyts

Узел моделирования временных рядов можно использовать для генерирования слепка модели временных рядов.4 Имя сценария этого слепка модели - *applyts*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства ts” на стр. 334.

Таблица 186. Свойства applyts		
Свойства applyts	**Значения**	**Описание свойства**
extend_records_into_future	*Логический*	
ext_future_num	*целое*	
compute_future_values_input	*Логический*	
forecastperiods	*целое*	
noise_res	*логический*	
conf_limits	*логический*	
target_fields	*список*	
target_series	*список*	
includeTargets	*поле*	

Свойства applytimeseriesnode (объявлено устаревшим)

Узел моделирования временных рядов можно использовать для генерирования слепка модели временных рядов.4 Имя сценария этого слепка модели - *applytimeseriesnode*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства timeseriesnode (объявлено устаревшим)” на стр. 342.

Таблица 187. Свойства applytimeseriesnode		
Свойства applytimeseriesnode	**Значения**	**Описание свойства**
calculate_conf	*флаг*	
calculate_residuals	*флаг*	

Свойства applytreeas

Узлы моделирования деревьев-AS можно использовать для генерации слепка модели деревьев-AS. Имя сценария этого слепка модели - *applytreenas*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства treeas” на стр. 345.

Таблица 188. Свойства applytreeas		
Свойства applytreeas	**Значения**	**Описание свойства**
calculate_conf	*флаг*	Это свойство включает в себя вычисления доверительных показателей в сгенерированном дереве.
display_rule_id	*флаг*	Добавляет поле в выводе данных скоринга, обозначающее ID для конечного узла, которому назначена каждая запись.

Таблица 188. Свойства applytreeas (продолжение)		
Свойства applytreeas	**Значения**	**Описание свойства**
enable_sql_generation	udf native	Используется для задания опций генерирования SQL во время выполнения потока. Choose either to pushback to the database and score using a SPSS Modeler Server scoring adapter (if connected to a database with a scoring adapter installed), or score within SPSS Modeler.

Свойства применения узла двухшаговых моделей (applytwostepnode)

Узлы двухшагового моделирования можно использовать для генерирования слепка двухшаговой модели. Имя сценария этого слепка модели - *applytwostepnode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства узла двухшаговых моделей (twostepnode)” на стр. 348.

Свойства applytwostepAS

Узлы двухшагового моделирования AS можно использовать для генерирования слепка двухшаговой модели AS. Имя сценария этого слепка модели - *applytwostepAS*. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства twostepAS” на стр. 349.

Таблица 189. Свойства applytwostepAS		
Свойства applytwostepAS	**Значения**	**Описание свойства**
enable_sql_generation	udf native	Используется для задания опций генерирования SQL во время выполнения потока. Возможные варианты - обратный перенос в базу данных и скоринг при помощи адаптера скоринга SPSS® Modeler Server (при соединении с базой данных с установленным адаптером скоринга) либо скоринг в SPSS Modeler. Значение по умолчанию - udf.

Свойства applyxgboosttreenode

Узел Дерево XGBoost можно использовать для генерирования слепка модели дерева XGBoost. Имя сценария этого слепка модели - *applyxgboosttreenode*. Свойства в следующей таблице добавлены в 18.2.1.1. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства xgboosttreenode” на стр. 472.

Таблица 190. свойства applyxgboosttreenode		
Свойства applyxgboosttreenode	**Тип переменной**	**Описание свойства**
use_model_name		

Таблица 190. свойства applyxgboosttreenode (продолжение)		
Свойства applyxgboosttreenode	**Тип переменной**	**Описание свойства**
model_name		

Свойства applyxgboostlinearnode

Узлы Линейный XGBoost можно использовать для генерирования слепка модели Линейный XGBoost. Имя сценария этого слепка модели - *applyxgboostlinearnode*. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства xgboostlinearnode” на стр. 471.

Свойства hdbscannugget

Узел HDBSCAN можно использовать для генерирования слепка модели HDBSCAN. Имя сценария этого слепка модели - hdbscannugget. Других свойств для этого слепка модели нет. Более подробную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства hdbscannode” на стр. 458.

Свойства kdeapply

Узел моделирования KDE можно использовать для генерирования слепка модели KDE. Имя сценария этого слепка модели - kdeapply. Дополнительную информацию о сценарии самого узла моделирования смотрите в разделе “Свойства kdemodel” на стр. 460.

Таблица 191. Свойства kdeapply		
Свойства kdeapply	**Тип переменной**	**Описание свойства**
outLogDensity Переименовано в out_log_density, начиная с версии 18.2.1.1	*логический*	Укажите значение True или False, чтобы включить или выключить значение плотности записи в журнал в выводе. Значение по умолчанию - False.

Глава 15. Свойства узла моделирования баз данных

IBM SPSS Modeler поддерживает интеграцию с исследованием данных и инструментами моделирования, доступными у поставщиков баз данных, в том числе Microsoft SQL Server Analysis Services, Oracle Data Mining и IBM Netezza Analytics. Вы можете построить и оценить модели с помощью собственных алгоритмов баз данных изнутри прикладной программы IBM SPSS Modeler. Модели баз данных можно создавать также и работать с ними через сценарии, используя описанные в этом разделе свойства.

Например, следующий отрывок сценария иллюстрирует создание модели деревьев решений Microsoft с использованием интерфейса сценариев 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([])
```

Свойства узлов для моделирования Microsoft

Свойства узлов моделирования Microsoft

Общие свойства

Следующие свойства общие для узлов моделирования баз данных Microsoft.

Таблица 192. Общие свойства узлов Microsoft

Общие свойства узлов Microsoft	Значения	Описание свойства
analysis_database_name	*string*	Имя базы данных Analysis Services.
analysis_server_name	*string*	Имя хоста Analysis Services.
use_transactional_data	*флаг*	Задает, в каком формате используются входные данные, в табличном или транзакционном.
inputs	*список*	Входные поля для табличных данных.

Таблица 192. Общие свойства узлов Microsoft (продолжение)		
Общие свойства узлов Microsoft	**Значения**	**Описание свойства**
target	*поле*	Предсказанное поле (неприменимо для узлов кластеризации MS или кластеризации последовательностей).
unique_field	*поле*	Ключевое поле.
msas_parameters	*структурированный*	Параметры алгоритмов. Дополнительную информацию смотрите в разделе “Параметры алгоритма” на стр. 377.
with_drillthrough	*флаг*	С опцией детализации.

Дерево решений MS

Не существует специфических свойств, определенных для узлов типа mstreenode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Кластеризация MS

Не существует специфических свойств, определенных для узлов типа msclusternode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Правила связывания MS

Следующие конкретные свойства доступны для узлов типа msassocnode:

Таблица 193. Свойства msassocnode		
Свойства msassocnode	**Значения**	**Описание свойства**
id_field	*поле*	Идентифицирует каждую транзакцию в данных.
trans_inputs	*список*	Входные поля для транзакционных данных.
transactional_target	*поле*	Предсказанное поле (транзакционные данные).

Наивный критерий Байеса MS

Не существует специфических свойств, определенных для узлов типа msbayesnode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Линейная регрессия MS

Не существует специфических свойств, определенных для узлов типа msregressionnode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Нейросеть MS

Не существует специфических свойств, определенных для узлов типа msneuralnetworknode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Логистическая регрессия MS

Не существует специфических свойств, определенных для узлов типа mslogisticnode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Временные ряды MS

Не существует специфических свойств, определенных для узлов типа mstimeseriesnode. Смотрите описание общих свойств Microsoft в начале этого раздела.

Кластеризация последовательностей MS

Следующие конкретные свойства доступны для узлов типа mssequenceclusternode:

Таблица 194. Свойства mssequenceclusternode

Свойства mssequenceclusternode	**Значения**	**Описание свойства**
id_field	*поле*	Идентифицирует каждую транзакцию в данных.
input_fields	*список*	Входные поля для транзакционных данных.
sequence_field	*поле*	Идентификатор последовательности.
target_field	*поле*	Предсказанное поле (табличные данные).

Параметры алгоритма

У каждого типа моделей баз данных Microsoft есть специфический параметр, который можно задать с использованием свойства msas_parameters, например:

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters",
[["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Эти параметры получены от SQL Server. Чтобы увидеть соответствующие параметры для каждого узла:

1. Поместите узел источника базы данных на холст.
2. Откройте узел источника базы данных.
3. Выберите допустимый источник из раскрывающегося списка **Источник данных**.
4. Выберите допустимую таблицу из списка **Имя таблицы**.
5. Нажмите кнопку **ОК**, чтобы закрыть узел источника базы данных.
6. Присоедините узел моделирования базы данных Microsoft, свойства которого вы хотите перечислить.
7. Откройте узел моделирования баз данных.
8. Выберите вкладку **Эксперт**.

Будут выведены доступные свойства msas_parameters для этого узла.

Свойства слепков моделей Microsoft

Следующие свойства предназначены для слепков моделей, созданных с использованием узлов моделирования баз данных Microsoft.

Дерево решений MS

Таблица 195. Свойства дерева решений MS

Свойства applymstreenode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.
datasource	*строка*	Имя источника данных (data source name, DSN) SQL Server ODBC.
sql_generate	*флаг* udf	Включает генерирование SQL.

Линейная регрессия MS

Таблица 196. Свойства линейной регрессии MS

Свойства applymsregressionnode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.

Нейросеть MS

Таблица 197. Свойства нейронной сети MS

Свойства applymsneuralnetworknode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.

Логистическая регрессия MS

Таблица 198. Свойства логистической регрессии MS		
Свойства applymslogisticnode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.

Временные ряды MS

Таблица 199. Свойства MS Time Series		
Свойства applymstimeseriesnode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.
start_from	new_prediction `historical_prediction`	Задает, какие предсказания будут делаться, будущие или хронологические.
new_step	*number*	Определяет начальный период времени для будущих предсказаний.
historical_step	*число*	Определяет начальный период времени для будущих предсказаний.
end_step	*число*	Определяет конечный период времени для предсказаний.

Кластеризация последовательностей MS

Таблица 200. Свойства кластеризации последовательностей MS		
Свойства applymssequenceclusternode	**Значения**	**Описание**
analysis_database_name	*строка*	Этот узел может быть оценен непосредственно в потоке. Это свойство используется для идентификации имени базы данных Analysis Services.
analysis_server_name	*строка*	Имя хоста сервера анализа.

Свойства узлов для моделирования Oracle

Свойства узлов моделирования Oracle

Следующие свойства общие для узлов моделирования баз данных Oracle.

Таблица 201. Общие свойства узлов Oracle		
Общие свойства узлов Oracle	**Значения**	**Описание свойства**
target	*поле*	
inputs	*Список полей*	
partition	*поле*	Поле, которое будет использоваться для разделения данных на отдельные выборки для стадий обучения, испытания и проверки при построении моделей.
datasource		
username		
password		
epassword		
use_model_name	*флаг*	
model_name	*string*	Пользовательское имя для новой модели.
use_partitioned_data	*флаг*	Если определено поле раздела, эта опция обеспечивает, что для построения модели используются только данные из раздела обучения.
unique_field	*поле*	
auto_data_prep	*флаг*	Включает или отключает возможность автоматической подготовки данных (только для баз данных 11g).

Таблица 201. Общие свойства узлов Oracle (продолжение)		
Общие свойства узлов Oracle	**Значения**	**Описание свойства**
costs	*структурированный*	Структурированное свойство в следующей форме: [[drugA drugB 1.5] [drugA drugC 2.1]], где аргументы в квадратных скобках [] - это фактические предсказанные стоимости.
mode	Простые Эксперт	Приводит к тому, что при заданной опции Простые некоторые свойства игнорируются, как отмечено в свойствах конкретных узлов.
use_prediction_probability	*флаг*	
prediction_probability	*string*	
use_prediction_set	*флаг*	

Наивный критерий Байеса Oracle

Следующие свойства доступны для узлов типа oranbnode.

Таблица 202. Свойства oranbnode		
Свойства oranbnode	**Значения**	**Описание свойства**
singleton_threshold	*число*	0,0–1,0.*
pairwise_threshold	*число*	0,0–1,0.*
priors	Data Equal Пользовательские	
custom_priors	*структурированный*	Структурированное свойство в следующей форме: set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Свойства, которые игнорируются, если для mode задано значение Простые.

Адаптивный критерий Байеса Oracle

Следующие свойства доступны для узлов типа oraabnnode.

Таблица 203. Свойства oraabnnode		
Свойства oraabnnode	**Значения**	**Описание свойства**
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*флаг*	*
execution_time_limit	*целое*	Значение должно быть больше 0,*
max_naive_bayes_predictors	*целое*	Значение должно быть больше 0,*
max_predictors	*целое*	Значение должно быть больше 0,*
priors	Data Equal Пользовательские	
custom_priors	*структурированный*	Структурированное свойство в следующей форме: set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Свойства, которые игнорируются, если для mode задано значение Простые.

Механизмы опорных векторов Oracle

Следующие свойства доступны для узлов типа orasvmnode.

Таблица 204. Свойства orasvmnode		
Свойства orasvmnode	**Значения**	**Описание свойства**
active_learning	Enable Отключить	
kernel_function	Линейная Gaussian Системная	
normalization_method	zscore minmax none	

Таблица 204. Свойства orasvmnode (продолжение)		
Свойства orasvmnode	**Значения**	**Описание свойства**
kernel_cache_size	*целое*	Только гауссовское ядро. Значение должно быть больше 0,*
convergence_tolerance	*число*	Значение должно быть больше 0,*
use_standard_deviation	*флаг*	Только гауссовское ядро.*
standard_deviation	*число*	Значение должно быть больше 0,*
use_epsilon	*флаг*	Только модели регрессии.*
epsilon	*число*	Значение должно быть больше 0,*
use_complexity_factor	*флаг*	*
complexity_factor	*число*	*
use_outlier_rate	*флаг*	Только вариант одного класса.*
outlier_rate	*число*	Только вариант одного класса. 0,0–1,0.*
weights	Data Equal Пользовательские	
custom_weights	*структурированный*	Структурированное свойство в следующей форме: set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* Свойства, которые игнорируются, если для mode задано значение Простые.

Обобщенные линейные модели Oracle

Следующие свойства доступны для узлов типа oraglmnode.

Таблица 205. Свойства oraglmnode		
Свойства oraglmnode	**Значения**	**Описание свойства**
normalization_method	zscore minmax none	

Таблица 205. Свойства oraglmnode (продолжение)		
Свойства oraglmnode	**Значения**	**Описание свойства**
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*флаг*	*
row_weights_field	*поле*	*
save_row_diagnostics	*флаг*	*
row_diagnostics_table	*string*	*
coefficient_confidence	*число*	*
use_reference_category	*флаг*	*
reference_category	*string*	*
ridge_regression	Авто Off Вкл	*
parameter_value	*число*	*
vif_for_ridge	*флаг*	*

* Свойства, которые игнорируются, если для mode задано значение Простые.

Дерево решений Oracle

Следующие свойства доступны для узлов типа oradecisiontreenode.

Таблица 206. Свойства oradecisiontreenode		
Свойства oradecisiontreenode	**Значения**	**Описание свойства**
use_costs	*флаг*	
impurity_metric	Энтропия Gini	
term_max_depth	*целое*	2–20.*
term_minpct_node	*число*	0,0–10,0.*
term_minpct_split	*число*	0,0–20,0.*
term_minrec_node	*целое*	Значение должно быть больше 0,*
term_minrec_split	*целое*	Значение должно быть больше 0,*
display_rule_ids	*флаг*	*

* Свойства, которые игнорируются, если для mode задано значение Простые.

O-кластер Oracle

Следующие свойства доступны для узлов типа oraoclusternode.

Таблица 207. Свойства oraoclusternode

Свойства oraoclusternode	Значения	Описание свойства
max_num_clusters	*целое*	Значение должно быть больше 0.
max_buffer	*целое*	Значение должно быть больше 0,*
sensitivity	*число*	0,0–1,0.*

* Свойства, которые игнорируются, если для mode задано значение Простые.

K-средние Oracle

Следующие свойства доступны для узлов типа orakmeansnode.

Таблица 208. Свойства orakmeansnode

Свойства orakmeansnode	Значения	Описание свойства
num_clusters	*целое*	Значение должно быть больше 0.
normalization_method	zscore minmax none	
distance_function	Евклидова Косинус	
iterations	*целое*	0–20.*
conv_tolerance	*число*	0,0–0,5.*
split_criterion	Дисперсия Размер	По умолчанию используется Variance (дисперсия).*
num_bins	*целое*	Значение должно быть больше 0,*
block_growth	*целое*	1–5.*
min_pct_attr_support	*число*	0,0–1,0.*

* Свойства, которые игнорируются, если для mode задано значение Простые.

NMF Oracle

Следующие свойства доступны для узлов типа oranmfnode.

Таблица 209. Свойства oranmfnode		
Свойства oranmfnode	**Значения**	**Описание свойства**
normalization_method	minmax none	
use_num_features	*флаг*	*
num_features	*целое*	0–1. Значение по умолчанию оценено алгоритмом по данным.*
random_seed	*число*	*
num_iterations	*целое*	0–500.*
conv_tolerance	*число*	0,0–0,5.*
display_all_features	*флаг*	*

* Свойства, которые игнорируются, если для mode задано значение Простые.

Априорный анализ Oracle

Следующие свойства доступны для узлов типа oraapriorinode.

Таблица 210. Свойства oraapriorinode		
Свойства oraapriorinode	**Значения**	**Описание свойства**
content_field	*поле*	
id_field	*поле*	
max_rule_length	*целое*	2–20.
min_confidence	*число*	0,0–1,0.
min_support	*число*	0,0–1,0.
use_transactional_data	*флаг*	

Минимальная длина описания (Minimum Description Length, MDL) Oracle

Не существует специфических свойств, определенных для узлов типа oramdlnode. Смотрите описание общих свойств Oracle в начале этого раздела.

Важность атрибутов Oracle (AI)

Следующие свойства доступны для узлов типа oraainode.

Таблица 211. Свойства oraainode		
Свойства oraainode	**Значения**	**Описание свойства**
custom_fields	*флаг*	Если значение флага true, это позволяет вам задать поле назначения, входные и другие поля для текущего узла. При значении false используются текущие параметры с вышележащего узла Тип.

Таблица 211. Свойства oraainode (продолжение)		
Свойства oraainode	**Значения**	**Описание свойства**
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли важные поля.
important_label	*string*	Задает метку для ранжирования "важно".
select_marginal	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли пограничные поля.
marginal_label	*string*	Задает метку для ранжирования "пограничное".
important_above	*число*	0,0–1,0.
select_unimportant	*флаг*	Когда для selection_mode задано значение ImportanceLevel, задает, выбрать ли не важные поля.
unimportant_label	*string*	Задает метку для ранжирования "не важно".
unimportant_below	*число*	0,0–1,0.
importance_value	*число*	Когда для selection_mode задано значение ImportanceValue, задает значение отсечения для использования. Принимает значения от 0 до 100.
top_n	*число*	Когда для selection_mode задано значение TopN, задает значение отсечения для использования. Принимает значения от 0 до 1000.

Свойства слепков моделей Oracle

Следующие свойства предназначены для слепков моделей, созданных с использованием моделей Oracle.

Наивный критерий Байеса Oracle

Не существует специфических свойств, определенных для узлов типа applyoranbnode.

Адаптивный критерий Байеса Oracle

Не существует специфических свойств, определенных для узлов типа applyoraabnnode.

Механизмы векторов поддержки Oracle

Не существует специфических свойств, определенных для узлов типа applyorasvmnode.

Дерево решений Oracle

Следующие свойства доступны для узлов типа applyoradecisiontreenode.

Таблица 212. Свойства applyoradecisiontreenode		
Свойства applyoradecisiontreenode	**Значения**	**Описание свойства**
use_costs	*флаг*	
display_rule_ids	*флаг*	

O-кластер Oracle

Не существует специфических свойств, определенных для узлов типа applyoraoclusternode.

K-средние Oracle

Не существует специфических свойств, определенных для узлов типа applyorakmeansnode.

NMF Oracle

Следующие конкретные свойства доступны для узлов типа applyoranmfnode:

Таблица 213. Свойства applyoranmfnode		
Свойства applyoranmfnode	**Значения**	**Описание свойства**
display_all_features	*флаг*	

Априорный анализ Oracle

Этот слепок модели нельзя применять в сценарии.

MDL Oracle

Этот слепок модели нельзя применять в сценарии.

Свойства узлов для моделирования IBM Netezza Analytics

Свойства узлов моделирования Netezza

Следующие свойства общие для узлов моделирования баз данных IBM Netezza.

Таблица 214. Свойства Common Netezza node		
Общие свойства узлов Netezza	**Значения**	**Описание свойства**
custom_fields	*флаг*	Если значение флага true, это позволяет вам задать поле назначения, входные и другие поля для текущего узла. При значении false используются текущие параметры с вышележащего узла Тип.
inputs	*[поле1 ... полеN]*	Входные (или предикторные) поля, используемые в модели.
target	*поле*	Поле назначения (количественное или категориальные).

Таблица 214. Свойства Common Netezza node (продолжение)		
Общие свойства узлов Netezza	**Значения**	**Описание свойства**
record_id	*поле*	Поле для использования в качестве уникального идентификатора записей.
use_upstream_connection	*флаг*	При значении true (по умолчанию) подробности соединения задаются на узле более высокого уровня. Не используется, если задано move_data_to_connection.
move_data_connection	*флаг*	При значении true перемещает данные в базу данных, заданную connection. Не используется, если задано use_upstream_connection.
connection	*структурированный*	Строка соединения для базы данных Netezza, где хранится модель. Структурированное свойство в следующей форме: ['odbc' '<dsn>' '<имя_пользователя>' '<пароль>' '<имя_каталога>' '<атрибуты_соединения>' [true|false]] где: <dsn> - это имя источника данных <имя_пользователя> и <пароль> - это имя пользователя и пароль для базы данных <имя_каталога> - это имя каталога <атрибуты_соединения> - это атрибуты соединения true | false обозначает, нужен ли пароль.
table_name	*string*	Имя таблицы базы данных, в которой будет храниться модель.
use_model_name	*флаг*	При значении true использует имя, заданное опцией имя_модели как имя модели, в противном случае имя модели создается системой.
model_name	*string*	Пользовательское имя для новой модели.
include_input_fields	*флаг*	При значении true передает все входные поля ниже уровнем, в противном случае передает только ID_записи и поля, сгенерированные моделью.

Дерево решений Netezza

Следующие свойства доступны для узлов типа netezzadectreenode.

Таблица 215. Свойства netezzadectreenode

Свойства netezzadectreenode	**Значения**	**Описание свойства**
impurity_measure	Энтропия Gini	Мера неоднородности используется для оценки лучшего положения для разветвления дерева.
max_tree_depth	*целое*	Максимальное количество уровней, до которых может расти дерево. Значение по умолчанию 62 (максимально возможное).
min_improvement_splits	*number*	Минимальное значение улучшения неоднородности, чтобы произошло разделение. Значение по умолчанию 0,01.
min_instances_split	*целое*	Минимальное количество остающихся неразделенных записей до возможного разделения. Значение по умолчанию 2 (минимально возможное).
weights	*структурированный*	Относительные значения веса для классов. Структурированное свойство в следующей форме: `set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]]` По умолчанию вес 1 задается для всех классов.
pruning_measure	Acc wAcc	Значение по умолчанию Acc (точность). Альтернативное значение wAcc (взвешенная точность) при применении усечения учитывает веса классов.
prune_tree_options	allTrainingData `partitionTrainingData` useOtherTable	По умолчанию для оценки точности модели используется allTrainingData. Используйте partitionTrainingData, чтобы задать для использования процентную долю данных обучения, или useOtherTable, чтобы использовать набор данных обучения из заданной таблицы базы данных.

Таблица 215. Свойства netezzadectreenode (продолжение)		
Свойства netezzadectreenode	**Значения**	**Описание свойства**
perc_training_data	*number*	Если для prune_tree_options задано значение partitionTrainingData, определяет процентную долю данных для использования при обучении.
prune_seed	*целое*	Начальное значение генератора псевдослучайных чисел, используемое для репликации результатов анализа, когда для prune_tree_options задано значение partitionTrainingData; значение по умолчанию равно 1.
pruning_table	*string*	Имя таблицы отдельного набора данных усечения для оценки точности модели.
compute_probabilities	*флаг*	При значении true создает поле доверительного интервала (вероятности), а также поле предсказания.

K-средние Netezza

Следующие свойства доступны для узлов типа netezzakmeansnode.

Таблица 216. Свойства netezzakmeansnode		
Свойства netezzakmeansnode	**Значения**	**Описание свойства**
distance_measure	Евклидова Манхеттен Канберра максимум	Способ, используемый для измерения расстояния между точками данных.
num_clusters	*целое*	Количество кластеров, которые будут созданы; значение по умолчанию 3.
max_iterations	*целое*	Количество итераций алгоритма, после которого останавливается обучение модели; значение по умолчанию 5.

Таблица 216. Свойства netezzakmeansnode (продолжение)		
Свойства netezzakmeansnode	**Значения**	**Описание свойства**
rand_seed	*целое*	Начальное значение генератора псевдослучайных чисел, используемое для репликации результатов анализа; значение по умолчанию 12345.

Байесовская сеть Netezza

Следующие свойства доступны для узлов типа netezzabayesnode.

Таблица 217. Свойства netezzabayesnode		
Свойства netezzabayesnode	**Значения**	**Описание свойства**
base_index	*целое*	Числовой идентификатор, назначенный первому входному полю для внутреннего управления; значение по умолчанию 777.
sample_size	*целое*	Размер выборки, которую требуется сделать при очень большом числе атрибутов; значение по умолчанию 10 000.
display_additional_information	*флаг*	При значении true в диалоговом окне сообщений выводит дополнительную информацию о ходе выполнения.
type_of_prediction	best соседи nn-neighbors	Тип алгоритма предсказания для использования: наилучший (наиболее скоррелированные соседи), соседи (взвешенное предсказание соседей) или nn-соседи (непустые соседние).

Наивный байесовский анализ Netezza

Следующие свойства доступны для узлов типа netezzanaivebayesnode.

Таблица 218. Свойства netezzanaivebayesnode		
Свойства netezzanaivebayesnode	**Значения**	**Описание свойства**
compute_probabilities	*флаг*	При значении true создает поле доверительного интервала (вероятности), а также поле предсказания.
use_m_estimation	*флаг*	При значении true использует способ m-оценки для исключения при оценке нулевых вероятностей.

KNN Netezza

Следующие свойства доступны для узлов типа netezzaknnnode.

Таблица 219. Свойства netezzaknnnode		
Свойства netezzaknnnode	**Значения**	**Описание свойства**
weights	*структурированный*	Структурированное свойство, используемое для назначения весов индивидуальным классам. Пример: `set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]`
distance_measure	Евклидова Манхеттен Канберра Максимум	Способ, используемый для измерения расстояния между точками данных.
num_nearest_neighbors	*целое*	Количество ближайших соседей для конкретного наблюдения; значение по умолчанию 3.
standardize_measurements	*флаг*	При значении true стандартизует измерения для количественных входных полей перед вычислением значений расстояния.
use_coresets	*флаг*	При значении true использует выборку базового набора для ускорения вычислений для больших наборов данных.

Разделительная кластеризация Netezza

Следующие свойства доступны для узлов типа netezzadivclusternode.

Таблица 220. Свойства netezzadivclusternode		
Свойства netezzadivclusternode	**Значения**	**Описание свойства**
distance_measure	Евклидова Манхеттен Канберра Максимум	Способ, используемый для измерения расстояния между точками данных.
max_iterations	*целое*	Максимальное количество итераций алгоритма для выполнения перед остановками обучения модели; значение по умолчанию - 5.
max_tree_depth	*целое*	Максимальное число уровней, на которые можно разделить набор данных; значение по умолчанию 3.

Таблица 220. Свойства netezzadivclusternode (продолжение)		
Свойства netezzadivclusternode	**Значения**	**Описание свойства**
rand_seed	*целое*	Начальное значение генератора псевдослучайных чисел, используемое для репликации результатов анализа; значение для умолчанию 12345.
min_instances_split	*целое*	Минимальное количество записей, которые можно разделить; значение по умолчанию 5.
level	*целое*	Уровень иерархии, до которого будут оцениваться записи; значение по умолчанию равно -1.

PCA Netezza

Следующие свойства доступны для узлов типа netezzapcanode.

Таблица 221. Свойства netezzapcanode		
Свойства netezzapcanode	**Значения**	**Описание свойства**
center_data	*флаг*	При значении true (по умолчанию), эта опция перед анализом выполняет центрирование данных (называемое также "извлечение среднего").
perform_data_scaling	*флаг*	При значении true выполняет масштабирование данных перед анализом. Это делает анализ менее произвольным, когда различные переменные измеряются разными единицами.
force_eigensolve	*флаг*	При значении true использует менее точный, но быстрый способ нахождения главных компонентов.
pc_number	*целое*	Количество главных компонентов, до которого будет сокращен набор данных; значение по умолчанию 1.

Дерево регрессии Netezza

Следующие свойства доступны для узлов типа netezzaregtreenode.

Таблица 222. Свойства netezzaregtreenode		
Свойства netezzaregtreenode	**Значения**	**Описание свойства**
max_tree_depth	*целое*	Максимальное количество уровней, до которых может расти дерево ниже корневого узла; значение по умолчанию 10.

Таблица 222. Свойства netezzaregtreenode (продолжение)

Свойства netezzaregtreenode	**Значения**	**Описание свойства**
split_evaluation_measure	Дисперсия	Показатель неоднородности классов, используется для оценки лучшего положения для разделения дерева; значение по умолчанию (в настоящее время единственное) - Дисперсия.
min_improvement_splits	*number*	Минимальное сокращение неоднородности перед созданием нового разделения в дереве.
min_instances_split	*целое*	Минимальное количество записей, которые можно разделить.
pruning_measure	mse r2 Пирсона Спирмана	Способ, который будет использоваться для усечения.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	По умолчанию для оценки точности модели используется allTrainingData. Используйте partitionTrainingData, чтобы задать для использования процентную долю данных обучения, или useOtherTable, чтобы использовать набор данных обучения из заданной таблицы базы данных.
perc_training_data	*number*	Если для prune_tree_options задано значение PercTrainingData, определяет процентную долю данных для использования при обучении.
prune_seed	*целое*	Начальное значение генератора псевдослучайных чисел, используемое для репликации результатов анализа, когда для prune_tree_options задано значение PercTrainingData; значение по умолчанию равно 1.

Таблица 222. Свойства netezzaregtreenode (продолжение)		
Свойства netezzaregtreenode	**Значения**	**Описание свойства**
pruning_table	*string*	Имя таблицы отдельного набора данных усечения для оценки точности модели.
compute_probabilities	*флаг*	При значении true задает, что дисперсии назначенных классов должны включаться в выходные данные.

Линейная регрессия Netezza

Следующие свойства доступны для узлов типа netezzalineregressionnode.

Таблица 223. Свойства netezzalineregressionnode		
Свойства netezzalineregressionnode	**Значения**	**Описание свойства**
use_svd	*флаг*	При значении true использует матрицу декомпозиции единого значения вместо исходной матрицы для повышения скорости и точности вычислений.
include_intercept	*флаг*	При значении true (по умолчанию) повышает общую точность решения.
calculate_model_diagnostics	*флаг*	При значении true вычисляет диагностику для модели.

Временные ряды Netezza

Следующие свойства доступны для узлов типа netezzatimeseriesnode.

Таблица 224. Свойства netezzatimeseriesnode		
Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
time_points	*поле*	Входное поле, содержащее значения даты или времени для временных рядов.
time_series_ids	*поле*	Входное поле, содержащее значения ID временных рядов; используется, если входные данные содержат больше одного временного ряда.
model_table	*поле*	Имя таблицы базы данных, где будет храниться модель временного ряда Netezza.
description_table	*поле*	Имя входной таблицы, содержащей имена и описания временных рядов.

Таблица 224. Свойства netezzatimeseriesnode (продолжение)

Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
seasonal_adjustment_table	*поле*	Имя выходной таблицы, где будут храниться значения, скорректированные по сезону, вычисленные по алгоритмам экспоненциального сглаживания или декомпозиции сезонных тенденций.
algorithm_name	SpectralAnalysis или спектральный ExponentialSmoothing или esmoothing АРПСС SeasonalTrendDecomposition или std	Алгоритмы, которые будут использованы для моделирования временных рядов.
trend_name	N A DA M DM	Тип тенденции для экспоненциального сглаживания: N - нет A - дополнительный DA - демпфированный аддитивный M - мультипликативный DM - демпфированный мультипликативный
seasonality_type	N A M	Тип сезонности для экспоненциального сглаживания: N - нет A - дополнительный M - мультипликативный

Таблица 224. Свойства netezzatimeseriesnode (продолжение)

Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
interpolation_method	Линейная cubicspline exponentialspline	Способ интерполяции, который будет использован.
timerange_setting	SD SP	Параметр диапазона времени для использования: SD - определяется системой (system-determined), использует полный диапазон данных временного ряда SP - задается пользователем в полях начальное_время и конечное_время

earliest_time
 	
latest_time
	целое
	
	дата
	
	время
	
	timestamp
		Начальное и конечное значения, если для timerange_setting задано значение SP.
		
		Формал должен следовать значению time_points.
		
		Например, если поле time_points содержит дату, формат тоже должен быть форматом даты.
		
		Пример:
		
		`set NZ_DT1.timerange_setting = 'SP'`
		
		`set NZ_DT1.earliest_time = '1921-01-01'`
		
		`set NZ_DT1.latest_time = '2121-01-01'`

Таблица 224. Свойства netezzatimeseriesnode (продолжение)		
Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
arima_setting	SD SP	Параметр для алгоритма ARIMA (используется только в том случае, если для имя_алгоритма задано значение ARIMA): SD - определяется системой SP - задается пользователем Если arima_setting = SP, используйте следующие параметры, чтобы задать сезонные и несезонные значения. Пример (только несезонные значения): 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 - оператор для параметров p, d, q, sp, sd и sq: less - меньше чем eq - равно lesseq - less than or equal to
p	*целое*	ARIMA - несезонные степени автокорреляции.

Таблица 224. Свойства netezzatimeseriesnode (продолжение)

Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
q	*целое*	ARIMA - несезонное значение отклонения.
d	*целое*	ARIMA - несезонное количество порядков скользящего среднего в модели.
sp	*целое*	ARIMA - сезонные степени автокорреляции.
sq	*целое*	ARIMA - сезонное значение отклонения.
sd	*целое*	ARIMA - сезонное количество порядков скользящего среднего в модели.

advanced_setting	SD
	
	SP
		Определяет, как будут обрабатываться расширенные параметры:
		
		SD - определяется системой
		
		SP - задается пользователем через период, единицы_периоды и параметр_прогноза.
		
		Пример:
		
		`set NZ_DT1.advanced_setting = 'SP'`
		
		`set NZ_DT1.period = 5`
		
		`set NZ_DT1.units_period = 'd'`

period	*целое*	Длина сезонного цикла, задается в сочетании с единицами_периода. Не применяется для спектрального анализа.

Таблица 224. Свойства netezzatimeseriesnode (продолжение)

Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
units_period	ms s min h d wk q y	Единицы, в которых выражается период: ms - миллисекунды s - секунды min - минуты h - часы d - дни wk - недели q - кварталы y - годы Например, для понедельных временных рядов используйте 1 для периода и wk для единиц_периода.
forecast_setting	forecasthorizon forecasttimes	Задает, как будет делаться прогноз.
forecast_horizon	*целое* *дата* *время* *отметка времени*	Если forecast_setting = forecasthorizon, задает значение конечной точки прогнозирования. Формал должен следовать значению time_points. Например, если поле time_points содержит дату, формат тоже должен быть форматом даты.

Таблица 224. Свойства netezzatimeseriesnode (продолжение)		
Свойства netezzatimeseriesnode	**Значения**	**Описание свойства**
forecast_times	*целое* *дата* *время* *timestamp*	Если forecast_setting = forecasttimes, задает значения для прогнозирования. Формал должен следовать значению time_points. Например, если поле time_points содержит дату, формат тоже должен быть форматом даты.
include_history	*флаг*	Обозначает, будут ли включены хронологические значения в выходные данные.
include_interpolated_values	*флаг*	Обозначает, будут ли включены интерполированные значения в выходные данные. Не применяется, если для include_history задано значение false.

Обобщенный линейный анализ Netezza

Следующие свойства доступны для узлов типа netezzaglmnode.

Таблица 225. Свойства netezzaglmnode		
Свойства netezzaglmnode	**Значения**	**Описание свойства**
dist_family	bernoulli Нормальное poisson negativebinomial wald gamma	Тип распределения; значение по умолчанию bernoulli.
dist_params	*number*	Значение параметра распределения для использования. Применимо только в том случае, если distribution - это Negativebinomial.

Таблица 225. Свойства netezzaglmnode (продолжение)

Свойства netezzaglmnode	Значения	Описание свойства
trials	*целое*	Применимо только в том случае, если distribution - это Binomial. Если отклик назначения - это количество событий, произошедших в наборе испытаний, поле target содержит количество событий, а поле trials - число испытаний.
model_table	*поле*	Имя таблицы базы данных, где будет храниться обобщенная линейная модель Netezza.
maxit	*целое*	Максимальное количество итераций, которые должны быть выполнены алгоритмом; значение по умолчанию 20.
eps	*number*	Максимальное значение ошибки (в экспоненциальном представлении), при котором алгоритм должен остановиться при поиске модели с наилучшей подгонкой. Значение по умолчанию -3, что означает 1E-3, или 0,001.
tol	*number*	Значение (в экспоненциальном представлении), ниже которого ошибки рассматриваются как имеющие нулевое значение. Значение по умолчанию -7, то есть значения ошибки меньше 1E-7 (или 0,0000001) рассматриваются как несущественные.

Таблица 225. Свойства netezzaglmnode (продолжение)

Свойства netezzaglmnode	**Значения**	**Описание свойства**
link_func	Тождество обратное invnegative invsquare sqrt Степень oddspower log clog loglog cloglog Логит Пробит gaussit cauchit canbinom cangeom cannegbinom	Функция связи, которая будет использоваться; значение по умолчанию logit.
link_params	*number*	Значение параметра функции связи для использования. Применимо только в том случае, если link_function - это power или oddspower.

Таблица 225. Свойства netezzaglmnode (продолжение)		
Свойства netezzaglmnode	**Значения**	**Описание свойства**
interaction	[[[*имена_столбцов1*],[*уровни1*]], [[*имена_столбцов2*],[*уровни2*]], ...,[[*имена_столбцовN*], [*уровниN*]],]	Задает взаимодействия между полями. *имена_столбцов* - это список входных полей, а *уровень* - это всегда 0 для каждого поля. Пример: `[[["K","BP","Sex","K"], [0,0,0,0]], [["Age","Na"],[0,0]]]`
intercept	*флаг*	При значении true включает в модель свободный член.

Свойства слепков моделей Netezza

Следующие свойства общие для слепков моделей баз данных Netezza.

Таблица 226. Общие свойства слепков моделей Netezza		
Общие свойства слепков моделей Netezza	**Значения**	**Описание свойства**
connection	*string*	Строка соединения для базы данных Netezza, где хранится модель.
table_name	*string*	Имя таблицы базы данных, в которой хранится модель.

Другие свойства слепков моделей - те же, что для соответствующего узла моделирования.

Имена сценариев слепков моделей следующие.

Таблица 227. Имена сценариев слепков моделей Netezza	
Образец модели	**Имя сценария**
дерево решений	applynetezzadectreenode
K-средних	applynetezzakmeansnode
Байесовская сеть	applynetezzabayesnode
наивная модель Байеса	applynetezzanaivebayesnode
KNN	applynetezzaknnnode
Разделительная кластеризация	applynetezzadivclusternode
PCA	applynetezzapcanode
дерево регрессии	applynetezzaregtreenode
Линейная регрессия	applynetezzalineregressionnode
Временные ряды	applynetezzatimeseriesnode

Таблица 227. Имена сценариев слепков моделей Netezza (продолжение)

Образец модели	**Имя сценария**
Обобщенная линейная	`applynetezzaglmnode`

Глава 16. Свойства узлов вывода

Свойства узлов вывода немного отличаются от свойств других типов узлов. Вместо того, чтобы относиться к определенной опции узла, свойства узлов вывода хранят ссылку на выходной объект. Это полезно, когда нужно взять значение из таблицы, а затем задать его как параметр потока.

В этом разделе описываются свойства сценариев, доступные для полей вывода.

Свойства analysisnode

Узел Анализ оценивает способность прогнозирующих моделей генерировать точные предсказания. Узлы анализа выполняют различные сравнения между предсказанными и фактическими значениями для одного или нескольких слепков моделей. Они могут сравнивать также прогнозирующие модели друг с другом.

Пример

```
node = stream.create("analysis", "My node")
# Вкладка "Анализ"
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)
# "Определить пользовательский показатель..."
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"])
# Вкладка "Выход"
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Таблица 228. Свойства analysisnode

Свойства analysisnode	**Тип переменной**	**Описание свойства**
output_mode	Screen File	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*строка*	Если для use_output_name задано true, определяет имя для использования.

Таблица 228. Свойства analysisnode (продолжение)		
Свойства analysisnode	**Тип переменной**	**Описание свойства**
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Используется для указания типа выходных данных.
by_fields	*список*	
full_filename	*строка*	Имя выходного файла для диска, данных или HTML.
coincidence	*флаг*	
performance	*флаг*	
evaluation_binary	*флаг*	
confidence	*флаг*	
threshold	*number*	
improve_accuracy	*number*	
field_detection_method	Метаданные Имя	Определяет способ сопоставления предсказанных полей с исходным полем назначения. Задайте Метаданные или Имя.
inc_user_measure	*флаг*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	
user_compute	[Среднее Сумма Мин Макс Среднеквадр_отклон]	

Свойства dataauditnode

Узел Аудит данных предоставляет всесторонний первый взгляд на данные, в том числе сводную статистику, гистограммы и распределение для каждого поля, а также информацию о выбросах, значениях отсутствия и экстремумах. Результаты выводятся в виде простой для чтения матрицы, которую можно отсортировать и использовать для генерирования узлов полноразмерных графиков и подготовки данных.

Пример

```
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")
```

Таблица 229. Свойства dataauditnode

Свойства dataauditnode	**Тип переменной**	**Описание свойства**
custom_fields	*флаг*	
fields	*[поле1 ... полеN]*	
overlay	*поле*	
display_graphs	*флаг*	Используется для включения или выключения вывода графиков в выходной матрице.
basic_stats	*флаг*	
advanced_stats	*флаг*	
median_stats	*флаг*	
calculate	Количество Breakdown	Используется для вычисления пропущенных значений. Выберите один из способов вычисления, оба или ни одного.
outlier_detection_method	std iqr	Используется для задания способа детектирования выбросов и экстремальных значений.
outlier_detection_std_outlier	*number*	Если способ_обнаружения_выбросов - это std, задает число, используемое для определения выбросов.
outlier_detection_std_extreme	*число*	Если способ_обнаружения_выбросов - это std, задает число, используемое для определения экстремальных значений.
outlier_detection_iqr_outlier	*number*	Если способ_обнаружения_выбросов - это iqr, задает число, используемое для определения выбросов.

Таблица 229. Свойства dataauditnode (продолжение)		
Свойства dataauditnode	**Тип переменной**	**Описание свойства**
outlier_detection_iqr_extreme	*number*	Если способ_обнаружения_выбросов - это iqr, задает число, используемое для определения экстремальных значений.
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Если для use_output_name задано true, определяет имя для использования.
output_mode	Screen File	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Используется для указания типа выхода.
paginate_output	*флаг*	Когда output_format - это HTML, выходные данные будут разделяться на страницы.
lines_per_page	*number*	При использовании с paginate_output задает количество строк на страницу в выходных данных.
full_filename	*string*	

Свойства extensionoutputnode

Узел вывода Extension дает возможность проанализировать данные и результаты скоринга модели, используя ваш собственный пользовательский сценарий R или Python for Spark. Вывод анализа может быть текстовым или графическим. Кроме того, вывод добавляется на вкладку **Вывод** панели менеджеров; другой вариант - направить вывод в файл.

Пример Python for Spark

```
#### пример сценария для 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)
```

Пример R

```
#### Пример сценария для R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

Таблица 230. свойства extensionoutputnode

Свойства extensionoutputnode	**Тип переменной**	**Описание свойства**
syntax_type	*R* *Python*	Указывает, какой сценарий работает – R или Python (R - значение по умолчанию).
r_syntax	*строка*	Синтаксис сценария R для скоринга моделей.
python_syntax	*строка*	Синтаксис сценария Python для скоринга моделей.
convert_flags	`StringsAndDoubles` `LogicalValues`	Опция для преобразования флаговых полей.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	`POSIXct` `POSIXlt`	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.
output_to	`Screen` `Файл`	Задайте тип вывода (`Screen` - экран - или `File` - файл).

Таблица 230. свойства extensionoutputnode (продолжение)		
Свойства extensionoutputnode	**Тип переменной**	**Описание свойства**
output_type	Диаграмма Текст	Укажите, каким должен быть вывод, графическим или текстовым.
full_filename	*строка*	Имя файла для использования в сгенерированном выводе.
graph_file_type	HTML COU	Тип выходного файла (.html или .cou).
text_file_type	HTML TEXT COU	Задайте тип файла для текстового вывода (.html, .txt или .cou).

свойства kdeexport

Ядерная оценка плотности (KDE)© использует алгоритмы дерева шаров или дерева KDE для эффективных запросов и сочетает идеи неконтролируемого обучения, инженерного проектирования и моделирования данных. Подходы на основе соседей, такие как KDE - это некоторые наиболее популярные и полезные методы оценки плотности. Узел Моделирование KDE и Имитация KDE в SPSS Modeler проявляют базовые функции и обычно используемые параметры библиотеки KDE. Узлы реализованы на языке Python.

Таблица 231. свойства kdeexport		
Свойства kdeexport	**Тип переменной**	**Описание свойства**
bandwidth	*double*	Значение по умолчанию - 1.
kernel	*строка*	Используемое ядро: гауссово или прямоугольное. Значение по умолчанию - Гауссово.
algorithm	*строка*	Алгоритм дерева, который нужно использовать: kd_tree, ball_tree или auto. Значение по умолчанию - auto.
metric	*строка*	Метрика для использования при вычислении расстояния. Для алгоритма kd_tree выбирайте из: Евклидова, Чебышева, Городских кварталов, Минковского, Манхеттен, Бесконечности, P, L2 или L1. Для алгоритма ball_tree выбирайте из: Евклидова, Брея-Кёртиса, Чебышева, Канберра, Городских кварталов, Дайса, Хэмминга, Бесконечности, Жаккарда, L1, L2, Минковского, Совпадения, Манхеттен, P, Роджерса-Танимото, Расселла-Рао, Сокала-Миченера, Сокала-Сниата или Кульчинского. Значение по умолчанию - Евклидова.

Таблица 231. свойства kdeexport (продолжение)		
Свойства kdeexport	**Тип переменной**	**Описание свойства**
atol	*плавающее*	Требуемый абсолютный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 0.0.
rtol	*плавающее*	Требуемый относительный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 1E-8.
breadthFirst	*логический*	Задайте значение True, чтобы использовать подход в ширину. Задайте значение False, чтобы использовать подход в глубину. Значение по умолчанию - True.
LeafSize	*целое*	Размер листа в базовом дереве. Значение по умолчанию - 40. Изменение этого значения может существенно повлиять на производительность.
pValue	*double*	Задайте P-значение, используемое для метрики Minkowski. Значение по умолчанию - 1,5.

Свойства matrixnode

Узел Матрица создает таблицу, показывающую взаимосвязи между полями. Чаще всего он используется для показа взаимосвязи между двумя символическими полями, но он же может показывать взаимосвязи между флаговыми или числовыми полями.

Пример

```
node = stream.create("matrix", "My node")
# Вкладка "Параметры"
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")
# Вкладка "Вид"
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)
# Вкладка "Выходные данные"
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)
```

Таблица 232. Свойства matrixnode		
Свойства matrixnode	**Тип переменной**	**Описание свойства**
fields	Выбранные Флаги Numerics	
строка	*поле*	
столбец	*поле*	
include_missing_values	*флаг*	Задает, включается ли в вывод строк и столбцов пользовательские значения отсутствия (пустые) и системные значения отсутствия (null).
cell_contents	CrossTabs Function	
function_field	*string*	
function	Sum Mean Min Max SDev	
sort_mode	Без сортировки По возрастанию По убыванию	
highlight_top	*number*	Если отлично от нуля, то true.
highlight_bottom	*number*	Если отлично от нуля, то true.

Таблица 232. Свойства matrixnode (продолжение)

Свойства matrixnode	**Тип переменной**	**Описание свойства**
display	[Количества Ожидаемые Остатки RowPct ColumnPct TotalPct]	
include_totals	*флаг*	
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Если для use_output_name задано true, определяет имя для использования.
output_mode	Screen File	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Используется для указания типа выхода. Оба формата, Formatted и Delimited, могут использовать модификатор transposed, который транспонирует строки и столбцы в таблице.
paginate_output	*флаг*	Когда output_format - это HTML, выходные данные будут разделяться на страницы.
lines_per_page	*number*	При использовании с paginate_output задает количество строк на страницу в выходных данных.
full_filename	*string*	

Свойства meansnode

Узел средних значений сравнивает независимые группы или пары связанных полей для проверки, существует ли между ними существенное различие. Например, можно сравнить средние прибыли до и после рекламной кампании, или сравнить прибыли от клиентов, не получавших рекламы, и клиентов, участвовавших в программе продвижения товара.

Пример

```
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")
```

Таблица 233. Свойства meansnode

Свойства meansnode	**Тип переменной**	**Описание свойства**
means_mode	BetweenGroups BetweenFields	Задает тип статистики средних для выполнения для данных.
test_fields	[поле1 ... полен]	Задает проверяемое поле, когда для means_mode задано BetweenGroups.
grouping_field	*поле*	Задает поле группировки.
paired_fields	[[поле1 поле2] [поле3 поле4] ...]	Задает пары полей, когда для means_mode задано BetweenFields.
label_correlations	*флаг*	Задает, показываются ли при выводе метки корреляции. Этот параметр применим, когда для means_mode задано BetweenFields.
correlation_mode	Вероятность Абсолютная	Задает, как метить корреляции, по вероятности или по абсолютному значению.
weak_label	*string*	
medium_label	*string*	
strong_label	*string*	

Таблица 233. Свойства meansnode (продолжение)

Свойства meansnode	Тип переменной	Описание свойства
weak_below_probability	*число*	Когда для correlation_mode задано значение Probability, определяет значение отсечения для слабых корреляций. Это должно быть значение от 0 до 1, например, 0,90.
strong_above_probability	*число*	Значение отсечения для сильных корреляций.
weak_below_absolute	*число*	Когда для correlation_mode задано значение Absolute, определяет значение отсечения для слабых корреляций. Это должно быть значение от 0 до 1, например, 0,90.
strong_above_absolute	*number*	Значение отсечения для сильных корреляций.
unimportant_label	*string*	
marginal_label	*строка*	
important_label	*string*	
unimportant_below	*number*	Значение отсечения для низкой важности полей. Это должно быть значение от 0 до 1, например, 0,90.
important_above	*number*	
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Имя для использования.
output_mode	Screen File	Задает положение назначения для выходных данных, сгенерированных на узле выходных данных.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Задает тип выходных данных.
full_filename	*string*	

Таблица 233. Свойства meansnode (продолжение)		
Свойства meansnode	**Тип переменной**	**Описание свойства**
output_view	Простые Дополнительные параметры	Задает, какое представление используется для вывода - простое или расширенное.

Свойства reportnode

Узел отчетов создает форматированные отчеты, содержащие фиксированный текст, а также данные и другие выражения, полученные из данных. Вы задаете формат отчета, используя текстовые шаблоны, чтобы определить конструкции фиксированного текста и вывода данных. Вы можете предоставить пользовательское форматирование текста с помощью тегов HTML в шаблоне и задав опции на вкладке Вывод. Значения данных и другой условный вывод можно включить в отчет с использованием выражений CLEM в шаблоне.

Пример

```
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", "Узел отчета, созданный сценарием")
node.setPropertyValue("highlights", False)
```

Таблица 234. Свойства reportnode		
Свойства reportnode	**Тип переменной**	**Описание свойства**
output_mode	Screen File	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Используется для указания типа выходного файла.
формат	Авто Пользовательское :	Служит для выбора, форматировать ли вывод автоматически или при помощи HTML, включенного в шаблон. Для использования в шаблоне форматирования HTML задайте Custom.
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Если для use_output_name задано true, определяет имя для использования.

Таблица 234. Свойства reportnode (продолжение)		
Свойства reportnode	**Тип переменной**	**Описание свойства**
text	*string*	
full_filename	*string*	
highlights	*флаг*	
title	*string*	
lines_per_page	*number*	

Свойства Routputnode

Узел вывода R дает возможность проанализировать данные и результаты оценки модели, используя пользовательский сценарий R. Вывод анализа может быть текстовым или графическим. Кроме того, вывод добавляется на вкладку **Вывод** панели менеджеров; другой вариант - направить вывод в файл.

Таблица 235. Свойства Routputnode		
Свойства Routputnode	**Тип переменной**	**Описание свойства**
синтаксис	*строка*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*флаг*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*флаг*	
output_name	Авто Пользовательское:	
custom_name	*строка*	
output_to	Screen Файл	
output_type	Диаграмма Текст	
full_filename	*строка*	
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

Свойства setglobalsnode

Узел задания глобальных значений просматривает данные и вычисляет сводные значения, которые могут использоваться в выражениях CLEM. Например, можно использовать этот узел для вычисления статистических показателей для поля с именем *age*, а затем использовать общее среднее *age* в выражениях CLEM, вставив функцию `@GLOBAL_MEAN(age)`.

Пример

```
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)
```

Таблица 236. Свойства setglobalsnode

Свойства setglobalsnode	**Тип переменной**	**Описание свойства**
globals	[Сумма Среднее Мин Макс Среднеквадр_откл он]	Структурированное свойство, где на поля, которые будут задаваться, нужно ссылаться с использованием следующего синтаксиса: node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	*флаг*	
show_preview	*флаг*	

Свойства simevalnode

Узел оценки имитации оценивает заданное предсказанное поле назначения и представляет информацию о распределении и корреляции этого поля назначения.

Таблица 237. Свойства simevalnode

Свойства simevalnode	**Тип переменной**	**Описание свойства**
назначение	*поле*	
iteration	*поле*	
presorted_by_iteration	*логическое*	
max_iterations	*число*	
tornado_fields	*[поле_1...поле_N]*	
plot_pdf	*логическое*	

Таблица 237. Свойства simevalnode (продолжение)		
Свойства simevalnode	**Тип переменной**	**Описание свойства**
plot_cdf	*логическое*	
show_ref_mean	*логическое*	
show_ref_median	*логическое*	
show_ref_sigma	*логическое*	
num_ref_sigma	*число*	
show_ref_pct	*логическое*	
ref_pct_bottom	*число*	
ref_pct_top	*число*	
show_ref_custom	*логическое*	
ref_custom_values	*[число_1...число_N]*	
category_values	`Категория Вероятности И те, и другие`	
category_groups	`Категории Итерации`	
create_pct_table	*логическое*	
pct_table	`Квартили Интервалы Пользовательское:`	
pct_intervals_num	*число*	
pct_custom_values	*[число_1...число_N]*	

Свойства simfitnode

Узел подгонки имитации исследует статистическое распределение данных в каждом поле и генерирует (или обновляет) узел генерирования имитации, используя для каждого поля оптимально подогнанное распределение. Затем узел генерирования имитации можно использовать для генерирования данных имитации.

Таблица 238. Свойства simfitnode		
Свойства simfitnode	**Тип переменной**	**Описание свойства**
build	`Узел XMLExport И те, и другие`	
use_source_node_name	*логическое*	
source_node_name	*строка*	Пользовательское имя генерируемого или изменяемого узла источника.

Таблица 238. Свойства simfitnode (продолжение)		
Свойства simfitnode	**Тип переменной**	**Описание свойства**
use_cases	`Все` `LimitFirstN`	
use_case_limit	*целое*	
fit_criterion	`AndersonDarling` `KolmogorovSmirnov`	
num_bins	*целое*	
parameter_xml_filename	*строка*	
generate_parameter_import	*логическое*	

Свойства statisticsnode

Узел Статистика предоставляет базовую сводную информацию о числовых полях. Здесь вычисляется сводная статистика для индивидуальных полей и корреляции между полями.

Пример

```
node = stream.create("statistics", "My node")
# Вкладка "Параметры"
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# Раздел "Метки корреляции..."
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")
# Вкладка "Выходные данные"
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Таблица 239. Свойства statisticsnode		
Свойства statisticsnode	**Тип переменной**	**Описание свойства**
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Если для use_output_name задано true, определяет имя для использования.
output_mode	`Screen` `File`	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.

Таблица 239. Свойства statisticsnode (продолжение)

Свойства statisticsnode	**Тип переменной**	**Описание свойства**
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Используется для указания типа выходных данных.
full_filename	*string*	
examine	*список*	
correlate	*список*	
statistics	[count mean sum min max range variance sdev semean median mode]	
correlation_mode	Вероятность Абсолютная	Задает, как метить корреляции, по вероятности или по абсолютному значению.
label_correlations	*флаг*	
weak_label	*строка*	
medium_label	*string*	
strong_label	*string*	
weak_below_probability	*число*	Когда для correlation_mode задано значение Probability, определяет значение отсечения для слабых корреляций. Это должно быть значение от 0 до 1, например, 0,90.
strong_above_probability	*число*	Значение отсечения для сильных корреляций.
weak_below_absolute	*число*	Когда для correlation_mode задано значение Absolute, определяет значение отсечения для слабых корреляций. Это должно быть значение от 0 до 1, например, 0,90.
strong_above_absolute	*число*	Значение отсечения для сильных корреляций.

Свойства узла выходных данных статистики (statisticsoutputnode)

Узел Вывод статистики позволяет вызвать процедуру IBM SPSS Statistics для анализа ваших данных IBM SPSS Modeler. Доступны разнообразные аналитические процедуры IBM SPSS Statistics. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Свойства этого узла описаны в разделе “Свойства узла выходных данных статистики (statisticsoutputnode)” на стр. 455.

Свойства tablenode

Узел Таблица выводит данные в табличном формате, которые можно также записать в файл. Это полезно всякий раз, когда вам нужно проверить значения своих данных или экспортировать их в просто читаемую форму.

Пример

```
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)
```

Таблица 240. Свойства tablenode

Свойства tablenode	**Тип переменной**	**Описание свойства**
full_filename	*string*	Имя выходного файла для диска, данных или HTML.
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*string*	Если для use_output_name задано true, определяет имя для использования.
output_mode	Screen File	Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Используется для указания типа выходных данных.

Таблица 240. Свойства tablenode (продолжение)

Свойства tablenode	**Тип переменной**	**Описание свойства**
transpose_data	*флаг*	Транспонирует данные перед экспортом, чтобы строки представляли поля, а столбцы - записи.
paginate_output	*флаг*	Когда output_format - это HTML, выходные данные будут разделяться на страницы.
lines_per_page	*number*	При использовании с paginate_output задает количество строк на страницу в выходных данных.
highlight_expr	*string*	
output	*string*	Свойство только для чтения, содержащее ссылку на последнюю таблицу, созданную узлом.
value_labels	*[[Value LabelString]* *[Value LabelString] ...]*	Используется для задания меток парам значений.
display_places	*целое*	Задает количество десятичных разрядов при выводе поля (применимо только к полям с системой хранения REAL). При значении –1 будут использоваться значения потока по умолчанию.
export_places	*целое*	Задает количество десятичных разрядов при экспорте (применимо только к полям с системой хранения REAL). При значении –1 будут использоваться значения потока по умолчанию.
decimal_separator	DEFAULT PERIOD COMMA	Задает десятичный разделитель для поля (применимо только к полям с системой хранения REAL).

Таблица 240. Свойства tablenode (продолжение)

Свойства tablenode	Тип переменной	Описание свойства
date_format	"ДДММГГ" "ММДДГГ" "ГГММДД" "ГГГГММДД" "ГГГГДД" DAY MONTH "ДД-ММ-ГГ" "ДД-ММ-ГГГГ" "ММ-ДД-ГГ" "ММ-ДД-ГГГГ" "ДД-МЕС-ГГ" "ДД-МЕС-ГГГГ" "ГГГГ-ММ-ДД" "ДД.ММ.ГГ" "ДД.ММ.ГГГГ" "ММ.ДД.ГГГГ" "ДД.МЕС.ГГ" "ДД.МЕС.ГГГГ" "ДД/ММ/ГГ" "ДД/ММ/ГГГГ" "ММ/ДД/ГГ" "ММ/ДД/ГГГГ" "ДД/МЕС/ГГ" "ДД/МЕС/ГГГГ" МЕС ГГГГ к К ГГГГ нн НД ГГГГ	Задает формат даты для поля (применимо только к полям с системой хранения DATE или TIMESTAMP).

Таблица 240. Свойства tablenode (продолжение)		
Свойства tablenode	**Тип переменной**	**Описание свойства**
time_format	"ЧЧMMCC" "ЧЧMM" "MMCC" "ЧЧ:MM:CC" "ЧЧ:MM" "MM:CC" "(Ч)Ч:(M)M:(C)C" "(Ч)Ч:(M)M" "(M)M:(C)C" "ЧЧ.MM.CC" "ЧЧ.MM" "MM.CC" "(Ч)Ч.(M)M.(C)C" "(Ч)Ч.(M)M" "(M)M.(C)C"	Задает формат времени для поля (применимо только к полям с системой хранения TIME или TIMESTAMP).
column_width	*целое*	Задает ширину столбца для поля. При значении –1 для ширины столбца будет задано Auto.
justify	AUTO CENTER LEFT RIGHT	Задает выравнивание столбцов для поля.

Свойства transformnode

Узел Преобразование позволяет выбрать и предварительно просмотреть результаты преобразований, прежде чем применить их к выбранным полям.

Пример

```
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)
```

Таблица 241. Свойства transformnode

Свойства transformnode	**Тип переменной**	**Описание свойства**
fields	[*поле1... полеп*]	Поля для использования в преобразовании.
formula	All Выбрать	Обозначает, какие преобразования будут вычисляться, все или выбранные.
formula_inverse	*флаг*	Обозначает, должно ли использоваться обратное преобразование.
formula_inverse_offset	*number*	Обозначает смещение данных для использования в формуле. По умолчанию задано 0, в противном случае указывается пользователем.
formula_log_n	*флаг*	Обозначает, должно ли использоваться преобразование $\log_n$.
formula_log_n_offset	*number*	
formula_log_10	*флаг*	Обозначает, должно ли использоваться преобразование $\log_{10}$.
formula_log_10_offset	*number*	
formula_exponential	*флаг*	Обозначает, должно ли использоваться экспоненциальное преобразование (e^x).
formula_square_root	*флаг*	Обозначает, должно ли использоваться преобразование квадратного корня.
use_output_name	*флаг*	Задает, используется ли пользовательское имя вывода.
output_name	*строка*	Если значение поля use_output_name - это true, задает имя для использования.

Таблица 241. Свойства transformnode (продолжение)		
Свойства `transformnode`	**Тип переменной**	**Описание свойства**

`output_mode`	`Screen`
	
	`File`
		Используется для задания положения назначения для выходных данных, сгенерированных на узле выходных данных.

`output_format`	`HTML` (*.html*) `Output` (*.cou*)	Используется для указания типа выходных данных.
`paginate_output`	*флаг*	Когда `output_format` - это `HTML`, выходные данные будут разделяться на страницы.
`lines_per_page`	*number*	При использовании с `paginate_output` задает количество строк на страницу в выходных данных.
`full_filename`	*строка*	Обозначает имя файла для использования при файловых выходных данных.

Глава 17. Свойства узла экспорта

Общие свойства узлов экспорта

Следующие свойства общие для всех узлов экспорта.

Таблица 242. Общие свойства узлов экспорта

Свойство	Значения	Описание свойства
publish_path	*string*	Введите имя rootname, которое будет использоваться для опубликованного образа и файлов параметров.
publish_metadata	*флаг*	Задает, будет ли создаваться файл метаданных, описывающий входные и выходные данные образа и их модели.
publish_use_parameters	*флаг*	Задает, включаются ли параметры потока в файл *.par.
publish_parameters	*список строк*	Задайте параметры, которые будут включены.
execute_mode	export_data publish	Задает, будет ли выполняться узел без публикации потока, или при выполнении узла поток публикуется автоматически.

Свойства asexport

При помощи экспорта Analytic Server поток можно выполнить в файловой системе HDFS (Hadoop Distributed File System).

Пример

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

Таблица 243. Свойства asexport

свойства asexport	Тип переменной	Описание свойства
data_source	*string*	Имя источника данных.
export_mode	*строка*	Задает, что нужно присоединить (**append**) экспортированные данные к существующему источнику данных или перезаписать (**overwrite**) существующий источник данных.

Таблица 243. Свойства asexport (продолжение)		
свойства asexport	**Тип переменной**	**Описание свойства**
use_default_as	*логический*	Если задано True, используется соединение Analytic Server по умолчанию, сконфигурированное в файле options.cfg сервера. Если задано False, используется соединение данного узла.
connection	["string","string","string", "string","string","string","string","string", "string" ,"string","string", "string" ,"string"]	Свойство списка, содержащее подробности соединения Analytic Server. Формат: ["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"] где: is_secure_connect: указывает, используется ли защищенное соединение; возможные значения - true или false. use-kerberos-auth: указывает, используется ли аутентификация kerberos; возможные значения - true или false. enable-kerberos-debug: указывает, используется ли отладочный режим аутентификации kerberos; возможные значения - true или false.

Свойства узла экспорта Cognos (cognosexportnode)

Узел экспорта IBM Cognos экспортирует данные в формате, который могут прочесть базы данных Cognos.

Для этого узла необходимо определить подключение Cognos и подключение ODBC.

Подключение Cognos

Свойства соединения Cognos следующие.

Таблица 244. Свойства cognosexportnode

Свойства cognosexportnode	**Тип переменной**	**Описание свойства**
cognos_connection	*["строка","флаг","строка","строка","строка"]*	Свойство списка, содержащего подробности соединения для сервера Cognos. Используется следующий формат: ["URL_сервера_Cognos", режим_регистрации, "пространство имен", "имя пользователя", "пароль"] где: URL_сервера_Cognos - это URL сервера Cognos, содержащего источник данных. режим_регистрации обозначает, используется ли анонимная регистрация, и может принимать значение true или false; если задано true, следующие поля должны быть заданы как "". пространство имен задает провайдера аутентификации защиты, используемого для регистрации на сервере. имя_пользователя и пароль - это значения для регистрации на сервере Cognos. Вместо режима режим_регистрации доступны также следующие режимы: • anonymousMode. Например: ['URL_сервера_Cognos', 'anonymousMode', "пространство имен", "имя пользователя", "пароль"] • credentialMode. Например: ['URL_сервера_Cognos', 'credentialMode', "пространство имен", "имя пользователя", "пароль"]

Таблица 244. Свойства cognosexportnode (продолжение)		
Свойства cognosexportnode	**Тип переменной**	**Описание свойства**
		• `storedCredentialMode`. Например: `['URL_сервера_Cognos', 'storedCredentialMode', " имя_хранимых_идентификац_данных"]` Где `имя_хранимых_идентификац_данных` - это имя идентификационных данных Cognos в репозитории.
`cognos_package_name`	*string*	Путь и имя пакета Cognos, в который экспортируются данные, например: `/Public Folders/MyPackage`
`cognos_datasource`	*string*	

`cognos_export_mode`	`Опубликовать`
	
	`ExportFile`	

`cognos_filename`	*string*	

соединение ODBC

Свойства соединения ODBC идентичны перечисленным для `databaseexportnode` в следующем разделе, за тем исключением, что свойство `datasource` не используется.

Свойства databaseexportnode

Узел экспорта баз данных записывает данные в совместимый с ODBC источник реляционных данных. Чтобы произвести запись в источник данных ODBC, этот источник данных должен существовать и у вас должны быть разрешения записи для него.

Пример

```
'''
Предполагается, что источник базы данных с именем "MyDatasource" был
сконфигурирован
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Вкладка Экспорт
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)

# Диалоговое окно Схема
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")

# Диалоговое окно Индексы
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP
INDEX <имя-индекса>
 ON <имя-таблицы> <(индекс-столбцы)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id",
"region"]])
```

Таблица 245. Свойства databaseexportnode

Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
datasource	*string*	
username	*string*	
password	*string*	
epassword	*string*	При выполнении этот слот предназначен только для чтения. Чтобы сгенерировать закодированный пароль, используйте Инструмент паролей из меню Инструменты. Дополнительную информацию смотрите в разделе “Генерирование закодированного пароля” на стр. 56.
table_name	*string*	
write_mode	Создать Добавлять в конец Merge	

Таблица 245. Свойства databaseexportnode (продолжение)

Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
map	*string*	Отображает имя поля потока на имя столбца базы данных (допустимо только в том случае, если режим_записи - это Слияние). При слиянии все поля должны быть отображены, чтобы их можно было экспортировать. Имена полей, не существующие в базе данных, добавляются как новые столбцы.
key_fields	*список*	Задает поле потока, используемое для ключа; свойство map показывает, чему оно соответствует в базе данных.
join	База данных Add	
drop_existing_table	*флаг*	
delete_existing_rows	*флаг*	
default_string_size	*целое*	
type		Структурированное свойство, используемое для задания типа схемы.
generate_import	*флаг*	
use_custom_create_table_command	*флаг*	Используйте слот *пользовательское_создание_таблицы* для изменения стандартной команды SQL CREATE TABLE.
custom_create_table_command	*string*	Задает строковую команду для использования вместо стандартной команды SQL CREATE TABLE.
use_batch	*флаг*	Следующие свойства - это расширенные опции для массовой загрузки базы данных. Значение True для опции Использовать_пакет выключает принятия строки за строкой в базу данных.

Таблица 245. Свойства databaseexportnode (продолжение)

Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
batch_size	*число*	Задает число записей, отправляемых в базу данных перед принятием в память.
bulk_loading	Off ODBC Внешнее	Задает тип массовой загрузки. Дополнительные опции для ODBC и Внешнее перечислены ниже.
not_logged	*флаг*	
odbc_binding	Строка Столбец	Задайте построчное или постолбцовое связывание для массовой загрузки через ODBC.
loader_delimit_mode	Табуляция Пробел Другое	Для массовой загрузки через внешнюю программу задайте тип разделителя. Выберите Другое в сочетании со свойством loader_other_delimiter для указания других разделителей, например, запятой (,).
loader_other_delimiter	*строка*	
specify_data_file	*флаг*	Значение флага True активирует свойство файл_данных ниже, где можно задать имя файла и путь для записи в него при массовой загрузке в базу данных.
data_file	*строка*	
specify_loader_program	*флаг*	Значение флага True активирует свойство программа_загрузчика ниже, где можно задать имя файла и положение сценария или программы внешнего загрузчика.
loader_program	*string*	
gen_logfile	*флаг*	Значение флага True активирует свойство имя_файла_журнала ниже, где можно задать имя файла на сервере для генерирования журнала ошибок.

Таблица 245. Свойства databaseexportnode (продолжение)

Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
logfile_name	*string*	
check_table_size	*флаг*	Флаг True позволяет проверить таблицу, чтобы увеличение размера таблицы базы данных соответствовало количеству строк, экспортированных из IBM SPSS Modeler.
loader_options	*string*	Задайте дополнительные аргументы, такие как -comment и -specialdir, в программу загрузчика.
export_db_primarykey	*флаг*	Указывает, представляет ли собой данное поле первичный ключ.
use_custom_create_index_command	*флаг*	Если true, включает пользовательский SQL для всех индексов.
custom_create_index_command	*string*	Задает команду SQL, используемую для создания индексов при включении пользовательского SQL. (Это значение можно переопределить для конкретных индексов, как описано ниже).
indexes.INDEXNAME.fields		При необходимости создает заданный индекс и перечисляет имена полей для включения в этот индекс.
INDEXNAME "use_custom_create_index_command"	*флаг*	Используется для включения или отключения пользовательского SQL для конкретного индекса. Смотрите примеры после следующей таблицы.
INDEXNAME "custom_create_index_command"	*string*	Задает пользовательский SQL, используемый для заданного индекса. Смотрите примеры после следующей таблицы.
indexes.INDEXNAME.remove	*флаг*	Если True, удаляет заданный индекс из набора индексов.
table_space	*строка*	Задает табличное пространство, которое будет создано.

Таблица 245. Свойства databaseexportnode (продолжение)		
Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
use_partition	*флаг*	Задает, что будет использоваться хеш-поле распределения.
partition_field	*string*	Задает содержимое хеш-поля распределения.

Прим.: Для некоторых баз данных можно задать, чтобы таблицы базы данных создавались для экспорта со сжатием (например, эквивалент CREATE TABLE MYTABLE (...) COMPRESS YES; в SQL). Для поддержки этой возможности предоставляются свойства использовать_сжатие и режим_сжатия, как это описано ниже.

Таблица 246. Свойства узла экспорта базы данных (databaseexportnode) с использованием возможностей сжатия		
Свойства databaseexportnode	**Тип переменной**	**Описание свойства**
use_compression	*Логический*	Если задано True, создает таблицы для экспорта со сжатием.
compression_mode	Строка Страница	Задает уровень сжатия для баз данных SQL Server.
	По умолчанию Direct_Load_Operations All_Operations Тип OLTP Query_High Query_Low Archive_High Archive_Low	Задает уровень сжатия для баз данных Oracle. Обратите внимание на то, что для использования значений OLTP, Query_High, Query_Low, Archive_High и Archive_Low требуется как минимум Oracle 11gR2.

Пример, показывающий изменение команды CREATE INDEX для конкретного индекса

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["custom_create_index_command",
 "CREATE BITMAP INDEX <имя-индекса> ON <имя-таблицы> <(индекс-столбцы)>"])
```

Другой вариант - сделать это через хеш-таблицу:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id",
"region"],
 "use_custom_create_index_command":True,
"custom_create_index_command":"CREATE INDEX <имя-индекса> ON
 <имя-таблицы> <(индекс-столбцы)>"])
```

Свойства узла экспорта собрания данных (datacollectionexportnode)

Узел экспорта Data Collection выводит данные в формате, используемом программным обеспечением изучения рынка Data Collection. Для использования этого узла должна быть установлена библиотека данных Data Collection.

Пример

```
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)
```

Таблица 247. Свойства datacollectionexportnode

Свойства datacollectionexportnode	**Тип переменной**	**Описание свойства**
metadata_file	*string*	Имя файла метаданных для экспорта.
merge_metadata	Перезаписать MergeCurrent	
enable_system_variables	*флаг*	Задает, должен ли экспортируемый файл *.mdd* включать в себя системные переменные Data Collection.
casedata_file	*string*	Имя файла *.sav*, в который экспортируются данные наблюдения.
generate_import	*флаг*	

Свойства узла экспорта Excel (excelexportnode)

Узел экспорта Excel выводит данные в формате файлов Microsoft Excel .xlsx. Дополнительно можно выбрать автоматический запуск Excel и открытие экспортированного файла при выполнении узла.

Пример

```
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)
```

Таблица 248. Свойства excelexportnode

Свойства excelexportnode	**Тип переменной**	**Описание свойства**
full_filename	*string*	
excel_file_type	Excel2007	
export_mode	Create Добавлять в конец	
inc_field_names	*флаг*	Укажите, следует ли включить имена полей в первую строку рабочей таблицы.
start_cell	*string*	Задает начальную ячейку для экспорта.
worksheet_name	*string*	Имя рабочего листа для записи.
launch_application	*флаг*	Задает, должен ли Excel вызываться для итогового файла. Обратите внимание на то, что путь для запуска Excel должен быть задан в диалоговом окне Прикладные программы помощника (меню Инструменты, прикладные программы помощника).
generate_import	*флаг*	Задает, должен ли быть сгенерирован узел импорта Excel, который будет читать файл экспортированных данных.

Свойства extensionexportnode

При помощи узла Extension Export можно запускать сценарии R или Python for Spark для экспорта данных.

Пример Python for Spark

```
#### пример сценария для 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)
```

Пример R

```
#### пример сценария для R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

Таблица 249. Свойства extensionexportnode

Свойства extensionexportnode	Тип переменной	Описание свойства
syntax_type	*R* *Python*	Указывает, какой сценарий работает – R или Python (R - значение по умолчанию).
r_syntax	*строка*	Синтаксис R для запускаемого сценария.
python_syntax	*строка*	Синтаксис Python для запускаемого сценария.
convert_flags	`StringsAndDoubles` `LogicalValues`	Опция для преобразования флаговых полей.
convert_missing	*флаг*	Опция для преобразования пропущенных значений в значение R NA.
convert_datetime	*флаг*	Опция для преобразования переменных с форматом данных даты или даты-времени в форматы даты/времени R.
convert_datetime_class	`POSIXct` `POSIXlt`	Опции для указания, в какой формат нужно конвертировать переменные из формата даты или даты-времени.

Свойства jsonexportnode

Узел экспорта JSON выводит данные в формате JSON.

Таблица 250. Свойства jsonexportnode		
Свойства jsonexportnode	**Тип переменной**	**Описание свойства**
full_filename	*строка*	Полное имя файла, включающее путь.
string_format	*records* *значения*	Задайте формат строки JSON. Значение по умолчанию - records.
generate_import	*флаг*	Задает, должен ли быть сгенерирован узел импорта JSON, который будет читать файл экспортированных данных. Значение по умолчанию - False.

Свойства узла выходного файла (outputfilenode)

Узел экспорта плоских файлов выводит данные в текстовом формате с разделителями. Он полезен для экспорта данных, которые может читать другое программное обеспечение анализа или электронных таблиц.

Пример

```
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)
```

Таблица 251. Свойства outputfilenode		
Свойства outputfilenode	**Тип переменной**	**Описание свойства**
full_filename	*string*	Имя выходного файла.
write_mode	Перезаписать Добавлять в конец	
inc_field_names	*флаг*	

Таблица 251. Свойства outputfilenode (продолжение)		
Свойства outputfilenode	**Тип переменной**	**Описание свойства**
use_newline_after_records	*флаг*	
delimit_mode	Запятая Табуляция Пробел Другое	
other_delimiter	*char*	
quote_mode	Нет Одиночный Двойной точности Другое	
other_quote	*флаг*	
generate_import	*флаг*	
encoding	StreamDefault SystemDefault "UTF-8"	

Свойства узла экспорта SAS (sasexportnode)

Узел экспорта SAS выводит данные в формате SAS, которые можно прочесть в программных пакетах SAS или SAS-совместимых. Доступно три формата файлов SAS: SAS для Windows/OS2, SAS для UNIX или SAS Версии 7/8.

Пример

```
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)
```

Таблица 252. Свойства sasexportnode		
Свойства sasexportnode	**Тип переменной**	**Описание свойства**
формат	Windows UNIX SAS7 SAS8	Поля меток различных свойств.
full_filename	*string*	
export_names	NamesAndLabels NamesAsLabels	Используется для отображения имен полей из IBM SPSS Modeler при экспорте на IBM SPSS Statistics или на имена переменных SAS.
generate_import	*флаг*	

Свойства узла экспорта статистики (statisticsexportnode)

Узел Экспорт статистики выводит данные в формате IBM SPSS Statistics *.sav* или *.zsav*. Файлы `.sav` или `.zsav` могут быть прочитаны модулем IBM SPSS Statistics Base и другими продуктами. Это формат, используемый также для файлов кэша в IBM SPSS Modeler.

Свойства этого узла описаны в разделе “Свойства узла экспорта статистики (statisticsexportnode)” на стр. 456.

Свойства узла tm1odataexport

Узел экспорта IBM Cognos TM1 экспортирует данные в формате, который могут прочесть базы данных Cognos TM1.

Таблица 253. Свойства узла tm1odataexport		
Свойства узла tm1odataexport	**Тип переменной**	**Описание свойства**
credential_type	inputCredential или storedCredential	Используется для указания типа идентификационных данных.
input_credential	*список*	Когда credential_type - inputCredential; задайте домен, имя пользователя и пароль.
stored_credential_name	*строка*	Когда credential_type - storedCredential; задайте имя регистрационных данных на сервере C&DS.

Таблица 253. Свойства узла tm1odataexport (продолжение)

Свойства узла tm1odataexport	**Тип переменной**	**Описание свойства**
selected_cube	*поле*	Имя куба, куда вы экспортируете данные. Например: TM1_export.setPropertyValue(" выбранный_куб", "план_BudgetPlan")

Таблица 253. Свойства узла tm1odataexport (продолжение)

Свойства узла tm1odataexport	**Тип переменной**	**Описание свойства**
spss_field_to_tm1_element_mapping	*список*	Элемент tm1, на который выполняется отображение, должен входить в состав измерения столбцов для выбранного представления кубов. Формат: [[[Поле_1, Измерение_1, False], [Элемент_1, Измерение_2, True], ...], [[Поле_2, Существующий_элемент_показаний, False], [Поле_3, Новый_элемент_показаний, True], ...]] Для описания информации отображения есть два списка. Отображение конечного элемента на измерение соответствует примеру 2 ниже: Пример 1: первый список: ([[Поле_1, Измерение_1, False], [Элемент_1, Измерение_2, True], ...]) используется для информации отображения измерения TM1. Каждая тройка значений списка указывает информацию отображения измерения. Третье значение - логическое, оно используется для указания, выбран ли элемент для измерения. Например, "[Поле_1, Измерение_1, False]" означает, что Поле_1отображается на Измерение_1; "[Элемент_1, Измерение_2, True]" означает, что Элемент_1 выбран для Измерения_2. Пример 2: Второй список: ([[Поле_2, ExistMeasureElement, False], [Поле_3, NewMeasureElement, True], ...]) используется для информации об отображении элемента измерения показаний TM1. Каждая тройка значений списка указывает информацию отображения элемента показаний. Третье логическое значение используется для указания на потребность создать новый элемент. "[Поле_2, ExistMeasureElement, False]" означает, что Поле_2 отображено на ExistMeasureElement; "[Поле_3, NewMeasureElement, True]" означает, что NewMeasureElement должен быть измерением показаний, выбранном в выбранное_показание, и Поле_3 отображается на него.

Таблица 253. Свойства узла tm1odataexport (продолжение)		
Свойства узла tm1odataexport	**Тип переменной**	**Описание свойства**
selected_measure	*строка*	Задайте измерение показаний. Пример: setPropertyValue("selected_measure", "Measures")
connection_type	AdminServer TM1Server	Указывает тип соединения. По умолчанию AdminServer.
admin_host	*string*	URL для имени хоста API REST. Требуется, если для connection_type задано AdminServer.
server name	*string*	Имя сервера TM1, выбранного на хост_admin. Требуется, если для connection_type задано AdminServer.
server_url	*string*	URL для API REST сервера TM1. Требуется, если для connection_type задано TM1Server.

Свойства узла tm1export (объявлено устаревшим)

Узел экспорта IBM Cognos TM1 экспортирует данные в формате, который могут прочесть базы данных Cognos TM1.

Прим.: Этот узел устарел в Modeler 18.0. Имя заменяющего сценария узла - *tm1odataexport*.

Таблица 254. Свойства узла tm1export		
Свойства узла tm1export	**Тип переменной**	**Описание свойства**
pm_host	*строка*	**Прим.:** Только для версий 16.0 и 17.0 Имя хоста. Например: TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	*["поле","поле", ... ," поле"]*	**Прим.:** Только для версий 16.0 и 17.0 Свойство списка, содержащего подробности соединения для сервера TM1. Формат: ["имя_сервера_TM1", "имя_пользователя_tm1", "пароль_tm1"] Например: TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])

Таблица 254. Свойства узла tm1export (продолжение)		
Свойства узла tm1export	**Тип переменной**	**Описание свойства**
selected_cube	*поле*	Имя куба, куда вы экспортируете данные. Например: `TM1_export.setPropertyValue(" выбранный_куб", "план_BudgetPlan")`

Таблица 254. Свойства узла tm1export (продолжение)

Свойства узла tm1export	Тип переменной	Описание свойства
spssfield_tm1element_mapping	*список*	Элемент tm1, на который выполняется отображение, должен входить в состав измерения столбцов для выбранного представления кубов. Формат: [[[Поле_1, Измерение_1, False], [Элемент_1, Измерение_2, True], ...], [[Поле_2, Существующий_элемент_показаний, False], [Поле_3, Новый_элемент_показаний, True], ...]] Для описания информации отображения есть два списка. Отображение конечного элемента на измерение соответствует примеру 2 ниже: Пример 1: первый список: ([[Поле_1, Измерение_1, False], [Элемент_1, Измерение_2, True], ...]) используется для информации отображения измерения TM1. Каждая тройка значений списка указывает информацию отображения измерения. Третье значение - логическое, оно используется для указания, выбран ли элемент для измерения. Например, "[Поле_1, Измерение_1, False]" означает, что Поле_1отображается на Измерение_1; "[Элемент_1, Измерение_2, True]" означает, что Элемент_1 выбран для Измерения_2. Пример 2: Второй список: ([[Поле_2, ExistMeasureElement, False], [Поле_3, NewMeasureElement, True], ...]) используется для информации об отображении элемента измерения показаний TM1. Каждая тройка значений списка указывает информацию отображения элемента показаний. Третье логическое значение используется для указания на потребность создать новый элемент. "[Поле_2, ExistMeasureElement, False]" означает, что Поле_2 отображено на ExistMeasureElement; "[Поле_3, NewMeasureElement, True]" означает, что NewMeasureElement должен быть измерением показаний, выбранном в выбранное_показание, и Поле_3 отображается на него.

Таблица 254. Свойства узла tm1export (продолжение)		
Свойства узла tm1export	**Тип переменной**	**Описание свойства**
selected_measure	*строка*	Задайте измерение показаний. Пример: setPropertyValue("selected_measure", "Measures")

Свойства узла экспорта XML (xmlexportnode)

Узел экспорта XML выводит данные в файл в формате XML. Дополнительно вы можете создать узел источника XML, чтобы прочесть экспортированные данные обратно в поток.

Пример

```
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"]])
```

Таблица 255. Свойства xmlexportnode		
Свойства xmlexportnode	**Тип переменной**	**Описание свойства**
full_filename	*string*	(обязательно) Полный путь и имя файла экспорта XML.
use_xml_schema	*флаг*	Задает, использовать ли схему XML (файл XSD или DTD) для управления структурой экспортированных данных.
full_schema_filename	*string*	Полный путь и имя файла XSD или DTD для использования. Обязательно, если для use_xml_schema задано значение true.
generate_import	*флаг*	Генерирует узел источника XML, который будет читать файл экспортированных данных обратно в поток.
records	*string*	Выражение XPath, обозначающее границу записи.
map	*string*	Отображает имя поля на структуру XML.

Глава 18. Свойства узла IBM SPSS Statistics

Свойства узла импорта статистики (statisticsimportnode)

Узел Файл статистики читает данные в формате файлов *.sav*, используемом IBM SPSS Statistics, а также файлы кэша, сохраненные IBM SPSS Modeler, которые также используют этот формат.

Пример

```
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)
```

Таблица 256. Свойства statisticsimportnode

Свойства statisticsimportnode	**Тип переменной**	**Описание свойства**
full_filename	*string*	Полное имя файла, включающее путь.
password	*string*	Пароль. Параметр password должен быть задан перед параметром file_encrypted.
file_encrypted	*флаг*	Защищен ли файл паролем.
import_names	NamesAndLabels LabelsAsNames	Способ для обработки имен и меток переменных.
import_data	DataAndLabels LabelsAsData	Способ для обработки значений и меток.
use_field_format_for_storage	*Логический*	Указывает, использовать ли при импорте информацию о формате полей IBM SPSS Statistics.

Свойства statisticstransformnode

Узел Преобразование статистики запускает разнообразные команды синтаксиса IBM SPSS Statistics для источников данных в IBM SPSS Modeler. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Пример

```
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)
```

Таблица 257. Свойства statisticstransformnode

Свойства statisticstransformnode	Тип переменной	Описание свойства
синтаксис	*string*	
check_before_saving	*флаг*	Проверяет введенный синтаксис перед сохранением записей. Выводит сообщение об ошибке, если синтаксис неправильный.
default_include	*флаг*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.
include	*флаг*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.
new_name	*string*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.

Свойства statisticsmodelnode

Узел Statistics Model позволяет анализировать данные и работать с ними, вызывая процедуру IBM SPSS Statistics, генерирующую PMML. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Пример

```
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")
```

Свойства statisticsmodelnode	Тип переменной	Описание свойства
синтаксис	*string*	
default_include	*флаг*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.
include	*флаг*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.

Свойства statisticsmodelnode	Тип переменной	Описание свойства
new_name	*string*	Дополнительную информацию смотрите в разделе “Свойства filternode” на стр. 174.

Свойства узла выходных данных статистики (statisticsoutputnode)

Узел Вывод статистики позволяет вызвать процедуру IBM SPSS Statistics для анализа ваших данных IBM SPSS Modeler. Доступны разнообразные аналитические процедуры IBM SPSS Statistics. Этому узлу требуется лицензированная копия IBM SPSS Statistics.

Пример

```
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")
```

Таблица 258. Свойства statisticsoutputnode

Свойства statisticsoutputnode	Тип переменной	Описание свойства
mode	Dialog Синтаксис	Выбирает опцию "Диалоговое окно IBM SPSS Statistics" или редактор синтаксиса
синтаксис	*string*	
use_output_name	*флаг*	
output_name	*string*	
output_mode	Screen File	
full_filename	*string*	
file_type	HTML SPV SPW	

Свойства узла экспорта статистики (statisticsexportnode)

Узел Экспорт статистики выводит данные в формате IBM SPSS Statistics *.sav* или *.zsav*. Файлы `.sav` или `.zsav` могут быть прочитаны модулем IBM SPSS Statistics Base и другими продуктами. Это формат, используемый также для файлов кэша в IBM SPSS Modeler.

Пример

```
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)
```

Таблица 259. Свойства statisticsexportnode

Свойства statisticsexportnode	**Тип переменной**	**Описание свойства**
`full_filename`	*string*	

`file_type`	`sav`
	
	`zsav`
		Сохранить файл в формате *sav* или *zsav*. Например:
		
		`statisticsexportnode.setPropertyValue("file_type","sav")`

`encrypt_file`	*флаг*	Защищен ли файл паролем.
`password`	*string*	Пароль.
`launch_application`	*флаг*	

`export_names`	`NamesAndLabels`
	
	`NamesAsLabels`
		Используется для отображения имен полей из IBM SPSS Modeler при экспорте на IBM SPSS Statistics или на имена переменных SAS.

`generate_import`	*флаг*	

Глава 19. Свойства узла Python

Свойства gmm

Модель Gaussian Mixture© (смешанная Гауссова) - это вероятностная модель, которая предполагает, что все точки данных создаются из смеси конечного числа нормальных распределений с неизвестными параметрами. Можно считать смешанные модели обобщением кластеризации k-средних для включения информации о структуре ковариаций данных, а также центров скрытых гауссианов. Смешанный Гауссов узел в SPSS Modeler проявляет базовые функции и обычно используемые параметры библиотеки смешанных Гауссовых функций. Этот узел реализован на языке Python.

Таблица 260. Свойства gmm

Свойства gmm	**Тип переменной**	**Описание свойства**
use_partition	*логический*	Задайте значение True или False, чтобы указать, следует ли использовать разделенные данные. Значение по умолчанию - False.
вид_ковариации	*строка*	Задайте тип ковариации Полная, Связанная, Диагональная или Сферическая.
number_component	*целое*	Задайте целое число для числа компонентов смеси. Минимальное значение - 1. Значение по умолчанию - 2.
component_lable	*логический*	Задайте True, чтобы метка кластера была строкой, или False, чтобы метка кластера была числом. Значение по умолчанию - False.
label_prefix	*строка*	Если используется метка кластера строк, можно указать префикс.
enable_random_seed	*логический*	Укажите значение True, если хотите использовать случайное начальное значение. Значение по умолчанию - False.
random_seed	*целое*	Если используется случайное начальное значение, то укажите целое число, которое будет использоваться для генерирования случайных выборок.
tol	*Двойной точности*	Укажите порог сходимости. Значение по умолчанию - 0.000.1.
max_iter	*целое*	Укажите максимальное число итераций, которое нужно выполнить. Значение по умолчанию - 100.

Таблица 260. Свойства gmm (продолжение)		
Свойства gmm	**Тип переменной**	**Описание свойства**
init_params	*строка*	Задайте используемый параметр инициализации. Опции: K-средние или Случайные.
warm_start	*логический*	Задайте True, чтобы использовать в качестве инициализации для следующей подгонки решение последней подгонки. Значение по умолчанию - False.

Свойства hdbscannode

Иерархическая пространственная кластеризация на основе плотности (Hierarchical Density-Based Spatial Clustering, HDBSCAN©) использует обучение без учителя для поиска кластером или областей высокой плотности в наборе данных. Узел HDBSCAN в SPSS Modeler проявляет базовые функции и обычно используемые параметры библиотеки HDBSCAN. Этот узел реализован на языке Python и может использоваться для кластеризации набора данных на отдельные группы, когда вы изначально не знаете, что собой представляют эти группы.

Таблица 261. Свойства hdbscannode		
Свойства hdbscannode	**Тип переменной**	**Описание свойства**
inputs	*поле*	Поля ввода для кластеризации.
useHPO	*логический*	Задайте true или false, чтобы включить или выключить гиперпараметрическую оптимизацию (Hyper-Parameter Optimization, HPO) на основе Rbfopt, автоматически обнаруживающую оптимальную комбинацию параметров так, чтобы модель достигала ожидаемого или более низкого коэффициента ошибок на выборках. Значение по умолчанию - false.
min_cluster_size	*целое*	Минимальный размер кластеров. Задайте целое число. Значение по умолчанию - 5.
min_samples	*целое*	Минимальный размер выборки в окрестности для точки, чтобы она рассматривалась как точка ядра. Задайте целое число. Если установлено в 0, то используетсяmin_cluster_size. Значение по умолчанию - 0.
algorithm	*string*	Задайте используемый алгоритм: best, generic, prims_kdtree, prims_balltree, boruvka_kdtree или boruvka_balltree. Значение по умолчанию - best.

Таблица 261. Свойства hdbscannode (продолжение)

Свойства hdbscannode	Тип переменной	Описание свойства
metric	*string*	Задайте, какую метрику использовать для вычисления расстояния между экземплярами в массиве функции: euclidean, cityblock, L1, L2, manhattan, braycurtis, canberra, chebyshev, correlation, minkowski или sqeuclidean. Значение по умолчанию - euclidean.
useStringLabel	*логический*	Задайте true, чтобы использовать строковую метку кластера, или false, чтобы использовать числовую метку кластера. Значение по умолчанию - false.
stringLabelPrefix	*string*	Если для параметра useStringLabel задано значение true, задайте значение для префикса строковой метки. Префикс по умолчанию - cluster.
approx_min_span_tree	*логический*	Задайте true, чтобы принять примерное минимальное дерево охвата, или false, если вы хотите пожертвовать скоростью ради точности. Значение по умолчанию - true.
cluster_selection_method	*string*	Задайте используемый метод для выбора кластеров в сжатом дереве: eom или leaf. Значение по умолчанию - eom (алгоритм Excess of Mass - избыток массы).
allow_single_cluster	*логический*	Задайте true, если вы хотите разрешить результаты в виде единственного кластера. Значение по умолчанию - false.
p_value	*double*	Задайте p-значение, используемое для метрики minkowski. Значение по умолчанию - 1,5.
leaf_size	*целое*	Если используется алгоритм пространственного дерева (boruvka_kdtree или boruvka_balltree), задайте число точек в конечном узле дерева. Значение по умолчанию - 40.
outputValidity	*логический*	Задайте true или false, чтобы включать или не включать диаграмму Индекс допустимости в вывод модели.
outputCondensed	*логический*	Задайте true или false, чтобы включать или не включать диаграмму Сжатое дерево в вывод модели.

Таблица 261. Свойства hdbscannode (продолжение)		
Свойства hdbscannode	**Тип переменной**	**Описание свойства**
outputSingleLinkage	*логический*	Задайте true или false, чтобы включать или не включать диаграмму Дерево ближайших соседей в вывод модели.
outputMinSpan	*логический*	Задайте true или false, чтобы включать или не включать диаграмму Минимальное дерево охвата в вывод модели.
is_split		Добавлено в версию 18.2.1.1.

Свойства kdemodel

Ядерная оценка плотности (KDE)© использует алгоритмы дерева шаров или дерева KDE для эффективных запросов и сочетает идеи неконтролируемого обучения, инженерного проектирования и моделирования данных. Подходы на основе соседей, такие как KDE - это некоторые наиболее популярные и полезные методы оценки плотности. Узел Моделирование KDE и Имитация KDE в SPSS Modeler проявляют базовые функции и обычно используемые параметры библиотеки KDE. Узлы реализованы на языке Python.

Таблица 262. Свойства kdemodel		
Свойства kdemodel	**Тип переменной**	**Описание свойства**
bandwidth	*double*	Значение по умолчанию - 1.
kernel	*строка*	Используемое ядро: гауссово, прямоугольное, Епанечникова, экспоненциальное, линейное или косинусное. Значение по умолчанию - Гауссово.
algorithm	*строка*	Алгоритм дерева, который нужно использовать: kd_tree, ball_tree или auto. Значение по умолчанию - auto.
metric	*строка*	Метрика для использования при вычислении расстояния. Для алгоритма kd_tree выбирайте из: Евклидова, Чебышева, Городских кварталов, Минковского, Манхеттен, Бесконечности, P, L2 или L1. Для алгоритма ball_tree выбирайте из: Евклидова, Брея-Кёртиса, Чебышева, Канберра, Городских кварталов, Дайса, Хэмминга, Бесконечности, Жаккарда, L1, L2, Минковского, Совпадения, Манхеттен, P, Роджерса-Танимото, Расселла-Рао, Сокала-Миченера, Сокала-Сниата или Кульчинского. Значение по умолчанию - Евклидова.

Таблица 262. Свойства kdemodel (продолжение)		
Свойства kdemodel	**Тип переменной**	**Описание свойства**
atol	*плавающее*	Требуемый абсолютный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 0.0.
rtol	*плавающее*	Требуемый относительный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 1E-8.
breadthFirst Переименовано в breadth_first, начиная с версии 18.2.1.1	*логический*	Задайте значение True, чтобы использовать подход в ширину. Задайте значение False, чтобы использовать подход в глубину. Значение по умолчанию - True.
LeafSize Переименовано в leaf_size, начиная с версии 18.2.1.1	*целое*	Размер листа в базовом дереве. Значение по умолчанию - 40. Изменение этого значения может существенно повлиять на производительность.
pValue	*double*	Задайте P-значение, используемое для метрики Minkowski. Значение по умолчанию - 1,5.
custom_name		
default_node_name		
use_HPO		

свойства kdeexport

Ядерная оценка плотности (KDE)© использует алгоритмы дерева шаров или дерева KDE для эффективных запросов и сочетает идеи неконтролируемого обучения, инженерного проектирования и моделирования данных. Подходы на основе соседей, такие как KDE - это некоторые наиболее популярные и полезные методы оценки плотности. Узел Моделирование KDE и Имитация KDE в SPSS Modeler проявляют базовые функции и обычно используемые параметры библиотеки KDE. Узлы реализованы на языке Python.

Таблица 263. свойства kdeexport		
Свойства kdeexport	**Тип переменной**	**Описание свойства**
bandwidth	*double*	Значение по умолчанию - 1.
kernel	*строка*	Используемое ядро: гауссово или прямоугольное. Значение по умолчанию - Гауссово.
algorithm	*строка*	Алгоритм дерева, который нужно использовать: kd_tree, ball_tree или auto. Значение по умолчанию - auto.

Таблица 263. свойства kdeexport (продолжение)		
Свойства kdeexport	**Тип переменной**	**Описание свойства**
metric	*строка*	Метрика для использования при вычислении расстояния. Для алгоритма kd_tree выбирайте из: Евклидова, Чебышева, Городских кварталов, Минковского, Манхеттен, Бесконечности, P, L2 или L1. Для алгоритма ball_tree выбирайте из: Евклидова, Брея-Кёртиса, Чебышева, Канберра, Городских кварталов, Дайса, Хэмминга, Бесконечности, Жаккарда, L1, L2, Минковского, Совпадения, Манхеттен, P, Роджерса-Танимото, Расселла-Рао, Сокала-Миченера, Сокала-Сниата или Кульчинского. Значение по умолчанию - Евклидова.
atol	*плавающее*	Требуемый абсолютный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 0.0.
rtol	*плавающее*	Требуемый относительный допуск результата. Чем выше уровень допуска, тем, как правило, быстрее выполнение. Значение по умолчанию - 1E-8.
breadthFirst	*логический*	Задайте значение True, чтобы использовать подход в ширину. Задайте значение False, чтобы использовать подход в глубину. Значение по умолчанию - True.
LeafSize	*целое*	Размер листа в базовом дереве. Значение по умолчанию - 40. Изменение этого значения может существенно повлиять на производительность.
pValue	*double*	Задайте P-значение, используемое для метрики Minkowski. Значение по умолчанию - 1,5.

Свойства gmm

Модель Gaussian Mixture© (смешанная Гауссова) - это вероятностная модель, которая предполагает, что все точки данных создаются из смеси конечного числа нормальных распределений с неизвестными параметрами. Можно считать смешанные модели обобщением кластеризации k-средних для включения информации о структуре ковариаций данных, а также центров скрытых гауссианов. Смешанный Гауссов узел в SPSS Modeler проявляет базовые функции и обычно используемые параметры библиотеки смешанных Гауссовых функций. Этот узел реализован на языке Python.

Таблица 264. Свойства gmm		
Свойства gmm	**Тип переменной**	**Описание свойства**
use_partition	*логический*	Задайте значение True или False, чтобы указать, следует ли использовать разделенные данные. Значение по умолчанию - False.
вид_ковариации	*строка*	Задайте тип ковариации Полная, Связанная, Диагональная или Сферическая.
number_component	*целое*	Задайте целое число для числа компонентов смеси. Минимальное значение - 1. Значение по умолчанию - 2.
component_lable	*логический*	Задайте True, чтобы метка кластера была строкой, или False, чтобы метка кластера была числом. Значение по умолчанию - False.
label_prefix	*строка*	Если используется метка кластера строк, можно указать префикс.
enable_random_seed	*логический*	Укажите значение True, если хотите использовать случайное начальное значение. Значение по умолчанию - False.
random_seed	*целое*	Если используется случайное начальное значение, то укажите целое число, которое будет использоваться для генерирования случайных выборок.
tol	*Двойной точности*	Укажите порог сходимости. Значение по умолчанию - 0.000.1.
max_iter	*целое*	Укажите максимальное число итераций, которое нужно выполнить. Значение по умолчанию - 100.
init_params	*строка*	Задайте используемый параметр инициализации. Опции: K-средние или Случайные.
warm_start	*логический*	Задайте True, чтобы использовать в качестве инициализации для следующей подгонки решение последней подгонки. Значение по умолчанию - False.

свойства ocsvmnode

Узел One-Class SVM использует неконтролируемый алгоритм обучения. Этот узел можно использовать для обнаружения новизны. Он будет обнаруживать нерезкую границу данного набора выборок, чтобы затем классифицировать новые точки как принадлежащие к этому набору или нет. Этот узел моделирования One-Class SVM в SPSS Modeler реализован на языке Python, и для него требуется библиотека Python scikit-learn©.

Таблица 265. свойства ocsvmnode		
Свойства ocsvmnode	**Тип переменной**	**Описание свойства**
role_use Переименовано в custom_fields, начиная с версии 18.2.1.1	*строка*	Задайте значение predefined для использования заранее заданных ролей или значение custom для использования пользовательских назначений полей. Значение по умолчанию - predefined.
splits	*поле*	Список имён полей для расщепления.
use_partition	*Логический*	Задайте true или false. Значение по умолчанию - true. Если задать значение true, при построении модели будут использоваться только данные обучения.
mode_type	*строка*	Режим. Возможные значения - simple или expert. Если задать simple, на вкладке Эксперт будут выведены все параметры.
stopping_criteria	*строка*	Строка в экспоненциальной записи. Возможные значения - 1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5 или 1.0E-6. Значение по умолчанию - 1.0E-3.
precision	*плавающее*	Точность регрессии (ню). Граница для доли ошибок обучения и опорных векторов. Задайте число больше 0 и не больше 1.0. Значение по умолчанию - 0.1.
kernel	*строка*	Тип ядра для использования в алгоритме. Возможные значения: linear, poly, rbf, sigmoid или precomputed. Значение по умолчанию - rbf.
enable_gamma	*Логический*	Разрешает использование параметра gamma. Задайте true или false. Значение по умолчанию - true.
gamma	*плавающее*	Поддержка этого параметра включается только для ядер rbf, poly и sigmoid. Если для параметра enable_gamma задать значение false, для этого параметра будет задано значение auto. Если задать значение true, будет использоваться значение по умолчанию 0,1.
coef0	*плавающее*	Независимый член в функции ядра. Поддержка этого параметра включается только для ядра poly и ядра sigmoid. Значение по умолчанию - 0.0.

Таблица 265. свойства ocsvmnode (продолжение)

Свойства ocsvmnode	Тип переменной	Описание свойства
degree	*целое*	Степень полиномиальной функции ядра. Поддержка этого параметра включается только для ядра poly. Задайте любое целое число. Значение по умолчанию - 3.
shrinking	*Логический*	Указывает, использовать ли опцию сокращения. Задайте true или false. Значение по умолчанию - false.
enable_cache_size	*Логический*	Разрешает использование параметра cache_size. Задайте true или false. Значение по умолчанию - false.
cache_size	*плавающее*	Размер кэша ядра в Мбайтах. Значение по умолчанию - 200.
pc_type	*строка*	Тип диаграммы параллельных координат. Возможные опции - independent или general.
lines_amount	*целое*	Максимальное число строк, включаемых в диаграмму. э Укажите целое число от 1 до 1000.
lines_fields_custom	*Логический*	Включает поддержку параметра lines_fields, разрешающего задавать пользовательские поля для включения в вывод диаграммы. Если задать значение false, будут выводиться все поля. Если задать значение true, будут выводиться только поля, заданные с параметром lines_fields. По причинам производительности будет показано не более 20 полей.
lines_fields	*поле*	Список имён полей, включаемых в диаграмму в качестве ординат.
enable_graphic	*Логический*	Задайте true или false. Включает графический вывод (отключите, чтобы сократить время и размер потокового файла).
enable_hpo	*Логический*	Задайте true или false, чтобы включить или отключить свойства HPO. Если задано true, будет применен Rbfopt для автоматического поиска "наилучшей" модели SVM с одним классом, которая достигает целевого значения показателя, определенного пользователем при помощи следующего параметра target_objval.

Таблица 265. свойства ocsvmnode (продолжение)		
Свойства ocsvmnode	**Тип переменной**	**Описание свойства**
target_objval	*плавающее*	Значение целевой функции (частоты ошибок модели на выборках), которого мы хотим достичь (например, значение неизвестного оптимума). Задайте для этого параметра подходящее значение, если оптимум неизвестен (например, 0.01).
max_iterations	*целое*	Максимальное число итераций для попыток построения модели. Значение по умолчанию - 1000.
max_evaluations	*целое*	Максимальное число оценок функций для попыток построения модели, если наша цель - точность, а не скорость. Значение по умолчанию - 300.

Свойства rfnode

Узел Случайный лес использует улучшенную реализацию алгоритма бэггинга с моделью дерева в качестве базовой. Этот узел моделирования Случайный лес в SPSS Modeler реализован на языке Python, и для него требуется библиотека Python scikit-learn©.

Таблица 266. Свойства rfnode		
Свойства rfnode	**Тип переменной**	**Описание свойства**
role_use	*строка*	Задайте значение predefined для использования заранее заданных ролей или значение custom для использования пользовательских назначений полей. Значение по умолчанию - predefined.
inputs	*поле*	Список имён полей для ввода.
splits	*поле*	Список имён полей для расщепления.
n_estimators	*целое*	Число создаваемых деревьев. Значение по умолчанию - 10.
specify_max_depth	*Логический*	Задать пользовательскую максимальную глубину. При значении false узлы раскрываются, пока все листья не станут однородными или пока каждый лист не будет содержать менее min_samples_split примеров. Значение по умолчанию - false.
max_depth	*целое*	Максимальная глубина дерева. Значение по умолчанию - 10.
min_samples_leaf	*целое*	Минимальный размер конечного узла. Значение по умолчанию - 1.

Таблица 266. Свойства rfnode (продолжение)

Свойства rfnode	Тип переменной	Описание свойства
max_features	*строка*	Число рассматриваемых возможностей при поиске наилучшего расщепления: • Если задано значение auto, используется max_features=sqrt(n_features) для классификатора и max_features=sqrt(n_features) для регрессии. • Если задано значение sqrt, используется max_features=sqrt(n_features). • Если задано значение log2, используется max_features=log2(n_features). Значение по умолчанию - auto.
bootstrap	*Логический*	Использовать выборки бутстрепа при построении деревьев. Значение по умолчанию - true.
oob_score	*Логический*	Применять неиспользуемую часть выборки для оценки точности генерализации. Значение по умолчанию - false.
extreme	*Логический*	Использовать предельно рандомизированные деревья. Значение по умолчанию - false.
use_random_seed	*Логический*	Задайте этот параметр, чтобы получить воспроизводимые результаты. Значение по умолчанию - false.
random_seed	*целое*	Начальное значение для генератора псевдослучайных чисел, используемое при построении деревьев. Задайте любое целое число.
cache_size	*плавающее*	Размер кэша ядра в Мбайтах. Значение по умолчанию - 200.
enable_random_seed	*Логический*	Разрешает использование параметра random_seed. Задайте true или false. Значение по умолчанию - false.
enable_hpo	*Логический*	Задайте true или false, чтобы включить или отключить свойства HPO. Если задано true, будет применен Rbfopt для автоматического определения "наилучшей" модели случайного леса, которая достигает целевого значения показателя, определенного пользователем при помощи следующего параметра target_objval.

Таблица 266. Свойства rfnode (продолжение)		
Свойства rfnode	**Тип переменной**	**Описание свойства**
target_objval	*плавающее*	Значение целевой функции (частоты ошибок модели на выборках), которого вы хотите достичь (например, значение неизвестного оптимума). Задайте для этого параметра подходящее значение, если оптимум неизвестен (например, 0.01).
max_iterations	*целое*	Максимальное число итераций для попыток построения модели. Значение по умолчанию - 1000.
max_evaluations	*целое*	Максимальное число оценок функций для попыток построения модели, если наша цель - точность, а не скорость. Значение по умолчанию - 300.

Свойства smotenode

Узел алгоритма синтетической дискретизации с уменьшением шага для миноритарного класса (Synthetic Minority Over-sampling Technique, SMOTE) представляет алгоритм дискретизации с уменьшением шага для работы с несбалансированными наборами данных. Он обеспечивает улучшенный способ балансировки данных. Узел процесса SMOTE в SPSS Modeler реализован на языке Python и требует библиотеки Python imbalanced-learn©.

Таблица 267. Свойства smotenode		
Свойства smotenode	**Тип переменной**	**Описание свойства**
target_field Переименовано в target, начиная с версии 18.2.1.1	*поле*	Поле назначения.
sample_ratio	*строка*	Включает поддержку пользовательского значения соотношения. Можно либо использовать Автоматический выбор (sample_ratio_auto), либо задать соотношение (sample_ratio_manual).
sample_ratio_value	*плавающее*	Этот показатель представляет собой отношение дискретизации выборки в миноритарном классе к дискретизации в мажоритарном классе. Оно должно быть больше 0 и не больше 1. Значение по умолчанию - auto.
enable_random_seed	*Логический*	Если задать значение true, будет включено свойство random_seed.
random_seed	*целое*	Начальное значение, используемое генератором случайных чисел.

Таблица 267. Свойства smotenode (продолжение)		
Свойства smotenode	**Тип переменной**	**Описание свойства**
k_neighbours	*целое*	Число ближайших соседей, которые будут использоваться для построения синтетических примеров. Значение по умолчанию - 5.
m_neighbours	*целое*	Число ближайших соседей, которые будут использоваться для определения, находится ли миноритарный пример в опасности. Эта опция включается только с алгоритмом SMOTE типов borderline1 и borderline2. Значение по умолчанию - 10.
algorithm_kind Переименовано в algorithm, начиная с версии 18.2.1.1	*строка*	Тип алгоритма SMOTE: regular, borderline1 или borderline2.
usepartition Переименовано в use_partition, начиная с версии 18.2.1.1	*Логический*	Если задать значение true, для построения модели будут использоваться только данные обучения. Значение по умолчанию - true.

Свойства tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) - инструмент для визуализации многомерных данных. Он преобразует аффинитеты точек данных в вероятности. Этот узел t-SNE SPSS Modeler реализован на языке Python, и для него требуется библиотека Python scikit-learn©.

Таблица 268. Свойства tsnenode		
Свойства tsnenode	**Тип переменной**	**Описание свойства**
mode_type	*строка*	Задайте режим - simple или expert.
n_components	*строка*	Размерность встроенного пространства (2D или 3D). Задайте 2 или 3. Значение по умолчанию - 2.
method	*строка*	Задайте barnes_hut или exact. Значение по умолчанию - barnes_hut.
init	*строка*	Инициализация встраивания. Задайте random или pca. Значение по умолчанию - random.
target_field Переименовано в target, начиная с версии 18.2.1.1	*строка*	Имя поля назначения. Это будет цветовая карта выходного изображения. Если поле назначения не указано, в изображении будет использоваться один цвет.

Таблица 268. Свойства tsnenode (продолжение)

Свойства tsnenode	Тип переменной	Описание свойства
perplexity	*плавающее*	Перплексия связана с числом ближайших соседей, используемых в других алгоритмах множественного обучения. Для больших наборов данных обычно требуется большая перплексия. Обдумайте выбор значения от 5 до 50. Значение по умолчанию - 30.
early_exaggeration	*плавающее*	Управляет тем, как тесно естественные кластеры в исходном пространстве находятся во вложенном пространстве, и сколько места будет между ними. Значение по умолчанию - 12,0.
learning_rate	*плавающее*	Значение по умолчанию - 200.
n_iter	*целое*	Максимальное число итераций для оптимизации. Задайте не меньше 250. Значение по умолчанию - 1000.
угол	*плавающее*	Угловой размер удаленного узла, измеренный с точки. Задайте значение в диапазоне от 0 до 1. Значение по умолчанию - 0,5.
enable_random_seed	*Логический*	Задайте true, чтобы включить параметр random_seed. Значение по умолчанию - false.
random_seed	*целое*	Используемое начальное значение для генератора псевдослучайных чисел. Значение по умолчанию - None.
n_iter_without_progress	*целое*	Максимальное число итераций, не дающих прогресса. Значение по умолчанию - 300.
min_grad_norm	*строка*	Если норма градиента ниже этого порога, оптимизация будет остановлена. Значение по умолчанию - 1.0E-7. Возможные значения: • 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	*Логический*	Задайте true, чтобы выполнить t-SNE с несколькими различными значениями перплексии. Значение по умолчанию - false.

Таблица 268. Свойства tsnenode (продолжение)		
Свойства tsnenode	**Тип переменной**	**Описание свойства**
output_Rename	*Логический*	Задайте true, если вы хотите указать пользовательское имя, или false, чтобы назвать вывод автоматически. Значение по умолчанию - false.
output_to	*строка*	Задайте Screen или Output. Значение по умолчанию - Screen.
full_filename	*строка*	Задайте имя файла вывода.
output_file_type	*строка*	Формат файл вывода. Задайте HTML или Output object. Значение по умолчанию - HTML.

Свойства xgboostlinearnode

Алгоритм Линейный XGBoost© - улучшенная реализация алгоритма градиентного бустинга с линейной моделью в качестве базовой. Алгоритмы бустинга итеративно обучают слабые классификаторы и затем добавляют их к итоговому сильному классификатору. Узел Линейный XGBoost в SPSS Modeler реализован на языке Python.

Таблица 269. Свойства xgboostlinearnode		
Свойства xgboostlinearnode	**Тип переменной**	**Описание свойства**
TargetField Переименовано в target, начиная с версии 18.2.1.1	*поле*	
InputFields Переименовано в inputs, начиная с версии 18.2.1.1	*поле*	
alpha	*Двойной точности*	Параметр альфа линейного бустера. Задайте любое число не меньше 0. Значение по умолчанию - 0.
лямбда	*Двойной точности*	Параметр лямбда линейного бустера. Задайте любое число не меньше 0. Значение по умолчанию - 1.
lambdaBias	*Двойной точности*	Смещение лямбда линейного бустера. Задайте любое число. Значение по умолчанию - 0.
numBoostRound Переименовано в num_boost_round, начиная с версии 18.2.1.1	*целое*	Число циклов бустинга для построения моделей. Укажите значение от 1 до 1000. Значение по умолчанию - 10.

Таблица 269. Свойства xgboostlinearnode (продолжение)		
Свойства xgboostlinearnode	**Тип переменной**	**Описание свойства**
objectiveType	*строка*	Целевой тип для задачи обучения. Возможны значения -reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic или multi. Имейте в виду, что для флаговых полей назначения можно использовать только binary:logistic или multi. Если используется multi, в результате скоринга будут выводится целевые типы XGBoost multi:softmax и multi:softprob.
random_seed	*целое*	Начальное значение для генератора псевдослучайных чисел. Любое число между 0 и 9999999. Значение по умолчанию - 0.
useHPO	*Логическое значение*	Задайте true или false, чтобы включить или отключить свойства HPO. Если задано true, будет применен Rbfopt для автоматического поиска "наилучшей" модели SVM с одним классом, которая достигает целевого значения показателя, определенного пользователем при помощи параметра target_objval.

Свойства xgboosttreenode

Алгоритм Дерево XGBoost© - улучшенная реализация алгоритма градиентного бустинга с моделью дерева в качестве базовой. Алгоритмы бустинга итеративно обучают слабые классификаторы и затем добавляют их к итоговому сильному классификатору. Дерево XGBoost - очень гибкий алгоритм, предоставляющий много параметров, которые могут показаться чрезмерными большинству пользователей, поэтому узел Дерево XGBoost в SPSS Modeler представляет основные возможности и чаще используемые параметры. Этот узел реализован на языке Python.

Таблица 270. Свойства xgboosttreenode		
Свойства xgboosttreenode	**Тип переменной**	**Описание свойства**
TargetField Переименовано в target, начиная с версии 18.2.1.1	*поле*	Поля назначения.
InputFields Переименовано в inputs, начиная с версии 18.2.1.1	*поле*	Входные поля.

Таблица 270. Свойства xgbosttreenode (продолжение)

Свойства xgbosttreenode	Тип переменной	Описание свойства
treeMethod Переименовано в tree_method, начиная с версии 18.2.1.1	*строка*	Метод дерева для построения моделей. Возможные значения - auto, exact или approx. Значение по умолчанию - auto.
numBoostRound Переименовано в num_boost_round, начиная с версии 18.2.1.1	*целое*	Число циклов бустинга для построения моделей. Укажите значение от 1 до 1000. Значение по умолчанию - 10.
maxDepth Переименовано в max_depth, начиная с версии 18.2.1.1	*целое*	Максимальная глубина для роста дерева. Укажите значение не меньше 1. Значение по умолчанию - 6.
minChildWeight Переименовано в min_child_weight, начиная с версии 18.2.1.1	*Двойной точности*	Минимальный вес дочерних для роста дерева. Укажите значение не меньше 0. Значение по умолчанию - 1.
maxDeltaStep Переименовано в max_delta_step, начиная с версии 18.2.1.1	*Двойной точности*	Максимальный шаг приращения для роста дерева. Укажите значение не меньше 0. Значение по умолчанию - 0.
objectiveType Переименовано в objective_type, начиная с версии 18.2.1.1	*строка*	Целевой тип для задачи обучения. Возможны значения -reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic или multi. Имейте в виду, что для флаговых полей назначения можно использовать только binary:logistic или multi. Если используется multi, в результате скоринга будут выводится целевые типы XGBoost multi:softmax и multi:softprob.
earlyStopping Переименовано в early_stopping, начиная с версии 18.2.1.1	*Логическое значение*	Использовать ли функцию ранней остановки. Значение по умолчанию - False.
earlyStoppingRounds Переименовано в early_stopping_rounds, начиная с версии 18.2.1.1	*целое*	Ошибки проверки должны уменьшаться по крайней мере на каждом раунде ранней остановки, чтоб продолжать обучение. Значение по умолчанию - 10.

Таблица 270. Свойства xgboosttreenode (продолжение)		
Свойства xgboosttreenode	**Тип переменной**	**Описание свойства**
evaluationDataRatio Переименовано в evaluation_data_ratio, начиная с версии 18.2.1.1	*двойной точности*	Доля входных данных, используемая для ошибок оценки. Значение по умолчанию - 0.3.
random_seed	*целое*	Начальное значение для псевдослучайных чисел. Любое число между 0 и 9999999. Значение по умолчанию - 0.
sampleSize Переименовано в sample_size, начиная с версии 18.2.1.1	*Двойной точности*	Подвыборка для управления избыточной подгонкой (переобучением). Укажите значение от 0.1 до 1.0. Значение по умолчанию - 0.1.
эта	*Двойной точности*	Эта для управления переобучением. Укажите значение от 0 до 1. Значение по умолчанию - 0.3.
gamma	*Двойной точности*	Гамма для управления переобучением. Задайте любое число не меньше 0. Значение по умолчанию - 6.
colsSampleRatio Переименовано в col_sample_ratio, начиная с версии 18.2.1.1	*Двойной точности*	Выборка столбцов по дереву для управления переобучением. Укажите значение от 0.01 до 1. Значение по умолчанию - 1.
colsSampleLevel Переименовано в col_sample_level, начиная с версии 18.2.1.1	*Двойной точности*	Выборка столбцов по уровню для управления переобучением. Укажите значение от 0.01 до 1. Значение по умолчанию - 1.
лямбда	*Двойной точности*	Лямбда для управления переобучением. Задайте любое число не меньше 0. Значение по умолчанию - 1.
alpha	*Двойной точности*	Альфа для управления переобучением. Задайте любое число не меньше 0. Значение по умолчанию - 0.
scalePosWeight Переименовано в scale_pos_weight, начиная с версии 18.2.1.1	*Двойной точности*	Положительный вес масштабирования для несбалансированных наборов данных. Значение по умолчанию - 1.
use_HPO Добавлено для версии 18.2.1.1		

Глава 20. Свойства узла Spark

Свойства isotonicasnode

Изотоническая регрессия принадлежит к семейству алгоритмов регрессии. Узел Isotonic-AS в SPSS Modeler реализован на языке Spark. Подробности об изотонической регрессии смотрите в разделе https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html.

Таблица 271. Свойства isotonicasnode

Свойства isotonicasnode	Тип переменной	Описание свойства
label	*строка*	Это свойство - зависимая переменная, для которой вычисляется изотоническая регрессия.
элементы	*строка*	Это свойство - независимая переменная.
weightCol	*строка*	Вес представляет количество применяемых мер. Значение по умолчанию - 1.
isotonic	*Логический*	Это свойство указывает, является ли тип Изотоническим или Антитоническим.
featureIndex	*целое*	Это свойство для индекса функции, если featuresCol - векторный столбец. Значение по умолчанию - 0.

Свойства kmeansasnode

K-Means - это один из наиболее часто используемых алгоритмов кластеризации. Он распределяет точки данных по заданному числу кластеров. Узел K-Means-AS в SPSS Modeler реализован на языке Spark. Подробности об алгоритмах K-Means смотрите в разделе https://spark.apache.org/docs/2.2.0/ml-clustering.html. Обратите внимание на то, что узел K-Means-AS автоматически применяет для категорийных переменных унитарное кодирование.

Таблица 272. свойства kmeansasnode

Свойства kmeansasnode	Значения	Описание свойства
roleUse	*строка*	Задайте значение predefined для использования заранее заданных ролей или значение custom для использования пользовательских назначений полей. Значение по умолчанию - predefined.

Таблица 272. свойства kmeansasnode (продолжение)

Свойства kmeansasnode	Значения	Описание свойства
autoModel	*Логический*	Задайте true, чтобы использовать имя по умолчанию ($S-prediction) для нового поля сгенерированной оценки, или false, чтобы использовать пользовательское имя. Значение по умолчанию - true.
элементы	*поле*	Список имён полей для ввода, когда для roleUse задано значение custom.
name	*строка*	Название нового поля сгенерированной оценки, когда для свойства autoModel задано значение false.
clustersNum	*целое*	Число создающихся кластеров. Значение по умолчанию - 5.
initMode	*строка*	Алгоритм инициализации. Возможные значения: k-means|| или random. Значение по умолчанию - k-means||.
initSteps	*целое*	Число шагов инициализации, когда для initMode задано значение k-means||. Значение по умолчанию - 2.
advancedSettings	*Логический*	Задайте true, чтобы сделать доступными следующие четыре свойства. Значение по умолчанию - false.
maxIteration	*целое*	Максимальное число итераций для кластеризации. Значение по умолчанию - 20.
tolerance	*строка*	Допуск для прекращения итераций. Возможные значения 1,0E-1, 1,0E-2, ..., 1,0E-6. Значение по умолчанию - 1,0E-4.
setSeed	*Логический*	Задайте true, чтобы использовать пользовательское начальное значение случайных чисел. Значение по умолчанию - false.
randomSeed	*целое*	Пользовательское начальное значение случайных чисел, когда для свойства setSeed задано значение true.

Свойства multilayerperceptronnode

Многослойный перцептрон - это классификатор, основанный на искусственной нейронной сети с прямой связью, состоящий из нескольких слоев. Каждый слой полностью соединен со следующим слоем в сети. Узел MultiLayerPerceptron-AS в SPSS Modeler реализован на языке Spark. Подробную информацию о классификаторе многослойного перцептрона смотрите в разделе https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier.

Таблица 273. Свойства multilayerperceptronnode

Свойства multilayerperceptronnode	**Тип переменной**	**Описание свойства**
элементы	*поле*	Одно или несколько полей, чтобы использовать как входные поля для предсказания.
label	*поле*	Поле в качестве назначения для предсказания.
layers[0]	*целое*	Число слоев перцептрона, которые нужно включить. Значение по умолчанию - 1.
слои[1…<последний-1>]	*целое*	Число скрытых слоев. Значение по умолчанию - 1.
слои[<последний>]	*целое*	Число слоев выходных данных. Значение по умолчанию - 1.
seed	*целое*	Пользовательское начальное значение для генератора псевдослучайных чисел.
maxiter	*целое*	Максимальное число итераций, которое нужно выполнить. Значение по умолчанию - 10.

Свойства xgboostasnode

Алгоритм XGBoost - это улучшенная реализация алгоритма градиентного бустинга. Алгоритмы бустинга итеративно обучают слабые классификаторы и затем добавляют их к итоговому сильному классификатору. XGBoost - очень гибкий алгоритм, предоставляющий много параметров, которые могут показаться чрезмерными большинству пользователей, поэтому узел XGBoost-AS в SPSS Modeler представляет основные возможности и чаще используемые параметры. Узел XGBoost-AS реализован на языке Spark.

Таблица 274. свойства xgboostasnode

Свойства xgboostasnode	**Тип переменной**	**Описание свойства**
target_field	*поле*	Список имен полей для назначения.
input_fields	*поле*	Список имен полей для ввода.

Таблица 274. свойства xgboostasnode (продолжение)

Свойства xgboostasnode	Тип переменной	Описание свойства
nWorkers	*целое*	Число обработчиков, используемых для обучения модели XGBoost. Значение по умолчанию - 1.
numThreadPerTask	*целое*	Число потоков, используемых на каждого обработчика. Значение по умолчанию - 1.
useExternalMemory	*Логический*	Использовать ли внешнюю память как кэш. По умолчанию - **false**.
boosterType	*строка*	Используемый тип бустера. Возможные опции: gbtree, gblinear и dart. Значение по умолчанию - gbtree.
numBoostRound	*целое*	Число циклов бустинга. Укажите значение не меньше 0. Значение по умолчанию - 10.
scalePosWeight	*Двойной точности*	Управление балансом положительных и отрицательных весов. Значение по умолчанию - 1.
randomseed	*целое*	Начальное значение, используемое генератором случайных чисел. Значение по умолчанию - **0**.
objectiveType	*строка*	Цель обучения. Возможны значения -reg:linear, reg:logistic, reg:gamma, reg:tweedie. rank:pairwise, binary:logistic или multi. Имейте в виду, что для флаговых полей назначения можно использовать только binary:logistic или multi. Если используется multi, в результате скоринга будут выводится целевые типы XGBoost multi:softmax и multi:softprob. Значение по умолчанию - reg:linear.
evalMetric	*строка*	Метрики оценки для данных проверки. Метрика по умолчанию будет назначена в соответствии с целевым показателем. Возможные значения: rmse, mae, logloss, error, merror, mlogloss, auc, ndcg, map и gamma-deviance. Значение по умолчанию - rmse.
lambda	*Двойной точности*	Член регуляризации L2 для весов. Увеличение этого значения сделает модель более консервативной. Задайте любое число не меньше 0. Значение по умолчанию - 1.

Таблица 274. свойства xgboostasnode (продолжение)

Свойства xgboostasnode	Тип переменной	Описание свойства
alpha	*Двойной точности*	Член регуляризации L1 для весов. Увеличение этого значения сделает модель более консервативной. Задайте любое число не меньше 0. Значение по умолчанию - 0.
lambdaBias	*Двойной точности*	Член регуляризации L2 для смещения. Параметр смещения лямбда линейного бустинга доступен, если используется тип бустинга gblinear. Задайте любое число не меньше 0. Значение по умолчанию - 0.
treeMethod	*строка*	Параметр метода дерева для роста дерева (как и следующие три параметра) доступен, если используется тип бустинга gbtree или dart. Задает алгоритм построения дерева XGBoost. Возможные опции: auto, exact и approx. Значение по умолчанию - auto.
maxDepth	*целое*	Максимальная глубина для деревьев. Укажите значение не меньше 2. Значение по умолчанию - 6.
minChildWeight	*Двойной точности*	Минимальная сумма весов экземпляров (гессиан), необходимая для дочернего элемента. Укажите значение не меньше 0. Значение по умолчанию - 1.
maxDeltaStep	*Двойной точности*	Шаг с максимальной дельтой, чтобы разрешить оценку веса каждого дерева. Укажите значение не меньше 0. Значение по умолчанию - 0.
sampleSize	*Двойной точности*	Подвыборка - это доля экземпляра обучения. Укажите значение от 0.1 до 1.0. Значение по умолчанию - 1.0.
эта	*Двойной точности*	Сокращение размера шага, используемое на шаге обновления для предотвращения переобучения. Укажите значение от 0 до 1. Значение по умолчанию - 0.3.
gamma	*Двойной точности*	Минимальное сокращение потерь, необходимое для выполнения следующего разделения на конечном узле дерева. Задайте любое число не меньше 0. Значение по умолчанию - 6.
colsSampleRatio	*Двойной точности*	Доля подмножества столбцов при построении каждого дерева. Укажите значение от 0,01 и 1. Значение по умолчанию - 1.

Таблица 274. свойства xgboostasnode (продолжение)

Свойства xgboostasnode	Тип переменной	Описание свойства
colsSampleLevel	*Двойной точности*	Доля подмножества для каждого разделения в каждом уровне. Укажите значение от 0,01 и 1. Значение по умолчанию - 1.
normalizeType	*строка*	Этот параметр DART (как и следующие три параметра DART) доступен, если используется тип бустинга dart. Этот параметр задает алгоритм нормализации. Укажите tree или forest. Значение по умолчанию - tree.
sampleType	*строка*	Тип алгоритма выборки. Укажите uniform или weighted. Значение по умолчанию - uniform.
rateDrop	*Двойной точности*	Параметр бустера DART доли отсева. Укажите значение от 0.0 до 1.0. Значение по умолчанию - 0.0.
skipDrop	*Двойной точности*	параметр бустинга DART для вероятности пропуска отсева. Укажите значение от 0.0 до 1.0. Значение по умолчанию - 0.0.

Глава 21. Свойства надузлов

В следующих таблицах описываются свойства, специфичные для надузлов. Обратите внимание на то, что общие свойства узлов применимы также к надузлам.

Таблица 275. Свойства конечных надузлов

Имя свойства	Тип свойства/Список значений	Описание свойства
execute_method	Сценарий Нормальный	
script	*string*	

Параметры надузлов

Сценарии можно использовать для создания или задания параметров надузлов в следующем общем формате:

```
mySuperNode.setParameterValue("minvalue", 30)
```

Значение параметра можно получить следующим образом:

```
value mySuperNode.getParameterValue("minvalue")
```

Нахождение существующих надузлов

Надузлы в потоках можно найти с помощью функции 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)
```

Задание свойств для инкапсулированных узлов

Можно задать свойства для конкретных узлов, инкапсулированных в надузел, обратившись к дочерней диаграмме в составе надузла. Например, допустим, что есть надузел источника с инкапсулированным узлом файлов переменных для чтения данных. Имя файла для чтения (заданное при помощи свойства полное_имя_файла) можно передать, перейдя к дочерней диаграмме и найдя соответствующий узел:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Создание надузлов

Если вы хотите создать надузел и его содержимое с нуля, это можно сделать аналогично, обратившись к дочерней диаграмме и создав нужные узлы. Необходимо обеспечить также, чтобы

узлы из диаграммы надузлов были связаны с узлами входящих и/или исходящих соединений. Например, если нужно создать надузел процесса:

```
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)
```

Приложение A. Ссылки на имена узлов

В этом разделе представлены ссылки на имена сценариев узлов в IBM SPSS Modeler.

Имена слепков моделей

На слепки моделей (также известные как сгенерированные модели) можно ссылаться по типу, как и на узел и выходные объекты. В следующей таблице перечислены имена ссылок на объекты моделей.

Обратите внимание на то, что эти имена используются в частности для ссылки на слепки моделей на палитре Модели (в верхнем правом углу окна IBM SPSS Modeler). Для ссылки на узлы модели, добавленные в поток для целей скоринга, используется другой набор имен с префиксом `apply...`.

Примечание: При обычных обстоятельствах рекомендуется, чтобы исключить путаницу, ссылаться на модели по имени *и* по типу.

Таблица 276. Имена слепков моделей (палитра моделирования)

Имя модели	**Модель**
anomalydetection	Аномалия
априори	Априорный анализ
autoclassifier	Автоклассификатор
autocluster	Автокластер
autonumeric	Автонумерация
bayesnet	Байесовская сеть
c50	C5.0
carma	Carma
cart	Дерево C&R
chaid	CHAID
coxreg	Регрессия Кокса
decisionlist	Список решений
discriminant	Дискриминантный
factor	PCA/фактор
featureselection	Отбор показателей
genlin	Обобщенная линейная регрессия
glmm	GLMM
kmeans	K-средних
knn	*k* ближайших соседей
kohonen	Коонена
Линейная	Линейное
logreg	Логистическая регрессия

Таблица 276. Имена слепков моделей (палитра моделирования) (продолжение)	
Имя модели	**Модель**
neuralnetwork	Нейросеть
quest	QUEST
регрессия	Линейная регрессия
последовательность	Порядковый номер
slrm	Самообучаемая модель откликов
statisticsmodel	Модель IBM SPSS Statistics
svm	Механизм опорных векторов
timeseries	Временные ряды
twostep	TwoStep

Таблица 277. Имена слепков моделей (палитра моделирования базы данных)	
Имя модели	**Модель**
db2imcluster	Кластеризация IBM ISW
db2imlog	Логистическая регрессия IBM ISW
db2imnb	Наивный критерий Байеса IBM ISW
db2imreg	Регрессия IBM ISW
db2imtree	Дерево решений IBM ISW
msassoc	Правила связывания MS
msbayes	Наивный критерий Байеса MS
mscluster	Кластеризация MS
mslogistic	Логистическая регрессия MS
msneuralnetwork	Нейросеть MS
msregression	Линейная регрессия MS
mssequencecluster	Кластеризация последовательностей MS
mstimeseries	Временные ряды MS
mstree	Дерево решений MS
netezzabayes	Байесовская сеть Netezza
netezzadectree	Дерево решений Netezza
netezzadivcluster	Разделительная кластеризация Netezza
netezzaglm	Обобщенный линейный анализ Netezza
netezzakmeans	K-средние Netezza
netezzaknn	KNN Netezza
netezzalineregression	Линейная регрессия Netezza
netezzanaivebayes	Наивный байесовский анализ Netezza
netezzapca	PCA Netezza

Таблица 277. Имена слепков моделей (палитра моделирования базы данных) (продолжение)	
Имя модели	**Модель**
netezzaregtree	Дерево регрессии Netezza
netezzatimeseries	Временные ряды Netezza
oraabn	Адаптивный критерий Байеса Oracle
oraai	AI Oracle
oradecisiontree	Дерево решений Oracle
oraglm	ОЛМ Oracle
orakmeans	*k*-средние Oracle
oranb	Наивный критерий Байеса Oracle
oranmf	NMF Oracle
oraocluster	O-кластер Oracle
orasvm	SVM Oracle

Исключение дублирования имен моделей

При использовании сценариев для работы со сгенерированными моделями имейте в виду, что разрешение дублирования имен моделей может привести к неоднозначным ссылкам. Для исключения этого рекомендуется при написании сценариев потребовать уникальности имен для сгенерированных моделей.

Чтобы задать опции для дублирования имен моделей:

1. Выберите в меню:

   Инструменты > **Пользовательские опции**
2. Щелкните по вкладке **Уведомления**.
3. Выберите **Заменить предыдущую модель**, чтобы ограничить дублирование имен для сгенерированных моделей.

Поведение выполнения сценариев может различаться для SPSS Modeler и IBM SPSS Collaboration and Deployment Services, когда есть неоднозначные ссылки на модель. Клиент SPSS Modeler включает в себя опцию "Заменить предыдущую модель", которая автоматически заменяет модели с одинаковыми именами (например, когда в сценарии происходит цикл для создания всякий раз новой модели). Однако эта опция недоступна, когда тот же сценарий запущен в IBM SPSS Collaboration and Deployment Services. Исключить эту ситуацию можно, или переименовав модель, сгенерированную при каждой итерации, чтобы исключить неоднозначные ссылки на модели, или очистив текущую модель (например, добавляя оператор `clear generated palette`) перед окончанием цикла.

Имена типов вывода

В следующей таблице перечислены все типы объектов вывода и узлы, которые их создали.

Таблица 278. Типы объектов вывода и создающие их узлы	
Тип объектов вывода	**Узел**
analysisoutput	Анализ
collectionoutput	Собрание

Таблица 278. Типы объектов вывода и создающие их узлы (продолжение)

Тип объектов вывода	**Узел**
dataauditoutput	Аудит данных
distributionoutput	Распределение
evaluationoutput	Оценка
histogramoutput	Гистограмма
matrixoutput	Матрица
meansoutput	Средние
multiplotoutput	Несколько графиков
plotoutput	График
qualityoutput	Качество
reportdocumentoutput	Этот тип объектов создан не узлом, такой вывод создается отчетом проекта
reportoutput	Отчет
statisticsprocedureoutput	Вывод Статистика
statisticsoutput	Статистика
tableoutput	Таблица
timeplotoutput	график зависимости от времени
weboutput	Web

Приложение B. Перенастройка от унаследованных сценарием к сценариям Python

Обзор перенастройки унаследованных сценариев

В этом разделе дается обзор различий между сценариями Python и унаследованными сценариями в IBM SPSS Modeler, а также информация о том, как перенастроить унаследованные сценарии в сценарии Python. В этом разделе вы найдете список унаследованных стандартных команд SPSS Modeler и эквивалентных команд Python.

Общие отличия

Во многом структура унаследованных сценариев происходит от командных сценариев операционной системы. Унаследованные сценарии ориентированы на строки, и отступы обычно не играют роли, несмотря на то, что используются некоторые блочные структуры, например, `if...then...else...endif` и `for...endfor`.

В сценариях Python отступы важны, и у строк одного логического блока должен быть один и тот же уровень отступа.

Прим.: Будьте осторожны при копировании кода Python. Строка с отступами, введенными клавишей Tab, может выглядеть в редакторе так же, как строка с отступами, введенными клавишей пробела. Однако сценарий Python сгенерирует ошибку, поскольку он не считает такие отступы строк одинаковыми.

Контекст сценариев

Контекст сценариев определяет среду, в которой выполняется сценарий, например, поток или надузел, вызывающий сценарий. В унаследованных сценариях контекст задается неявно; например, ссылки на узлы в сценарии потока считаются ссылками на узлы в потоке, запустившем сценарий.

В сценариях Python контекст сценариев задается явным образом через модуль `modeler.script`. Например, сценарий потока Python может обратиться к потоку, который выполняет сценарий с таким кодом:

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

После этого через возвращенный объект можно вызывать функции, связанные с потоком.

Команды или функции

Унаследованные сценарии ориентированы на команды. Это значит, что обычно строка сценария начинается с выполняемой команды, за которой следуют параметры, например:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python uses functions that are usually invoked through an object (a module, class or object) that defines the function, for example:

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Литералы и комментарии

Для некоторых команд литералов и комментариев, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 279. Отображение унаследованных сценариев на сценарии Python для литералов и комментариев

Унаследованный сценарий	**Сценарий Python**
Целое число, например, 4	То же
Число с плавающей точкой, например, `0.003`	То же
Строки в одинарных кавычках, например `'Привет'`	То же **Прим.:** Перед строковыми литералами, содержащими не входящие в ASCII символы, надо указывать u, чтобы они интерпретировались как символы Unicode.
Строки в двойных кавычках, например, `"Привет еще раз"`	То же **Прим.:** Перед строковыми литералами, содержащими не входящие в ASCII символы, надо указывать u, чтобы они интерпретировались как символы Unicode.
Длинные строки, например `"""Это строковое значение, которое занимает несколько строк"""`	То же
Списки, например, `[1 2 3]`	`[1, 2, 3]`
Ссылка на переменную, например, `set x = 3`	`x = 3`
Продолжение строки (\), например `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
Блок комментариев, например `/* Это длинный комментарий с переносом на другую строку. */`	`""" это длинный комментарий с переносом на другую строку. """`
Однострочные комментарии, например, `set x = 3 # задать для x значение 3`	`x = 3 # задать для x значение 3`
`undef`	`Нет`
`true`	`True`
`false`	`False`

Операторы

Для некоторых операционных команд, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших

существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 280. Отображение унаследованных сценариев на сценарии Python для операций

Унаследованный сценарий	Сценарий 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`

`и`
`или`
`not(EXPR)`
	`и`
	`или`
	`not EXPR`

Условное выполнение и циклы

Для некоторых команд условного выполнения и циклов, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 281. Отображение унаследованных сценариев на сценарии Python для условных выполнений и циклов

Унаследованный сценарий	Сценарий Python

`for VAR from INT1 to INT2`
`  …`
`endfor`
	`for VAR in range(INT1, INT2):`
	`  …`
	
	или
	
	`VAR = INT1`
	`while VAR <= INT2:`
	`  ...`
	`  VAR += 1`

Таблица 281. Отображение унаследованных сценариев на сценарии Python для условных выполнений и циклов (продолжение)

Унаследованный сценарий	Сценарий 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`
	Нет эквивалента

`var VAR1`	Объявление переменной не требуется

Переменные

В унаследованных сценариях переменные объявляются до их использования, например:

```
var mynode
set mynode = create typenode at 96 96
```

В сценариях Python переменные создаются при первом использовании, например:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

В унаследованных сценариях ссылки на переменные нужно удалять явным образом при помощи операции ^, например:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

В сценариях Python, как и в большинстве языков сценариев, такой необходимости нет, например:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Типы узлов, объектов вывода и моделей

В унаследованных сценариях к конкретному типу узла, объекта вывода и модели обычно добавляется общий тип (node, output и model). Например, узел вычислений имеет тип `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

В API Python IBM SPSS Modeler не добавляет суффикс node, так что тип узла вычислений - derive, например:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

Отсутствие суффикса типа - единственное различие между именами типов в унаследованных сценариях и языке сценариев Python.

Имена свойств

Имена свойств в унаследованных сценариях и в сценариях Python одни и те же. Так, на узле файла переменных свойство, задающее положение файла, называется full_filename в обеих языковых средах.

Ссылки на узлы

Во многих унаследованных сценариях для доступа к редактируемому узлу используется неявный поиск. Например, приведенные ниже команды ищут в текущем потоке узел типа с меткой "Type", а затем задают направление (или роль моделирования) поля "Age" как Входное поле и поля "Drug" - как Поле назначения, то есть предсказываемое:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

В сценариях Python поиск объектов нужно выполнять явно и до того, как вызывается функция, задающая значение свойства, например:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Прим.: В этом случае "Target" берется в кавычки.

Другой вариант - в сценариях Python может использоваться нумерация ModelingRole в пакете modeler.api.

Сценарии Python могут получаться более громоздкими, зато производительность при их выполнении выше, поскольку поиск узла в них, как правило, производится только один раз. В примере унаследованных сценариев поиск узла выполняется для каждой команды.

Поддерживается также поиск узлов по ID (ID узла можно видеть в диалоговом окне узла на вкладке аннотаций). Например, в унаследованных сценариях:

```
# id65EMPB9VL87 - это ID узла типа
set @id65EMPB9VL87.direction."Age" = Input
```

В следующем сценарии показан пример на языке Python:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Получение и задание свойств

Унаследованные сценарии используют команду set для задания значения. Элемент после команды set может быть определением свойства. В приведенном ниже сценарии показано два возможных формата для задания свойства в сценарии:

```
set <ссылка на узел>.<свойство> = <значение>
set <ссылка на узел>.<ключевое свойство>.<ключ> = <значение>
```

В сценариях Python тот же результат достигается использованием функций `setPropertyValue()` и `setKeyedPropertyValue()`, например:

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

В унаследованных сценариях для доступа к значениям свойств можно было использовать команду `get`, например:

```
var n v
set n = get node :filternode
set v = ^n.name
```

В сценариях Python тот же результат достигается использованием функции `getPropertyValue()`, например:

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Редактирование потоков

В унаследованных сценариях для создания нового узла служит команда `create`, например:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

В сценариях Python есть ряд методов для создания узлов в потоках, например:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

В унаследованных сценариях для создания связей между узлами служит команда `connect`, например:

```
connect ^agg to ^select
```

В сценариях Python для создания связей между узлами служит метод `link`, например:

```
stream.link(agg, select)
```

В унаследованных сценариях для удаления связей между узлами служит команда `disconnect`, например:

```
disconnect ^agg from ^select
```

В сценариях Python для удаления связей между узлами служит метод `unlink`, например:

```
stream.unlink(agg, select)
```

В унаследованных сценариях для размещения узлов на холсте потока или между другими узлами служит команда `position`, например:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

В сценариях Python тот же результат достигается использованием двух различных методов - `setXYPosition` и `setPositionBetween`. Например:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Операции с узлами

Для некоторых команд операций с узлами, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 282. Отображение унаследованных сценариев на сценарии Python для операций с узлами

Унаследованный сценарий	Сценарий 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*)

Циклы

В унаследованных сценариях поддерживаются следующие две основных опции:

- *Циклы с подсчетом*, в которых переменная индекса изменяется в диапазоне между двумя целыми числами.
- *Циклы последовательности*, в которых перебирается последовательность значений, которые привязывают текущее значение к переменной цикла.

Следующий сценарий служит примером цикла с подсчетом в унаследованных сценариях:

```
for i from 1 to 10
    println ^i
endfor
```

Следующий сценарий служит примером цикла последовательности в унаследованных сценариях:

```
var items
set items = [a b c d]

for i in items
    println ^i
endfor
```

Доступны и другие типы циклов:

- Итерация по моделям на палитре моделей или по объектам вывода на палитре объектов вывода.
- Итерация по входным или выходным полям узла.

В сценариях Python тоже поддерживаются различные типы циклов. Следующий сценарий служит примером цикла с подсчетом в языке сценариях Python:

```
i = 1
while i <= 10:
    print i
    i += 1
```

Следующий сценарий служит примером цикла последовательности в сценариях Python:

```
items = ["a", "b", "c", "d"]
for i in items:
    print i
```

Цикл последовательности - весьма гибкая конструкция, а в сочетании с методами API IBM SPSS Modeler способен поддерживать большинство ситуаций в унаследованных сценариях. Приведенный ниже пример показывает, как использовать цикл последовательности в сценариях Python для итерации по выходным полям узла:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
    print column.getColumnName()
```

Выполнение потоков

Во время выполнения потока генерируемые объекты моделей и выходные объекты добавляются в один из менеджеров объектов. В унаследованных сценариях построенные объекты нужно либо найти в менеджере объектов, либо вызвать как последний сгенерированный вывод соответствующего узла.

В сценариях Python выполнение потока отличается в том отношении, что все модели и выходные объекты, сгенерированные при выполнении потока, возвращаются в виде списка, передаваемого внешней функции. Это упрощает доступ к результатам выполнения потока.

В унаследованных сценариях поддерживаются три команды выполнения потока:

- `execute_all` выполняет все выполняемые конечные узлы потока.
- `execute_script` выполняет сценарий потока независимо от того, задано ли выполнение сценариев.
- `execute` *`узел`* выполняет указанный узел.

В сценариях Python поддерживается аналогичный набор функций:

- *`поток`*`.runAll(`*`список-результатов`*`)` выполняет все выполняемые конечные узлы в потоке.
- *`поток`*`.runScript(`*`список-результатов`*`)` выполняет сценарий потока независимо от того, задано ли выполнение сценариев.
- *`поток`*`.runSelected(`*`массив-узлов, список-результатов`*`)` выполняет заданный набор узлов в том порядке, в котором они заданы.
- *`поток`*`.run(`*`список-результатов`*`)` выполняет указанный узел.

В унаследованных сценариях можно завершить выполнение потока командой `exit` с необязательным целочисленным кодом, например:

```
exit 1
```

В языке Python того же результата можно достичь таким сценарием:

```
modeler.script.exit(1)
```

Доступ к объектам через файловую систему и репозиторий

В унаследованных сценариях можно открыть существующий поток, модель или выходной объект командой open, например:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

В языке сценариев Python есть класс TaskRunner, доступный из сеанса и поддерживающий аналогичные задачи, например:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Чтобы сохранить объект в унаследованных сценариях, можно использовать команду save, например:

```
save stream s as "c:/my streams/new_modeling.str"
```

Эквивалентный подход в языке сценариев Python - использовать класс TaskRunner, например:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Операции на основе IBM SPSS Collaboration and Deployment Services Repository поддерживаются в унаследованных сценариях через команды retrieve и store, например:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

В языке сценариев Python более удобен доступ к эквивалентным возможностям через объект репозитория, связанный с сеансом:

```
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)
```

Прим.: Для доступа к репозиторию требуется, чтобы сеанс был сконфигурирован с использованием допустимого соединения с репозиторием.

Операции с потоками

Для некоторых команд операций с потоками, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 283. Отображение унаследованных сценариев на сценарии Python для операций с потоками

Унаследованный сценарий	Сценарий Python
`create stream` *DEFAULT_FILENAME*	*taskrunner*`.createStream(`*name*`,` *autoConnect*`,` *autoManage*`)`
`close stream`	*stream*`.close()`
`clear stream`	*stream*`.clear()`

Таблица 283. Отображение унаследованных сценариев на сценарии Python для операций с потоками (продолжение)	
Унаследованный сценарий	**Сценарий Python**
`get stream` *stream*	Нет эквивалента
`load stream` *path*	Нет эквивалента
`open stream` *path*	*taskrunner*`.openStreamFromFile(`*path*`,` *autoManage*`)`
`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*`)`

Операции с моделями

Для некоторых команд операций с моделями, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 284. Отображение унаследованных сценариев на сценарии Python для операций с моделями	
Унаследованный сценарий	**Сценарий 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*`)`

Операции вывода документов

Для некоторых команд операций вывода документов, широко используемых в IBM SPSS Modeler, есть эквивалентные команды в сценариях Python. Это может помочь при преобразовании ваших существующих сценариев SPSS Modeler прежних версий в сценарии Python для использования в IBM SPSS Modeler 17.

Таблица 285. Отображение унаследованных сценариев на сценарии Python для операций вывода документов	
Унаследованный сценарий	**Сценарий 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*`)`

Таблица 285. Отображение унаследованных сценариев на сценарии Python для операций вывода документов (продолжение)	
Унаследованный сценарий	**Сценарий Python**
`store output as path`	`repository.storeDocument(output, path, label)`

Другие различие между унаследованными сценариями и сценариями Python

В унаследованных сценариях есть поддержка работы с проектами IBM SPSS Modeler. Сценарии Python в настоящее время это не поддерживают.

В унаследованных сценариях до некоторой степени поддерживается загрузка *объектов состояния* (сочетание потоков и моделей). Объекты состояния устарели, начиная с версии IBM SPSS Modeler 8.0. Сценарии Python не поддерживают объекты состояния.

В сценариях Python предлагаются следующие дополнительные возможности, недоступные в унаследованных сценариях:

- Определения классов и функций
- Обработка ошибок
- Более современная поддержка ввода-вывода
- Внешние модули и модули других производителей

Уведомления

Эта информация относится к продуктам и сервису, предлагаемым в США. Этот материал может быть доступен от IBM на других языках. Однако для его получения может понадобиться приобрести продукт или версию продукта на нужном языке.

IBM может не предоставлять в других странах продукты, услуги и аппаратные средства, описанные в данном документе. За информацией о продуктах и услугах, предоставляемых в вашей стране, обращайтесь к местному представителю IBM. Ссылки на продукты, программы или услуги IBM не означают и не предполагают, что можно использовать только указанные продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любого продукта, программы или сервиса, не произведенного корпорацией IBM, лежит на пользователе.

IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Предъявление данного документа не предоставляет какую-либо лицензию на эти патенты. Вы можете послать письменный запрос о лицензии по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US ASCII

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ "КАК ЕСТЬ", БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, КАК ЯВНЫХ, ТАК И ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ТАКОВЫМИ, ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОБЛЮДЕНИЯ ЧЬИХ-ЛИБО АВТОРСКИХ ПРАВ, ВОЗМОЖНОСТИ КОММЕРЧЕСКОГО ИСПОЛЬЗОВАНИЯ ИЛИ ПРИГОДНОСТИ ДЛЯ КАКИХ-ЛИБО ЦЕЛЕЙ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ. В некоторых странах для ряда сделок не допускается отказ от явных или предполагаемых гарантий; в таком случае данное положение к вам не относится.

Эта информация может содержать технические неточности и типографские ошибки. В представленную здесь информацию периодически вносятся изменения; эти изменения будут включаться в новые издания данной публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в этой публикации на сайты, не принадлежащие IBM, приведены только для удобства и никоим образом не означают их поддержки. Материалы на этих сайтах не входят в число материалов по данному продукту IBM, и весь риск пользования этими сайтами несете вы сами.

Любую предоставленную вами информацию IBM может использовать или распространять любым способом, какой сочтет нужным, не беря на себя никаких обязательств по отношению к вам.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US ASCII

Такая информация может быть доступна при соответствующих условиях и соглашениях, включая в некоторых случаях взимание платы.

Описанную в данном документе лицензионную программу и все прилагаемые к ней лицензированные материалы IBM предоставляет на основе положений Соглашения между IBM и Заказчиком, Международного Соглашения о Лицензиях на Программы IBM или любого эквивалентного соглашения между IBM и заказчиком.

Данные производительности и примеры клиентов представлены только для иллюстрации. Фактическая производительность зависит от конкретной конфигурации и условий работы.

Информация о продуктах других компаний (не IBM) получена от поставщиков этих продуктов, из их опубликованных объявлений или из иных общедоступных источников. IBM не производила тестирование этих продуктов и никак не может подтвердить информацию о их точности работы и совместимости, а также прочие заявления относительно продуктов других компаний (не IBM). Вопросы о возможностях продуктов других компаний (не IBM) следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эти сведения содержат примеры данных и отчетов, используемых в повседневных деловых операциях. Чтобы проиллюстрировать их настолько полно, насколько это возможно, данные примеры включают имена индивидуумов, названия компаний, брендов и продуктов. Все эти имена являются вымышленными и любое их сходство с реальными именами и адресами предприятий является случайным.

Товарные знаки

IBM, логотип IBM, и ibm.com являются товарными знаками или зарегистрированными товарными знаками компании International Business Machines Corp., зарегистрированными во многих странах мира. Прочие наименования продуктов и услуг могут быть товарными знаками, принадлежащими IBM или другим компаниям. Текущий список товарных знаков IBM смотрите на веб-сайте "Copyright and trademark information" (Информация об авторских правах и товарных знаках) по адресу www.ibm.com/legal/copytrade.shtml.

Adobe, логотип Adobe, PostScript и логотип PostScript являются либо зарегистрированными товарными знаками, либо товарными знаками корпорации Adobe Systems в Соединенных Штатах и/или других странах.

Intel, логотип Intel, Intel Inside, логотип Intel Inside, Intel Centrino, логотип Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium и Pentium являются товарными знаками или зарегистрированными товарными знаками компании Intel или ее дочерних компаний в Соединенных Штатах и других странах.

Linux является зарегистрированным товарным знаком Linus Torvalds в Соединенных Штатах и других странах.

Microsoft, Windows, Windows NT и логотип Windows являются товарными знаками корпорации Microsoft в Соединенных Штатах и других странах.

UNIX является зарегистрированным товарным знаком The Open Group в Соединенных Штатах и других странах.

Java и все основанные на Java товарные знаки и логотипы - товарные знаки или зарегистрированные товарные знаки Oracle и/или его филиалов.

Правила и условия для документации продукта

Разрешения для использования этих публикаций предоставляются на следующих условиях.

Применимость

Данные правила и условия являются дополнением к правилам использования для сайта IBM.

Персональное использование

Вы можете воспроизводить эти публикации для персонального некоммерческого использования при условии сохранения всех замечаний о правах собственности. Вам запрещается распространять эти публикации, полностью или по частям, демонстрировать их или создавать из них производные продукты без явного на то согласия от IBM.

Коммерческое использование

Вам предоставляется право воспроизводить эти публикации исключительно в пределах своего предприятия при условии, что будут воспроизведены все замечания об авторских правах. За пределами вашего предприятия вам запрещается распространять эти публикации, полностью или по частям, демонстрировать их или создавать из них производные продукты без явного на то согласия от IBM.

Права

За исключением прав, явным образом предоставляемых настоящим разрешением, никаких иных разрешений, лицензий и прав, ни явных, ни подразумеваемых, в отношении публикаций и любой содержащейся в них информации, данных, программ или иной интеллектуальной собственности, не предоставляется.

IBM оставляет за собой право отозвать разрешения, предоставленные этим документом, если, по мнению IBM, использование публикаций наносит ущерб IBM или, как это установлено IBM, вышеприведенные инструкции не соблюдаются должным образом.

Запрещается загружать, экспортировать или реэкспортировать эту информацию, если при этом не будут полностью соблюдаться все применимые законы и постановления, включая все законы и постановления США, касающиеся экспорта.

IBM НЕ ДАЕТ НИКАКИХ ГАРАНТИЙ ОТНОСИТЕЛЬНО СОДЕРЖАНИЯ ЭТИХ ПУБЛИКАЦИЙ. ПУБЛИКАЦИИ ПРЕДСТАВЛЯЮТСЯ "КАК ЕСТЬ", БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ (НО НЕ ОГРАНИЧИВАЯСЬ ТАКОВЫМИ) ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ ОТСУТСТВИЯ НАРУШЕНИЙ, КОММЕРЧЕСКОЙ ПРИГОДНОСТИ ИЛИ СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ.

Индекс

A

API сценариев
- автономные сценарии 52
- введение 41
- глобальные значения 51
- доступ к сгенерированным объектам 45
- метаданные 42
- несколько потоков 52
- обработка ошибок 46
- параметры надузла 47
- параметры потока 47
- параметры сеанса 47
- поиск 41
- получение каталога 41
- пример 41

C

CLEM
- сценарий 1

D

derive_stbnode
- свойства 133

E

exportModelToFile 45

I

IBM SPSS Modeler
- выполнение из командной строки 67

J

Jython 15

N

Netezza K-Means models
- Свойства node scripting 388, 405

O

O-кластер Oracle
- Свойства node scripting 380, 387

P

Python
- сценарий 16

S

SuperNodes
- сценарии 6

А

автоматическая подготовка данных
- свойства 160

автономные сценарии 1, 4, 29
адаптивные модели Байеса Oracle
- Свойства node scripting 380, 387

априорные модели
- Свойства node scripting 232, 353

аргументы
- командный файл 73
- система 68
- соединение с репозиторием сервера IBM SPSS Analytic 72
- соединение с репозиторием IBM SPSS Collaboration and Deployment Services 72
- соединение с сервером 70

Б

Байесовские модели сети
- Свойства node scripting 244

блоки кода 20

В

Временные ряды MS
- Свойства node scripting 377

Выполнение потоков 30
выполнение сценариев 12
выходные узлы
- свойства сценариев 407

Г

геопространственный узел источника
- свойства 112

Д

двухшаговые модели
- Свойства node scripting 348, 373

Дерево решений MS
- Свойства node scripting 375, 377

диаграммы 29
дискриминантные модели
- Свойства node scripting 259, 360

добавление атрибутов 25
доступ к результатам выполнения потока
- модель содержимого JSON 62
- модель содержимого XML 60

доступ к результатам выполнения потока *(продолжение)*
модель табличного содержимого 58

З

задание свойств 32
закодированные пароли
добавление к сценариям 56
защита
закодированные пароли 56, 70

И

идентификаторы 19
изменение потоков 33, 36
имена полей
изменение регистра 53
исходные узлы
свойства 85
исходный узел IBM Cognos TM1
свойства 115
Исходный узел IBM Cognos TM1
свойства 116
Исходный узел JSON
свойства 112

К

Кластеризация последовательностей MS
Свойства node scripting 377
ключ итерации
циклы в сценариях 8
команда хранилища 53
команда clear generated palette 57
Команда for 53
Команда multiset 75
Команда retrieve 53
командная строка
выполнение IBM SPSS Modeler 67
несколько аргументов 73
параметры 70
список аргументов 68, 70, 72
сценарий 57
комментарии 19

Л

Линейная регрессия MS
Свойства node scripting 375, 377
линейные модели
Свойства node scripting 294, 365
линейные модели метода опорных векторов
свойства сценариев узлов 366
линейные-AS модели
свойства сценариев узлов 296, 366
линейные-AS свойства 296
Логистическая регрессия MS
Свойства node scripting 375, 377

М

математические методы 21
Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM.
Матрица ошибок, иногда называемая таблицей сводки, содержит число наблюдений, правильно и неправильно отнесенных к каждой из групп в анализе LSVM. 304
модели
имена сценариев 483, 485
модели автоклассификации
Свойства node scripting 355
модели автокластеризации
Свойства node scripting 357
модели автонумерации
Свойства node scripting 242, 357
модели априори Oracle
Свойства node scripting 380, 387
модели Байесовской сети
Свойства node scripting 357
модели Байесовской сети Netezza
Свойства node scripting 388, 405
модели ближайших соседей
Свойства node scripting 291
модели временных рядов
свойства сценариев узлов 334, 371
Свойства node scripting 342, 372
модели временных рядов Netezza
Свойства node scripting 388
модели выбора возможностей
Свойства node scripting 266, 362
Модели выбора функций
применение 5
сценарий 5
модели дерева регрессии Netezza
Свойства node scripting 388, 405
модели дерева решений Oracle
Свойства node scripting 380, 387
модели дерева C&R
Свойства node scripting 249, 358
модели деревьев решений Netezza
Свойства node scripting 388, 405
модели деревьев-AS
свойства сценариев узлов 345, 372
модели Коонена
Свойства node scripting 293, 365
модели линейной регрессии
Свойства node scripting 315, 369, 370
модели линейной регрессии Netezza
Свойства node scripting 388, 405
модели логистической регрессии
Свойства node scripting 298, 366
модели механизмов опорных векторов
Свойства node scripting 327, 371
модели механизмов опорных векторов Oracle
Свойства node scripting 380, 387
модели нейронной сети
Свойства node scripting 305, 367
модели обнаружения аномалий
Свойства node scripting 230, 353
Модели откликов самообучения
Свойства node scripting 319, 370
модели последовательности
Свойства node scripting 318, 370

модели потоковых временных рядов
свойства сценариев узлов 148
модели разделительной кластеризации Netezza
Свойства node scripting 388, 405
модели регрессии Кокса
Свойства node scripting 255, 359
модели случайных деревьев
свойства сценариев узлов 313, 369
модели списка решений
Свойства node scripting 257, 360
модели AI Oracle
Свойства node scripting 380
модели C5.0
Свойства node scripting 246, 358
модели CARMA
Свойства node scripting 248, 358
модели CHAID
Свойства node scripting 252, 359
модели GLE
свойства сценариев узлов 280, 364
модели GLMM
Свойства node scripting 274, 363
Модели IBM SPSS Statistics
Свойства node scripting 454
модели k-средних
Свойства node scripting 288, 365
модели K-средних Oracle
Свойства node scripting 380, 387
модели K-Means-AS
свойства node scripting 290, 475
Модели KDE
свойства node scripting 374
модели KNN
Свойства node scripting 365
модели KNN Netezza
Свойства node scripting 388, 405
модели MDL Oracle
Свойства node scripting 380, 387
модели Microsoft
Свойства node scripting 375, 377
модели Netezza
Свойства node scripting 388
модели NMF Oracle
Свойства node scripting 380, 387
модели Oracle
Свойства node scripting 380
модели PCA
Свойства node scripting 263, 362
модели PCA Netezza
Свойства node scripting 388, 405
модели PCA/факторов
Свойства node scripting 263, 362
модели Python
Свойства сценариев смешанного Гауссова узла 364
свойства сценариев узлов 368
свойства node scripting 373
Свойства node scripting 374
Модели QUEST
Свойства node scripting 310, 368
модели SLRM
Свойства node scripting 319, 370
модели SVM
Свойства node scripting 327
модели tcm
модели tcm *(продолжение)*
свойства сценариев узлов 371
модели TwoStep AS
Свойства node scripting 349, 373
моделирование базы данных 375
модель содержимого JSON 62
модель содержимого XML 60
модель табличного содержимого 58

Н

надузел 75
Надузел
поток 29
Надузлы
задание свойств в 481
параметры 481
потоки 29
свойства 481
сценарии 1, 5, 29
сценарий 481
наивные модели Байеса
Свойства node scripting 405
наивные модели Байеса Netezza
Свойства node scripting 388
наивные модели Байеса Oracle
Свойства node scripting 380, 387
направленный узел Web
свойства 225
наследование 26
нейросети
Свойства node scripting 308, 367
Нейросеть MS
Свойства node scripting 375, 377

О

обобщенные линейные модели
Свойства node scripting 268, 363
обобщенные линейные модели Netezza
Свойства node scripting 388
обобщенные линейные модели Oracle
Свойства node scripting 380
объектно-ориентированное 24
объекты вывода
имена сценариев 485
объекты моделей
имена сценариев 483
объекты модели
имена сценариев 485
операторы 19
операции 16
определение атрибутов 26
определение класса 25
определение методов 26
Особый узел
свойства 135

П

параметры
Надузлы 481
сценарий 16

Параметры slot 5, 75, 77
пароли
  добавление к сценариям 56
  закодированные 70
передача аргументов 20
переменная итерации
  циклы в сценариях 9
переменные
  сценарий 16
перемещение по узлам 36
перенастройка
  выполнение потоков 494
  доступ к объектам 495
  задание свойств 491
  имена свойств 491
  команды 487
  контекст сценариев 487
  обзор 487
  общие отличия 487
  очистка менеджеров потоков, выводов и моделей 37
  переменные 490
  получение свойств 491
  разное 497
  редактирование потоков 492
  репозиторий 495
  ссылки на узлы 491
  типы вывода 490
  типы моделей 490
  типы узлов 490
  файловая система 495
  функции 487
  циклы 493
перепроектирование системы координат
  свойства 179
поиск узлов 31
поля
  выключение в сценариях 201
порядок выполнения
  изменение через сценарии 53
порядок выполнения потока
  изменение через сценарии 53
потоки
  выполнение 30
  изменение 33
  Команда multiset 75
  свойства 79
  сценарий 1, 29
  условное выполнение 6, 10
  циклы 6, 7
прерывание сценариев 12
примеры 21
причинные модели времени
  свойства сценариев узлов 329
проверка ошибок
  сценарий 56

P

репозиторий сервера IBM SPSS Analytic
  аргументы командной строки 72
Репозиторий IBM SPSS Collaboration and Deployment Services
  аргументы командной строки 72
  сценарий 53

C

свойства
  Надузлы 481
  общие сценарии 78
  поток 79
  сценарий 75, 77, 229, 353, 431
  узлы моделирования баз данных 375
  узлы фильтрования 75
Свойства узла cognosimport 94
свойства узла gsdata_import 112
Свойства узла Space-Time-Boxes 133
свойства узла tm1import 116
Свойства узла tm1odataimport 115
свойства узла twcimport 117
Свойства aggregatenode 127
Свойства analysisnode 407
Свойства anomalydetectionnode 230
Свойства anonymizenode 159
Свойства appendnode 127
Свойства applyanomalydetectionnode 353
Свойства applyapriorinode 353
свойства applyassociationrulesnode 354
Свойства applyautoclassifiernode 355
свойства applyautoclusternode 357
Свойства applyautonumericnode 357
Свойства applybayesnetnode 357
Свойства applyc50node 358
Свойства applycarmanode 358
Свойства applycartnode 358
Свойства applychaidnode 359
Свойства applycoxregnode 359
Свойства applydecisionlistnode 360
Свойства applydiscriminantnode 360
свойства applyextension 361
Свойства applyfactornode 362
Свойства applyfeatureselectionnode 362
Свойства applygeneralizedlinearnode 363
свойства applygle 364
Свойства applyglmmnode 363
Свойства applygmm 364
Свойства applykmeansnode 365
Свойства applyknnnode 365
Свойства applykohonennode 365
свойства applylinearasnode 366
Свойства applylinearnode 365
Свойства applylogregnode 366
свойства applylsvmnode 366
Свойства applymslogisticnode 377
Свойства applymsneuralnetworknode 377
Свойства applymsregressionnode 377
Свойства applymssequenceclusternode 377
Свойства applymstimeseriesnode 377
Свойства applymstreenode 377
Свойства applynetezzabayesnode 405
Свойства applynetezzadectreenode 405
Свойства applynetezzadivclusternode 405
Свойства applynetezzakmeansnode 405
Свойства applynetezzaknnnode 405
Свойства applynetezzalineregressionnode 405
Свойства applynetezzanaivebayesnode 405
Свойства applynetezzapcanode 405
Свойства applynetezzaregtreenode 405
Свойства applyneuralnetnode 367

Свойства applyneuralnetworknode 367
свойства applyocsvm 368
Свойства applyoraabnnode 387
Свойства applyoradecisiontreenode 387
Свойства applyorakmeansnode 387
Свойства applyoranbnode 387
Свойства applyoranmfnode 387
Свойства applyoraoclusternode 387
Свойства applyorasvmnode 387
Свойства applyquestnode 368
свойства applyr 369
свойства applyrandomtrees 369
свойства applyregressionnode 370
Свойства applyselflearningnode 370
Свойства applysequencenode 370
свойства applystpnode 371
Свойства applysvmnode 371
свойства applytcmnode 371
Свойства applytimeseriesnode 372
свойства applytreeas 372
свойства applyts 371
свойства applytwostepAS 373
Свойства applytwostepnode 373
свойства applyxgboostlinearnode 374
свойства applyxgboosttreenode 373
Свойства apriorinode 232
свойства asexport 431
свойства asimport 93
свойства associationrulesnode 233
свойства astimeintervalsnode 164
Свойства autoclassifiernode 237
Свойства autoclusternode 240
Свойства autodataprepnode 160
Свойства autonumericnode 242
Свойства balancenode 129
Свойства bayesnet 244
Свойства binningnode 165
свойства buildr 245
Свойства c50node 246
Свойства carmanode 248
Свойства cartnode 249
Свойства chaidnode 252
Свойства collectionnode 202
Свойства coxregnode 255
Свойства cplexoptnode 129
Свойства dataauditnode 408
Свойства databaseexportnode 434
Свойства databasenode 98
Свойства datacollectionexportnode 440
Свойства datacollectionimportnode 100
Свойства decisionlist 257
Свойства derivenode 169
Свойства directedwebnode 225
Свойства discriminantnode 259
Свойства distinctnode 135
Свойства distributionnode 203
Свойства ensemblenode 172
свойства eplotnode 222
Свойства evaluationnode 204
Свойства excelexportnode 440, 443
Свойства excelimportnode 104
свойства extensionexportnode 441
свойства extensionimportnode 105
свойства extensionmodelnode 261
свойства extensionoutputnode 410
свойства extensionprocessnode 137
Свойства factornode 263
Свойства featureselectionnode 5, 266
Свойства fillernode 173
Свойства filternode 174
Свойства fixedfilenode 107
Свойства flatfilenode 443
Свойства genlinnode 268
свойства gle 280
Свойства glmmnode 274
Свойства gmm 457, 462
Свойства graphboardnode 206
Свойства hdbscannode 458
Свойства hdbscannugget 374
Свойства histogramnode 211
Свойства historynode 175
Свойства isotonicasnode 475
Свойства jsonimportnode 112
Свойства kdeapply 374
свойства kdeexport 412, 461
Свойства kdemodel 460
свойства kmeansasnode 290, 475
Свойства kmeansnode 288
Свойства knnnode 291
Свойства kohonennode 293
Свойства linear 294
Свойства logregnode 298
Свойства mapvisualization 212
Свойства matrixnode 413
Свойства meansnode 416
Свойства mergenode 138
Свойства msassocnode 375
Свойства msbayesnode 375
Свойства msclusternode 375
Свойства mslogisticnode 375
Свойства msneuralnetworknode 375
Свойства msregressionnode 375
Свойства mssequenceclusternode 375
Свойства mstimeseriesnode 375
Свойства mstreenode 375
Свойства multilayerperceptronnode 477
Свойства multiplotnode 217
Свойства netezzabayesnode 388
Свойства netezzadectreenode 388
Свойства netezzadivclusternode 388
Свойства netezzaglmnode 388
Свойства netezzakmeansnode 388
Свойства netezzaknnnode 388
Свойства netezzalineregressionnode 388
Свойства netezzanaivebayesnode 388
Свойства netezzapcanode 388
Свойства netezzaregtreenode 388
Свойства netezzatimeseriesnode 388
Свойства neuralnetnode 305
Свойства neuralnetworknode 308
Свойства node scripting
  слепки модели 353
  узлы моделирования 229
  узлы экспорта 431
Свойства numericpredictornode 242
свойства ocsvmnode 463
Свойства oraabnnode 380
Свойства oraainode 380

Свойства oraapriorinode 380
Свойства oradecisiontreenode 380
Свойства oraglmnode 380
Свойства orakmeansnode 380
Свойства oramdlnode 380
Свойства oranbnode 380
Свойства oranmfnode 380
Свойства oraoclusternode 380
Свойства orasvmnode 380
Свойства outputfilenode 443
Свойства partitionnode 176
Свойства plotnode 218
Свойства questnode 310
свойства randomtrees 313
Свойства reclassifynode 177
Свойства regressionnode 315
Свойства reordernode 178
Свойства reportnode 418
свойства reprojectnode 179
Свойства restructurenode 179
Свойства rfmaggregatenode 140
Свойства rfmanalysisnode 180
свойства rfnode 466
свойства routputnode 419
свойства Rprocessnode 142
Свойства samplenode 143
Свойства sasexportnode 444
Свойства sasimportnode 113
Свойства selectnode 145
Свойства sequencenode 318
Свойства setglobalsnode 420
Свойства settoflagnode 182
свойства simevalnode 420
свойства simfitnode 421
свойства simgennode 113
Свойства slrmnode 319
свойства smotenode 468
Свойства sortnode 146
Свойства spacetimeboxes 146
Свойства statisticsexportnode 456
Свойства statisticsimportnode 5, 453
Свойства statisticsmodelnode 454
Свойства statisticsnode 422
Свойства statisticsoutputnode 455
Свойства statisticstransformnode 453
свойства stpnode 321
свойства streamingtimeseries 148
Свойства streamingts 155
Свойства svmnode 327
Свойства tablenode 424
свойства tcmnode 329
Свойства timeintervalsnode 183
Свойства timeplotnode 221
Свойства timeseriesnode 342
Свойства transformnode 427
Свойства transposenode 189
свойства treeas 345
свойства ts 334
свойства tsnenode 223, 469
свойства twostepAS 349
Свойства twostepnode 348
Свойства typenode 5, 191
Свойства userinputnode 118
Свойства variablefilenode 119
Свойства webnode 225
свойства xgboostasnode 477
свойства xgboostlinearnode 471
свойства xgboosttreenode 472
Свойства xmlexportnode 451
Свойства xmlimportnode 125
свойство stream.nodes 53
сгенерированное ключевое слово 57
сгенерированные модели
  имена сценариев 483, 485
сервер
  аргументы командной строки 70
символы не из кодового набора ASCII 23
системные
  аргументы командной строки 68
скрытые переменные 26
слепки
  Свойства node scripting 353
слепки модели
  имена сценариев 483, 485
  Свойства node scripting 353
слепок узла Правила связывания
  свойства 354
слепок узла STP
  свойства 371
Смешанный гауссов узел
  свойства 457, 462
создание класса 25
создание узлов 33–35
списки 16
ссылки на узлы
  задание свойств 32
  поиск узлов 31
строки
  изменение регистра 53
структурированные свойства 75
сценарии
  в надузлах 5
  выбор полей 10
  импорт из текстовых файлов 1
  ключ итерации 8
  переменная итерации 9
  пользовательский интерфейс 4, 5
  сохранение 1
  условное выполнение 6, 10
  циклы 6, 7
сценарий
  автономные сценарии 1, 29
  выбор полей 10
  выполнение 12
  выходные узлы 407
  диаграммы 29
  из командной строки 57
  используемые сокращения 76
  ключ итерации 8
  контекст 30
  Модели выбора функций 5
  наглядное изображение циклов 6, 7
  обзор 1, 15
  общие свойства 78
  переменная итерации 9
  пользовательский интерфейс 1
  порядок выполнения потока 53
  потоки 1, 29

сценарий *(продолжение)*
потоки надузлов 29
прерывание 12
проверка ошибок 56
синтаксис 16, 17, 19–21, 23–26
совместимость с более старыми версиями 57
сценарии надузла 1, 29
Сценарий Python 488, 489, 493, 495, 496
узлы диаграмм 201
унаследованные сценарии 488, 489, 493, 495, 496
условное выполнение 6, 10

У

узел автоклассификации
Свойства node scripting 237
узел автокластеризации
Свойства node scripting 240
Узел агрегации
свойства 127
узел агрегации RFM
свойства 140
Узел Анализ
свойства 407
узел анализа RFM
свойства 180
узел анонимизации
свойства 159
узел ансамбля
свойства 172
Узел аудита данных
свойства 408
узел базы данных
свойства 98
узел балансировки
свойства 129
Узел визуализации карт
свойства 212
узел выбора
свойства 145
Узел выборки
свойства 143
Узел вывода IBM SPSS Statistics
свойства 455
узел вывода R
свойства 419
узел генерирования имитации
свойства 113
узел Гистограмма
свойства 211
узел График
свойства 218
узел График зависимости от времени
свойства 221
Узел дерева XGBoost
свойства 472
узел добавления
свойства 127
узел Задать глобальные значения
свойства 420
узел Задать как флаг
свойства 182
Узел заполнения
свойства 173
узел извлечения
свойства 169
Узел Имитация KD
свойства 412, 461
узел интервалов времени
свойства 183
узел интервалов времени AS
свойства 164
узел источника Analytic Server
свойства 93
узел источника Data Collection
свойства 100
узел источника Excel
свойства 104
узел источника IBM Cognos
свойства 94
узел источника IBM SPSS Statistics
свойства 453
узел источника SAS
свойства 113
узел источника TWC Import
свойства 117
узел источника XML
свойства 125
узел Линейный XGBoost
свойства 471
узел Матрица
свойства 413
Узел моделирования KDE
свойства 460
узел Несколько графиков
свойства 217
Узел Оптимизация CPLEX
свойства 129
узел отчета
свойства 418
узел Оценка
свойства 204
узел оценки имитации
свойства 420
узел Панель выбора диаграмм
свойства 206
Узел переклассификации
свойства 177
узел перепроектирования
свойства 179
узел переупорядочения
свойства 178
узел переупорядочения полей
свойства 178
узел плоского файла
свойства 443
узел подгонки имитации
свойства 421
узел пользовательского ввода
свойства 118
Узел потоковых временных рядов
свойства 155
узел Правила связывания
свойства 233
узел преобразования
свойства 427
узел преобразования IBM SPSS Statistics
свойства 453

узел преобразования R
свойства 142
узел пространственно-временного предсказания
свойства 321
узел раздела
свойства 176
Узел разделения на интервалы
свойства 165
узел Распределение
свойства 203
узел реструктуризации
свойства 179
узел сборки R
Свойства node scripting 245
узел слияния
свойства 138
узел Случайный лес
свойства 466
узел Собрание
свойства 202
Узел сортировки
свойства 146
узел средних
свойства 416
узел статистики
свойства 422
узел таблицы
свойства 424
узел типа
свойства 191
узел транспонирования
свойства 189
узел файла переменных
свойства 119
узел фиксированного файла
свойства 107
Узел фильтра
свойства 174
узел Хронология
свойства 175
узел экспорта базы данных
свойства 434
узел экспорта Data Collection
свойства 440
узел экспорта Excel
свойства 440, 443
узел экспорта IBM SPSS Statistics
свойства 456
узел экспорта SAS
свойства 444
узел экспорта XML
свойства 451
узел Электронный график
свойства 222
узел Extension Export
свойства 441
узел Extension Import
свойства 105
узел Extension Model
свойства сценариев узлов 261
узел Extension Output
свойства 410
узел Extension Transform
свойства 137
Узел HDBSCAN
свойства 458
узел Isotonic-AS
свойства 475
Узел MultiLayerPerceptron-AS
свойства 477
узел One-Class SVM
свойства 463
узел simeval (оценка имитации)
свойства 420
узел SimFit (подгонка имитации)
свойства 421
узел SimGen
свойства 113
узел SMOTE
свойства 468
узел STP
свойства 321
узел t-SNE
свойства 223, 469
узел Web
свойства 225
узел XGBoost-AS
свойства 477
узлы
замена 35
импорт 35
информация 38
отсоединение узлов 34
соединение узлов 34
ссылки на имена 483
удаление 35
цикл по сценариям 53
узлы диаграмм
свойства сценариев 201
узлы моделирования
Свойства node scripting 229
узлы пространственно-временных интервалов
свойства 133, 146
узлы экспорта
Свойства node scripting 431
условное выполнение потоков 6, 10

Ф

флаги
аргументы командной строки 67
объединение нескольких флагов 73
функции
комментарии 488
литералы 488
операторы 488
операции вывода документов 496
операции с моделями 496
операции с потоками 495
операции с узлами 493
ссылки на объекты 488
условное выполнение 489
циклы 489
функции для работы с текстовыми значениями 53
Функция lowertoupper 53

Ц

циклы
    использование в сценариях 53
циклы в потоках 6, 7