

*IBM SPSS Modeler 19.0 Python Scripting  
and Automation Guide*



## 注

在使用本资料及其支持的产品之前，请阅读第 415 页的『声明』中的信息。

## 产品信息

本版本适用于 IBM® SPSSModeler 的版本 19、第 0 版、第 0 次修改，以及所有后续版本和修改，除非新版本另有说明。

© **Copyright International Business Machines Corporation .**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Chapter 1. 脚本编写和脚本编写语言.....</b>	<b>1</b>
脚本编写概述.....	1
脚本类型.....	1
流脚本.....	1
流脚本示例：训练神经网络.....	2
Jython 代码大小限制.....	3
独立脚本.....	3
独立脚本示例：保存和加载模型.....	3
独立脚本示例：生成特征选择模型.....	4
超节点脚本.....	4
超节点脚本示例.....	5
流中的循环和条件执行.....	5
流中的循环.....	6
流中的条件执行.....	8
执行和中断脚本.....	9
查找和替换.....	9
 <b>Chapter 2. 脚本语言.....</b>	 <b>13</b>
脚本编写语言概述.....	13
Python 和 Jython.....	13
Python 脚本编制.....	13
操作.....	13
列表.....	14
字符串.....	15
备注.....	16
语句语法.....	16
标识.....	17
代码块.....	17
将参数传递给脚本.....	17
示例.....	18
数学方法.....	18
使用非 ASCII 字符.....	20
面向对象的程序设计.....	21
定义类.....	21
创建类实例.....	21
向类实例添加属性.....	21
定义类属性和方法.....	22
隐藏变量.....	22
继承.....	22
 <b>Chapter 3. 在 IBM SPSS Modeler 中进行脚本编制.....</b>	 <b>25</b>
脚本类型.....	25
流、SuperNode 流和图表.....	25
流.....	25
SuperNode 流.....	25
图表.....	25
执行流.....	25
脚本编制上下文.....	25
引用现有节点.....	26
查找节点.....	26

设置属性.....	27
创建节点以及修改流.....	28
创建节点.....	28
链接和取消链接节点.....	29
导入、替换和删除节点.....	30
遍历流中的节点.....	30
清除或移除项.....	31
获取节点的相关信息.....	31
<b>Chapter 4. 脚本编制 API.....</b>	<b>33</b>
脚本编制 API 简介.....	33
示例 1：使用定制过滤器搜索节点.....	33
示例 2：允许用户基于其权限获取目录或文件信息.....	33
元数据：有关数据的信息.....	34
访问已生成的对象.....	36
处理错误.....	37
流、会话和超节点参数.....	38
全局值.....	41
使用多个流 - 独立脚本.....	41
<b>Chapter 5. 脚本编写技巧.....</b>	<b>43</b>
修改流执行.....	43
对节点执行循环.....	43
访问 IBM SPSS 协作与部署服务资源库 中的对象.....	43
生成经过编码的密码.....	45
脚本检查.....	46
从命令行执行脚本编写.....	46
与先前版本的兼容性.....	46
访问流执行结果.....	46
表内容模型.....	47
XML 内容模型.....	48
JSON 内容模型.....	50
列统计内容模型和成对统计内容模型.....	51
<b>Chapter 6. 命令行自变量.....</b>	<b>55</b>
调用软件.....	55
使用命令行参数.....	55
系统自变量.....	56
参数自变量.....	57
服务器连接参数.....	58
IBM SPSS 协作与部署服务资源库 连接参数.....	59
IBM SPSS 分析服务器 连接参数.....	59
组合多个参数.....	60
<b>Chapter 7. 属性参考信息.....</b>	<b>61</b>
属性参考概述.....	61
属性语法.....	61
节点和流属性示例.....	62
节点属性概述.....	63
通用节点属性.....	63
<b>Chapter 8. 流属性.....</b>	<b>65</b>
<b>Chapter 9. 源节点属性.....</b>	<b>69</b>
源节点公共属性.....	69
asimport 属性.....	76

cognosimport 节点属性.....	77
databasenode 属性.....	79
datacollectionimportnode 属性.....	81
excelimportnode 属性.....	84
extensionimportnode 属性.....	86
fixedfilenode 属性.....	88
gsdata_import 节点属性.....	92
jsonimportnode 属性.....	92
sasimportnode 属性.....	92
simgennode 属性.....	93
statisticsimportnode 属性.....	95
tm1odataimport 节点属性.....	95
tm1import 节点属性（不推荐） .....	96
twcimport 节点属性.....	97
userinputnode 属性.....	98
variablefilenode 属性.....	99
xmlimportnode 属性.....	104
<b>Chapter 10. 记录操作节点属性.....</b>	<b>105</b>
appendnode 属性.....	105
aggregatenode 属性.....	105
balancenode 属性.....	106
cplexoptnode 属性.....	107
derive_stbnode 属性.....	109
distinctnode 属性.....	111
extensionprocessnode 属性.....	113
mergenode 属性.....	114
rfmaggregatenode 属性.....	116
samplenode 属性.....	118
selectnode 属性.....	120
sortnode 属性.....	120
spacetimeboxes 属性.....	121
streamingtimeseries 属性.....	122
<b>Chapter 11. 字段操作节点属性.....</b>	<b>131</b>
anonymizenode 属性.....	131
autodataprepnode 属性.....	132
astimeintervalsnode 属性.....	136
binningnode 属性.....	136
derivenode 属性.....	139
ensemblenode 属性.....	142
fillernode 属性.....	143
filternode 属性.....	144
historynode 属性.....	145
partitionnode 属性.....	146
reclassifynode 属性.....	147
reordernode 属性.....	147
reprojectnode 属性.....	148
restructurenode 属性.....	149
rfanalysisnode 属性.....	149
settoflagnode 属性.....	150
statisticstransformnode 属性.....	151
timeintervalsnode 属性（不推荐） .....	151
transposenode 属性.....	157
typenode 属性.....	159
<b>Chapter 12. 图形节点属性.....</b>	<b>167</b>

图形节点公共属性.....	167
collectionnode 属性.....	168
distributionnode 属性.....	169
evaluationnode 属性.....	170
graphboardnode 属性.....	172
histogramnode 属性.....	176
mapvisualization 属性.....	177
multiplotnode 属性.....	180
plotnode 属性.....	181
timeplotnode 属性.....	183
eplotnode 属性.....	185
tsnode 属性.....	185
webnode 属性.....	187

## **Chapter 13. 建模节点属性..... 189**

公共建模节点属性.....	189
anomalydetectionnode 属性.....	189
apriorinode 属性.....	191
associationrulesnode 属性.....	192
autoclassifiernode 属性.....	195
设置算法属性.....	197
autoclusternode 属性.....	197
autonumericnode 属性.....	199
bayesnetnode 属性.....	201
c50node 属性.....	203
carmanode 属性.....	204
cartnode 属性.....	205
chaidnode 属性.....	208
coxregnode 属性.....	210
decisionlistnode 属性.....	213
discriminantnode 属性.....	214
extensionmodelnode 属性.....	216
factornode 属性.....	218
featureselectionnode 属性.....	220
genlinnode 属性.....	222
glmmnode 属性.....	227
gle 属性.....	231
kmeansnode 属性.....	238
kmeansasnode 属性.....	239
knnnode 属性.....	240
kohonenode 属性.....	241
linearnode 属性.....	243
linearasnode 属性.....	244
logregnode 属性.....	246
lsvmnode 属性.....	251
neuralnetnode 属性.....	252
neuralnetworknode 属性.....	255
questnode 属性.....	257
randomtrees 属性.....	259
regressionnode 属性.....	261
sequencenode 属性.....	263
slrmnode 属性.....	264
statisticsmodelnode 属性.....	265
stpnode 属性.....	266
svminode 属性.....	271
tcminode 属性.....	273
ts 属性.....	278

treeas 属性.....	288
twostepnode 属性.....	290
twostepAS 属性.....	291
<b>Chapter 14. 模型块节点属性.....</b>	<b>295</b>
applyanomalydetectionnode 属性.....	295
applyapriorinode 属性.....	295
applyassociationrulesnode 属性.....	296
applyautoclassifiernode 属性.....	296
applyautoclusternode 属性.....	298
applyautonumericnode 属性.....	298
applybayesnetnode 属性.....	298
applyc50node 属性.....	299
applycarmanode 属性.....	299
applycartnode 属性.....	299
applychaidnode 属性.....	299
applycoxregnode 属性.....	300
applydecisionlistnode 属性.....	300
applydiscriminantnode 属性.....	301
applyextension 属性.....	301
applyfactornode 属性.....	302
applyfeatureselectionnode 属性.....	303
applygeneralizedlinearnode 属性.....	303
applyglmnode 属性.....	303
applygle 属性.....	304
applygmm 属性.....	304
applykmeansnode 属性.....	304
applyknnnode 属性.....	304
applykohonennode 属性.....	305
applylinearnode 属性.....	305
applylinearasnode 属性.....	305
applylogregnode 属性.....	305
applysvmnode 属性.....	306
applyneuralnetnode 属性.....	306
applyneuralnetworknode 属性.....	306
applyocsvmnode 属性.....	307
applyquestnode 属性.....	307
applyrandomtrees 属性.....	308
asapplyregressionnode 属性.....	308
applyselflearningnode 属性.....	308
applysequencenode 属性.....	309
appliesvmnode 属性.....	309
applystpnode 属性.....	309
applytcmnode 属性.....	309
applyts 属性.....	310
applytimeseriesnode 属性（不推荐） .....	310
applytreeas 属性.....	310
applytwostepnode 属性.....	311
applytwostepAS 属性.....	311
applyxgboosttreenode 属性.....	311
applyxgboostlinearnode 属性.....	311
hdbscannugget 属性.....	312
kdeapply 属性.....	312
<b>Chapter 15. 数据库建模节点属性.....</b>	<b>313</b>
Microsoft 建模的节点属性.....	313
Microsoft 建模节点属性.....	313

Microsoft 模型块属性.....	315
Oracle 建模的节点属性.....	317
Oracle 建模节点属性.....	317
Oracle 模型块属性.....	324
IBM Netezza 分析 建模节点属性.....	325
Netezza 建模节点属性.....	325
Netezza 模型块属性.....	338
<b>Chapter 16. 输出节点属性.....</b>	<b>339</b>
analysisnode 属性.....	339
dataauditnode 属性.....	340
extensionoutputnode 属性.....	342
kdeexport 属性.....	343
matrixnode 属性.....	344
meansnode 属性.....	346
reportnode 属性.....	348
setglobalsnode 属性.....	350
simevalnode 属性.....	350
simfitnode 属性.....	351
statisticsnode 属性.....	352
statisticsoutputnode 属性.....	353
tablenode 属性.....	354
transformnode 属性.....	356
<b>Chapter 17. 导出节点属性.....</b>	<b>359</b>
通用导出节点属性.....	359
asexport 属性.....	359
cognosexportnode 属性.....	360
databaseexportnode 属性.....	362
datacollectionexportnode 属性.....	366
excelexportnode 属性.....	367
extensionexportnode 属性.....	368
jsonexportnode 属性.....	369
outputfilenode 属性.....	369
sasexportnode 属性.....	370
statisticsexportnode 属性.....	371
tm1odataexport 节点属性.....	371
tm1export 节点属性（不推荐）.....	373
xmlexportnode 属性.....	375
<b>Chapter 18. IBM SPSS Statistics 节点属性.....</b>	<b>377</b>
statisticsimportnode 属性.....	377
statisticstransformnode 属性.....	377
statisticsmodelnode 属性.....	378
statisticsoutputnode 属性.....	378
statisticsexportnode 属性.....	379
<b>Chapter 19. Python 节点属性.....</b>	<b>381</b>
gmm 属性.....	381
hdbscannode 属性.....	381
kdemodel 属性.....	383
kdeexport 属性.....	384
gmm 属性.....	385
ocsvmnode 属性.....	385
rfnode 属性.....	387
smotenode 属性.....	388
tsnnode 属性.....	389



xgboostlinearnode 属性.....	391
xgboosttreenode 属性.....	392
<b>Chapter 20. Spark 节点属性.....</b>	<b>395</b>
isotonicasnode 属性.....	395
kmeansasnode 属性.....	395
multilayerperceptronnode 属性.....	396
xgboostasnode 属性.....	396
<b>Chapter 21. 超节点属性.....</b>	<b>399</b>
<b>Appendix A. 节点名引用.....</b>	<b>401</b>
模型块名称.....	401
避免重复的模型名称.....	403
输出类型名称.....	403
<b>Appendix B. 从旧脚本编制迁移到 Python 脚本编制.....</b>	<b>405</b>
旧脚本迁移概述.....	405
一般差异.....	405
脚本编制上下文.....	405
命令与函数.....	405
文字和注释.....	405
运算符.....	406
条件语句和循环.....	407
变量.....	408
节点、输出和模型类型.....	408
属性名.....	408
节点引用.....	409
获取并设置属性.....	409
编辑流.....	410
节点操作.....	410
循环.....	411
执行流.....	412
通过文件系统和存储库访问对象.....	412
流操作.....	413
模型操作.....	413
文档输出操作.....	414
旧脚本编制与 Python 脚本编制之间的其他差异.....	414
<b>声明.....</b>	<b>415</b>
商标.....	416
产品文档的条款和条件.....	416
<b>Index.....</b>	<b>417</b>



---

# 第 1 章 脚本编写和脚本编写语言

## 脚本编写概述

---

IBM SPSSModeler 中的脚本编写是用于在用户界面上实现过程自动化的强大工具。您使用鼠标或键盘进行的操作,借助脚本同样可以完成,而且使用脚本可以自动化那些手动执行将造成大量重复操作且高耗时的任务。

脚本的作用包括：

- 限制在流中执行节点的特定顺序。
- 设置节点属性并使用 CLEM（表达式操作控制语言）的子集来执行派生。
- 指定通常包含用户交互的操作的自动执行顺序,例如您可以构建一个模型,然后对其进行测试。
- 设置需要实际用户交互的复杂过程,例如需要重复模型生成和测试的交叉验证步骤。
- 设置流操纵过程 - 例如,您可以提取一个模型训练流,运行它,然后自动生成相应的模型测试流。

本章提供流级脚本、独立脚本以及 IBM SPSSModeler 用户界面超节点内脚本的高级说明和示例。有关脚本编写语言、语法和命令的更多信息,请参阅 章后的章节。

注:

您无法导入和运行在 IBM SPSSModeler 中的 IBM SPSS Statistics 中创建的脚本。

## 脚本类型

---

IBM SPSSModeler 使用三种类型的脚本：

- **流脚本**存储为流属性然后和指定流一起保存和加载。例如,可以编写自动化训练和应用模型块流程的流脚本。您还可以指定何时执行特定流,脚本应代替流工作区内容运行。
- **独立脚本**不与保存在外部文本文件中的所有特定流关联。例如,可以使用独立脚本同时操作多个流。
- **超节点脚本**存储为超节点流属性。超节点只在终端超节点中可用。您可以使用超节点脚本控制超节点内容的执行序列。对于非终端（源或过程）超节点,可以为超节点定义属性或定义这种超节点直接在流脚本中包含的节点。

## 流脚本

---

脚本可用于定制特定流中的操作并与该流一起保存。流脚本可用于指定某个流中终端节点的特定执行顺序。可以使用“流脚本”对话框来编辑与当前流一起保存的脚本。

从流属性对话框访问流“脚本”选项卡：

1. 从工具菜单中,选择：

**流属性 > 执行**

2. 单击**执行**选项卡以处理当前流的脚本。

使用流脚本对话框顶部的工具栏图标来执行以下操作：

- 将先前存在的独立脚本内容导入到窗口。
- 将脚本保存为文本文件。
- 打印脚本。
- 追加缺省脚本。
- 编辑脚本（撤销、剪切、复制、粘贴及其他常见的编辑功能）。
- 执行整个当前脚本。
- 执行某个脚本中的选定行。

- 在执行期间停止脚本。（只有在脚本处于运行状态的情况下，才会启用此图标。）
- 检查脚本的语法，如果发现任何错误，就将其显示在对话框的下部窗格中复查。

注：从 V16.0 开始，SPSS 建模器使用 Python 脚本语言。16.0 之前的所有版本都使用 SPSS 建模器所特有的脚本语言，现称为遗存脚本。根据您所使用的脚本类型不同，在**执行**选项卡上，选择**缺省（可选脚本）**执行方式，然后选择 **Python** 或**遗存**。

您可以指定执行流时是否运行脚本。要在每次执行流时都运行脚本，并且按照脚本的执行顺序运行，请选择**运行该脚本**。此设置为快速构建模型提供流一级的自动化。但是，缺省设置为在执行流的过程中忽略此脚本。即使选择选项 **忽略此脚本**，也可以直接从此对话框运行脚本。

脚本编辑器提供了下列功能，这些功能有助于脚本编写：

- 语法突出显示；将突出显示关键字、文字值（例如字符串和数字）以及注释。
- 行编号。
- 块匹配；当光标处于程序块的开始位置时，还将突出显示相应的结束块。
- 建议的自动补全。

可以使用 IBM SPSSModeler 显示首选项来定制语法突出显示器使用的颜色和文本样式。要访问显示首选项，请选择**工具 > 选项 > 用户选项**并选择**语法**选项卡。

通过从上下文菜单中选择**自动建议**或者按 **Ctrl + Space**，可以访问建议语法补全的列表。使用光标键在列表中上下移动，然后按 **Enter** 键可插入所选文本。要退出自动建议方式而不修改现有文本，请按 **Esc** 键。

**调试**选项卡显示调试消息，并且可以用于在执行脚本后立即对脚本状态进行评估。**调试**选项卡包含一个只读文本区域和单行输入文本字段。文本区域显示由脚本发送到标准输出或标准错误（例如，通过错误消息文本）的文本。输入文本字段将接收来自用户的输入。然后，将在对话框内最近执行的脚本上下文（称为脚本编制上下文）中对此输入进行评估。文本区域包含命令和生成的输出，以便用户能够查看命令跟踪。文本输入字段始终包含命令提示符（对于遗存脚本编制而言为 **-->**）。

在下列情况下，将创建新的脚本编制上下文：

- 使用**运行此脚本**或**运行所选行**来执行脚本。
- 脚本语言会发生更改。

如果创建了新的脚本编制上下文，那么将清除文本区域。

注：在脚本窗格外执行流不会修改脚本窗格的脚本上下文。在该执行过程中创建的任何变量的值在脚本对话框中都不可见。

## 流脚本示例：训练神经网络

当执行时，流可用于训练神经网络模型。通常，要检验模型，可以运行建模节点以便将该模型添加到流中，建立适当的连接，然后执行“分析”节点。

借助 IBM SPSSModeler 脚本，您可以在创建模型块之后，实现模型块测试过程的自动化。例如，可以从“流属性”对话框（**工具 > 流属性 > 脚本**）运行用于测试演示流 **druglearn.str** 的以下流脚本（位于 IBM SPSSModeler 安装的 **/Demos/streams/** 文件夹下）：

```
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([])
```

以下带着重号的句子说明此脚本示例中的每一行。

- 第一行定义指向当前流的变量。
- 在第二行中，脚本将查找“神经网络”构建器节点。

- 在第三行中，脚本将创建可以在其中存储执行结果的列表。
- 在第四行中，将创建“神经网络”模型块。此模型块存储在第三行中定义的列表内。
- 在第五行中，将为此模型块创建模型应用节点并将此节点放入流画布中。
- 在第六行中，将创建称为 **Drug** 的分析节点。
- 在第七行中，脚本将查找类型节点。
- 在第八行中，脚本将连接第五行中在类型节点与分析节点之间创建的模型应用节点。
- 最后，执行分析节点以生成分析报告。

可以使用脚本从头开始（从空画布开始）构建并运行流。

## Jython 代码大小限制

Jython 将每个脚本编译为 Java 字节码，然后由 Java 虚拟机 (JVM) 执行。但是，Java 对单个字节码文件的大小施加了限制。因此，当 Jython 尝试加载字节码时，可能会导致 JVM 崩溃。IBM SPSSModeler 无法阻止这种情况发生。

请确保您使用适当的编码做法（例如，使用变量或函数计算公共中间值，从而尽量减少重复代码）编写 Jython 脚本。如有需要，可能需要将代码拆分成若干个源文件，或者使用模块进行定义，这些模块继而编译为单独的字节码文件。

## 独立脚本

“独立脚本”对话框用于创建或编辑保存为文本文件的脚本。它显示了文件名称，提供了用于加载、保存、导入和执行脚本的实用程序。

要访问“独立脚本”对话框，请执行以下操作：

在主菜单中，选择：

**工具 > 独立脚本**

对流脚本可用的工具栏和脚本语法检查选项对独立脚本同样适用。有关更多信息，请参阅主题 [第 1 页的『流脚本』](#)。

## 独立脚本示例：保存和加载模型

独立脚本可用于流操纵。假设有两个流，第一个流创建模型并生成规则集，第二个流则通过现有数据字段，采用图示的方式对规则集进行探索。该方案的独立脚本可能具有如下形式：

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

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

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

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

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

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivencode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
modelapplier.setPositionBetween(derivencode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivencode, plotnode)
```

```
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

注: 要了解有关常规脚本编写语言的更多信息, 请参阅第 13 页的『脚本编写语言概述』。

## 独立脚本示例：生成特征选择模型

首先打开一个空工作区, 在此示例中将构建一个流, 该流生成一个特征选择模型, 应用此模型并创建一个表, 该表包含有对于指定目标而言重要性最高的 15 个字段。

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

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

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

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

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

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

该脚本会创建要读取数据的源节点, 使用类型节点将 **response\_01** 字段的角色 (方向) 设置为 **Target**, 然后创建并执行“特征选择”节点。此脚本还连接流画布上的各个节点和位置以生成可读的布局。然后结果模型块与表节点相连接, “表”节点列出了属性 **selection\_mode** 和 **top\_n** 所确定的 15 个最重要的字段。请参阅主题第 220 页的『featureselectionnode 属性』, 了解更多信息。

## 超节点脚本

通过使用 IBM SPSS Modeler 的脚本语言, 可以创建和保存所有终端超节点中的脚本。这些脚本只在终端超节点中可用, 并且常在创建模板流或用于强制超节点内容以特定顺序执行时使用。使用超节点脚本, 您也可以在流中运行多个脚本。

例如, 假设需要指定一个复杂流的执行顺序, 并且超节点包含若干个包括设置全局量节点的节点, 而执行设置全局量节点又需要在派生用于散点图节点的新字段之前进行。这种情况下, 可以创建一个首先执行设置全局量节点的超节点脚本。接着, 可在执行“散点图”节点时使用此节点所计算的值 (例如, 平均值或标准偏差)。

在超节点脚本中也可以指定节点属性, 操作方法与在其他脚本中的进行的操作一样。另外, 为所有超节点或直接来自流脚本的超节点的封装节点更改和定义属性。有关更多信息, 请参阅主题第 399 页的『第 21 章超节点属性』。此方法适用于源和过程超节点以及终端超节点。

注: 因为只有终端超节点能够执行自身脚本, 所以“超节点”对话框的“脚本”选项卡只在用于终端超节点时可用。

## 从主工作区打开“超节点脚本”对话框：

从流工作区选择终端超节点，然后从“超节点”菜单选择：

超节点脚本...

## 从放大超节点工作区打开“超节点脚本”对话框：

右键单击超节点工作区，然后从上下文菜单中选择：

超节点脚本...

## 超节点脚本示例

以下超节点脚本声明超节点中终端节点的执行顺序。此顺序可确保首先执行设置全局量节点，以便随后执行其他节点时可使用由此节点计算的值。

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

## 锁定和解锁超节点

以下示例说明了如何锁定和解锁超节点：

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

## 流中的循环和条件执行

从 V16.0 开始，通过 SPSS 建模器，您可以选择各个对话框中的值在流中创建一些基本脚本，而无需使用脚本编制语言直接编写指令。可通过此方式创建的两种主要类型的脚本是简单循环以及在满足条件时执行节点的方式。

可以组合流中的循环规则和条件执行规则。例如，您可能具有来自世界各地制造商的汽车销售相关数据。您可以在流中设置一个用于处理数据的循环，从而按制造国家或地区标识详细信息，并将数据输出到各个显示了详细信息（例如，按型号排列的销售量，按制造商和引擎大小排列的排放级别等）的图形。如果您希望仅分析欧洲信息，那么还可以向循环添加条件，以阻止针对总部设在美国和亚洲的制造商创建图形。

**注：**由于循环和条件执行均以后台脚本为基础，因此它们仅适用于运行的整个流。

- **循环** 使用循环可自动化重复任务。例如，这可能意味着向流添加给定数目的节点，并且每次更改一个节点参数。另外，您可以将流或分支的运行控制为反复运行给定数目的次数，如以下示例所示：
  - 运行流给定数目的次数，并且每次都对源进行更改。
  - 运行流给定数目的次数，并且每次都对变量的值进行更改。
  - 运行流给定数目的次数，并且在每次执行时都输入一个额外的字段。
  - 构建模型给定数目的次数，并且每次都对模型设置进行更改。
- **条件执行** 您可以使用它根据预定义的条件来控制终端节点的运行方式，可能的示例如下：
  - 根据给定值是 **true** 还是 **false**，控制是否将运行节点。
  - 定义节点循环将以并行方式运行还是按顺序运行。



循环和条件执行都是在“流属性”对话框中的“执行”选项卡中设置的。任何在条件或循环要求中使用的节点都随附加到这些节点的附加符号一起显示在流画布上，此符号用于指示这些节点将参与循环和条件执行。

您可以通过下列三种方式中的其中一种来访问“执行”选项卡：

- 使用主对话框顶部的菜单：
  1. 从“工具”菜单中，选择：  
**流属性 > 执行**
  2. 单击“执行”选项卡以处理当前流的脚本。
- 从流中：
  1. 右键单击节点，然后选择**循环/条件执行**。
  2. 选择相关子菜单选项。
- 从主对话框顶部的图形工具栏中，单击流属性图标。

如果这是您第一次设置循环或条件执行详细信息，请在“执行”选项卡上选择**循环/条件执行**执行方式，然后选择**条件**或**循环**子选项卡。

## 流中的循环

通过循环，您可以自动化流中的重复任务；可能的示例如下：

- 运行流给定数目的次数，并且每次都对源进行更改。
- 运行流给定数目的次数，并且每次都对变量的值进行更改。
- 运行流给定数目的次数，并且在每次执行时都输入一个额外的字段。
- 构建模型给定数目的次数，并且每次都对模型设置进行更改。

可以在流“执行”选项卡的**循环**子选项卡上设置要满足的条件。要显示该子选项卡，请选择**循环/条件执行**执行方式。

如果设置了**循环/条件执行**执行方式，那么在您运行流时，您定义的所有循环要求都将生效。（可选）您可以根据循环需求生成脚本代码，并通过单击“循环”子选项卡右下角的**粘贴...**，将其粘贴到脚本编辑器中；主“执行”选项卡显示更改为显示**缺省值（可选脚本）**执行方式，并且脚本显示在选项卡的顶部。这意味着，您可以先使用多个循环对话框选项来定义循环结构，然后再生成可在脚本编辑器中进行进一步定制脚本。请注意，当您单击**粘贴...**时，您定义的任何条件执行需求也将会显示在生成的脚本中。

**要点:** 如果您在 IBM SPSS 协作和部署服务 作业中运行某个 SPSS 建模器 流，那么可以覆盖您在此流中设置的循环变量。这是因为，IBM SPSS 协作和部署服务 作业编辑器条目将覆盖 SPSS 建模器 条目。例如，如果您在流中设置了某个循环变量以便为每个循环创建不同的输出文件名称，那么这些文件将在 SPSS 建模器中正确命名，但由 IBM SPSS 协作和部署服务 Deployment Manager 的“结果”选项卡中输入的固定条目覆盖。

### 要设置循环，请完成下列步骤：

1. 创建迭代关键字以定义将在流中执行的主要循环结构。有关更多信息，请参阅[创建迭代关键字](#)。
2. 在需要时，定义一个或多个迭代变量。有关更多信息，请参阅[创建迭代变量](#)。
3. 您创建的迭代和所有变量都将显示在该子选项卡的主要部分中。缺省情况下，将按显示顺序执行迭代；要在列表中上下移动迭代，请单击迭代以将其选中，然后使用该子选项卡右侧的向上或向下箭头更改顺序。

## 创建用于流中的循环的迭代关键字

使用迭代关键字可以定义将在流中执行的主要循环结构。例如，如果要对汽车销售进行分析，那么可以创建流参数制造国家或地区，并将其用作迭代关键字；在运行流时，此关键字将设置为各个迭代过程中您的数据中的各个不同的国家或地区值。使用“定义迭代关键字”对话框可以设置关键字。



要打开此对话框，请选择“循环”子选项卡左下角的**迭代键...**按钮，或右键单击流中的任何节点，然后选择**循环/条件执行 > 定义迭代键（字段）**或**循环/条件执行 > 定义迭代键（值）**。如果是从流中打开此对话框，那么系统可能会自动为您填写一些字段，例如，节点的名称。

要设置迭代关键字，请填写下列字段：

**迭代依据。**您可以选择下列其中一个选项：

- **流参数 - 字段。**使用此选项可创建一个循环，用于将现有流参数的值依次设置为各个指定字段。
- **流参数 - 值。**使用此选项可创建一个循环，用于将现有流参数的值依次设置为各个指定值。
- **节点属性 - 字段。**使用此选项可创建一个循环，用于将节点属性的值依次设置为各个指定字段。
- **节点属性 - 值。**使用此选项可创建一个循环，用于将节点属性的值依次设置为各个指定值。

**设置内容。**选择将在每次执行循环时设置其值的项。您可以选择下列其中一个选项：

- **参数。**仅当您选择**流参数 - 字段**或**流参数 - 值**时才可用。从可用列表中选择所需参数。
- **节点。**仅当您选择**节点属性 - 字段**或**节点属性 - 值**时才可用。选择要对其设置循环的节点。单击浏览按钮以打开“选择节点”对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点：“源”、“进程”、“图形”、“建模”、“输出”、“导出”或“应用模型”节点。
- **属性。**仅当您选择**节点属性 - 字段**或**节点属性 - 值**时才可用。从可用列表中选择节点的属性。

**要使用的字段。**仅当您选择**流参数 - 字段**或**节点属性 - 字段**时才可用。选择节点中要用于提供迭代值的字段。您可以选择下列其中一个选项：

- **节点。**仅当您选择**流参数 - 字段**时才可用。选择要对其设置循环且包含详细信息的节点。单击浏览按钮以打开“选择节点”对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点：“源”、“进程”、“图形”、“建模”、“输出”、“导出”或“应用模型”节点。
- **字段列表。**单击右边列中的列表按钮可显示“选择字段”对话框，您可以在此对话框中选择节点中要用于提供迭代数据的字段。请参阅第 8 页的『[选择用于迭代的字段](#)』以获取更多信息。

**要使用的值。**仅当您选择**流参数 - 值**或**节点属性 - 值**时才可用。选择所选字段内要用作迭代值的值。您可以选择下列其中一个选项：

- **节点。**仅当您选择**流参数 - 值**时才可用。选择要对其设置循环且包含详细信息的节点。单击浏览按钮以打开“选择节点”对话框并选择所需节点；如果列出的节点过多，那么可以通过选择下列其中一个类别对显示结果进行过滤，以仅显示特定类型的节点：“源”、“进程”、“图形”、“建模”、“输出”、“导出”或“应用模型”节点。
- **字段列表。**选择节点中用于提供迭代数据的字段。
- **值列表。**单击右边列中的列表按钮可显示“选择值”对话框，您可以在此对话框中选择节点中要用于提供迭代数据的字段。

## 创建用于流中的循环的迭代变量

您可以使用迭代变量在每次执行循环时更改流中的流参数值或选定节点的属性值。例如，如果流循环将对汽车销售数据进行分析并使用制造国家或地区作为迭代关键字，那么您可能会具有一个按型号显示销售额的图形输出，以及另一显示了废气排放信息的图形输出。在这些情况下，您可以创建迭代变量，这些变量将为生成的图形创建新标题，例如瑞典汽车排放和按型号排列的日本汽车销售额。使用“定义迭代变量”对话框可以设置任何您需要的变量。

要打开对话框，请选择“循环”子选项卡左下角中的**添加变量...**按钮，或者右键单击流中的任何节点并选择：**循环/条件执行 > 定义迭代变量。**

要设置迭代变量，请填写下列字段：

**更改。**选择要修改的属性的类型。可以从**流参数**或**节点属性**中进行选择。

- 如果选择**流参数**，请选择所需参数，然后通过循环的各个迭代，使用下列其中一个选项（如果在流中可用）定义应该将该参数设置为的值。
  - **全局变量。**选择应该将流参数设置为的全局变量。

- **表输出单元。** 要将流参数设置为表输出单元中的值，请从列表中选择表，然后输入要使用的**行和列**。
- **手动输入。** 如果要手动为此参数输入将在各个迭代中采用的值，请选择此选项。返回到“循环”子选项卡时，将创建一个可在其中输入所需文本的新列。
- 如果选择**节点属性**，请选择所需节点以及该节点的其中一个属性，然后选择要用于该属性的值。通过使用下列其中一个选项，可以设置新属性值：
  - **单独。** 属性值将使用迭代关键字值。请参阅第 6 页的『创建用于流中的循环的迭代关键字』以获取更多信息。
  - **作为资源前缀。** 使用迭代关键字值作为在**资源**字段中输入的内容的前缀。
  - **作为资源后缀。** 使用迭代关键字值作为在**资源**字段中输入的内容的后缀。

如果选择前缀或后缀选项，那么系统将提示您向**资源**字段添加附加文本。例如，如果迭代关键字值为制造国家或地区并且您选择**作为资源前缀**，那么可以在此字段中输入 - 按型号排列的销售额。

## 选择用于迭代的字段

创建迭代时，您可以使用“选择字段”对话框选择一个或多个字段。

**排序依据** 可以通过选择下列其中一个选项对可供查看的字段进行排序：

- **自然** 数据流向下遍历数据时，当前节点接收字段的顺序即为字段的查看顺序。
- **名称** 采用字母顺序对字段进行排序以便于查看。
- **类型** 查看字段时按其测量级别排序。此选项在选择具有特定测量级别的字段时非常有用。

一次从列表中选择一个字段，或采用按住 Shift 并单击和按住 Ctrl 并单击的方法选择多个字段。此外，也可以使用列表下面的按钮根据测量级别选择多组字段，或选择或取消选择表中所有字段。

请注意，可供选择的字段将进行过滤，以仅显示适用于您使用的流参数或节点属性的字段。例如，如果您使用的是存储类型为字符串的流参数，那么将仅显示存储类型为字符串的字段。

## 流中的条件执行


通过条件执行，您可以根据与您所定义的条件相匹配的流内容来控制终端节点的运行方式；可能的示例如下：

- 根据给定值是 true 还是 false，控制是否将运行节点。
- 定义节点循环将以并行方式运行还是按顺序运行。

可以在流“执行”选项卡的**条件**子选项卡上设置要满足的条件。要显示该子选项卡，请选择**循环/条件执行**执行方式。

如果设置了**循环/条件执行**执行方式，那么在您运行流时，您定义的所有条件执行要求都将生效。（可选）您可以根据条件执行需求生成脚本代码，并通过单击“条件”子选项卡右下角的**粘贴...**，将其粘贴到脚本编辑器中；显示的主“执行”选项卡会发生更改，以显示**缺省值（可选脚本）**执行方式，且脚本位于选项卡的顶部。这意味着，您可以先使用多个循环对话框选项来定义条件，然后再生成可在脚本编辑器中进行进一步定制脚本。请注意，当您单击**粘贴...**时，您定义的任何循环需求也将显示在生成的脚本中。

要设置条件，请完成下列步骤：

1. 在“条件”子选项卡的右侧列中，单击“添加新条件”按钮 ，以打开“添加条件执行语句”对话框。在此对话框中，可以指定执行节点所必须满足的条件。
2. 在“添加条件执行语句”对话框中，指定以下内容：
  - a. **节点。** 选择要对其设置条件执行的节点。单击浏览按钮以打开“选择节点”对话框并选择所需节点；如果列出的节点过多，那么可以对显示结果进行过滤，以按下列其中一个类别显示节点：“导出”、“图形”、“建模”或“输出”节点。
  - b. **作为依据的条件。** 指定执行节点所必须满足的条件。您可以从下列四个选项中选择其中一个：**流参数**、**全局变量**、**表输出单元**或**始终满足**。在对话框下半部分中输入的详细信息由您选择的条件控制。

- **流参数** 从提供的列表中选择参数，然后选择该参数的**运算符**；例如，运算符可以是大于、等于、小于和介于之间等等。然后，输入**值**或最小值和最大值，具体取决于运算符。
  - **全局变量**。从提供的列表中选择变量；例如，这可能包括平均值、总和、最小值、最大值或标准差。然后，选择**运算符**及所需值。
  - **表输出单元**。从可用列表中选择表节点，然后选择表中的**行和列**。然后，选择**运算符**及所需值。
  - **始终满足**。如果必须始终执行节点，请选择此选项。选择此选项后，将无需选择其他参数。
3. 重复步骤 1 和 2 所需次数，直到您设置了所有需要的条件。所选节点和执行该节点前所必须满足的条件将分别显示在该子选项卡主要部分中的**执行节点**和**如果满足此条件**列中。
  4. 缺省情况下，将按显示顺序执行节点和条件；要在列表中上下移动节点和条件，请单击节点或条件以将其选中，然后使用该子选项卡右侧的向上或向下箭头更改顺序。

另外，您可以在“条件”子选项卡的底部设置下列选项：

- **按顺序对所有条件进行求值**。选择此选项可按条件在该子选项卡上的显示顺序对各个条件进行求值。对所有条件进行求值后，将立即执行那些条件求值为“true”的节点。
- **一次执行一个节点**。只有选中**按顺序对所有条件进行求值**时才可用。选中此选项表示，如果某个条件求值为“true”，那么将先执行与该条件关联的节点，然后再对下一个条件进行求值。
- **在首次命中之前进行求值**。选中此选项表示，将仅运行第一个根据您的条件返回“true”求值的节点。

## 执行和中断脚本

可以通过多种方法来执行脚本。例如，在流脚本或独立脚本对话框中，“运行此脚本”按钮将执行完整的脚本：



图 1: “运行此脚本”按钮

“运行选定的行”按钮用于执行您在脚本中选择的单一行或者相邻行所组成的块：



图 2: “运行选定的行”按钮

可以使用以下方式执行脚本：

- 在流脚本或独立脚本对话框中，单击“运行此脚本”或“运行选定的行”按钮。
- 在**运行此脚本**设置为缺省执行方式的情况下运行流。
- 启动后以交互模式使用 **-execute** 标志。有关更多信息，请参阅主题 [第 55 页的『使用命令行参数』](#)。

注: 如果在“超节点脚本”对话框中选择**运行此脚本**，则将在执行超节点时执行超节点脚本。

### 中断脚本执行

“流脚本”对话框中的红色“停止”按钮将在脚本执行过程中被激活。使用此按钮可以放弃脚本和任何当前流的执行。

## 查找和替换

可在编辑脚本或表达式文本的位置（包括脚本编辑器和 CLEM 表达式构建器）或定义“报告”节点中的模板时使用“查找/替换”对话框。在上述任一区域编辑文本时，按 **Ctrl+F** 可访问此对话框，并确保光标的焦点位于文本区域中。例如，处理填充节点时，可以通过“设置”选项卡的任一文本区域或表达式构建器中的文本字段访问此对话框。

1. 使用文本区域中的光标，按 **Ctrl+F** 可访问“查找/替换”对话框。
2. 输入要搜索的文本，或从最近搜索项下拉列表中选择。

3. 输入替换文本（如果有的话）。
4. 单击**查找下一个**开始搜索。
5. 单击**替换**替换当前选定的内容，或单击**全部替换**更新所有项或选定的实例。
6. 每次操作完成后，此对话框将关闭。从任一文本区域中按 F3 键，可重复上一次查找操作，或按 Ctrl+F，可再次访问该对话框。

### 搜索选项

**匹配大小写。** 指定查找操作是否区分大小写；例如 *myvar* 是否与 *myVar* 匹配。无论怎样设置，替换文本始终完全按照输入插入。

**全字匹配。** 指定查找操作是否匹配单词中内嵌的文本。如果选中，*spider* 的搜索结果将不会包括 *spiderman* 或 *spider-man*。

**正则表达式。** 指定是否使用正则表达式语法（请参阅下一节）。如果选中，**仅限于整个单词**选项将禁用并且会忽略其值。

**仅所选文本。** 控制使用**全部替换**选项时的搜索范围。

### 正则表达式语法

使用正则表达式，您可以搜索特殊字符（如选项卡或换行字符）、字符的类或范围（如 *a* 到 *d*）、任何数字或非数字以及边界（如行首或行尾）。支持的表达式类型如下。

表 1: 字符匹配	
字符	匹配
x	字符 x
\\	反斜杠字符
\0n	含八进制值的字符 0n (0 <= n <= 7)
\0nn	含八进制值的字符 0nn (0 <= n <= 7)
\0mnn	含八进制值的字符 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	含十六进制值的字符 0xhh
\uhhhh	含十六进制值的字符 0xhhhh
\t	制表符 ('\u0009')
\n	换行符 ('\u000A')
\r	回车符 ('\u000D')
\f	换页符 ('\u000C')
\a	警报（蜂鸣）符 ('\u0007')
\e	转义符 ('\u001B')
\cx	x 对应的控制字符

表 2: 匹配字符类	
字符类	匹配
[abc]	a、b、或 c（简单类）
[^abc]	除 a、b、或 c 之外的所有字符（相减）
[a-zA-Z]	a 到 z 或 A 到 Z，包含（范围）
[a-d[m-p]]	a 到 d 或 m 到 p（合并）。也可指定为 [a-dm-p]

表 2: 匹配字符类 (继续)	
字符类	匹配
[a-z&&[def]]	a 到 z 和 d、e、或 f（交集）
[a-z&&[^bc]]	a 到 z，除 b 和 c 外（相减）。也可指定为 [ad-z]
[a-z&&[^m-p]]	a 到 z，而非 m 到 p（相减）。也可指定为 [a-lq-z]

表 3: 预定义字符类	
预定义字符类	匹配
.	任意字符（可能或不可能与行终止符匹配）
\d	任意数字：[0-9]
\D	非数字：[^0-9]
\s	空格字符：[\t\n\x0B\f\r]
\S	非空格字符：[^s]
\w	单词字符：[a-zA-Z_0-9]
\W	非单词字符：[^w]

表 4: 边界匹配	
边界匹配符	匹配
^	行首
\$	行尾
\b	单词边界
\B	非单词边界
\A	输入的开头
\Z	除最后终止符外（如果有），输入的结尾
\z	输入的结尾





---

## 第 2 章 脚本语言

### 脚本编写语言概述

---

通过 IBM SPSS Modeler 的脚本编制工具，您可以创建一些脚本，这些脚本可以在 SPSS Modeler 用户界面上运行、处理输出对象并运行命令语法。您可以在 SPSS Modeler 中直接运行这些脚本。

IBM SPSS Modeler 中的脚本以脚本语言 Python 编写。IBM SPSS Modeler 所使用的基于 Java 的 Python 实现称为 Jython。脚本语言包含下列功能部件：

- 用于引用节点、流、工程、输出和其他 IBM SPSS Modeler 对象的格式。
- 可用于处理这些对象的一组脚本编制语句或命令。
- 用于设置变量、参数和其他对象的值的脚本编制表达式语言。
- 注释、连接符和文字文本块的支持。

以下各节描述了 Python 脚本语言、Python 的 Jython 实现以及在 IBM SPSS Modeler 内进行脚本编制的入门基本语法。具体属性和命令的有关信息则在随后的章节中提供。

### Python 和 Jython

---

Jython 是 Python 脚本语言的实现，它以 Java 语言编写并与 Java 平台进行集成。Python 是一种面向对象的功能强大的脚本语言。Jython 具有成熟脚本语言的生产力特征，而且与 Python 不同的是，Jython 可以在任何支持 Java 虚拟机 (JVM) 的环境中运行。这意味着您在编写程序时可以使用 JVM 上的 Java 库。通过 Jython，您可以利用此差异并使用 Python 语言的语法和大部分功能

作为一种脚本语言，Python（及其 Jython 实现）易于学习并能够高效地进行编码，而且具备创建运行程序所需要的最小结构。可以在交互方式下输入代码，即一次输入一行。Python 是一种解释性脚本语言；它没有 Java 中的预编译步骤。Python 程序仅仅是文本文件，系统将在输入这些文件时对其进行解释（在解析语法错误后）。简单表达式（例如已定义的值）以及更加复杂的操作（例如函数定义）将立即执行并可供使用。可以快速测试任何对代码进行的更改。但是，脚本解释确实存在一些缺点。例如，由于使用未定义的变量不是编译器错误，因此只有在执行使用了该变量的语句的情况下，才会检测到此错误。在这种情况下，可以编辑并运行程序以调试错误。

Python 将所有内容（包括所有数据和代码）视为对象。因此，您可以使用多行代码来处理这些对象。某些选择类型（例如数字和字符串）将被更加方便地视为值而不是对象；Python 支持此行为。有一个受支持的 Null 值。此 Null 值具有保留名称 None。

有关 Python 和 Jython 脚本编写的更深入介绍以及一些示例脚本，请参阅 <http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html> 和 <http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html>。

### Python 脚本编制

---

本 Python 脚本语言指南介绍了在 IBM SPSS Modeler 中编制脚本时最可能使用的组件，其中包括概念和编程基础。这将为您提供足够的知识来开发自己的 Python 脚本，以便在 IBM SPSS Modeler 中使用。

### 操作

赋值是使用等号 (=) 完成的。例如，要将值“3”赋值给名为“x”的变量，您可以使用以下语句：

```
x = 3
```

等号还可用于将字符串类型数据赋值给变量。例如，要将值“a string value”赋值给变量“y”，您可以使用以下语句：

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

下表列出了一些常用的比较运算和数值运算及其描述。

表 5: 常用的比较运算和数值运算	
运算	描述
<code>x &lt; y</code>	x 是否小于 y ?
<code>x &gt; y</code>	x 是否大于 y ?
<code>x &lt;= y</code>	x 是否小于或等于 y ?
<code>x &gt;= y</code>	x 是否大于或等于 y ?
<code>x == y</code>	x 是否等于 y ?
<code>x != y</code>	x 是否不等于 y ?
<code>x &lt;&gt; y</code>	x 是否不等于 y ?
<code>x + y</code>	将 y 与 x 相加
<code>x - y</code>	从 x 中减去 y
<code>x * y</code>	将 x 乘以 y
<code>x / y</code>	将 x 除以 y
<code>x ** y</code>	求 x 的 y 次幂

## 列表

列表是元素序列。列表可以包含任意数目的元素，而列表的元素可以是任何类型的对象。也可以将列表视为阵列。随着添加、移除或替换元素，列表中元素的数目可能会增加或减少。

示例

<code>[]</code>	任何空列表。
<code>[1]</code>	包含单个元素（整数）的列表。
<code>["Mike", 10, "Don", 20]</code>	包含 4 个元素（两个字符串元素和两个整数元素）的列表。
<code>[[], [7], [8, 9]]</code>	列表的列表。每个子列表都是一个空列表或整数元素列表。
<pre>x = 7; y = 2; z = 3; [1, x, y, x + y]</pre>	整数列表。此示例说明了变量和表达式的使用。

您可以向变量分配列表，例如：

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

然后，可以访问列表的特定元素，例如：

```
mylist[0]
```

这将生成以下输出：

```
one
```

方括号 ([]) 中的数字称为索引，它指向列表中的某个特定元素。将从 0 开始对列表中的元素编制索引。

您也可以选择列表中的一系列元素；这称为切割。例如，`x[1:3]` 会选择 x 的第二个和第三个元素。结束索引是经过选择的索引。



# 字符串

字符串是一个被视为值的不可变字符序列。字符串支持所有生成新字符串的不可变序列函数和运算符。例如, "abcdef"[1:4] 将生成输出 "bcd"。

在 Python 中, 字符由长度为 1 的字符串表示。

字符串字面值通过使用单重引用或三重引用来定义。使用单引号定义的字符串不能跨行, 而使用三重引号定义的字符串可以跨行。字符串可以用单引号 (') 或双引号 (") 括起来。引用字符可包含其他未转义的引用字符或者转义的引用字符 (前置反斜杠 (\) 字符)。

示例

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

Python 解析器将自动合并多个以空格分隔的字符串。这样您可以更轻松地输入长字符串, 并且更容易混合单个字符串中的引号类型, 例如:

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

这将生成以下输出:

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

字符串支持一些有用的方法。下表列出了其中一些方法。

表 6: 字符串方法	
Method	用途
s.capitalize()	对 s 执行首字母大写
s.count(ss {,start {,end}})	计算 ss 在 s[start:end] 中的出现次数
s.startswith(str {, start {, end}}) s.endswith(str {, start {, end}})	测试以查看 s 是否以 str 开头 测试以查看 s 是否以 str 结尾
s.expandtabs({size})	将制表符替换为空格, 缺省 size 为 8
s.find(str {, start {, end}}) s.rfind(str {, start {, end}})	在 s 中查找 str 的第一个索引; 如果找不到, 那么结果为 -1。rfind 从右到左进行搜索。
s.index(str {, start {, end}}) s.rindex(str {, start {, end}})	在 s 中查找 str 的第一个索引; 如果找不到, 那么将引起 ValueError。rindex 从右到左进行搜索。
s.isalnum	测试以查看字符串是否为字母数字字符串
s.isalpha	测试以查看字符串是否为字母字符串
s.isnum	测试以查看字符串是否为数字字符串
s.isupper	测试以查看字符串是否为全部大写
s.islower	测试以查看字符串是否为全部小写
s.isspace	测试以查看字符串是否全是空格
s.istitle	测试以查看字符串是否为首字母大写的字母数字字符串序列

表 6: 字符串方法 (继续)	
Method	用途
<code>s.lower()</code> <code>s.upper()</code> <code>s.swapcase()</code> <code>s.title()</code>	转换为全部小写 转换为全部大写 转换为大小写颠倒 转换为全标题形式
<code>s.join(seq)</code>	将 <code>seq</code> 中的字符串连接起来，以 <code>s</code> 作为分隔符
<code>s.splitlines({keep})</code>	将 <code>s</code> 分割为多行，如果 <code>keep</code> 为 <code>true</code> ，那么将使用换行
<code>s.split({sep}, max)</code>	使用 <code>sep</code> （缺省 <code>sep</code> 为空格）将 <code>s</code> 分割为“字”，最多分割 <code>max</code> 次数
<code>s.ljust(width)</code> <code>s.rjust(width)</code> <code>s.center(width)</code> <code>s.zfill(width)</code>	在宽度为 <code>width</code> 的字段中，将字符串左对齐 在宽度为 <code>width</code> 的字段中，将字符串右对齐 在宽度为 <code>width</code> 的字段中，将字符串居中对齐 用 0 进行填充。
<code>s.lstrip()</code> <code>s.rstrip()</code> <code>s.strip()</code>	移除前导空格 移除尾部空格 移除前导和尾部空格
<code>s.translate(str, delc)</code>	除去 <code>delc</code> 中的任何字符后，使用表转换 <code>s</code> 。 <code>str</code> 应该是长度为 <code>256</code> 的字符串。
<code>s.replace(old, new, max)</code>	使用字符串 <code>new</code> 全部替换或按照 <code>max</code> 出现次数替换字符串 <code>old</code>

## 备注

备注是由井号 (#) 引入的注释。同一行上井号后面的所有文本都被视为备注的一部分，并且会被忽略。备注可以开始于任何列。以下示例说明了备注的使用：

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

## 语句语法

Python 的语句语法非常简单。通常，每个源代码行都是单一语句。除 `expression` 和 `assignment` 语句外，每个语句都由一个关键字名称（例如 `if` 或 `for`）引入。可以在代码中任何语句之间的任意位置插入空白行或备注行。如果某一行中有多个语句，那么必须使用分号 (;) 来分隔每个语句。

超长语句可以分为多行。在这种情况下，要分到下一行的语句必须以反斜杠 (\) 结尾，例如：

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

如果某个结构括在圆括号 (())、方括号 ([]) 或花括号 ({} ) 内，那么语句可以在任何逗号后面分为新行，而不必插入反斜杠，例如：

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

## 标识

标识用于对变量、函数、类和关键字进行命名。标识可以是任何长度，但必须以大写或小写的字母字符或下划线字符(\_)开头。以下划线开头的名称通常保留用于内部名称或专用名称。在第一个字符后面，标识可以包含任意数目的字母字符、0 到 9 的数字以及下划线字符，并且这些字符和数字可以任意组合。

Jython 中的一些保留字不可用于对变量、函数或类进行命名。这些保留字分为下列类别：

- **语句引导词**：assert、break、class、continue、def、del、elif、else、except、exec、finally、for、from、global、if、import、pass、print、raise、return、try 和 while
- **参数引导词**：as、import 和 in
- **运算符**：and、in、is、lambda、not 和 or

关键字使用不当通常会生成 `SyntaxError`。

## 代码块

代码块是在期望单个语句的位置使用的语句组。代码块可以跟随下列任何语句：if、elif、else、for、while、try、except、def 和 class。这些语句将引入带有冒号字符(:)的代码块，例如：

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

使用缩进对代码块进行定界，而不是像 Java 一样使用花括号。代码块中的所有行都必须缩进到同一位置。这是因为对缩进的更改指示代码块结束。通常，每一级缩进四个空格。建议使用空格而不是制表符来缩进行。不得混用空格和制表符。模块的最外层块中的行必须从第一列开始，否则将发生 `SyntaxError`。

组成代码块的语句（以及冒号后面的语句）也可以包括在一行中，并以分号分隔，例如：

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

## 将参数传递给脚本

将参数传递给脚本非常有用，因为这表示可以重复使用脚本而无需进行修改。在命令行中传递的参数将作为列表 `sys.argv` 中的值进行传递。使用命令 `len(sys.argv)` 可以获取所传递的值的数目。例如：

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

在此示例中，`import` 命令用于导入整个 `sys` 类，以便可以使用这个类中存在的方法，例如 `argv`。

可以使用以下行调用此示例中的脚本：

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

结果为以下输出：

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

# 示例

`print` 关键字将打印紧跟其后的参数。如果语句后跟逗号，那么不会在输出中新增一行。例如：

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

这将生成以下输出：

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

`for` 语句用于迭代代码块。例如：

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

在此示例中，将为列表 `mylist1` 分配 3 个字符串。然后，将打印该列表的元素，每个元素占用一行。这将生成以下输出：

```
one
two
three
```

在此示例中，迭代器 `lv` 将依次采用列表 `mylist1` 中每个元素的值，因为 `for` 循环用于实现每个元素的代码块。迭代器可以是任意长度的任何有效标识。

`if` 语句是条件语句。该语句将对条件进行求值，并根据求值结果返回 `true` 或 `false`。例如：

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

在此示例中，对迭代器 `lv` 进行了求值。如果 `lv` 的值为 `two`，那么将返回一个字符串，该字符串不同于 `lv` 的值不是 `two` 时返回的字符串。这将生成以下输出：

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

# 数学方法

您可以从 `math` 模块访问有用的数学方法。下表列出了其中一些方法。除非另有说明，否则所有值将作为浮点数返回。

表 7: 数学方法	
Method	用途
<code>math.ceil(x)</code>	将 <code>x</code> 的上限作为浮点数返回，即大于或等于 <code>x</code> 的最小整数
<code>math.copysign(x, y)</code>	返回带有 <code>y</code> 符号的 <code>x</code> 。 <code>copysign(1, -0.0)</code> 返回 <code>-1</code>
<code>math.fabs(x)</code>	返回 <code>x</code> 的绝对值
<code>math.factorial(x)</code>	返回 <code>x</code> 阶乘。如果 <code>x</code> 是负数或非整数，那么将发生 <code>ValueError</code> 。

表 7: 数学方法 (继续)	
Method	用途
<code>math.floor(x)</code>	将 $x$ 的下限作为浮点数返回, 即小于或等于 $x$ 的最大整数
<code>math.frexp(x)</code>	将 $x$ 的尾数 ( $m$ ) 和指数 ( $e$ ) 作为对 ( $m$ , $e$ ) 返回。 $m$ 是浮点数, $e$ 是整数, 这样刚好满足 $x == m * 2^{**}e$ 。如果 $x$ 为零, 那么返回 $(0.0, 0)$ , 否则返回 $0.5 \leq \text{abs}(m) < 1$ 。
<code>math.fsum(iterable)</code>	返回 <code>iterable</code> 中值的精确浮点总和
<code>math.isinf(x)</code>	检查浮点数 $x$ 是正不定式还是负不定式
<code>math.isnan(x)</code>	检查浮点数 $x$ 是否为 NaN (非数字)
<code>math.ldexp(x, i)</code>	返回 $x * (2^{**}i)$ 。此方法本质上是函数 <code>frexp</code> 的反函数。
<code>math.modf(x)</code>	返回 $x$ 的小数和整数部分。这两个结果都带有 $x$ 的符号, 并且都是浮点数。
<code>math.trunc(x)</code>	返回已截断为 Integral 的 Real 值 $x$ 。
<code>math.exp(x)</code>	返回 $e^{**}x$
<code>math.log(x[, base])</code>	返回以给定值 <code>base</code> 为底的 $x$ 的对数。如果未指定 <code>base</code> , 那么将返回 $x$ 的自然对数。
<code>math.log1p(x)</code>	返回 $1+x$ ( <code>base e</code> ) 的自然对数
<code>math.log10(x)</code>	返回以 10 为底的 $x$ 的对数
<code>math.pow(x, y)</code>	返回 $x$ 的 $y$ 次幂。 <code>pow(1.0, x)</code> 和 <code>pow(x, 0.0)</code> 始终返回 1, 即使 $x$ 为零或 NaN 也如此。
<code>math.sqrt(x)</code>	返回 $x$ 的平方根

除数学函数外, 还提供了一些有用的三角函数法。下表列出了这些方法。

表 8: 三角函数法	
Method	用途
<code>math.acos(x)</code>	返回以弧度表示的 $x$ 的反余弦
<code>math.asin(x)</code>	返回以弧度表示的 $x$ 的正弦
<code>math.atan(x)</code>	返回以弧度表示的 $x$ 的正切
<code>math.atan2(y, x)</code>	返回以弧度表示的 <code>atan(y / x)</code> 。
<code>math.cos(x)</code>	返回以弧度表示的 $x$ 的余弦。
<code>math.hypot(x, y)</code>	返回欧几里得范数 $\text{sqrt}(x*x + y*y)$ 。这是从原点到点 $(x, y)$ 的向量的长度。
<code>math.sin(x)</code>	返回以弧度表示的 $x$ 的正弦
<code>math.tan(x)</code>	返回以弧度表示的 $x$ 的正切
<code>math.degrees(x)</code>	将角度 $x$ 从弧度转换为度
<code>math.radians(x)</code>	将角度 $x$ 从度转换为弧度
<code>math.acosh(x)</code>	返回 $x$ 的反双曲余弦值

表 8: 三角函数法 (继续)	
Method	用途
<code>math.asinh(x)</code>	返回 x 的反双曲正弦值
<code>math.atanh(x)</code>	返回 x 的反双曲正切值
<code>math.cosh(x)</code>	返回 x 的双曲正弦值
<code>math.sinh(x)</code>	返回 x 的双曲正弦值
<code>math.tanh(x)</code>	返回 x 的双曲正切值

还有两个数学常量。math.pi 的值为数学常量 pi。math.e 的值为数学常量 e。

## 使用非 ASCII 字符

要使用非 ASCII 字符，Python 需要明确地将字符串编码和解码为 Unicode。在 IBM SPSSModeler 中，假定 Python 脚本采用 UTF-8 进行编码，这是支持非 ASCII 字符的标准 Unicode 编码。以下脚本将执行编译，这是因为 SPSS 建模器 已将 Python 编译器设置为 UTF-8。

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

但是，生成的节点将具有不正确的标签。



图 3: 错误显示的包含非 ASCII 字符的节点标签

标签不正确，因为 Python 已将字符串面值自身转换为 ASCII 字符串。

Python 通过在字符串面值前添加 u 字符前缀来支持指定 Unicode 字符串面值：

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

这将创建 Unicode 字符串，并且将正确显示标签。



图 4: 正确显示的包含非 ASCII 字符的节点标签

使用 Python 和 Unicode 是一个非常大的主题，它超出了本文档的范围。提供了许多对此主题进行了更详细介绍的书籍和在线资源。

# 面向对象的程序设计

面向对象的程序设计基于在程序中创建目标问题模型的概念。面向对象的程序设计减少了编程错误并促进了代码的复用。Python 是一种面向对象的语言。以 Python 定义的对象具有下列特征：

- **身份。** 每个对象都必须截然不同，并且必须可以对此特征进行测试。is 和 is not 测试可用于此目的。
- **状态。** 每个对象都必须能够存储状态。属性（例如字段和实例变量）可用于此目的。
- **行为。** 每个对象都必须能够处理其状态。方法可用于此目的。

Python 提供了支持面向对象的程序设计的下列特征：

- **基于类的对象创建。** 类是用于创建对象的模板。对象是具有关联行为的数据结构。
- **具有多态性的继承。** Python 支持单继承和多重继承。所有 Python 实例方法都具有多态性，并且可以由子类覆盖。
- **隐藏数据的封装。** Python 允许隐藏属性。隐藏后，将只能通过类的方法从类外部访问这些属性。类实现了用于修改数据的方法。

## 定义类

在 Python 类中，可以定义变量和方法。与 Java 不同，您可以在 Python 中对每个源文件（或模块）定义任意数目的公共类。因此，可以认为 Python 中的模块类似于 Java 中的软件包。

在 Python 中，类是使用 class 语句定义的。class 语句的格式如下：

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

或者

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

定义类时，您可以选择提供零个或零个以上的赋值语句。这些赋值语句将创建该类的所有实例共享的类属性。您还可以提供零个或零个以上的函数定义。这些函数定义将创建方法。超类列表是可选的。

在同一作用域中（即模块、函数或类中），类名应该唯一。您可以将多个变量定义为引用同一类。

## 创建类实例

类用于保存类（或共享）属性，或者用于创建类实例。要创建某个类的实例，请将该类作为函数进行调用。例如，请考虑以下类：

```
class MyClass:  
    pass
```

这里使用了 pass 语句，因为需要一个语句来完成类，但不需要以编程方式执行任何操作。

以下语句将创建类 MyClass 的实例：

```
x = MyClass()
```

## 向类实例添加属性

与 Java 不同，客户机可以在 Python 中向类实例添加属性。只有一个实例会发生更改。例如，要向实例 x 添加属性，请在该实例上设置新值：

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

```
x.attrN = n
```

## 定义类属性和方法

任何绑定在类中的变量都是类属性。任何在类中定义的函数都是方法。方法接收类的实例（通常称为 `self`）作为第一个自变量。例如，要定义一些类属性和方法，您可以输入以下代码：

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

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

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

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

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

在类中，您应该使用类名限定所有对类属性的引用，例如 `MyClass.attr1`。应该使用 `self` 变量限定所有对实例属性的引用；例如 `self.text`。在类外部，您应该使用类名（例如 `MyClass.attr1`）或使用类的实例（例如 `x.attr1`，其中 `x` 是类的实例）限定所有对类属性的引用。在类外部，应该使用类的实例限定所有对实例变量的引用；例如 `x.text`。

## 隐藏变量

可以通过创建专用变量来隐藏数据。专用变量只能由类自身进行访问。如果您声明格式为 `__xxx` 或 `__xxx_yyy`（即，带有两个前置下划线）的名称，那么 Python 解析器将自动向声明的名称添加类名以创建隐藏变量，例如：

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

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

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

与 Java 不同，在 Python 中必须使用 `self` 限定所有对实例变量的引用；未隐含地使用 `this`。

## 继承

从类进行继承的功能是面向对象的程序设计的基础。Python 同时支持单继承和多重继承。单继承表示只能存在一个超类。多重继承表示可以存在多个超类。

继承通过对其他类划分子类来实现。许多 Python 类都可以是超类。在 Python 的 Jython 实现中，只能从一个 Java 类进行直接或间接继承。无需提供超类。

超类中的任何属性或方法也包含在任何子类中，并且只要属性或方法未隐藏，类自身或任何客户机就可以使用这些属性或方法。可以在任何位置使用子类的任何实例，并且可以使用超类的实例；这是多态性示例。这些特征支持复用和轻松扩展。

示例

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass    #single inheritance
```



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



---

## 第 3 章 在 IBM SPSS Modeler 中进行脚本编制

### 脚本类型

---

在 IBM SPSS Modeler 中，有 3 种类型的脚本：

- 流脚本，用于控制单个流的执行并且它存储在流中。
- 超节点脚本，用于控制超节点的行为。
- 独立或会话脚本，可用于跨多个不同的流协调执行。

提供了在 IBM SPSS Modeler 中的脚本内使用的多种方法，您可以使用这些方法来访问各种 SPSS Modeler 功能。另外，这些方法还在第 33 页的『第 4 章 脚本编制 API』中用于创建更高级的函数。

### 流、SuperNode 流和图表

---

多数情况下，流这个词表示同一内容，无论是从文件加载的流还是 SuperNode 中使用的流都是如此。通常情况下，它表示连接在一起并可执行的节点集合。但是，脚本中的操作并非在所有地方都受支持，这表示脚本编写人员应该知道他们正在使用哪些流变体。

#### 流

流是主 IBM SPSS Modeler 文档类型。可对其进行保存、加载、编辑和执行。流还可以有参数、全局值、脚本、以及与其相关联的其他信息。

#### SuperNode 流

SuperNode 流是 SuperNode 中所用的流类型。与标准流一样，它包含链接在一起的节点。SuperNode 流与标准流存在若干不同之处：

- 参数以及任何脚本与具有 SuperNode 流的 SuperNode 相关联，而不是与 SuperNode 流本身相关联。
- SuperNode 流有附加的输入和输出连接器节点，这取决于 SuperNode 的类型。这些连接器节点用于将信息导入和导出 SuperNode 流，并且在创建 SuperNode 之后会自动创建这些连接器节点。

#### 图表

图表这个词涵盖标准流和 SuperNode 流均支持的功能，例如添加和删除节点，以及修改节点之间的连接。

### 执行流

---

以下示例将运行流中的所有可执行节点，并且它是最简单的流脚本类型：

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

以下示例也将运行流中的所有可执行节点：

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

在此示例中，流存储在名为 `stream` 的变量中。将流存储在变量中非常有用，因为脚本通常用于修改流或流中的节点。创建用于存储流的变量将生成一个更加简洁的脚本。

### 脚本编制上下文

---

`modeler.script` 模块提供了在其中执行脚本的上下文。此模块将在运行时自动导入 SPSS Modeler 脚本中。此模块定义了以下 4 个函数，这些函数提供可以访问其执行环境的脚本：

- `session()` 函数，用于返回脚本的会话。此会话定义语言环境等信息以及将用于运行任何流的 SPSS Modeler 后端（本地进程或联网 SPSS Modeler Server）。
  - `stream()` 函数，此函数可以与流脚本及超节点脚本配合使用。此函数将返回拥有正在运行的流脚本或超节点脚本的流。
  - `diagram()` 函数，此函数可以与超节点脚本配合使用。此函数将返回超节点内的图。对于其他脚本类型，此函数返回的内容与 `stream()` 函数相同。
  - `supernode()` 函数，此函数可以与超节点脚本配合使用。此函数将返回拥有正在运行的脚本的超节点。
- 下表概述了这四个函数及其输出。

表 9: <i>modeler.script</i> 函数摘要				
脚本类型	<code>session()</code>	<code>stream()</code>	<code>diagram()</code>	<code>supernode()</code>
独立	返回会话	在调用该脚本时返回当前受管流（例如，通过批处理方式 - <code>stream</code> 选项传递的流）或 <code>None</code> 。	与 <code>stream()</code> 相同	不适用
流	返回会话	返回流	与 <code>stream()</code> 相同	不适用
SuperNode	返回会话	返回流	返回超节点流	返回超节点

`modeler.script` 模块还定义了一种使用退出码终止脚本的方式。`exit(exit-code)` 函数将停止执行脚本并返回所提供的整数退出码。

为流定义的其中一种方法是 `runAll(List)`。此方法将运行所有可执行节点。执行节点时生成的任何模型或输出都将添加到所提供的列表中。

流执行通常生成模型和图形等输出以及其他输出。要捕获此输出，脚本可以提供一个初始化为列表的变量，例如：

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

执行完成后，可以从 `results` 列表访问执行所生成的任何对象。

## 引用现有节点

- 流通常是使用一些参数预先构建的，在执行流之前必须先修改这些参数。修改这些参数涉及下列任务：
1. 在相关流中找到节点。
  2. 更改节点和/或流设置。

## 查找节点

流提供了多种查找现有节点的方式。下表概述了这些方法。

表 10: 查找现有节点的方法		
Method	返回类型	描述
<code>s.findAll(type, label)</code>	集合	返回具有指定类型和标签的所有节点的列表。类型或标签可以为 <code>None</code> ，在这种情况下将使用其他参数。

表 10: 查找现有节点的方法 (继续)		
Method	返回类型	描述
<code>s.findAll(filter, recursive)</code>	集合	返回指定过滤器所接受的所有节点的集合。如果递归标志为 <code>True</code> ，那么还将搜索指定流内的任何超节点。
<code>s.findByID(id)</code>	节点	返回具有所提供标识的节点或 <code>None</code> （如果不存在此类节点）。搜索范围限制为当前流。
<code>s.findByType(type, label)</code>	节点	返回具有所提供类型和/或标签的节点。类型或名称可以为 <code>None</code> ，在这种情况下将使用其他参数。如果有多个节点匹配，那么将选择并返回任意一个节点。如果没有匹配的节点，那么返回值为 <code>None</code> 。
<code>s.findDownstream(fromNodes)</code>	集合	从所提供的节点列表中进行搜索，并返回所提供节点下游的节点集合。返回的列表包括最初提供的节点。
<code>s.findUpstream(fromNodes)</code>	集合	从所提供的节点列表中进行搜索，并返回所提供节点上游的节点集合。返回的列表包括最初提供的节点。
<code>s.findProcessorForID(String id, boolean recursive)</code>	节点	返回具有所提供标识的节点或 <code>None</code> （如果不存在此类节点）。如果递归标志为 <code>true</code> ，那么还将搜索此图中的任何组合节点。

例如，如果流包含脚本需要访问的单个“过滤”节点，那么可以使用以下脚本来找到该“过滤”节点：

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

另外，如果已经知道节点的标识（如节点对话框的“注释”选项卡中所示），那么可以使用此标识来查找节点，例如：

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

## 设置属性

节点、流、模型和输出都具有可以访问并在大多数情况下可以设置的属性。这些属性通常用于修改对象的行为或外观。下表概述了可用于访问和设置对象属性的方法。

表 11: 用于访问和设置对象属性的方法		
Method	返回类型	描述
<code>p.getPropertyValue(propertyName)</code>	对象	返回指定属性的值或 <code>None</code> （如果不存在此属性）。
<code>p.setPropertyValue(propertyName, value)</code>	不适用	设置指定属性的值。

表 11: 用于访问和设置对象属性的方法 (继续)

Method	返回类型	描述
<code>p.setPropertyValues(properties)</code>	不适用	设置指定属性的值。属性图中的每个条目都包含一个表示属性名的键，以及应指定给该属性的值。
<code>p.getKeyedPropertyValue(propertyName, keyName)</code>	对象	返回指定属性的值及关联的键或 <code>None</code> （如果不存在此属性或键）。
<code>p.setKeyedPropertyValue(propertyName, keyName, value)</code>	不适用	设置指定属性的值和键。

例如，如果要设置位于流的开始位置的“变量文件”节点的值，那么可以使用以下脚本：

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

或者，您可能希望根据“过滤”节点来过滤字段。在这种情况下，还将根据字段名称键入值，例如：

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

## 创建节点以及修改流

在某些情况下，您可能希望向现有流添加新节点。向现有流添加节点通常涉及下列任务：

1. 创建节点。
2. 将节点链接到现有流。

### 创建节点

流提供了多种创建节点的方式。下表概述了这些方法。

表 12: 创建节点的方法		
Method	返回类型	描述
<code>s.create(nodeType, name)</code>	节点	创建具有指定类型的节点并将其添加到指定的流中。
<code>s.createAt(nodeType, name, x, y)</code>	节点	创建具有指定类型的节点并将其添加到指定流中的指定位置。如果 $x < 0$ 或 $y < 0$ ，那么未设置位置。
<code>s.createModelApplier(modelOutput, name)</code>	节点	创建派生自所提供的模型输出对象的模型应用器节点。

例如，要在流中创建新的“类型”节点，您可以使用以下脚本：

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

# 链接和取消链接节点

在流中创建新节点时，这个新节点必须先连接到节点序列，然后才可使用。流提供了多种链接节点和取消链接节点的方法。下表概述了这些方法。

表 13: 用于链接和取消链接节点的方法		
Method	返回类型	描述
s.link(source, target)	不适用	在源节点与目标节点之间创建新链接。
s.link(source, targets)	不适用	在源节点与所提供列表中的每个目标节点之间创建新链接。
s.linkBetween(inserted, source, target)	不适用	连接两个其他节点实例（源节点和目标节点）之间的节点，并将已插入节点的位置设置为位于这两个节点实例之间。将首先移除源节点与目标节点之间的任何直接链接。
s.linkPath(path)	不适用	在节点实例之间创建新路径。第一个节点将链接到第二个节点，而第二个节点将链接到第三个节点，依此类推。
s.unlink(source, target)	不适用	移除源节点与目标节点之间的任何直接链接。
s.unlink(source, targets)	不适用	移除源节点与目标列表中每个对象之间的任何直接链接。
s.unlinkPath(path)	不适用	移除节点实例之间存在的任何路径。
s.disconnect(node)	不适用	移除所提供节点与指定流中任何其他节点之间的任何链接。
s.isValidLink(source, target)	BOOLEAN	如果在指定的源节点与目标节点之间创建链接是有效的，那么此方法将返回 True。此方法将检查这两个对象是否属于指定流，源节点是否可以提供链接以及目标节点是否可以接收链接，并确认创建此类链接不会在流中引起循环。

下面的示例脚本将执行 5 项任务：

1. 创建“变量文件”输入节点、“过滤”节点和“表”输出节点。
2. 将这些节点连接到一起。
3. 在“变量文件”输入节点上设置文件名。
4. 根据生成的输出过滤字段“药品”。
5. 执行“表”节点。

```
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)
```

## 导入、替换和删除节点

与创建和连接节点一样，通常需要替换和删除流中的节点。下表概述了可用于导入、替换和删除节点的方法。

表 14: 用于导入、替换和删除节点的方法		
Method	返回类型	描述
<code>s.replace(originalNode, replacementNode, discardOriginal)</code>	不适用	替换指定流中的指定节点。原始节点和替换节点都必须属于指定流。
<code>s.insert(source, nodes, newIDs)</code>	列表	在所提供的列表中插入节点的副本。假定所提供列表中的所有节点都包含在指定流中。 <code>newIDs</code> 标志指示应该为每个节点生成新标识，还是应该复制并使用现有标识。假定流中所有节点都具有唯一标识，这样在源流与指定流相同的情况下，必须将此标志设置为 <code>True</code> 。此方法将返回新插入节点的列表，此列表中未定义节点的顺序（即，顺序不一定与输入列表中节点的顺序相同）。
<code>s.delete(node)</code>	不适用	从指定流中删除指定节点。该节点必须属于指定流。
<code>s.deleteAll(nodes)</code>	不适用	从指定流中删除所有指定节点。集合中的所有节点都必须属于指定流。
<code>s.clear()</code>	不适用	从指定流中删除所有节点。

## 遍历流中的节点

标识特定节点上游或下游的节点是一项常见需求。流提供了一些可用于标识这些节点的方法。下表概述了这些方法。

表 15: 用于标识上游节点和下游节点的方法		
Method	返回类型	描述
<code>s.iterator()</code>	迭代器	返回针对指定流中包含的节点对象的迭代器。如果在 <code>next()</code> 函数的两次调用之间对流进行了修改，那么将取消定义迭代器的行为。
<code>s.predecessorAt(node, index)</code>	节点	返回所提供节点的指定直接前趋或 <code>None</code> （如果索引超出范围）。
<code>s.predecessorCount(node)</code>	<code>int</code>	返回所提供节点的直接前趋数。
<code>s.predecessors(node)</code>	列表	返回所提供节点的直接前趋。
<code>s.successorAt(node, index)</code>	节点	返回所提供节点的指定直接后继或 <code>None</code> （如果索引超出范围）。
<code>s.successorCount(node)</code>	<code>int</code>	返回所提供节点的直接后继数。



表 15: 用于标识上游节点和下游节点的方法 (继续)		
Method	返回类型	描述
s.successors(node)	列表	返回所提供节点的直接后继。

## 清除或移除项

旧脚本编制支持 `clear` 命令的各种用法，例如：

- `clear outputs` 用于从管理器选用板中删除所有输出项。
- `clear generated palette` 用于从模型选用板中清除所有模型块。
- `clear stream` 用于移除流的内容。

Python 脚本编制支持一组相似的函数；`removeAll()` 命令用于清除流、输出和模型管理器。例如：

- 清除流管理器：

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

- 清除输出管理器：

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

- 清除模型管理器：

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

## 获取节点的相关信息

节点分为多种不同的类别，例如数据导入和导出节点、模型构建节点和其他类型的节点。每个节点都提供了一些可用于查找该节点的相关信息的方法。

下表概述了可用于获取节点的标识、名称和标签的方法。

表 16: 用于获取节点的标识、名称和标签的方法		
Method	返回类型	描述
n.getLabel()	<i>string</i>	返回指定节点的显示标签。只有在属性 <code>custom_name</code> 是非空字符串并且属性 <code>use_custom_name</code> 未进行设置的情况下，标签才是前一属性的值；否则，标签是 <code>getName()</code> 的值。
n.setLabel(label)	不适用	设置指定节点的显示标签。如果新标签为非空字符串，那么会将其指定给属性 <code>custom_name</code> ，并且会将 <code>False</code> 指定给属性 <code>use_custom_name</code> ，以使指定的标签优先；否则，会将空字符串指定给属性 <code>custom_name</code> ，并将 <code>True</code> 指定给属性 <code>use_custom_name</code> 。
n.getName()	<i>string</i>	返回指定节点的名称。

表 16: 用于获取节点的标识、名称和标签的方法 (继续)

Method	返回类型	描述
n.getID()	string	返回指定节点的标识。每次创建新节点时都会创建一个新标识。将节点作为流的组成部分进行保存时，标识将随节点一起持久存储，以便在打开流时保留节点标识。但是，如果将已保存的节点插入流中，那么会将已插入的节点视为新对象并为其分配一个新标识。

下表概述了可用于获取节点的其他相关信息的方法。

表 17: 用于获取节点的相关信息的方法

Method	返回类型	描述
n.getTypeName()	string	返回此节点的脚本编写名称。此名称即为可用于创建该节点的新实例的名称。
n.isInitial()	BOOLEAN	如果此节点是初始节点（即，位于流的开始位置的节点），那么此方法将返回 True。
n.isInline()	BOOLEAN	如果此节点是内嵌节点（即，位于流中部的节点），那么此方法将返回 True。
n.isTerminal()	BOOLEAN	如果此节点是终端节点（即，位于流的结束位置的节点），那么此方法将返回 True。
n.getXPosition()	int	返回节点在流中的 x 位置偏移量。
n.getYPosition()	int	返回节点在流中的 y 位置偏移量。
n.setXYPosition(x, y)	不适用	设置节点在流中的位置。
n.setPositionBetween(source, target)	不适用	设置节点在流中的位置，以使其位于所提供的节点之间。
n.isCacheEnabled()	BOOLEAN	如果已启用缓存，那么此方法将返回 True，否则将返回 False。
n.setCacheEnabled(val)	不适用	对此对象启用或禁用缓存。如果缓存已满并且缓存功能进入禁用状态，那么缓存将进行清空。
n.isCacheFull()	BOOLEAN	如果已禁用缓存，那么此方法将返回 True，否则将返回 False。
n.flushCache()	不适用	清空此节点的缓存。如果缓存未启用或者未滿，那么此方法不起任何作用。

## 第 4 章 脚本编制 API

### 脚本编制 API 简介

脚本编写 API 提供对各种不同的 SPSS Modeler 功能的访问。目前描述的所有方法是 API 的组成部分，可以在脚本内隐式地访问这些方法而不执行进一步的导入。但是，如果要引用 API 类，那么必须使用以下语句显式导入 API：

```
import modeler.api
```

此导入语句是许多脚本编制 API 示例所必需的。

在《*IBM SPSS Modeler Python 脚本编写 API 参考指南*》文档中，可以找到通过脚本编写 API 提供的类、方法和参数的完整指南。

### 示例 1：使用定制过滤器搜索节点

第 26 页的『查找节点』一节提供了关于使用节点的类型名称作为搜索条件在流中搜索节点的示例。在某些情况下，需要执行更加通用的搜索，并且此搜索可使用 `NodeFilter` 类和流 `findAll()` 方法来实现。这种搜索包括以下两个步骤：

1. 创建用于扩展 `NodeFilter` 并实现 `accept()` 方法的定制版本的新类。
2. 使用此新类的实例来调用流 `findAll()` 方法。这将返回所有满足 `accept()` 方法中定义的条件节点。

以下示例显示如何在流中搜索已启用节点高速缓存的节点。所返回的节点列表可用于清空或禁用这些节点的高速缓存。

```
import modeler.api

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

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

### 示例 2：允许用户基于其权限获取目录或文件信息

为避免向用户打开 PSAPI，可以通过调用 `PSAPI` 函数创建文件系统对象来使用名为 `session.getServerFileSystem()` 的方法。

以下示例显示了如何允许用户基于连接到 IBM SPSSModeler 服务器的用户的权限来获取目录或文件信息。

```
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()
```

# 元数据：有关数据的信息

由于流中的节点相互连接在一起，因此能够提供每个节点中可用列或字段的信息。例如，在 **Modeler UI** 中，您可以使用此信息来选择要排序或汇总的字段。此信息被称为数据模型。

脚本还可以访问数据模型，方法是查看进出节点的字段。对于某些节点，输入和输出数据模型是相同的，例如，“排序”节点仅对记录进行重新排序，但不更改数据模型。某些节点（例如，“派生”节点）可添加新的字段。而其他节点（例如，“过滤器”节点）可重命名或移除字段。

在以下示例中，脚本采用标准 **IBM SPSS Modeler druglearn.str** 流，并且对于每个字段，构建其中一个输入字段已删除的模型。按以下操作执行：

- 1. 从“类型”节点访问输出数据模型。
- 2. 在输出数据模型中遍历每个字段。
- 3. 修改每个输入字段的“过滤器”节点。
- 4. 更改要构建的模型名称。
- 5. 运行模型构建节点。

**注：**在 **druglearn.str** 流中运行脚本之前，请记住将脚本语言设置为 **Python**（流是在先前版本的 **IBM SPSS Modeler** 中创建的，因此流脚本语言设置为“旧版”）。

```
import modeler.api

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

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

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

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

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

**DataModel** 对象提供了访问数据模型中字段信息或列信息的多种方法。下表概述了这些方法。

表 18: 用于访问字段信息或列信息的 <i>DataModel</i> 对象方法		
Method	返回类型	描述
d.getColumnCount()	int	返回数据模型中的列数。
d.columnIterator()	迭代器	返回按“缺省”插入顺序返回各列的迭代器。迭代器返回列实例。
d.nameIterator()	迭代器	返回按“缺省”插入顺序返回各列名称的迭代器。
d.contains(name)	BOOLEAN	如果此 <b>DataModel</b> 中存在含有有所供名称的列，那么返回 <b>True</b> ，否则返回 <b>False</b> 。
d.getColumn(name)	列	返回含有指定名称的列。

表 18: 用于访问字段信息或列信息的 <i>DataModel</i> 对象方法 (继续)		
Method	返回类型	描述
<code>d.getColumnGroup(name)</code>	<code>ColumnGroup</code>	返回指定的列组或 <code>None</code> （如果不存在此类列组）。
<code>d.getColumnGroupCount()</code>	<code>int</code>	返回此数据模型中的列组数。
<code>d.columnGroupIterator()</code>	迭代器	依次返回用于返回各个列组的迭代器。
<code>d.toArray()</code>	列[]	按列数组返回数据模型。按“缺省”插入顺序对列进行排序。

每个字段（列对象）包括用于访问列信息的多种方法。下表显示这些方法选择。

表 19: 用于访问列信息的列对象方法		
Method	返回类型	描述
<code>c.columnName()</code>	<code>string</code>	返回列名称。
<code>c.columnLabel()</code>	<code>string</code>	如果不存在与列相关联的任何标签，那么返回列的标签或返回空字符串。
<code>c.measureType()</code>	<code>MeasureType</code>	返回列的测量类型。
<code>c.storageType()</code>	<code>StorageType</code>	返回列的存储类型。
<code>c.isMeasureDiscrete()</code>	<code>BOOLEAN</code>	如果是离散列，那么返回 <code>True</code> 。会将作为集合或标记的列视为离散列。
<code>c.isModelOutputColumn()</code>	<code>BOOLEAN</code>	如果是模型输出列，那么返回 <code>True</code> 。
<code>c.isStorageDatetime()</code>	<code>BOOLEAN</code>	如果列的存储为时间、日期或时间戳记值，那么返回 <code>True</code> 。
<code>c.isStorageNumeric()</code>	<code>BOOLEAN</code>	如果列的存储是整数或实数，那么返回 <code>True</code> 。
<code>c.isValidValue(value)</code>	<code>BOOLEAN</code>	返回 <code>True</code> （如果指定的值对于此存储器有效）和 <code>valid</code> （在有效列值已知时）。
<code>c.modelingRole()</code>	<code>ModelingRole</code>	返回列的建模角色。
<code>c.getSetValues()</code>	对象[]	返回列的有效值数组或 <code>None</code> （如果值未知或者列不是集合）。
<code>c.getValueLabel(value)</code>	<code>string</code>	如果不存在与值相关联的任何标签，那么返回列中值的标签或返回空字符串。
<code>c.getFalseFlag()</code>	对象	返回列的“false”指示符值或 <code>None</code> （如果值未知或者列不是标志）。
<code>c.getTrueFlag()</code>	对象	返回列的“true”指示符值或 <code>None</code> （如果值未知或者列不是标志）。
<code>c.getLowerBound()</code>	对象	返回列中值的下限值或 <code>None</code> （如果值未知或者列不连续）。

表 19: 用于访问列信息的列对象方法 (继续)		
Method	返回类型	描述
c.getUpperBound()	对象	返回列表中值的上限值或 None（如果值未知或者列不连续）。

请注意，访问列信息的大多数方法在 `DataModel` 对象本身中定义了等效方法。例如，以下两个语句是等效的。

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

## 访问已生成的对象

执行某个流通常涉及生成附加输出对象。这些附加对象可能是新模型，也可能是提供要在后续执行中使用的信息的输出。

在以下示例中，`druglearn.str` 流再次用作流的起始点。在此示例中，将执行流中的所有节点，并且结果将存储在列表中。然后，脚本循环遍历结果，并且从执行生成的任何模型输出都保存为 IBM SPSS Modeler 模型 (.gm) 文件，并且模型是导出的 PMML。

```
import modeler.api

stream = modeler.script.stream()

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

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

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

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

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

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

任务运行器类提供了一种运行各项常见任务的便捷方式。下表概述了此类中提供的方法。

表 20: 用于执行常见任务的任务运行器类方法		
Method	返回类型	描述
t.createStream(name, autoConnect, autoManage)	流	创建并返回新的流。请注意，必须以不公开方式创建流而不向用户显示这些流的代码应该将 <code>autoManage</code> 标志设置为 <code>False</code> 。
t.exportDocumentToFile(documentOutput, filename, fileFormat)	不适用	使用指定的文件格式将流描述导出至文件。

表 20: 用于执行常见任务的任务运行器类方法 (继续)

Method	返回类型	描述
<code>t.exportModelToFile(modelOutput, filename, fileFormat)</code>	不适用	使用指定的文件格式将模型导出至文件。
<code>t.exportStreamToFile(stream, filename, fileFormat)</code>	不适用	使用指定的文件格式将流导出至文件。
<code>t.insertNodeFromFile(filename, diagram)</code>	节点	从指定文件中读取并返回节点，然后将其插入所提供的图中。请注意，此方法可用于同时读取节点对象和超节点对象。
<code>t.openDocumentFromFile(filename, autoManage)</code>	DocumentOutput	从指定文件读取并返回文档。
<code>t.openModelFromFile(filename, autoManage)</code>	ModelOutput	从指定文件读取并返回模型。
<code>t.openStreamFromFile(filename, autoManage)</code>	流	从指定文件读取并返回流。
<code>t.saveDocumentToFile(documentOutput, filename)</code>	不适用	将文档保存到指定的文件位置。
<code>t.saveModelToFile(modelOutput, filename)</code>	不适用	将模型保存到指定的文件位置。
<code>t.saveStreamToFile(stream, filename)</code>	不适用	将流保存到指定的文件位置。

## 处理错误

Python 语言提供了通过 `try...except` 代码块执行的错误处理方法。可以在脚本内使用此方法来捕获异常，并处理将导致脚本终止的问题。

在以下示例脚本中，已尝试从 IBM SPSS Collaboration and Deployment Services Repository 中检索模型。此操作可能会导致抛出异常，例如可能未正确设置存储库登录凭证或者存储库路径错误。在脚本中，这可能导致抛出 `ModelerException`（IBM SPSS Modeler 生成的所有异常派生自 `modeler.api.ModelerException`）。

```
import modeler.api

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

注: 某些脚本编制操作可能会导致抛出标准 Java 异常；这些异常并非派生自 `ModelerException`。要捕获这些异常，可以使用附加的 `except` 块来捕获所有 Java 异常，例如：

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

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
```

```

print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()

```

## 流、会话和超节点参数

参数提供了一种在运行时传递值而不是在脚本中直接对这些值进行硬编码的有用方式。参数及其值的定义方式与流相同，即定义为流或超节点的参数表中的条目或命令行中的参数。流类和超节点类实现了一组由 `ParameterProvider` 对象定义的函数，如下表所示。会话提供了 `getParameters()` 调用，此调用将返回定义这些函数的对象。

表 21: 由 `ParameterProvider` 对象定义的函数

Method	返回类型	描述
<code>p.parameterIterator()</code>	迭代器	返回此对象的参数名的迭代器。
<code>p.getParameterDefinition(parameterName)</code>	<code>ParameterDefinition</code>	返回具有指定名称的参数的参数定义或 <code>None</code> （如果此提供程序中不存在此类参数）。结果可以是调用此方法时的定义快照，并且不需要反映通过此提供程序对该参数进行的任何后续修改。
<code>p.getParameterLabel(parameterName)</code>	<code>string</code>	返回指定参数的标签或 <code>None</code> （如果不存在此类参数）。
<code>p.setParameterLabel(parameterName, label)</code>	不适用	设置指定参数的标签。
<code>p.getParameterStorage(parameterName)</code>	<code>ParameterStorage</code>	返回指定参数的存储或 <code>None</code> （如果不存在此类参数）。
<code>p.setParameterStorage(parameterName, storage)</code>	不适用	设置指定参数的存储。
<code>p.getParameterType(parameterName)</code>	<code>ParameterType</code>	返回指定参数的类型或 <code>None</code> （如果不存在此类参数）。
<code>p.setParameterType(parameterName, type)</code>	不适用	设置指定参数的类型。
<code>p.getParameterValue(parameterName)</code>	对象	返回指定参数的值或 <code>None</code> （如果不存在此类参数）。
<code>p.setParameterValue(parameterName, value)</code>	不适用	设置指定参数的值。

在以下示例中，脚本汇总了一些 Telco 数据以查找具有最低平均收入数据的区域。然后，将使用此区域设置一个流参数。接下来，将在“选择”节点中使用该流参数从数据中排除此区域，然后根据剩余的数据构建流失模型。

此示例并不真实，这是因为脚本本身生成了“选择”节点，并因此已经在“选择”节点表达式中直接生成正确的值。但是，流通常是预先构建的，因此通过这种方式设置参数提供了有用的示例。

示例脚本的第一部分用于创建流参数，该流参数将包含平均收入最低的区域。另外，此脚本还将在汇总分支和模型构建分支中创建节点，并将这些节点连接在一起。

```

import modeler.api

stream = modeler.script.stream()

```



```

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

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

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

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

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

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

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

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

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

```

此示例脚本将创建以下流。

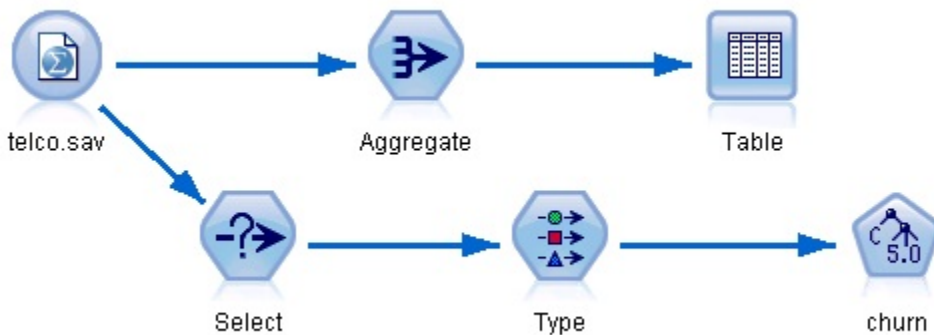


图 5: 示例脚本生成的流

示例脚本的以下部分用于执行位于汇总分支末尾的“表”节点。

```

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

```

示例脚本的以下部分用于访问执行“表”节点所生成的表输出。随后，此脚本将对表中的各行执行迭代，以查找平均收入最低的区域。

```

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

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

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

```

```

        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

```

脚本的以下部分使用平均收入最低的区域来设置先前创建的“LowestRegion”流参数。然后，在从训练数据中排除了指定区域的情况下，此脚本将运行模型构建器。

```

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

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

```

完整的示例脚本如下所示。

```

import modeler.api

stream = modeler.script.stream()

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

else:
    stream.setParameterValue("LowestRegion", -1)

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

```

## 全局值

全局值用于计算指定字段的各项汇总统计。可以在流内部的任何位置访问这些汇总值。在流中，可以按名称访问全局值，这一点与流参数相似。全局值与流参数的差异在于，关联值将在“设置全局值”节点运行时自动进行更新，而不是通过脚本编制或命令行来指定。可以通过调用流的 `getGlobalValues()` 方法来访问该流的全局值。

GlobalValues 对象定义下表中显示的函数。

表 22: GlobalValues 对象所定义的函数		
Method	返回类型	描述
<code>g.fieldNameIterator()</code>	迭代器	返回至少具有一个全局值的每个字段名称的迭代器。
<code>g.getValue(type, fieldName)</code>	对象	返回指定类型和字段名称的全局值或 <code>None</code> （如果找不到值）。虽然未来的功能可能会返回各种值类型，但通常期望返回值为数字。
<code>g.getValues(fieldName)</code>	映射	返回包含指定字段名称的已知条目的图或 <code>None</code> （如果该字段没有现有条目）。

GlobalValues.Type 定义可用的汇总统计的类型。可用的汇总统计如下：

- MAX：字段的最大值。
- MEAN：字段的均值。
- MIN：字段的最小值。
- STDDEV：字段的标准差。
- SUM：字段中值的总和。

例如，以下脚本将访问“收入”字段的均值，此均值由“设置全局值”节点计算：

```

import modeler.api

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

```

## 使用多个流 - 独立脚本

要使用多个流，必须使用独立脚本。可以在 IBM SPSSModeler UI 内编辑和运行独立脚本，也可以在批处理方式下将独立脚本作为命令行参数进行传递。

以下独立脚本将打开两个流。其中一个流用于构建模型，而第二个流用于绘制预测值的分布。

```

# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/19.0.0 /DEMOS/streams/"

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

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

```

```

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

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

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

```

以下示例显示还可以通过哪种方式对打开的流执行迭代（所有流都在“流”选项卡中打开）。请注意，这种方式仅在独立脚本中受支持。

```

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

```

## 第 5 章 脚本编写技巧

本章简要介绍使用脚本的技巧和方法，包括修改流执行、在脚本中采用加密密码以及访问 IBM SPSS 协作与部署服务资源库 中的对象等。

### 修改流执行

运行流时，将按缺省情形下的优化顺序来执行其终端节点。某些情况下，您可能更喜欢以其他顺序来执行。要修改流的执行顺序，请在流属性对话框的“执行”选项卡上完成以下步骤：

1. 打开一个空脚本。
2. 单击工具栏上的**追加缺省脚本**按钮来添加缺省流脚本。
3. 将缺省流脚本中语句的顺序更改为您希望的执行顺序。

### 对节点执行循环

您可以使用 `for` 循环对流中的所有节点进行循环。例如，以下两个脚本示例用于对所有节点进行循环并将“过滤”节点中的字段名更改为大写。

可以在具有“过滤”节点的任何流中使用此脚本，即使实际上不过滤任何字段也是如此。只需添加传递所有字段的过滤节点即可在整个面板上将字段名称更改为大写。

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

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

此脚本在当前流的所有节点中进行循环，并检查每个节点是否为过滤节点。如果是，那么脚本将循环该节点中的每个字段，并使用 `field.upper()` 或 `field.getColumnName().upper()` 函数将名称更改为大写。

### 访问 IBM SPSS 协作与部署服务资源库 中的对象

如果您有 IBM SPSS 协作与部署服务资源库 的许可证，那么可以使用脚本命令在存储库中存储和检索对象。通过使用此存储库，您可以在企业应用程序、工具和解决方案环境中管理数据挖掘模型和相关预测对象的生命周期。

#### 连接到 IBM SPSS 协作与部署服务资源库

要访问存储库，必须首先通过 SPSS 建模器 用户界面的工具菜单或通过命令行建立与该存储库的有效连接。有关更多信息，请参阅第 59 页的『[IBM SPSS 协作与部署服务资源库 连接参数](#)』。

## 访问存储库

可以通过会话访问存储库，例如：

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

## 从存储库中检索对象

在脚本中，使用 `retrieve*` 函数来访问各种对象，包括流、模型、输出和节点。下表中显示了关于检索函数的摘要。

表 23: 检索脚本编制函数	
对象类型	存储库函数
流	<code>repo.retrieveStream(String path, String version, String label, Boolean autoManage)</code>
模型	<code>repo.retrieveModel(String path, String version, String label, Boolean autoManage)</code>
输出	<code>repo.retrieveDocument(String path, String version, String label, Boolean autoManage)</code>
节点	<code>repo.retrieveProcessor(String path, String version, String label, ProcessorDiagram diagram)</code>

例如，您可以使用下列函数从存储库中检索流：

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

此示例将从指定的文件夹中检索 `risk_score.str` 流。标签 `production` 标识要检索的流版本，而最后一个参数指定 SPSS 建模器 将管理流（例如，如果 SPSS 建模器 用户界面可见，那么流将出现在流选项卡中）。作为替代方法，要使用未添加标签的特定版本，请使用下列函数：

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

注: 如果版本和标签参数都为 `None`，那么将返回最新版本。

## 在存储库中存储对象

要使用脚本编写在存储库中存储对象，请使用 `store*` 函数。下表中显示了关于存储函数的摘要。

表 24: 存储脚本编写功能	
对象类型	存储库函数
流	<code>repo.storeStream(ProcessorStream stream, String path, String label)</code>
模型	<code>repo.storeModel(ModelOutput modelOutput, String path, String label)</code>
输出	<code>repo.storeDocument(DocumentOutput documentOutput, String path, String label)</code>
节点	<code>repo.storeProcessor(Processor node, String path, String label)</code>

例如，您可以使用下列函数存储新版本的 `risk_score.str` 流：

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

此示例将存储新版本的流，并将 `"test"` 标签与此流进行关联，然后返回新创建的版本的版本标记。

注: 如果您不想将标签与该新版本进行关联，请对该标签传递 `None`。

## 管理存储库文件夹

通过使用存储库中的文件夹，您可以将对象编入逻辑组中，从而更易于查看相关的对象。通过使用 `createFolder()` 函数创建文件夹，如以下示例中所示：

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

此示例将在 `/projects` 文件夹中创建名为 `"cross-sell"` 的新文件夹。该函数将返回新文件夹的完整路径。

要重命名某个文件夹，请使用 `renameFolder()` 函数：

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

第一个参数是要重命名的文件夹的完整路径，而第二个参数是指定给该文件夹的新名称。

要删除空文件夹，请使用 `deleteFolder()` 函数：

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

## 锁定和解锁对象

对于脚本，您可以锁定一个对象，以防止其他用户更新任一现有版本或新建版本。还可以解锁已锁定的对象。

锁定和解锁对象的语法为：

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

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

对于存储和检索对象，`REPOSITORY_PATH` 指出对象在存储库中的位置。路径必须用英文引号引起并以正斜杠作为分隔符。路径不区分大小写。

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

除此之外，还可以使用统一资源标识 (URI) 而非存储库路径来给出对象的位置。URI 必须包含前缀 `spsscr:`，同时必须完全括在引号中。只有正斜杠可以作为路径分隔符，空格必须以编码形式出现。即在路径中以 `%20` 代替空格。URI 不区分大小写。示例如下：

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

注意，对象锁定适用于对象的所有版本 - 您无法锁定或解锁单个版本。

## 生成经过编码的密码

某些情况下，可能需要在脚本中包含密码，例如，您可能需要访问受密码保护的数据源。加密密码可用在：

- 数据库源和输出节点的节点属性
- 登录到服务器的命令行自变量
- 存储在 `.par` 文件（由导出节点的“发布”选项卡生成的参数文件）中的数据库连接属性

通过此用户界面，可以使用一个工具根据 **Blowfish** 算法来生成加密密码（有关详细信息，请参阅 <http://www.schneier.com/blowfish.html>）。编码后，可以复制密码并将其存储到脚本文件和命令行参数中。用于 `databasenode` 和 `databaseexportnode` 的节点属性 `epassword` 存储加密密码。

1. 要生成加密密码，请从“工具”菜单中选择：

对密码进行编码...

2. 在“密码”文本框中指定一个密码。
3. 单击**编码**，以便为您的密码生成随机编码。
4. 单击“复制”按钮，以便将经过编码的密码复制到剪贴板。
5. 将此密码粘贴到所需的脚本或参数中。

## 脚本检查

通过单击“独立脚本”对话框工具栏上的红色检查按钮，可以快速检查所有类型脚本的语法。



图 6: 流脚本工具栏图标

脚本检查将就您编码中的错误发出警报并给出改进建议。要查看错误行，请单击该对话框下半部分的反馈。此时将以红色突出显示错误。

## 从命令行执行脚本编写

通过编写脚本可以运行通常在用户界面中执行的操作。启动 IBM SPSSModeler 时，只需在命令行中指定和运行一个独立脚本。例如：

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

-script 标记表示加载指定脚本，而 -execute 标记表示执行该脚本文件中的所有命令。

## 与先前版本的兼容性

在以前版本的 IBM SPSSModeler 中创建的脚本通常应该无需更改就可以在当前版本中运行。不过，模型块现在可以自动插入到流中（此为缺省设置），并可替代或补充流中此类型的现有模型块。此情况是否实际发生取决于**向流添加模型**和**替换先前模型**选项的设置（**工具 > 选项 > 用户选项 > 通知**）。例如，您可能需要修改以前版本中的脚本，在该版本中模型块替换是通过删除现有模型块并插入新的模型块来完成。

在当前版本中创建的脚本在以前的版本中可能无法正常运行。

如果在旧版本中创建的脚本使用了已被替换（或不被支持）的命令，那么使用旧形式命令的脚本仍然会得到支持，但将显示一条警告消息。例如，旧的 **generated** 关键字已被 **model** 替换，且 **clear generated** 已被 **clear generated palette** 替换。沿用旧形式的脚本仍然可以运行，但将显示一条警告消息。

## 访问流执行结果

许多 IBM SPSSModeler 节点生成输出对象，例如模型、图表和表格数据。其中许多输出包包含有用的值，脚本可以使用这些值来指导后续运行。这些值分组到内容容器（简称为容器）中，您可以通过用于识别各个容器的标记或标识来访问这些容器。访问这些值的方式取决于该容器所使用的格式或“内容模型”。

例如，许多预测模型输出使用称为 PMML 的 XML 变体来表示关于模型的信息，例如决策树在每个拆分点使用哪些字段，或者神经网络中的神经元如何连接以及以何种强度进行连接。使用 PMML 的模型输出提供了可用于访问信息的 XML 内容模型。例如：

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

# check how many contents are there and what are their names
tags = modeloutput.getContentModelTags()
```



```
print "Content Model Tags :" , tags

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")
if (cm != None) :
    # The PMML content model is a generic XML-based content model that
    # uses XPath syntax. Use that to find the names of the data fields.
    # The call returns a list of strings match the XPath values
    dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField",
    "name")
    print "Data Field Names:" , dataFieldNames
```

在脚本中，IBM SPSSModeler 支持下列内容模型：

- **表内容模型**用于访问表示为行和列的简单表格数据。
- **XML 内容模型**用于访问以 XML 格式存储的内容。
- **JSON 内容模型**用于访问以 JSON 格式存储的内容。
- **列统计内容模型**用于访问有关特定字段的汇总统计。
- **成对列统计内容模型**用于访问两个字段的汇总统计或者两个单独字段的值。

请注意，下列节点未包含这些内容模型：

- 时间序列
- 判别
- SLRM
- TCM
- 所有 Python 节点
- 所有 Spark 节点
- 所有数据库建模节点
- 扩展模型
- STP

表内容模型

表内容模型提供了一个简单的模型，用于访问简单的行数据和列数据。特定列中的值必须全都采用同一类型的存储（例如字符串或整数）。

API

表 25: API		
返回	Method	描述
int	getRowCount()	返回这个表中的行数。
int	getColumnCount()	返回这个表中的列数。
String	getColumnName(int columnIndex)	返回指定列索引处的列的名称。 列索引起始于 0。
StorageType	getStorageType(int columnIndex)	返回指定索引处的列的存储类型。 列索引起始于 0。
Object	getValueAt(int rowIndex, int columnIndex)	返回指定行索引和列索引处的值。 行索引和列索引起始于 0。
void	reset()	将任何与此内容模型相关联的内部存储器清空。

## 节点和输出

下表列出一些节点，这些节点将构建包含此类内容模型的输出。

表 26: 节点和输出		
节点名称	输出名称	容器标识
table	table	"table"

## 示例脚本

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

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

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

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

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

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

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

# For each column, print column name, type and the first row
# of values from the table content
col = 0
while col < tablecontent.getColumnCount():
    print tablecontent.getColumnName(col), \
          tablecontent.getStorageType(col), \
          tablecontent.getValueAt(0, col)
    col = col + 1
```

在脚本“调试”选项卡中，输出将类似于：

```
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
```

## XML 内容模型

XML 内容模型用于访问基于 XML 的内容。

XML 内容模型支持访问基于 XPath 表达式的组成部分。XPath 表达式是一些字符串，这些字符串定义调用者所需的元素或属性。XML 内容模型隐藏了构造各种对象以及编译表达式的细节，而 XPath 支持人员通常需要这些细节。这使得从 Python 脚本中进行调用更加简单。

XML 内容模型包含一个以字符串形式返回 XML 文档的函数。这使 Python 脚本用户能够使用其偏爱的 Python 库来解析 XML。

API

表 27: API		
返回	Method	描述
String	getXMLAsString()	以字符串形式返回 XML。
number	getNumericValue(String xpath)	返回评估路径的结果，返回类型为数字（例如，计算与路径表达式匹配的元素数目）。
boolean	getBooleanValue(String xpath)	返回对所指定路径表达式求值得到的布尔结果。
String	getStringValue(String xpath, String attribute)	返回与指定路径匹配的属性值或 XML 节点值。
List of strings	getStringValues(String xpath, String attribute)	返回一个列表，其中包含所有与指定路径匹配的属性值或 XML 节点值。
List of lists of strings	getValuesList(String xpath, <List of strings> attributes, boolean includeValue)	返回一个列表，其中包含所有与指定路径匹配的属性值，有需要时还包含 XML 节点值。
Hash table (key:string, value:list of string)	getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)	返回一个散列表，这个表使用键属性或 XML 节点值作为键，并使用一系列指定的属性值作为表值。
boolean	isNamespaceAware()	返回 XML 解析器是否应了解名称空间。缺省值为 False。
void	setNamespaceAware(boolean value)	设置 XML 解析器是否应了解名称空间。这还将调用 reset()，以确保后续调用采用所作的更改。
void	reset()	将任何与此内容模型相关联的内部存储器（例如缓存的 DOM 对象）清空。

节点和输出

下表列出一些节点，这些节点将构建包含此类内容模型的输出。

表 28: 节点和输出		
节点名称	输出名称	容器标识
Most model builders	Most generated models	"PMML "
"autodataprep"	n/a	"PMML "

## 示例脚本

用于访问内容的 Python 脚本代码可能如下所示：

```
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")
```

## JSON 内容模型

JSON 内容模型用于提供对 JSON 格式内容的支持。此模型提供了基本 API，用于在假定调用者知道所要访问的值的值的情况下，允许调用者抽取值。

### API

表 29: API		
返回	Method	描述
String	getJSONAsString()	以字符串形式返回 JSON 内容。
Object	getObjectAt(<List of object> path, JSONArtifact artifact) throws Exception	返回指定路径处的对象。提供的根工件可能为 Null，在这种情况下，将使用内容的根。返回的值可能是字符串、整数、实数或布尔值，或者是 JSON 工件（JSON 对象或 JSON 数组）。
Hash table (key:object, value:object)	getChildValuesAt(<List of object> path, JSONArtifact artifact) throws Exception	如果指定的路径指向 JSON 对象，那么返回该路径的子代值，否则返回 Null。表中的键是字符串，而相关联的值可以是字符串、整数、实数或布尔值，或者是 JSON 工件（JSON 对象或 JSON 数组）。
List of objects	getChildrenAt(<List of object> path path, JSONArtifact artifact) throws Exception	如果指定的路径指向 JSON 数组，那么返回该路径处的对象的列表，否则返回 Null。返回的值可能是字符串、整数、实数或布尔值，或者是 JSON 工件（JSON 对象或 JSON 数组）。
void	reset()	将任何与此内容模型相关联的内部存储器（例如缓存的 DOM 对象）清空。

### 节点和输出

下表列出一些节点，这些节点将构建包含此类内容模型的输出。

表 30: 节点和输出		
节点名称	输出名称	容器标识
"applykmeansas"	n/a	"JSON_MV"
"applyxgboosttree"	n/a	"JSON_MV"

## 示例脚本

以下脚本用于检索 JSON 文件：

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

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

## 列统计内容模型和成对统计内容模型

列统计内容模型用于访问那些可以为每个字段计算的统计（单变量统计）。成对统计内容模型用于访问可以在一对字段之间或者某个字段中的一对值之间计算的统计。

可能的统计度量如下所示：

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

某些值仅适用于单列统计，而其他值仅适用于成对统计。

生成这些值的节点包括：

- **“统计”**节点生成列统计，在指定了相关性字段时，还可以生成成对统计。
- **“数据审核”**节点生成列统计，在指定了覆盖字段时，还可以生成成对统计。
- **“均值”**节点在比较一对字段或者将某个字段的值与其他字段摘要进行比较时生成成对统计。

可用的内容模型和统计既取决于特定节点的功能，也取决于该节点内的设置。

## ColumnStatsContentModel API

表 31: ColumnStatsContentModel API		
返回	Method	描述
List<StatisticType>	getAvailableStatistics()	返回此模型中的可用统计。并非所有字段都必定具有用于所有统计的值。
List<String>	getAvailableColumns()	返回已计算其统计的列名。
Number	getStatistic(String column, StatisticType statistic)	返回与该列相关联的统计值。
void	reset()	将任何与此内容模型相关联的内部存储器清空。

## PairwiseStatsContentModel API

表 32: PairwiseStatsContentModel API		
返回	Method	描述
List<StatisticType>	getAvailableStatistics()	返回此模型中的可用统计。并非所有字段都必定具有用于所有统计的值。
List<String>	getAvailablePrimaryColumns()	返回已计算其统计的主列名。
List<Object>	getAvailablePrimaryValues()	返回已计算其统计的主列的值。
List<String>	getAvailableSecondaryColumns()	返回已计算其统计的辅助列名。
Number	getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)	返回与各个列相关联的统计值。
Number	getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)	返回与主列值和辅助列相关联的统计值。
void	reset()	将任何与此内容模型相关联的内部存储器清空。

## 节点和输出

下表列出一些节点，这些节点将构建包含此类内容模型的输出。

表 33: 节点和输出			
节点名称	输出名称	容器标识	注释
"means" (“均值”节点)	"means"	"columnStatistics"	

表 33: 节点和输出 (继续)			
节点名称	输出名称	容器标识	注释
"means" (“均值”节点)	"means"	"pairwiseStatistics"	
"dataaudit" (“数据审核”节点)	"means"	"columnStatistics"	
"statistics" (“统计”节点)	"statistics"	"columnStatistics"	仅当检查特定字段时才会生成。
"statistics" (“统计”节点)	"statistics"	"pairwiseStatistics"	仅当关联字段时才会生成。

## 示例脚本

```

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

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

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

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

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

```





# 第 6 章 命令行自变量

## 调用软件

您可以使用操作系统的命令行启动 "IBM SPSSModeler，如下所示。

### Microsoft Windows

1. 在安装了 IBM SPSSModeler 的计算机上，打开 DOS 或命令提示符窗口。
2. 切换到 IBM SPSSModeler 的安装路径（例如 [Installpath]\Program Files\IBM\SPSS\Modeler\19.0\bin）。
3. 要以交互模式启动 "IBM SPSSModeler 界面，请键入 "modelerclient 命令，然后输入所需的参数，例如

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

您可以使用可用参数（标志）连接服务器、加载数据流、运行脚本或根据需要指定其他参数。

### Mac OS

1. 找到 IBM SPSSModeler 的 Mac OS 命令路径（例如 [Installpath]/Applications/IBM/SPSS/Modeler/19.0/IBM SPSS Modeler.app/Contents/MacOS）。
2. 要以交互模式启动 "IBM SPSSModeler 界面，请运行建模器命令，然后输入所需的参数，例如

```
./modeler -stream report.str -execute
```

## 使用命令行参数

您可以将命令行参数（也称为标志）附加到初始 modelerclient 命令，以改变对 IBM SPSSModeler 的调用。

存在多种可用的命令行自变量类型，本节的随后内容将对其进行描述。

表 34: 命令行自变量的类型	
自变量类型	描述位置
系统自变量	请参阅主题第 56 页的『系统自变量』，了解更多信息。
参数自变量	请参阅主题第 57 页的『参数自变量』，了解更多信息。
服务器连接参数	请参阅主题第 58 页的『服务器连接参数』，了解更多信息。
IBM SPSS 协作与部署服务资源库 连接参数	有关更多信息，请参阅主题 第 59 页的『IBM SPSS 协作与部署服务资源库 连接参数』。
IBM SPSS 分析服务器 连接参数	有关更多信息，请参阅主题 第 59 页的『IBM SPSS 分析服务器 连接参数』。

例如，可以使用 -server、-stream 和 -execute 标记来连接到服务器，然后加载并运行流，如下所示：

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

请注意，当运行本地客户机安装程序时，不需要输入服务器连接参数。

可以用双引号括起包含空格的参数值，例如：

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

您还可以通过此方式，分别使用 `-state` 和 `-script` 标志来执行 IBM SPSSModeler 状态和脚本。

注: 如果在命令中使用结构化参数，那么必须在引号前面加上反斜杠。这将阻止在解释字符串期间移除该引号。

## 调试命令行自变量

要调试命令行，请使用 `modelerclient` 命令及所需自变量来启动 IBM SPSSModeler。这样可以验证命令是否可像期望中的那样运行。您还可以在“会话参数”对话框（“工具”菜单 -> “设置会话参数”）中对由命令行传递的参数进行确认。

## 系统自变量

下表描述可用于用户界面命令行调用的系统自变量。

表 35: 系统自变量	
自变量	行为/描述
@ <commandFile>	@ 符号后跟文件名，此文件用于指定命令列表。当 <code>modelerclient</code> 遇到以 @ 开头的参数时，它将在该文件中对命令进行操作，就如同在命令行中一样。有关更多信息，请参阅主题 <a href="#">第 60 页的『组合多个参数』</a> 。
-directory <dir>	设置缺省工作目录。在本地模式下，该目录将同时用于数据操作和输出。示例： <code>-directory c:/</code> 或 <code>-directory c:\\</code>
-server_directory <dir>	为数据设置缺省服务器目录。通过 <code>-directory</code> 标记指定的工作目录将用于输出。
-execute	在启动后执行启动时所加载的流、状态或脚本。如果在流或状态之外还加载了脚本，则脚本将单独执行。
-stream <stream>	启动时加载指定的流。可以指定多个流，但是最后一个指定的流将被设置为当前流。
-stream_password <password>	在启动时，装入已加密密码的流。例如，可以运行如下命令： <code>clemb.exe -stream enc-stream1.str -stream_password Pass1234 -execute</code>
-script <script>	启动时加载指定的独立脚本。如下所述，除流或状态之外，此标记还可用于指定脚本，但在启动时仅可加载一个脚本。
-model <model>	在启动时加载指定的已生成模型（.gm 格式的文件）。
-state <state>	在启动时，加载指定的已保存状态。
-project <project>	加载指定工程。在启动时仅可加载一个工程。
-output <output>	在启动时加载已保存的输出项目（.cou 格式的文件）。
-help	显示命令行自变量列表。指定此选项后，将忽略所有其它参数并显示帮助屏幕。
-P <name>=<value>	用于设置启动参数。还可用于设置节点属性（通道参数）。

注: 也可以在用户界面中设置缺省目录。要访问上述选项，请在“文件”菜单中选择[设置工作目录](#)或[设置服务器目录](#)。

## 加载多个文件

命令行模式下，您可以通过在启动时重复输入每个加载对象的相关参数来加载多个流、状态和输出。例如，要加载和运行两个称为 `report.str` 和 `train.str` 的流，您可以使用如下命令：

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

## 从 IBM SPSS 协作与部署服务资源库 加载对象

因为您可以从文件或 IBM SPSS 协作与部署服务资源库（如果已获得许可）装入某些对象，所以文件名前缀 `spsscr:` 和（可选）`file:`（对于磁盘上的对象）会告知 IBM SPSSModeler 在哪里查找对象。前缀可与以下标记配合使用：

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

您可以使用前缀创建 URI 以指定对象的位置，例如 `-stream "spsscr:///folder_1/scoring_stream.str"`。如果存在 `spsscr:` 前缀，就需要在同一命令中指定与 IBM SPSS 协作与部署服务资源库的有效连接。因此，完整的命令应形如以下的示例：

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

注意，在命令行中必须使用 URI。不支持像 `REPOSITORY_PATH` 这样的简单路径。（此种路径仅适用于脚本。）有关 IBM SPSS 协作与部署服务资源库中对象的 URI 的详细信息，请参阅第 43 页的『访问 IBM SPSS 协作与部署服务资源库中的对象』主题。

## 参数自变量

参数可用作在 IBM SPSSModeler 的命令行执行期间的标记。在命令行自变量中，`-P` 标志用于表示 `-P <name>=<value>` 格式的参数。

形式参数可以是：

- **简单参数**（即，直接在 CLEM 表达式中使用的参数）。
- **槽参数**，也称为节点属性。此类参数可用于修改流中节点的设置。请参阅主题第 63 页的『节点属性概述』，以获取更多信息。
- **命令行自变量**，用于更改对 IBM SPSSModeler 的调用。

例如，您可以提供数据源用户名和密码作为命令行标志，如下所示：

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

其格式与 `databasenode` 节点属性的 `datasource` 参数相同。有关更多信息，请参阅：第 79 页的『`databasenode` 属性』。

如果正在传递经过编码的密码，那么最后一个参数应设置为 `true`。另请注意，在数据库用户名和密码前面不应使用前导空格（除非，用户名或密码实际包含前导空格）。

**注：**如果指定此节点，那么必须将节点名括在双引号内，并使用反斜杠对引号进行转义。例如，如果上述示例中的数据源节点名为 `Source_ABC`，那么此条目如下所示：

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

用于标识结构化参数的引号前还需要有反斜杠，如以下 TM1 数据源示例中所示：

```
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%\"}
```

注: 如果 datasource 属性中的数据库名称包含一个或多个空格、句点（也称为“句号”）或下划线，那么您可以使用“反斜杠双引号”格式将其作为字符串进行处理。例如：“{\"db2v9.7.6\_linux\"}”或“{\"TDATA 131\"}”。此外，始终将 datasource 字符串值括在双引号和花括号中，如以下示例中所示：“{\"SQL Server\", spssuser, abcd1234, false}”。

## 服务器连接参数

-server 标志告知 IBM SPSSModeler 它应该连接到公用服务器，而 -hostname、-use\_ssl、-port、-username、-password 和 -domain 标志则用于告知 IBM SPSSModeler 如何连接到公用服务器。如果未指定 -server 参数，则使用缺省或本地服务器。

### 示例

连接到公共服务器：

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

连接到服务器集群：

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

请注意，连接到服务器集群需要通过 IBM SPSS 协作和部署服务的进程协调程序，因此必须将 -cluster 参数与存储库连接选项（spsscr\_\*）结合使用。请参阅主题第 59 页的『IBM SPSS 协作与部署服务资源库连接参数』，以获取更多信息。

表 36: 服务器连接参数	
自变量	行为/描述
-server	以服务器模式运行 IBM SPSSModeler，同时使用标志 -hostname、-port、-username、-password 和 -domain 连接到公共服务器。
-hostname <name>	服务器的主机名称。仅在服务器模式下可用。
-use_ssl	指定连接应采用 SSL（安全套接字层）。此标记为可选项，缺省设置为不使用 SSL。
-port <number>	指定服务器的端口号。仅在服务器模式下可用。
-cluster <name>	指定指向服务器集群（而不是已命名的服务器）的连接；此参数可用来替代 hostname、port 和 use_ssl 参数。名称为聚类名，或标识 IBM SPSS 协作与部署服务资源库中聚类的唯一 URI。服务器集群由 IBM SPSS 协作和部署服务中的过程协调器管理。有关更多信息，请参阅第 59 页的『IBM SPSS 协作与部署服务资源库连接参数』主题。
-username <name>	这是用于登录服务器的用户名。仅在服务器模式下可用。
-password <password>	这是用于登录服务器的密码。仅在服务器模式下可用。 注: 如果未使用 -password 参数，那么系统将提示您输入密码。

表 36: 服务器连接参数 (继续)	
自变量	行为/描述
-epassword <encodedpasswordstring>	这是用于登录服务器的经过编码的密码。 仅在服务器模式下可用。 注: 可以从 IBM SPSSModeler 应用程序的“工具”菜单中生成加密密码。
-domain <name>	这是用于登录到服务器的域。 仅在服务器模式下可用。
-P <name>=<value>	用于设置启动参数。 还可用于设置节点属性（通道参数）。

## IBM SPSS 协作与部署服务资源库 连接参数

如果想通过命令行来存储或检索 IBM SPSS 协作和部署服务 中的对象，则必须指定一个指向该 IBM SPSS 协作与部署服务资源库的有效连接。 例如：

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

下表列出了可用于建立连接的自变量。

表 37: IBM SPSS 协作与部署服务资源库 连接参数	
自变量	行为/描述
-spsscr_hostname <hostname or IP address>	安装 IBM SPSS 协作与部署服务资源库 的服务器的主机名或 IP 地址。
-spsscr_port <number>	IBM SPSS 协作与部署服务资源库 接受连接的端口号（通常，缺省值为 8080）。
-spsscr_use_ssl	指定连接应采用 SSL（安全套接字层）。 此标记为可选项，缺省设置为不使用 SSL。
-spsscr_username <name>	登录到 IBM SPSS 协作与部署服务资源库 的用户名。
-spsscr_password <password>	登录到 IBM SPSS 协作与部署服务资源库 的密码。
-spsscr_epassword <encoded password>	登录到 IBM SPSS 协作与部署服务资源库 的加密密码。
-spsscr_providername <name>	用于登录到 IBM SPSS 协作与部署服务资源库（Active Directory 或 LDAP）的认证服务提供程序。 如果使用本机（本地存储库）提供程序，那么这不是必需的。

## IBM SPSS 分析服务器 连接参数

如果要通过命令行存储或检索 IBM SPSS 分析服务器 中的对象，那么必须指定与 IBM SPSS 分析服务器的有效连接。

注: 分析服务器的缺省位置是从 SPSSModeler 服务器 获取的。 用户也可以通过工具 > **Analytic Server** 连接定义自己的分析服务器连接。

下表列出了可用于建立连接的自变量。

表 38: IBM SPSS 分析服务器 连接参数	
自变量	行为/描述
-analytic_server_username	用于登录 IBM SPSS 分析服务器 的用户名。
-analytic_server_password	用于登录 IBM SPSS 分析服务器 的密码。

表 38: IBM SPSS 分析服务器 连接参数 (继续)	
自变量	行为/描述
-analytic_server_epassword	用于登录 IBM SPSS 分析服务器 的经过编码的密码。
-analytic_server_credential	用于登录 IBM SPSS 分析服务器 的凭证。

## 组合多个参数

通过在文件名后使用 @ 标记，可以在调用时指定的命令文件中合并多个参数。这将使您可以缩短命令行调用，并且可以克服操作系统关于命令长度的限制。例如，以下 startup 命令使用 <commandFileName> 引用的文件中指定的参数。

```
modelerclient @<commandFileName>
```

如果需用空格，则请用引号将命令文件的文件名和路径括起来，如下所示：

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

命令文件中可以包含在之前启动中单独指定的所有参数，每行一个参数。例如：

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

当写入和引用命令文件时，必须遵循以下限制：

- 每条命令占用一行。
- 不要在命令文件中嵌入 @CommandFile 参数。

## 第 7 章 属性参考信息

### 属性参考概述

可以为节点、流、工程和超节点指定多个不同的属性。某些属性在所有节点中通用，例如“名称”、“注释”和“工具提示”，有些属性则只针对某些特定的节点类型。其它属性涉及高级流操作，例如高速缓存或“超节点”行为。可通过标准用户界面（例如当打开对话框编辑节点选项时）访问属性，还可以多种其它方式使用属性。

- 可通过脚本修改属性（如本章所述）。有关更多信息，请参阅第 61 页的『属性语法』。
- 可在“超节点”参数中应用节点属性。
- 在启动 IBM SPSSModeler 时，还可以将节点属性用作命令行选项的一部分（使用 -P 标志）。

在 IBM SPSSModeler 的脚本编写环境中，节点和流属性通常称为 **通道参数**。在本指南中，它们指的是节点或流的属性。

### 属性语法

可以使用下列语法设置的属性

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

或者：

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

可以使用下列语法检索的属性的值：

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

或者：

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

其中 OBJECT 是节点或输出，PROPERTY 是表达式引用的节点属性的名称，而 KEY 是键控属性的键值。例如，以下语法用于查找过滤节点，然后设置缺省值以包含所有字段并根据下游数据过滤 Age 字段：

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

IBM SPSSModeler 中使用的所有节点都可以使用流 `findByType(TYPE, LABEL)` 函数进行定位。必须至少指定一个 TYPE 或 LABEL。

### 结构化属性

脚本编写通过结构化属性，增强语法解析清晰度的方式有二：

- 指定复杂节点属性名称的结构，例如类型节点、过滤节点或平衡节点。
- 提供一种可一次指定多种属性的格式。

### 复杂接口的结构化

带有表和其它复杂接口的节点（类型、过滤和平衡节点）的脚本必须遵循特定的结构以便正确解析。这些属性需要的名称比单个标识的名称更复杂，此名称称为关键字。例如，过滤节点内，每个可用字段（在其上游）均处于开或关的状态。为了引用此信息，“过滤”节点为每个字段（无论此字段为 true 还是 false）存储

一个信息项。此属性的值可能为（或指定为）True 或 False。假设名为 mynode 的过滤节点（在其上游端）有一个名为 Age 的字段。要关闭此字段，请将具有关键字 Age 的属性 include 的值设置为 False，如下所示：

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

为设置多重属性而结构化

对于许多节点而言，您可以一次指定节点或流的多个属性。这称为**多重集合命令或设置块**。  
在某些情况下，结构化属性相当复杂。示例如下：

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

结构化属性的另一个优势在于，在某个节点稳定之前可以在该节点上设置若干个属性。缺省情况下，多重集合将在基于单个属性设置的操作运行之前，在块中设置所有属性。例如，如果在定义固定文件节点时分二阶设置字段属性，由于节点在两个设置均生效之前是不一致的，将会导致发生错误。以多重集合方式定义属性，可使得在更新数据模型前设置上述两个属性，从而避免上述问题发生。

缩写词

在节点属性语法中使用标准缩写。了解缩写有助于构建脚本。

表 39: 语法中使用的标准缩写	
缩写	含义
abs	绝对值
len	长度
最小值	最小值
最大值	最大值
correl	相关性
covar	协方差
num	数字或数值
pct	百分比
transp	透明度
xval	交叉验证
var	方差或变量（源节点中）

节点和流属性示例

在 IBM SPSSModeler 中可以各种方式使用节点和流属性。此类属性经常用作脚本的一部分，作为**独立脚本**的一部分用以实现多个流或操作的自动化；或用作**流脚本**的一部分用以实现单个流内部的过程自动化。还可通过在“超节点”内使用节点参数来指定节点参数。就最基础的水平而言，属性还可用作命令行选项来启动 IBM SPSSModeler。将 -p 参数用作命令行调用的一部分时，可以使用流属性来更改流设置。

表 40: 节点和流属性示例	
属性	含义
s.max_size	涉及到节点 s 的属性 max_size。



表 40: 节点和流属性示例 (继续)	
属性	含义
s:samplenode.max_size	涉及到节点 s 的属性 max_size, 其必须为样本节点。
:samplenode.max_size	涉及到当前流中样本节点的属性 max_size (只能有一个样本流)。
s:sample.max_size	涉及到节点 s 的属性 max_size, 其必须为样本节点。
t.direction.Age	涉及到“类型”节点 t 中年龄字段的角色。
:.max_size	*** 非法操作 *** 必须指定节点名或节点类型。

示例 s:sample.max\_size 说明不一定要写出节点类型的全称。

示例 t.direction.Age 说明, 当某个节点属性比带有单个值的单个通道复杂时, 某些通道名称将自行结构化。此类通道称为 **结构化** 或 **复杂** 属性。

## 节点属性概述

每种类型的节点均有其合法属性集, 每种属性也有其类型。该类型可以是常用类型数字、标志或字符串, 此时属性设置将强制至正确类型。如果无法进行强制转换, 那么将发生错误。另外, 通过属性引用可以指定合法值的范围, 例如 Discard、PairAndDiscard 和 IncludeAsText, 此时如果采用其它值, 则将出现错误。应通过采用值 true 或 false 来读取或设置标志属性。(设置值时也可识别如下变异值: Off、OFF、off、No、NO、no、n、N、f、F、false、False、FALSE 或 0, 但在某些情况下读取属性值时会出错。所有其他值都将被当作 true。使用 true 和 false 时保持一致将可以避免混淆。) 在本指南的参考表中, **属性描述**列对结构化属性进行了说明, 并给出了属性的使用格式。

## 通用节点属性

IBM SPSSModeler 中的很多属性通用于所有节点 (包括超节点)。

表 41: 公共节点属性		
属性名称	数据类型	属性描述
use_custom_name	标志	
name	string	读取工作区中某个节点名称的只读属性 (自动或自定义)。
custom_name	string	指定节点的自定义名称。
tooltip	string	
annotation	string	
keywords	string	指定与对象关联的关键字列表的结构化通道 (例如 ["Keyword1" "Keyword2"] )。
cache_enabled	标志	
node_type	source_supernode process_supernode terminal_supernode 所有为进行 脚本编制	按类型引用节点的只读属性。例如, 除按名称引用节点 (例如 real_income) 之外, 还可以指定类型 (例如 userinputnode 或 filternode)。

超节点属性以及所有其它节点的属性均将单独讨论。有关更多信息，请参阅主题 [第 399 页的『第 21 章 超节点属性』](#)。

# 第 8 章 流属性

通过脚本编写可以控制多种流属性。要引用流属性，必须将执行方法设置为使用脚本：

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

示例

node 属性用于引用当前流中的节点。如下所示的流脚本可作为一个示例：

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

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

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

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

此示例使用 node 属性创建了一个包含流中所有节点的列表，并将该列表写入流的注解中。生成的注解具有如下形式：

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

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

流属性的具体说明见于下表。

表 42: 流属性		
属性名称	数据类型	属性描述
execute_method	Normal	
	Script	

表 42: 流属性 (继续)

属性名称	数据类型	属性描述
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	成员	
date_2digit_baseline	成员	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	标志	
import_datetime_as_string	标志	
decimal_places	成员	
decimal_symbol	Default  Period  Comma	
angles_in_radians	标志	
use_max_set_size	标志	
max_set_size	成员	

表 42: 流属性 (继续)		
属性名称	数据类型	属性描述
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	标志	用于在流执行中自动刷新源节点。
script	string	
annotation	string	
name	string	注: 此属性为只读。如果想要更改流的名称, 您应该使用其它名称加以保存。
parameters		使用此属性可以从独立脚本中更新流参数。
nodes		详细信息参见下方。
encoding	SystemDefault "UTF-8"	
stream_rewriting	BOOLEAN	
stream_rewriting_maximise_sql	BOOLEAN	
stream_rewriting_optimise_cl em_ execution	BOOLEAN	
stream_rewriting_optimise_sy ntax_ execution	BOOLEAN	
enable_parallelism	BOOLEAN	
sql_generation	BOOLEAN	
database_caching	BOOLEAN	
sql_logging	BOOLEAN	
sql_generation_logging	BOOLEAN	
sql_log_native	BOOLEAN	
sql_log_prettyprint	BOOLEAN	
record_count_suppress_inp ut	BOOLEAN	
record_count_feedback_int erval	整数	
use_stream_auto_create_node_ settings	BOOLEAN	如果值为 true, 那么将使用特定于流的设置, 否则将使用用户首选项。

表 42: 流属性 (继续)

属性名称	数据类型	属性描述
<code>create_model_applier_for_new_models</code>	<i>BOOLEAN</i>	如果值为 <code>true</code> ，那么在模型构建器创建新模型并且没有处于活动状态的更新链接时，将添加一个新的模型应用器。  <b>注:</b> 如果您使用的是 IBM SPSSModeler 批次 V15，那么必须在脚本中显式地添加模型应用器。
<code>create_model_applier_update_links</code>	<code>createEnabled</code> <code>createDisabled</code> <code>doNotCreate</code>	定义自动添加模型应用器节点时创建的链接类型。
<code>create_source_node_from_builders</code>	<i>BOOLEAN</i>	如果值为 <code>true</code> ，那么在源构建器创建新的源输出并且没有处于活动状态的更新链接时，将添加一个新的源节点。
<code>create_source_node_update_links</code>	<code>createEnabled</code> <code>createDisabled</code> <code>doNotCreate</code>	定义自动添加源节点时创建的链接类型。
<code>has_coordinate_system</code>	<i>BOOLEAN</i>	如果设置为 <code>true</code> ，那么将坐标系应用于整个流。
<code>coordinate_system</code>	<i>string</i>	选择的投影坐标系的名称。
<code>deployment_area</code>	<code>ModelRefresh</code> <code>Scoring</code> <code>None</code>	选择流的部署方式。如果此值设置为 <code>None</code> ，那么不会使用其他部署条目。
<code>scoring_terminal_node_id</code>	<i>string</i>	选择流中的评分分支。它可以是流中的任何终端节点。
<code>scoring_node_id</code>	<i>string</i>	选择评分分支中的块。
<code>model_build_node_id</code>	<i>string</i>	选择流中的建模节点。

---

## 第 9 章 源节点属性

### 源节点公共属性

---

所有源节点的通用属性如下所示，后面的主题是具体节点的相关信息。

#### 示例 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")
```

#### 示例 2

此脚本假定指定的数据文件包含名为 Region 的字段，该字段表示多行字符串。

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

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

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

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

表 43: 源节点公共属性

属性名称	数据类型	属性描述
direction	Input Target Both None Partition Split Frequency RecordID	<p>字段角色的键控属性。</p> <p>用法格式：</p> <p><code>NODE.direction.FIELDNAME</code></p> <p><b>注：</b>现在不推荐使用值 In 和 Out 。在将来的版本中可能取消对这些值的支持。</p>
type	Range Flag Set Typeless Discrete Ordered Set Default	<p>字段类型。如果将该属性设置为 <i>Default</i>，那么将清除所有 <i>values</i> 属性设置，如果将 <i>value_mode</i> 设置为 <i>Specify</i>，那么它将重新设置为 <i>Read</i>。如果 <i>value_mode</i> 已设置为 <i>Pass</i> 或 <i>Read</i>，那么它将不受 <i>type</i> 设置的影响。</p> <p>用法格式：</p> <p><code>NODE.type.FIELDNAME</code></p>
storage	Unknown String Integer Real Time Date Timestamp	<p>字段存储类型的只读键控属性。</p> <p>用法格式：</p> <p><code>NODE.storage.FIELDNAME</code></p>



表 43: 源节点公共属性 (继续)		
属性名称	数据类型	属性描述
check	None  Nullify  Coerce  Discard  Warn  Abort	字段类型和范围检查的键控属性。  用法格式：  NODE.check.FIELDNAME
values	[value value]	对于连续型（范围）字段而言，第一个是最小值，后一个是最大值。对于名义（集合）字段，请指定所有值。对标志字段而言，第一个值代表 <i>false</i> ，后一个值代表 <i>true</i> 。设置该属性将自动把 <i>value_mode</i> 属性设置为 <i>Specify</i> 。存储器是根据列表中的第一个值确定的，例如，如果第一个值为字符串，那么存储器将设置为 String。  用法格式：  NODE.values.FIELDNAME
value_mode	Read  Pass  Read+  Current  Specify	确定下一次数据传递中设置某个字段值的方式。  用法格式：  NODE.value_mode.FIELDNAME  注意，不能将此属性直接设置为 <i>Specify</i> ；要使用特定值，需设置 <i>values</i> 属性。
default_value_mode	Read  Pass	指定用缺省方式设置所有字段值。  用法格式：  NODE.default_value_mode  该设置可以通过使用 <i>value_mode</i> 属性，用特定字段进行覆盖。
extend_values	标志	当 <i>value_mode</i> 设置为 <i>Read</i> 时将应用。设为 <i>T</i> 则将新读取的值添加到任意现有字段值。设置为 <i>F</i> 则丢弃现有值并添加新读取值。  用法格式：  NODE.extend_values.FIELDNAME

表 43: 源节点公共属性 (继续)		
属性名称	数据类型	属性描述
value_labels	<i>string</i>	用于指定值标签。 请注意，必须先指定值。
enable_missing	标志	<p>当设置为 <i>T</i> 时，则激活对字段缺失值的跟踪。</p> <p>用法格式：</p> <p>NODE.enable_missing.FIELDNAME</p>
missing_values	[ <i>value value ...</i> ]	<p>指定表示缺失数据的数据值。</p> <p>用法格式：</p> <p>NODE.missing_values.FIELDNAME</p>
range_missing	标志	<p>此属性设置为 <i>T</i> 时，指定是否为字段定义缺失值（空白）范围。</p> <p>用法格式：</p> <p>NODE.range_missing.FIELDNAME</p>
missing_lower	<i>string</i>	<p>当 range_missing 为真时，指定缺失值范围的下限。</p> <p>用法格式：</p> <p>NODE.missing_lower.FIELDNAME</p>
missing_upper	<i>string</i>	<p>当 range_missing 为真时，指定缺失值范围的上限。</p> <p>用法格式：</p> <p>NODE.missing_upper.FIELDNAME</p>
null_missing	标志	<p>当此属性设置为 <i>T</i> 时，将用空（在本软件中显示为 \$null\$ 的未定义值）表示缺失值。</p> <p>用法格式：</p> <p>NODE.null_missing.FIELDNAME</p>
whitespace_missing	标志	<p>当该属性设置为 <i>T</i> 时，仅包含空白（空格、制表符和换行符）的值将被当成缺失值。</p> <p>用法格式：</p> <p>NODE.whitespace_missing.FIELDNAME</p>
description	<i>string</i>	用于指定字段标签或描述。

表 43: 源节点公共属性 (继续)		
属性名称	数据类型	属性描述
default_include	标志	<p>用于指定缺省行为是传递还是过滤字段的键控属性：</p> <p>NODE.default_include</p> <p>示例：</p> <p>set mynode:filternode.default_include = false</p>
include	标志	<p>用于指出是包含还是过滤单个字段的键控属性：</p> <p>NODE.include.FIELDNAME.</p>
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	<p>这个键控属性与 type 的相似之处在于，它可以用于定义与字段相关联的测量。区别之处在于，在 Python 脚本编写中，还可以向 setter 函数传递其中一个 MeasureType 值，而 getter 将始终返回 MeasureType 值。</p>

表 43: 源节点公共属性 (继续)

属性名称	数据类型	属性描述
collection_measure	Range / MeasureType.RANGE  Flag / MeasureType.FLAG  Set / MeasureType.SET  OrderedSet / MeasureType.ORDERED_SET  Typeless / MeasureType.TYPELESS	对于集合字段（深度为 0 的列表），此键控属性定义与底层值相关联的测量类型。
geo_type	Point  MultiPoint  LineString  MultiLineString  Polygon  MultiPolygon	对于地理空间字段，此键控属性定义此字段所表示的地理空间对象的类型。这应该与这些值的列表深度一致。
has_coordinate_system	BOOLEAN	对于地理空间字段，此属性定义此字段是否具有坐标系
coordinate_system	string	对于地理空间字段，此键控属性定义此字段的坐标系。

表 43: 源节点公共属性 (继续)

属性名称	数据类型	属性描述
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	这个键控属性与 custom_storage 的相似之处在于，它可以用于定义字段的覆盖存储。区别之处在于，在 Python 脚本编写中，还可以向 setter 函数传递其中一个 StorageType 值，而 getter 将始终返回 StorageType 值。
custom_list_storage_type	String / MeasureType.STRING  Integer / MeasureType.INTEGER  Real / MeasureType.REAL  Time / MeasureType.TIME  Date / MeasureType.DATE  Timestamp / MeasureType.TIMESTAMP	对于列表字段，此键控属性指定底层值的存储类型。
custom_list_depth	整数	对于列表字段，此键控属性指定字段的深度。

表 43: 源节点公共属性 (继续)		
属性名称	数据类型	属性描述
max_list_length	整数	仅适用于测量级别为地理空间或集合的数据。通过指定列表可以包含的元素数目来设置列表的最大长度。
max_string_length	整数	仅适用于无类型数据，而且在生成 SQL 以创建表时使用。输入数据中最大字符串的值；这样会在表中生成一个足够大的列以包含该字符串。

## asimport 属性

分析服务器 源使您可以在 Hadoop 分布式文件系统 (HDFS) 上运行流。

### 示例

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

表 44: asimport 属性		
asimport 个属性	数据类型	属性描述
data_source	string	数据源名称。
use_default_as	BOOLEAN	如果设置为 True，那么将使用服务器 options.cfg 文件中配置的缺省 分析服务器 连接。如果设置为 False，请使用此节点的连接。
connection	["string", "string", "string", "string", "string", "string", "string", "string", "string", "string", "string"]	这是包含 分析服务器 连接详细信息的列表属性。格式为：["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"], 其中： is_secure_connect: 指示是否使用安全连接，并且是 true 还是 false。 use-kerberos-auth: 指示 是否使用 Kerberos 认证，其值为 true 或 false。 enable-kerberos-debug: 指示是否使用 Kerberos 认证的调试模式，其值为 true 或 false。

# cognosimport 节点属性



The IBM Cognos 源节点从 Cognos Analytics 数据库导入数据。

示例

```
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]"])
```

表 45: cognosimport 节点属性		
cognosimport 节点属性	数据类型	属性描述
mode	Data	指定是导入 Cognos 数据（缺省）还是报告。
	Report	

表 45: cognosimport 节点属性 (继续)

cognosimport 节点属性	数据类型	属性描述
cognos_connection	<code>["string",flag,"string", "string","string"]</code>	<p>列表属性，其中包含 Cognos 服务器的连接详细信息。格式为：<code>["Cognos_server_URL", login_mode, "namespace", "username", "password"]</code></p> <p>其中：</p> <p>Cognos_server_URL 是包含源的 Cognos 服务器的 URL。</p> <p>login_mode 指示是否使用匿名登录，并且为 <code>true</code> 或 <code>false</code>；如果设置为 <code>true</code>，那么应将以下字段设置为 <code>""</code>。</p> <p>namespace 指定用于登录服务器的安全认证提供程序。</p> <p>username 和 password 是用于登录 Cognos 服务器的用户名和密码。</p> <p>您还可以使用以下方式替代 login_mode：</p> <ul style="list-style-type: none"> <li>anonymousMode。例如： <code>['Cognos_server_url', 'anonymousMode', 'namespace', 'username', 'password']</code></li> <li>credentialMode。例如： <code>['Cognos_server_url', 'credentialMode', 'namespace', 'username', 'password']</code></li> <li>storedCredentialMode。例如： <code>['Cognos_server_url', 'storedCredentialMode', 'stored_credential_name']</code></li> </ul> <p>其中，stored_credential_name 是存储库中 Cognos 凭证的名称。</p>



表 45: *cognosimport* 节点属性 (继续)

<b>cognosimport</b> 节点属性	数据类型	属性描述
cognos_package_name	<i>string</i>	您要将数据对象导入其中的 Cognos 数据包的路径和名称，例如：  /Public Folders/GOSALES  注: 只有正斜杠有效。
cognos_items	<i>["field","field", ...,"field"]</i>	要导入的一个或多个数据对象的名称。 <i>field</i> 格式为 [namespace]. [query_subject].[query_item]
cognos_filters	<i>field</i>	导入数据前要应用的一个或多个过滤器的名称。
cognos_data_parameters	列表	数据的提示参数的值。“名称/值”对括在花括号内，并且多个对以逗号分隔，而整个字符串括在方括号内。  格式：  [["param1", "value"], ..., ["paramN", "value"]]
cognos_report_directory	<i>field</i>	要从中导入报告的 文件夹或包的 Cognos 路径，例如：  /Public Folders/GOSALES  注: 只有正斜杠有效。
cognos_report_name	<i>field</i>	要导入的报告的报告位置中的路径和名称。
cognos_report_parameters	列表	报告参数的值。“名称/值”对括在花括号内，并且多个对以逗号分隔，而整个字符串括在方括号内。  格式：  [["param1", "value"], ..., ["paramN", "value"]]

## databasenode 属性



“数据库”节点可用于使用 ODBC（开放数据库连接）从多种其他数据包中导入数据，这些数据包包括 Microsoft SQL Server、Db2 和 Oracle 等。

示例

```
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")

```

表 46: databasenode 属性

databasenode 个属性	数据类型	属性描述
mode	Table Query	借助对话框控件，指定将 <i>Table</i> 连接到数据库表，或借助 SQL 来指定用 <i>Query</i> 查询选定数据库。
datasource	string	数据库名称（另请参阅下面的注释）。
username	string	数据库连接详细信息（另请参阅下面的注释）。
password	string	
credential	string	IBM SPSS 协作和部署服务 中存储的凭证的名称。此属性可用于替代 <b>username</b> 和 <b>password</b> 属性。此凭证的用户名和密码必须与访问数据库所需的用户名和密码匹配。
use_credential		设置为 True 或 False。
epassword	string	指定一个加密密码作为脚本中硬加密密码的备选项。  有关更多信息，请参阅主题 第 45 页的『 <a href="#">生成经过编码的密码</a> 』。在执行过程中，此属性为只读。
tablename	string	要访问的表名称。
strip_spaces	None Left Right Both	丢弃字符串中前端和尾部空格的选项。
use_quotes	AsNeeded Always Never	指定在向数据库发送查询时是否用引号括起表和列的名称（例如在包含空格或标点的情况下）。
query	string	指定要提交查询所对应的 SQL 编码。

注: 如果 **datasource** 属性中的数据库名称包含空格，那么您还可以使用以下格式的单个数据源属性，以代替使用 **datasource**、**username** 和 **password** 的各个属性：

表 47: databasenode 属性 - 特定于数据源

databasenode 个属性	数据类型	属性描述
datasource	string	<p>格式：</p> <p>[database_name,username,password[,true   false]]</p> <p>最后一个参数与经过加密的密码配合使用。如果将其设为 true，将会在使用之前对密码进行解密。</p>

如果您要更改数据源，也可使用此格式；不过，如果您只想更改用户名或密码，则可使用 username 或 password 属性。

## datacollectionimportnode 属性



数据收集 数据导入节点根据市场研究产品所使用的 数据收集 数据模型来导入调查数据。 必须安装 数据收集 数据库才可使用此节点。

示例

```
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")
```

表 48: <i>datacollectionimportnode</i> 属性		
<b>datacollectionimportnode</b> 个属性	数据类型	属性描述
metadata_name	<i>string</i>	<p>MDSC 的名称。特殊值 DimensionsMDD 表示应使用标准 数据收集 元数据文档。 其他可能的值包括：</p> <p>mrADODsc</p> <p>mrI2dDsc</p> <p>mrLogDsc</p> <p>mrQdiDrsDsc</p> <p>mrQvDsc</p> <p>mrSampleReportingMDSC</p> <p>mrSavDsc</p> <p>mrSCDsc</p> <p>mrScriptMDSC</p> <p>特殊值 none 指示不存在 MDSC。</p>
metadata_file	<i>string</i>	存储元数据的文件的名称。

表 48: datacollectionimportnode 属性 (继续)		
datacollectionimportnode 个属性	数据类型	属性描述
casedata_name	string	CDSC 的名称。可能的值包括：  mrADODsc  mrI2dDsc  mrLogDsc  mrPunchDSC  mrQdiDrsDsc  mrQvDsc  mrRdbDsc2  mrSavDsc  mrScDSC  mrXmlDsc  特殊值 none 指示不存在 CDSC。
casedata_source_type	Unknown  File  Folder  UDL  DSN	指出 CDSC 的源类型。
casedata_file	string	当 casedata_source_type 为 <i>File</i> 时， 则指定包含观测值数据的文件。
casedata_folder	string	当 casedata_source_type 为 <i>Foder</i> 时，则指定包含观测值数据的文件夹。
casedata_udl_string	string	当 casedata_source_type 为 <i>UDL</i> 时， 则为包含观测值数据的数据源指定 OLD-DB 连接字符串。
casedata_dsn_string	string	当 casedata_source_type 为 <i>DSN</i> ，则 为数据源指定 ODBC 连接字符串。
casedata_project	string	从 数据收集 数据库中读取观测值数据时， 可以输入工程的名称。对于所有其他的观 测值数据类型，应将此设置留空。

表 48: datacollectionimportnode 属性 (继续)

datacollectionimportnode 个属性	数据类型	属性描述
version_import_mode	All Latest Specify	定义版本处理方式。
specific_version	string	当 version_import_mode 为 Specify 时，则定义要导入观测值数据的版本。
use_language	string	定义是否应使用指定语言的标签。
language	string	如果 use_language 的值为 True，则定义导入时要使用的语言代码。语言代码应为观测值数据中的某一可用代码。
use_context	string	定义是否应导入特定的上下文。环境可用于区分与响应相关的描述。
context	string	如果 use_context 的值为真，则定义导入环境。环境应是观测值数据中的某一可用环境。
use_label_type	string	定义是否应导入指定标签类型。
label_type	string	如果 use_label_type 的值为真，则定义要导入的标签类型。标签类型应是观测值数据中的某一可用标签类型。
user_id	string	对于要求显式登录的数据库，可通过提供用户标识和密码来访问数据源。
password	string	
import_system_variables	Common None All	指定要导入哪些系统变量。
import_codes_variables	标志	
import_sourcefile_variables	标志	
import_multi_response	MultipleFlags Single	

## excelimportnode 属性



Excel 导入节点可从 Microsoft Excel 以 .xlsx 文件格式导入数据。不要求指定 ODBC 数据源。

## 示例

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

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

表 49: *excelimportnode* 属性

<b>excelimportnode</b> 个属性	数据类型	属性描述
excel_file_type	Excel2007	
full_filename	<i>string</i>	完整文件名（包括路径）。
use_named_range	<i>BOOLEAN</i>	是否使用指定范围。如果为真，则将用 <b>named_range</b> 属性来指定读取范围，但忽略其它工作表和数据范围设置。
named_range	<i>string</i>	
worksheet_mode	Index Name	指定是否通过索引或名称来定义工作表。
worksheet_index	整数	要读取工作表的索引，开始时第一个工作表的索引为 0，第二个工作表的索引为 1，依此类推。
worksheet_name	<i>string</i>	要读取工作表的名称。
data_range_mode	FirstNonBlank ExplicitRange	指定确定范围的方式。
blank_rows	StopReading ReturnBlankRows	当 <b>data_range_mode</b> 为 <i>FirstNonBlank</i> 时，指定空行处理方式。
explicit_range_start	<i>string</i>	当 <b>data_range_mode</b> 为 <i>ExplicitRange</i> 时，指定要读取范围的起点。
explicit_range_end	<i>string</i>	
read_field_names	<i>BOOLEAN</i>	指定是否应将指定范围的第一行用作字段（列）名称。
scanLineCount	整数	指定要扫描的列和存储类型的行数。缺省值是 200。



通过“扩展导入”节点，您可以运行 R 或 Python for Spark 脚本来导入数据。

### Python for Spark 示例

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

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

cxt = spss.pyspark.runtime.getContext()

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

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

    cxt.setSparkOutputData(df)
"""

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

### R 示例

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

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



```

    require(x,character.only=TRUE)
  }
}
# packages
packages(RJSONIO)
packages(plyr)
### This function is used to generate automatically the dataModel
getMetaData <- function(data) {
  if (dim(data)[1]<=0) {

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

  } else {

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

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

```

表 50: *extensionimportnode* 属性

<b>extensionimportnode</b> 个属性	数据类型	属性描述
syntax_type	<i>R</i> <i>Python</i>	指定运行哪个脚本 - R 还是 Python（R 是缺省值）。
r_syntax	<i>string</i>	要运行的 R 脚本编写语法。
python_syntax	<i>string</i>	要运行的 Python 脚本编写语法。

## fixedfilenode 属性



固定文件节点会从固定字段文本文件（即文件字段不定界，而是从相同的位置开始且长度固定）中导入数据。机器生成的数据或遗传数据通常以固定字段格式存储。

示例

```
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)
```

表 51: *fixedfilenode* 属性

<b>fixedfilenode</b> 个属性	数据类型	属性描述
record_len	成员	指定每条记录中的字符数。
line_oriented	标志	跳过每条记录尾部的换行符。
decimal_symbol	Default Comma Period	用于数据源中的十进制分隔符的类型。
skip_header	成员	指定每条记录开头要忽略的行数。用于忽略列标题。
auto_recognize_datetime	标志	指定在源数据中是否自动标识日期或时间。
lines_to_scan	成员	
fields	列表	结构化属性。
full_filename	<i>string</i>	要读取文件的全称（包括目录）。

表 51: <i>fixedfilenode</i> 属性 (继续)		
<b>fixedfilenode</b> 个属性	数据类型	属性描述
strip_spaces	None  Left  Right  Both	在导入时丢弃字符串中前端和尾部的空格。
invalid_char_mode	Discard  Replace	从数据输入中移除无效字符（空值、0 或当前编码中所没有的字符），或用指定的单字符符号替换无效字符。
invalid_char_replacement	<i>string</i>	
use_custom_values	标志	
custom_storage	Unknown  String  Integer  Real  Time  Date  Timestamp	

表 51: *fixedfilenode* 属性 (继续)

<b>fixedfilenode</b> 个属性	数据类型	属性描述
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	此属性仅在指定自定义存储器后适用。

表 51: <i>fixedfilenode</i> 属性 (继续)		
<b>fixedfilenode</b> 个属性	数据类型	属性描述
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	此属性仅在指定自定义存储器后适用。

表 51: <i>fixedfilenode</i> 属性 (继续)		
<b>fixedfilenode</b> 个属性	数据类型	属性描述
custom_decimal_symbol	<i>field</i>	只有在指定了定制存储器的情况下才适用。
encoding	StreamDefault SystemDefault "UTF-8"	指定文本编码方法。

## gsdata\_import 节点属性



您可以使用“地理空间”源节点将地图或空间数据引入到数据挖掘会话中。

表 52: <i>gsdata_import</i> 节点属性		
<b>gsdata_import</b> 节点属性	数据类型	属性描述
full_filename	<i>string</i>	请输入要加载的 .shp 文件的文件路径。
map_service_URL	<i>string</i>	请输入要连接到地图服务 URL。
map_name	<i>string</i>	仅当使用了 map_service_URL 时，此属性才有效，并且包含地图服务的顶级文件夹结构。

## jsonimportnode 属性



JSON 源节点从 JSON 文件导入数据。

表 53: <i>jsonimportnode</i> 属性		
<b>jsonimportnode</b> 个属性	数据类型	属性描述
full_filename	<i>string</i>	完整文件名（包括路径）。
string_format	<i>records</i> 值	指定 JSON 字符串格式。缺省值为 records。
auto_label		在 V18.2.1.1 中新增。

## sasimportnode 属性



SAS 导入节点可将 SAS 数据导入到 IBM SPSSModeler 中。

示例

```
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)
```

表 54: sasimportnode 属性		
sasimportnode 个属性	数据类型	属性描述
format	Windows  UNIX  Transport  SAS7  SAS8  SAS9	要导入文件的格式。
full_filename	string	输入的完整文件名（包括路径）。
member_name	string	指定要从特定 SAS 传输文件中导入的成员。
read_formats	标志	从指定格式文件中读取数据格式（例如变量标签）。
full_format_filename	string	
import_names	NamesAndLabels  LabelsasNames	指定在导入时映射变量名称和标签的方法。

simgennode 属性



“模拟生成”节点提供了一种生成模拟数据的简单方法 - 使用用户指定的统计分布从头开始生成数据，或者使用对现有历史数据运行“模拟拟合”节点而获取的分布自动生成数据。当您想要在模型输入存在不确定性的情况下评估预测模型的结果时，这十分有用。

表 55: simgennode 属性		
simgennode 个属性	数据类型	属性描述
fields	结构化属性	请参阅示例
correlations	结构化属性	请参阅示例
keep_min_max_setting	BOOLEAN	
refit_correlations	BOOLEAN	
max_cases	整数	最小值为 1000，最大值为 2,147,483,647

表 55: *simgennode* 属性 (继续)

<b>simgennode</b> 个属性	数据类型	属性描述
create_iteration_field	<i>BOOLEAN</i>	
iteration_field_name	<i>string</i>	
replicate_results	<i>BOOLEAN</i>	
random_seed	整数	
parameter_xml	<i>string</i>	以字符串形式返回参数 XML

## fields 示例

这是结构化槽参数，其语法如下：

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

**distribution** 是分布名称的声明，此名称后跟包含属性名称/值对的列表。每项分布都以如下方式定义：

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

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

例如，要创建用于生成具有二项式分布的单个字段的节点，您可以使用以下脚本：

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

二项式分布使用两个参数：**n** 和 **prob**。由于二项式分布不支持最小值和最大值，因此这两个参数将作为空字符串提供。

**注：**您不能直接设置 **distribution**；可以将其与 **fields** 属性一起使用。

以下示例显示所有可能的分发类型。请注意，阈值在 **NegativeBinomialFailures** 和 **NegativeBinomialTrial** 中均输入为 **thresh**。

```
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
])
```

## 相关示例

这是结构化槽参数，其语法如下：

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

相关性可以是介于 +1 与 -1 之间的任何数字。您可以根据需要指定相关性。任何未指定的相关性都将设置为 0。如果存在任何未知字段，那么应该在相关性矩阵（或表）上设置相关性值，并以红色文本显示该值。如果存在未知字段，那么无法执行节点。

## statisticsimportnode 属性



IBM SPSS Statistics 文件节点从 IBM SPSS Statistics 使用的 .sav 文件格式以及保存在 IBM SPSSModeler 中的缓存文件（其也使用相同格式）读取数据。

有关此节点属性的信息，请参阅第 377 页的『statisticsimportnode 属性』。

## tm1odataimport 节点属性



IBM Cognos TM1 源节点从 Cognos TM1 数据库导入数据。

表 56: tm1odataimport 节点属性		
tm1odataimport 节点属性	数据类型	属性描述
credential_type	inputCredential 或 storedCredential	用于指示凭证类型。
input_credential	列表	当 credential_type 为 inputCredential 时；指定域、用户名和密码。
stored_credential_name	string	在 credential_type 为 storedCredential 时，指定 C&DS 服务器上凭证的名称。
selected_view	["field" "field"]	列表属性，其中包含所选 TM1 多维数据集的详细信息以及要从中将数据导入到 SPSS 的多维数据集视图的名称。 TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input']) 例如：
is_private_view	标志	指定 selected_view 是否是私有视图。缺省值为 false。

表 56: *tm1odataimport* 节点属性 (继续)

<b>tm1odataimport</b> 节点属性	数据类型	属性描述
selected_columns	<code>["field"]</code>	指定所选列；只能指定一个项。 例如： <code>setProperty("selected_columns", ["Measures"])</code>
selected_rows	<code>["field" "field"]</code>	指定所选行。 例如： <code>setProperty("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])</code>
connection_type	AdminServer TM1Server	指示连接类型。缺省值为 AdminServer。
admin_host	<code>string</code>	REST API 的主机名的 URL。如果 <code>connection_type</code> 是 AdminServer，则此属性是必需的。
server_name	<code>string</code>	从 <code>admin_host</code> 中选择的 TM1 服务器的名称。如果 <code>connection_type</code> 是 AdminServer，则此属性是必需的。
server_url	<code>string</code>	TM1 服务器 REST API 的 URL。如果 <code>connection_type</code> 是 TM1Server，则此属性是必需的。

## tm1import 节点属性 (不推荐)



IBM Cognos TM1 源节点从 Cognos TM1 数据库导入数据。

注: 该节点在 Modeler 18.0 中不推荐使用。替换节点脚本名称是 *tm1odataimport*。

表 57: *tm1import* 节点属性

<b>tm1import</b> 节点属性	数据类型	属性描述
pm_host	<code>string</code>	注: 仅适用于 V16.0 和 V17.0。  主机名。 <code>TM1_import.setProperty("pm_host", 'http://9.191.86.82:9510/pmhub/pm')</code> 例如:
tm1_connection	<code>["field","field", ...,"field"]</code>	注: 仅适用于 V16.0 和 V17.0。  列表属性，其中包含 TM1 服务器的连接详细信息。格式为: <code>["TM1_Server_Name","tm1_username","tm1_password"]</code>  <code>TM1_import.setProperty("tm1_connection", ['Planning Sample', "admin", "apple"])</code> 例如:

表 57: *tm1import* 节点属性 (继续)

tm1import 节点属性	数据类型	属性描述
selected_view	<code>["field" "field"]</code>	列表属性，其中包含所选 TM1 多维数据集的详细信息以及要从中将数据导入到 SPSS 的多维数据集视图的名称。 TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input']) 例如：
selected_column	<code>["field"]</code>	指定所选列；只能指定一个项。  setPropertyValue("selected_columns", ["Measures"]) 例如：
selected_rows	<code>["field" "field"]</code>	指定所选行。  setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"]) 例如：

## twcimport 节点属性



TWC 源节点从 The Weather Company, an IBM Business 导入天气数据。您可以使用它来获取某一位置的历史或预测天气。这可以帮助制定天气驱动型业务解决方案，以使用可用的最准确天气数据做出更明智的决定。

表 58: *twcimport* 节点属性

twcimport 节点属性	数据类型	属性描述
TWCDataImport.latitude	REAL	以格式 [-90.0~90.0] 指定纬度值
TWCDataImport.longitude	REAL	以格式 [-180.0~180.0] 指定经度值。
TWCDataImport.licenseKey	string	指定从 The Weather Company 获取的许可证密钥。
TWCDataImport.measurementUnit	English Metric Hybrid	指定度量单位。可能的值为 English、Metric 或 Hybrid。Metric 是缺省值。
TWCDataImport.dataType	Historical Forecast	指定要输入的天气数据类型。可能的值是 Historical 或 Forecast。Historical 是缺省值。
TWCDataImport.startDate	整数	如果为 TWCDataImport.dataType 指定 Historical，则以格式 yyyyMMdd 指定开始日期。
TWCDataImport.endDate	整数	如果为 TWCDataImport.dataType 指定 Historical，则以格式 yyyyMMdd 指定结束日期。

表 58: twcimport 节点属性 (继续)

twcimport 节点属性	数据类型	属性描述
TwCDataImport.forecastHour	6 12 24 48	如果为 TwCDataImport.dataType 指定了 Forecast, 则指定 6、12、24 或 48 作为小时值。

## userinputnode 属性



用户输入节点提供了一种用于创建综合数据的简单方式 - 可以从头开始创建也可以通过更改现有数据进行创建。此节点非常有用, 例如, 在希望为建模创建测试数据集时, 即可使用此节点。

### 示例

```
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")
```

表 59: userinputnode 属性

userinputnode 个属性	数据类型	属性描述
data		
names		设置或返回节点所生成的字段名称列表的结构化通道。
custom_storage	Unknown String Integer Real Time Date Timestamp	可用于设置或返回某个字段存储的通道。

表 59: <i>userinputnode</i> 属性 (继续)		
<b>userinputnode</b> 个属性	数据类型	属性描述
data_mode	Combined  Ordered	如果指定了 Combined, 那么设定值以及最小/最大值的每个组合都将生成一个记录。生成的记录数等于每个字段中值的数量的乘积。如果指定了 Ordered, 那么将从每个记录的每一列中获取一个值, 以便生成一行数据。生成的记录数等于一个与字段相关的最大数值。 将为所有数据值较少的字段添加空值。
values		注: 此属性已由 <i>userinputnode.data</i> 取代, 不应继续使用。

## variablefilenode 属性



自由格式文件节点读取自由格式字段文本文件中的数据, 即, 其记录包含固定数量的字段, 但包含不定数量字符的文件。此节点对于具有固定长度标题文本和某些特定类型注解的文件也非常有用。

示例

```
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])
```

表 60: <i>variablefilenode</i> 属性		
<b>variablefilenode</b> 个属性	数据类型	属性描述
skip_header	成员	指定每条记录开头要忽略的字符数。
num_fields_auto	标志	自动确定每条记录中的字段数。记录必须以换行符终止。
num_fields	成员	手动指定每条记录中的字段数。
delimit_space	标志	指定文件中用于划定字段边界的字符。
delimit_tab	标志	
delimit_new_line	标志	
delimit_non_printing	标志	
delimit_comma	标志	在逗号既是字段分隔符又是流的小数点分隔符的情况下, 请将 <i>delimit_other</i> 设置为 <i>true</i> , 并使用 <i>other</i> 属性指定逗号作为分隔符。

表 60: *variablefilenode* 属性 (继续)

<b>variablefilenode</b> 个属性	数据类型	属性描述
<code>delimit_other</code>	标志	允许您使用 <code>other</code> 属性指定定制分隔符。
<code>other</code>	<i>string</i>	在 <code>delimit_other</code> 为 <i>true</i> 时，指定要使用的定界符。
<code>decimal_symbol</code>	Default Comma Period	指定用于数据源中的十进制分隔符。
<code>multi_blank</code>	标志	将多个相邻空格定界符视为一个单一定界符处理。
<code>read_field_names</code>	标志	将数据文件的第一行作为列的标签。
<code>strip_spaces</code>	None Left Right Both	在导入时丢弃字符串中前端和尾部的空格。
<code>invalid_char_mode</code>	Discard Replace	从数据输入中移除无效字符（空值、0 或当前编码中所没有的字符），或用指定的单字符符号替换无效字符。
<code>invalid_char_replacement</code>	<i>string</i>	
<code>break_case_by_newline</code>	标志	指定行定界符为换行符。
<code>lines_to_scan</code>	成员	指定具体数据类型的扫描行数。
<code>auto_recognize_datetime</code>	标志	指定在源数据中是否自动标识日期或时间。
<code>quotes_1</code>	Discard PairAndDiscard IncludeAsText	指定导入后单引号的处理方式。
<code>quotes_2</code>	Discard PairAndDiscard IncludeAsText	指定导入后双引号的处理方式。
<code>full_filename</code>	<i>string</i>	要读取的文件全称（包括目录）。
<code>use_custom_values</code>	标志	

表 60: variablefilenode 属性 (继续)		
variablefilenode 个属性	数据类型	属性描述
custom_storage	Unknown  String  Integer  Real  Time  Date  Timestamp	

表 60: variablefilenode 属性 (继续)

variablefilenode 个属性	数据类型	属性描述
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	只有在指定了定制存储器的情况下才适用。



表 60: variablefilenode 属性 (继续)		
variablefilenode 个属性	数据类型	属性描述
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	只有在指定了定制存储器的情况下才适用。

表 60: variablefilenode 属性 (继续)

variablefilenode 个属性	数据类型	属性描述
custom_decimal_symbol	field	只有在指定了定制存储器的情况下才适用。
encoding	StreamDefault SystemDefault "UTF-8"	指定文本编码方法。

## xmlimportnode 属性



“XML 源”节点将 XML 格式的数据导入到流中。可以导入单个文件，也可以导入某个目录中的所有文件。您可以选择性地指定模式文件，以便从中读取 XML 结构。

示例

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

表 61: xmlimportnode 属性

xmlimportnode 个属性	数据类型	属性描述
read	single directory	读取单个数据文件（缺省），或目录中的所有 XML 文件。
recurse	标志	指定是否另外读取指定目录的所有子目录中的 XML 文件。
full_filename	string	（必需）要导入的 XML 文件的完整路径和文件名（如果 read = single）。
directory_name	string	（必需）要从中导入 XML 文件的目录的完整路径和名称（如果 read = directory）。
full_schema_filename	string	要从中读取 XML 结构的 XSD 或 DTD 文件的完整路径和文件名。如果您省略了此参数，将从 XML 源文件中读取结构。
records	string	XPath 表达式（例如，/author/name），用以定义记录边界。每次在源文件中遇到此元素时，都将创建新的记录。
mode	read specify	读取所有数据（缺省），或指定要读取的项目。
fields		要导入的项目（元素和属性）列表。列表中的每项为 XPath 表达式。

# 第 10 章 记录操作节点属性

## appendnode 属性



“追加”节点用于连接多组记录。另外，也可以用于将结构类似但内容不同的数据集组合到一起。

示例

```
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")
```

表 62: appendnode 属性

appendnode 个属性	数据类型	属性描述
match_by	Position Name	可以根据字段在主数据源中的位置或输入数据集中字段的名称来附加数据集。
match_case	标志	匹配字段名称时启用区分大小写。
include_fields_from	Main All	
create_tag_field	标志	
tag_field_name	string	

## aggregatenode 属性



“汇总”节点将一系列输入记录替换为经过摘要和汇总的输出记录。

示例

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
node.setPropertyValue("inc_record_count", True)
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

表 63: aggregatenode 属性

aggregatenode 个属性	数据类型	属性描述
keys	列表	列出可用作汇总键的字段。例如，如果 Sex 和 Region 是键字段，那么具有区域 N 和 S 的 M 和 F 的每个唯一组合（四个唯一组合）都将具有一条汇总记录。
contiguous	标志	如果您知道输入中具有相同键值的所有记录都分为一组，那么可以选择此选项（例如，如果对关键字段上的输入进行排序）。这样做有助于提高性能。
aggregates		一种结构化属性，它列出其值将进行汇总的数字字段以及所选的汇总模式。
aggregate_exprs		键控属性，输入派生字段名称以及用于计算此名称的汇总表达式。例如： <pre>aggregatenode.setKeyedPropertyValue("aggregate_exprs", "Na_MAX", "MAX('Na')")</pre>
extension	string	为重复汇总字段指定前缀或后缀（样本如下）。
add_as	Suffix Prefix	
inc_record_count	标志	创建一个额外字段，该字段指定为形成每条汇总记录汇总了多少条输入记录。
count_field	string	指定记录计数字段的名称。
allow_approximation	BOOLEAN	在分析服务器中执行汇总时允许进行近似顺序统计。
bin_count	整数	指定要在近似统计中使用的分级数

## balancenode 属性



“均衡”节点用于纠正数据集中的不平衡，以使其遵循指定的条件。“均衡”伪指令根据指定系数调整条件成立的记录所占的比例。

示例

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

表 64: balancenode 属性

balancenode 个属性	数据类型	属性描述
directives		根据指定数字平衡字段值比例的结构化属性（参阅下面的示例）。

表 64: <i>balancenode</i> 属性 (继续)		
<b>balancenode</b> 个属性	数据类型	属性描述
training_data_only	标志	指定仅平衡训练数据。如果流中不存在分区字段，则忽略该选项。

此节点属性使用以下格式：

`[[ number, string ] \ [ number, string ] \ ... [ number, string ] ]`。

注: 如果在表达式中嵌入了字符串（使用双引号），那么必须在这些字符串前面添加转义字符 " \ "。" \ " 字符也是行连续字符，为清晰起见，您可以使用该字符使自变量保持一致。

## cplexoptnode 属性



借助 CPLEX Optimization 节点，可以通过优化编程语言 (OPL) 模型文件来使用基于优化的复杂数学。此功能可在不再受支持的 IBM Analytical Decision Management 产品中使用。但是，您也可以使用 SPSS 建模器 中使用 CPLEX 节点，而无需 IBM Analytical Decision Management。

表 65: <i>cplexoptnode</i> 属性		
<b>cplexoptnode</b> 个属性	数据类型	属性描述
opl_model_text	<i>string</i>	将由 CPLEX Optimization 节点运行并生成优化结果的 OPL（优化编程语言）脚本程序。
opl_tuple_set_name	<i>string</i>	这是 OPL 模型中对应于传入数据的元组集合名称。这不是必需的，并且通常未通过脚本进行设置。它应仅用于编辑所选数据源的字段映射。
data_input_map	结构化属性的列表	数据源的输入字段映射。这不是必需的，并且通常未通过脚本进行设置。它应仅用于编辑所选数据源的字段映射。

表 65: cplexoptnode 属性 (继续)

cplexoptnode 个属性	数据类型	属性描述
md_data_input_map	结构化属性的列表	<p>OPL 中定义的包含每个对应字段数据源（入局数据）的每个元组之间的字段映射。用户可以逐个数据源单独对其进行编辑。通过此脚本，可以直接设置属性以一次性设置所有映射。此设置未显示在用户界面中。</p> <p>列表中的每个实体都是结构化数据：</p> <p><b>数据源标记。</b> 可在数据源下拉列表中找到数据源的标记。例如，对于 0_Products_Type，标记为 0。</p> <p><b>数据源索引。</b> 数据源的物理序列（索引）。这由连接顺序确定。</p> <p><b>源节点。</b> 数据源的源节点（注释）。这可以在数据源下拉列表中找到。例如，对于 0_Products_Type，源节点为 Products。</p> <p><b>连接节点。</b> 用于连接当前 CPLEX Optimization 节点的先验节点（注释）。这可以在数据源下拉列表中找到。例如，对于 0_Products_Type，已连接节点为 Type。</p> <p><b>元组集合名称。</b> 数据源的元组集合名称。它必须与 OPL 中定义的内容匹配。</p> <p><b>元组字段名称。</b> 数据源的元组集合字段名称。它必须与 OPL 元组集合定义中定义的内容匹配。</p> <p><b>存储类型。</b> 字段存储类型。可能的值为 int、float 或 string。</p>
		<p><b>数据字段名称。</b> 数据源的字段名称。</p> <p><b>示例：</b></p> <pre>[[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']]</pre>
opl_data_text	string	用于 OPL 的一些变量或数据的定义。
output_value_mode	string	可能的值为 raw 或 dvar。如果指定了 dvar，那么用户必须在“输出”选项卡上指定 OPL 中用于输出的对象函数变量名称。如果指定了 raw，那么将直接输出目标函数，而与名称无关。

表 65: cplexoptnode 属性 (继续)		
cplexoptnode 个属性	数据类型	属性描述
decision_variable_name	string	OPL 中定义的目标函数变量名称。仅当 output_value_mode 属性设置为 dvar, 才会启用此属性。
objective_function_value_fieldname	string	要在输出中使用的目标函数值的字段名称。缺省值为 _OBJECTIVE。
output_tuple_set_names	string	<p>入局数据中的预定义元组的名称。此属性充当决策变量的索引, 并且预期通过“变量输出”进行输出。“输出元组”必须与 OPL 中的决策变量定义一致。如果有多个索引, 那么元组名称必须以逗号 (,) 连接。</p> <p>单个元组的一个示例为 Products, 相应的 OPL 定义为 dvar float+ Production[Products];</p> <p>多个元组的一个示例为 Products,Components, 相应的 OPL 定义为 dvar float+ Production[Products][Components];</p>
decision_output_map	结构化属性的列表	<p>OPL 中定义的将作为输出和输出字段的变量之间的字段映射。列表中的每个实体都是结构化数据:</p> <p><b>变量名称。</b> OPL 中要输出的变量名称。</p> <p><b>存储类型。</b> 可能的值为 int、float 或 string。</p> <p><b>输出字段名称。</b> 结果 (输出或导出) 中的预期字段名称。</p> <p><b>示例:</b></p> <pre>[[['Production','int','res'],   ['Remark','string','res_1']]['Cost',   'float','res_2']]</pre>

## derive\_stbnode 属性



“空间时间限制”节点根据纬度、经度和时间戳记字段派生了空间时间限制。您还可以将频繁的空间时间限制标识为逗留。

示例

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

# Individual Records mode
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")
```

```
# Hangouts mode
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

表 66: “空间时间限制”节点属性

derive_stbnode 个属性	数据类型	属性描述
mode	IndividualRecords Hangouts	
latitude_field	field	
longitude_field	field	
timestamp_field	field	
hangout_density	密度	单一密度。请参阅 densities 以了解有效的密度值。
densities	[density,density,..., density]	<p>每个密度都是一个字符串，例如 STB_GH8_1DAY。</p> <p>注: 对于哪些密度有效，存在限制。对于 geohash，可以使用 GH1 到 GH15 中的值。对于 temporal 部分，可以使用下列值：</p> <div>           EVER            1YEAR            1MONTH            1DAY            12HOURS            8HOURS            6HOURS            4HOURS            3HOURS            2HOURS            1HOUR            30MINS            15MINS            10MINS            5MINS            2MINS            1MIN            30SECS            15SECS            10SECS            5SECS            2SECS            1SEC         </div>
id_field	field	



表 66: “空间时间限制”节点属性 (继续)

derive_stbnode 个属性	数据类型	属性描述
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	必须是字符串。
min_events	整数	最小有效整数值为 2。
qualifying_pct	整数	必须介于 1 与 100 之间。
add_extension_as	Prefix Suffix	
name_extension	string	

## distinctnode 属性



“区分”节点通过将第一个区分记录传递到数据流，或者通过丢弃第一个记录并将任何重复记录传递到数据流，移除重复的记录。

示例

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

表 67: distinctnode 属性

distinctnode 个属性	数据类型	属性描述
mode	Include Discard	既可以将第一条区分记录包括在数据流中，也可以丢弃第一条区分记录并将任何重复记录传送到数据流。
grouping_fields	列表	列出用于确定记录是否相同的字段。 <b>注:</b> 从 IBM SPSSModeler 16 起，不推荐使用此属性。
composite_value	结构化槽	请参阅以下示例。
composite_values	结构化槽	请参阅以下示例。

表 67: <i>distinctnode</i> 属性 (继续)		
<b>distinctnode</b> 个属性	数据类型	属性描述
inc_record_count	标志	创建一个额外字段，该字段指定为形成每条汇总记录汇总了多少条输入记录。
count_field	<i>string</i>	指定记录计数字段的名称。
sort_keys	结构化槽。	<b>注:</b> 从 IBM SPSSModeler 16 起，不推荐使用此属性。
default_ascending	标志	
low_distinct_key_count	标志	指定您只具有少量记录和/或少量键字段唯一值。
keys_pre_sorted	标志	指定具有相同键值的所有记录在输入中分组在一起。
disable_sql_generation	标志	

## composite\_value 属性的示例

composite\_value 属性的一般格式如下：

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION 的格式为 [ FillType, Option1, Option2, ...]。

示例：

```
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"])
```

定制选项需要多个自变量，这些自变量以列表形式添加，例如：

```
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"])
```

## composite\_values 属性的示例

composite\_values 属性的一般格式如下：

```
node.setPropertyValue("composite_values", [
    [FIELD1, [FILLOPTION1]],
    [FIELD2, [FILLOPTION2]],
    .
    .
    .
])
```

示例：

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

## extensionprocessnode 属性



通过“扩展变换”节点，您可以从流中提取数据，并使用 R 脚本编写或 Python for Spark 脚本编写对该数据应用变换。

### Python for Spark 示例

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

process_script = """
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = StructType([StructField("Age", LongType(), nullable=True), \
        StructField("Sex", StringType(), nullable=True), \
        StructField("BP", StringType(), nullable=True), \
        StructField("Na", DoubleType(), nullable=True), \
        StructField("K", DoubleType(), nullable=True), \
        StructField("Drug", StringType(), nullable=True)])
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    print df.dtypes[:]
    _newDF = df.select("Age", "Sex", "BP", "Na", "K", "Drug")
    print _newDF.dtypes[:]
    cxt.setSparkOutputData(_newDF)
"""

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

### R 示例

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

表 68: *extensionprocessnode* 属性

<b>extensionprocessnode</b> 个属性	数据类型	属性描述
syntax_type	<i>R</i> <i>Python</i>	指定运行哪个脚本 - R 还是 Python（R 是缺省值）。
r_syntax	<i>string</i>	要运行的 R 脚本编写语法。
python_syntax	<i>string</i>	要运行的 Python 脚本编写语法。
use_batch_size	标志	允许使用批处理。
batch_size	整数	指定要包含在每个批次中的数据记录数量。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为哪种格式。

## mergenode 属性



“合并”节点使用多个输入记录，并创建包含某些或全部输入字段的单个输出记录。这对于合并来源不同的数据 非常有用，例如内部客户数据和已购买人群统计数据。

### 示例

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

表 69: mergenode 属性		
mergenode 个属性	数据类型	属性描述
method	Order  Keys  Condition  Rankedcondition	指定记录是否按它们在数据文件中的列示顺序进行合并，是否使用一个或多个键字段来合并键字段中包含相同值的记录，或者是否在满足指定条件时合并记录；使用排名表达式可以按从低到高顺序对任意多个匹配项进行排序。
condition	string	如果 method 设置为 Condition，指定包括或丢弃记录的条件。
key_fields	列表	
common_keys	标志	
join	Inner  FullOuter  PartialOuter  Anti	
outer_join_tag.n	标志	在此属性中， <i>n</i> 是“选择数据集”对话框中显示的标记名。注意，可以指定多个标记名，因为任何数量的数据集都无法提供完整记录。
single_large_input	标志	指定是否进行优化，以使一个输入与其他输入相比具有一个相对较大的输入值。
single_large_input_tag	string	按“选择较大数据集”对话框中的显示指定标记名。请注意，该属性的用法与 <b>outer_join_tag</b> 属性的用法略有不同（标记与字符串），因为前者只能指定一个输入数据集。
use_existing_sort_keys	标志	指定输入值是否已根据一个或多个关键字段进行排序。
existing_sort_keys	[['string', 'Ascending']\n['string', 'Descending']]	指定已排序的字段及其排序方向。
primary_dataset	string	如果 method 为 Rankedcondition，请选择用于合并的主数据集。您可以将其视为外连接合并的左侧。
rename_duplicate_fields	BOOLEAN	如果 method 为 Rankedcondition，此属性设置为 Y，并且所生成的合并数据集包含来自不同数据源的多个同名字段，那么来自这些数据源的相应标记将添加到列字段标题开头。
merge_condition	string	
ranking_expression	string	

表 69: <i>mergenode</i> 属性 (继续)		
<b>mergenode</b> 个属性	数据类型	属性描述
Num_matches	整数	要根据 <i>merge_condition</i> 和 <i>ranking_expression</i> 返回的匹配项数。最小值为 1，最大值为 100。

## rfmaggregatenode 属性



“近因、频率和货币 (RFM) 总量”节点使您能够接受客户的历史交易数据、剥离任何未使用的数据，并将所有余下的交易数据合并到一行中，其中列出客户上次与您交易的时间、进行的交易数量以及这些交易的总货币价值。

示例

```
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")
```

表 70: <i>rfmaggregatenode</i> 属性		
<b>rfmaggregatenode</b> 个属性	数据类型	属性描述
relative_to	Fixed	指定计算交易近因的日期。
	Today	
reference_date	<i>date</i>	仅在 <i>relative_to</i> 中选择 <i>Fixed</i> 时才可用。
contiguous	标志	如果您的数据进行了预先排序，以便所有标识相同的记录一起出现在数据流中，那么选择此选项可以加快处理速度。
id_field	<i>field</i>	指定该字段以用来识别客户及其交易。
date_field	<i>field</i>	指定将要用来计算近因的日期字段。
value_field	<i>field</i>	指定该字段以用来计算货币值。
extension	<i>string</i>	为重复汇总字段指定前缀或后缀。
add_as	Suffix	指定是否应作为前缀或后缀来添加 <i>extension</i> 。
	Prefix	
discard_low_value_records	标志	支持使用 <i>discard_records_below</i> 设置。
discard_records_below	成员	可在计算 RFM 总计时，指定一个最小值，凡低于该值的交易详细信息都不再被使用。值的单位与所选的 <i>value</i> 字段相关。
only_recent_transactions	标志	启用使用 <i>specify_transaction_date</i> 或 <i>transaction_within_last</i> 设置。

表 70: <i>rfmaggregatenode</i> 属性 (继续)		
<b>rfmaggregatenode</b> 个属性	数据类型	属性描述
specify_transaction_date	标志	
transaction_date_after	date	只有选中 <code>specify_transaction_date</code> 时才可用。指定交易日期以在分析时包含其之后的记录。
transaction_within_last	成员	只有选中 <code>transaction_within_last</code> 时才可用。指定从计算相对于以下内容的近因日期字段所返回的周期数和周期类型（天、周、月或年），在此日期之后的记录将被包含在您的分析中。
transaction_scale	Days Weeks Months Years	只有选中 <code>transaction_within_last</code> 时才可用。指定从计算相对于以下内容的近因日期字段所返回的周期数和周期类型（天、周、月或年），在此日期之后的记录将被包含在您的分析中。
save_r2	标志	显示每个客户第二个最近交易的日期。
save_r3	标志	只有选中 <code>save_r2</code> 时才可用。显示每个客户第三个最近交易的日期。

## Rprocessnode 属性



通过使用您自己的定制 R 脚本，可以使用“R 变换”节点从 IBM(r) SPSS(r) Modeler 流中获取数据并进行修改。修改数据后，会将数据返回到流中。

示例

```
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")
```

表 71: <i>Rprocessnode</i> 属性		
<b>Rprocessnode</b> 个属性	数据类型	属性描述
syntax	string	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	标志	

表 71: Rprocessnode 属性 (继续)

Rprocessnode 个属性	数据类型	属性描述
convert_datetime_class	POSIXct POSIXlt	
convert_missing	标志	
use_batch_size	标志	启用批处理
batch_size	整数	指定要包括在每个批次的数据记录数

## samplenode 属性



“样本”节点用于选择一部分记录。受支持的样本类型有许多，其中包括分层、聚类和非随机（结构化）样本。采样对于提高性能以及选择相关记录组或事务组进行分析十分有用。

示例

```
/* Create two Sample nodes to extract
   different samples from the same data */

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"], ["M",
"Normal", "Default"],
["F", "High", 0.3], ["F", "Normal", 0.3]])
```

表 72: samplenode 属性

samplenode 个属性	数据类型	属性描述
method	简单	
	Complex	
mode	Include	包括或丢弃满足指定条件的记录。
	Discard	
sample_type	First	指定抽样方法。
	OneInN	
	RandomPct	
first_n	整数	将包括或丢弃直到指定截止点的记录。
one_in_n	成员	每隔 $n-1$ 条记录包括或丢弃一条记录。



表 72: *samplenode* 属性 (继续)

<b>samplenode</b> 个属性	数据类型	属性描述
rand_pct	成员	指定要包括或丢弃记录的百分比。
use_max_size	标志	支持使用 maximum_size 设置。
maximum_size	整数	指定要包括在数据流中或丢弃的最大样本量。此选项是冗余选项，因此指定 First 和 Include 时会被禁用。
set_random_seed	标志	启用随机种子设置。
random_seed	整数	指定用作随机种子的值。
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	成员	
fixed_counts	整数	
variable_proportions	<i>field</i>	
variable_counts	<i>field</i>	
use_min_stratum_size	标志	
minimum_stratum_size	整数	仅当使用 Sample units=Proportions 获取复杂样本时，此选项才适用。
use_max_stratum_size	标志	
maximum_stratum_size	整数	仅当使用 Sample units=Proportions 获取复杂样本时，此选项才适用。
clusters	<i>field</i>	
stratify_by	<i>[field1 ... fieldN]</i>	
specify_input_weight	标志	
input_weight	<i>field</i>	
new_output_weight	<i>string</i>	

表 72: *samplenode* 属性 (继续)

<b>samplenode</b> 个属性	数据类型	属性描述
sizes_proportions	[[string string value] [string string value]...]	如果 sample_units=proportions 且 sample_size_proportions=Custom, 指定层字段值每个可能组合的值。
default_proportion	成员	
sizes_counts	[[string string value] [string string value]...]	指定层字段值每个可能的组合值。用法与 sizes_proportions 的用法相似, 但指定的是整数, 而非比例。
default_count	成员	

## selectnode 属性



“选择”节点根据特定条件从数据流中选择或废弃一部分记录。例如, 可以选择与特定销售区域相关的记录。

示例

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

表 73: *selectnode* 属性

<b>selectnode</b> 个属性	数据类型	属性描述
mode	Include	指定是包括还是丢弃选定记录。
	Discard	
condition	string	包括或丢弃记录的条件。

## sortnode 属性



“排序”节点根据一个或多个字段的值按升序或降序对记录进行排序。

示例

```
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"]])
```

表 74: *sortnode* 属性

<b>sortnode</b> 个属性	数据类型	属性描述
keys	列表	指定要作为排序依据的字段。如果未指定方向, 则会使用缺省值。

表 74: <i>sortnode</i> 属性 (继续)		
<b>sortnode</b> 个属性	数据类型	属性描述
default_ascending	标志	指定缺省排序顺序。
use_existing_keys	标志	指定是否使用以前已排序字段的排序顺序来优化现在的排序。
existing_keys		指定已排序的字段及其排序方向。使用的格式与 <b>keys</b> 属性相同。

## spacetimeboxes 属性



空间时间限制 (STB) 是进行了 Geohash 计算的空间位置的扩展。更具体地说, STB 是一个字母数字字符串, 它表示形状规则的空间和时间区域。

表 75: <i>spacetimeboxes</i> 属性		
<b>spacetimeboxes</b> 个属性	数据类型	属性描述
mode	<i>IndividualRecords</i> <i>Hangouts</i>	
latitude_field	<i>field</i>	
longitude_field	<i>field</i>	
timestamp_field	<i>field</i>	
densities	<i>[density, density, density...]</i>	<p>每个密度都是一个字符串。STB_GH8_1DAY 例如：</p> <p>请注意, 对于哪些密度有效, 存在一些限制。</p> <p>对于 geohash, 可以使用 GH1-GH15 中的值。</p> <p>对于 temporal 部分, 可以使用下列值：</p> <pre> EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2 MINS 1 MIN 30SECS 15SECS 10SECS 5 SECS 2 SECS 1SEC </pre>

表 75: *spacetimeboxes* 属性 (继续)

<b>spacetimeboxes</b> 个属性	数据类型	属性描述
field_name_extension	<i>string</i>	
add_extension_as	<i>Prefix</i> <i>Suffix</i>	
hangout_density	密度	单一密度（请参见上文）
id_field	<i>field</i>	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 2HOURS 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	这必须是字符串。
min_events	整数	最小值为 2
qualifying_pct	整数	必须在范围 1-100 中

## streamingtimeseries 属性



“流式时间序列”节点在一个步骤中对时间序列模型同时进行构建和评分。

注: 此“流式时间序列”节点替换 SPSS 建模器 V18 中不推荐使用的原始“流式 TS”节点。

表 76: *streamingtimeseries* 属性

<b>streamingtimeseries</b> 属性	值	属性描述
targets	<i>field</i>	“流式时间序列”节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率和权重字段。有关更多信息，请参阅主题第 189 页的『公共建模节点属性』。
candidate_inputs	[ <i>field1</i> ... <i>fieldN</i> ]	模型所使用的输入或预测变量字段。
use_period	标志	
date_time_field	<i>field</i>	

表 76: <i>streamingtimeseries</i> 属性 (继续)		
<b>streamingtimeseries</b> 属性	值	属性描述
input_interval	None  Unknown  Year  Quarter  Month  Week  Day  Hour  Hour_nonperiod  Minute  Minute_nonperiod  Second  Second_nonperiod	
period_field	<i>field</i>	
period_start_value	整数	
num_days_per_week	整数	
start_day_of_week	Sunday  Monday  Tuesday  Wednesday  Thursday  Friday  Saturday	
num_hours_per_day	整数	
start_hour_of_day	整数	
timestamp_increments	整数	

表 76: <i>streamingtimeseries</i> 属性 (继续)		
<b>streamingtimeseries</b> 属性	值	属性描述
cyclic_increments	整数	
cyclic_periods	列表	
output_interval	None  Year  Quarter  Month  Week  Day  Hour  Minute  Second	
is_same_interval	标志	
cross_hour	标志	
aggregate_and_distribute	列表	
aggregate_default	Mean  Sum  Mode  Min  Max	
distribute_default	Mean  Sum	
group_default	Mean  Sum  Mode  Min  Max	

表 76: <i>streamingtimeseries</i> 属性 (继续)		
<b>streamingtimeseries</b> 属性	值	属性描述
missing_imput	Linear_interp  Series_mean  K_mean  K_median  Linear_trend	
k_span_points	整数	
use_estimation_period	标志	
estimation_period	Observations  Times	
date_estimation	列表	仅在使用 date_time_field 时可用
period_estimation	列表	仅在使用 use_period 时可用
observations_type	Latest  Earliest	
observations_num	整数	
observations_exclude	整数	
method	ExpertModeler  Exsmooth  Arima	
expert_modeler_method	ExpertModeler  Exsmooth  Arima	
consider_seasonal	标志	
detect_outliers	标志	
expert_outlier_additive	标志	
expert_outlier_level_shift	标志	
expert_outlier_innovational	标志	
expert_outlier_level_shift	标志	
expert_outlier_transient	标志	

表 76: *streamingtimeseries* 属性 (继续)

<b>streamingtimeseries</b> 属性	值	属性描述
expert_outlier_seasonal_additive	标志	
expert_outlier_local_trend	标志	
expert_outlier_additive_patch	标志	
consider_newesmodels	标志	
exsmooth_model_type	Simple  HoltsLinearTrend  BrownsLinearTrend  DampedTrend  SimpleSeasonal  WintersAdditive  WintersMultiplicative  DampedTrendAdditive  DampedTrendMultiplicative  MultiplicativeTrendAdditive  MultiplicativeSeasonal  MultiplicativeTrendMultiplicative  MultiplicativeTrend	
futureValue_type_method	Compute  specify	
exsmooth_transformation_type	None  SquareRoot  NaturalLog	
arma.p	整数	
arma.d	整数	



表 76: <i>streamingtimeseries</i> 属性 (继续)		
<b>streamingtimeseries</b> 属性	值	属性描述
arima.q	整数	
arima.sp	整数	
arima.sd	整数	
arima.sq	整数	
arima_transformation_type	None  SquareRoot  NaturalLog	
arima_include_constant	标志	
tf_arima.p. <i>fieldname</i>	整数	用于转换函数。
tf_arima.d. <i>fieldname</i>	整数	用于转换函数。
tf_arima.q. <i>fieldname</i>	整数	用于转换函数。
tf_arima.sp. <i>fieldname</i>	整数	用于转换函数。
tf_arima.sd. <i>fieldname</i>	整数	用于转换函数。
tf_arima.sq. <i>fieldname</i>	整数	用于转换函数。
tf_arima.delay. <i>fieldname</i>	整数	用于转换函数。
tf_arima.transformation_type. <i>fieldname</i>	None  SquareRoot  NaturalLog	用于转换函数。
arima_detect_outliers	标志	
arima_outlier_additive	标志	
arima_outlier_level_shift	标志	
arima_outlier_innovational	标志	
arima_outlier_transient	标志	
arima_outlier_seasonal_additive	标志	
arima_outlier_local_trend	标志	
arima_outlier_additive_patch	标志	
conf_limit_pct	<i>real</i>	
events	字段	
forecastperiods	整数	
extend_records_into_future	标志	
conf_limits	标志	
noise_res	标志	

# streamingts 属性（不推荐）



注: 此原始“流式时间序列”节点在 SPSS 建模器 V18 中已不推荐使用，并且替换为旨在利用 IBM SPSS 分析服务器的能力并处理大数据的新的“流式时间序列”节点。

“流式 TS”节点在某个步骤中构建时间序列模型并对其进行评估，而不需要“时间间隔”节点。

示例

```
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")
```

表 77: *streamingts* 属性

streamingts 个属性	数据类型	属性描述
custom_fields	标志	如果 custom_fields=false, 那么将使用上游“类型”节点的当前设置。如果 custom_fields=true, 那么必须指定 targets 和 inputs。
targets	[field1...fieldN]	
inputs	[field1...fieldN]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	标志	
conf_limit_pct	real	
use_time_intervals_node	标志	如果 use_time_intervals_node=true, 那么将使用上游“时间间隔”节点的设置。如果 use_time_intervals_node=false, 那么必须指定 interval_offset_position、interval_offset 和 interval_type。
interval_offset_position	LastObservation LastRecord	LastObservation 是指最后一个有效观测值。LastRecord 是指从最后一个记录计数。
interval_offset	成员	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	字段	

表 77: <i>streamingts</i> 属性 (继续)		
streamingts 个属性	数据类型	属性描述
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	标志	
detect_outliers	标志	
expert_outlier_additive	标志	
expert_outlier_level_shift	标志	
expert_outlier_innovational	标志	
expert_outlier_transient	标志	
expert_outlier_seasonal_additive	标志	
expert_outlier_local_trend	标志	
expert_outlier_additive_patch	标志	
exsmooth_model_type	Simple HoltsLinearTrend BrownLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	整数	对于“时间序列”建模节点是同一属性
arima_d	整数	对于“时间序列”建模节点是同一属性
arima_q	整数	对于“时间序列”建模节点是同一属性
arima_sp	整数	对于“时间序列”建模节点是同一属性
arima_sd	整数	对于“时间序列”建模节点是同一属性
arima_sq	整数	对于“时间序列”建模节点是同一属性
arima_transformation_type	None SquareRoot NaturalLog	对于“时间序列”建模节点是同一属性
arima_include_constant	标志	对于“时间序列”建模节点是同一属性
tf_arima_p. <i>fieldname</i>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。

表 77: *streamingts* 属性 (继续)

<b>streamingts</b> 个属性	数据类型	属性描述
<code>tf_arima_d.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_q.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_sp.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_sd.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_sq.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_delay.fieldname</code>	整数	对于“时间序列”建模节点是同一属性。用于转换函数。
<code>tf_arima_transformation_type.fieldname</code>	None SquareRoot NaturalLog	
<code>arima_detect_outlier_mode</code>	None Automatic	
<code>arima_outlier_additive</code>	标志	
<code>arima_outlier_level_shift</code>	标志	
<code>arima_outlier_innovational</code>	标志	
<code>arima_outlier_transient</code>	标志	
<code>arima_outlier_seasonal_additive</code>	标志	
<code>arima_outlier_local_trend</code>	标志	
<code>arima_outlier_additive_patch</code>	标志	
<code>deployment_force_rebuild</code>	标志	
<code>deployment_rebuild_mode</code>	Count Percent	
<code>deployment_rebuild_count</code>	成员	
<code>deployment_rebuild_pct</code>	成员	
<code>deployment_rebuild_field</code>	<field>	

# 第 11 章 字段操作节点属性

## anonymizenode 属性



“匿名化”节点用于转换字段名和字段值在下游的表示方式，从而掩饰原始数据。如果要允许其他用户构建含有敏感数据（例如客户名称或其他详细信息）的模型，那么这种节点十分有用。

示例

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

表 78: anonymizenode 属性		
anonymizenode 个属性	数据类型	属性描述
enable_anonymize	标志	设置为 True 时，将实现字段值的匿名化（相当于在“对值进行匿名化”列中为该字段选择是）。
use_prefix	标志	设置为 True 时，如果已指定定制前缀，那么将使用该前缀。适用于将通过散列法匿名化的字段，而且等效于在“替换值”对话框中为该字段选中自定义单选按钮。
prefix	string	等效于在“替换值”对话框的文本框中键入前缀。如果未指定其他任何值，那么缺省前缀是该缺省值。
transformation	Random Fixed	确定通过变换法匿名化的字段的变换参数是随机的还是固定的。
set_random_seed	标志	设置为 True 时，将使用指定的种子值（如果 transformation 也设置为 Random）。
random_seed	整数	当 set_random_seed 设置为 True 时，该值是随机数的种子。
scale	成员	当 transformation 设置为 Fixed 时，此值用于“乘数”。通常，最大乘数值为 10，但可能会减小以避免溢出。
translate	成员	当 transformation 设置为 Fixed 时，此值用于“转换”。通常，最大转换值为 1000，但可能会减小以避免溢出。

# autodatapreprenode 属性



“自动数据准备 (ADP)”节点可分析您的数据并标识修正，筛选出存在问题或可能无用的字段，并在适当的情况下派生新的属性，通过智能筛选和抽样技术改进性能。您可以采用完全自动化方式使用此节点，从而允许此节点选择并应用修订，另外也可以在应用修订前预览更改并根据需要接受、拒绝或进行修改。

示例

```
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")
```

表 79: autodatapreprenode 属性		
autodatapreprenode 个属性	数据类型	属性描述
objective	Balanced  Speed  Accuracy  Custom	
custom_fields	标志	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
target	field	指定单个目标字段。
inputs	[field1 ... fieldN]	模型所使用的输入或预测变量字段。
use_frequency	标志	
frequency_field	field	
use_weight	标志	
weight_field	field	
excluded_fields	Filter  None	
if_fields_do_not_match	StopExecution  ClearAnalysis	
prepare_dates_and_times	标志	控制对所有日期字段和时间字段的访问
compute_time_until_date	标志	
reference_date	Today  Fixed	
fixed_date	date	

表 79: autodatapreprenode 属性 (继续)

autodatapreprenode 个属性	数据类型	属性描述
units_for_date_durations	Automatic	
	Fixed	
fixed_date_units	Years	
	Months	
	Days	
compute_time_until_time	标志	
reference_time	CurrentTime	
	Fixed	
fixed_time	时间	
units_for_time_durations	Automatic	
	Fixed	
fixed_date_units	Hours	
	Minutes	
	Seconds	
extract_year_from_date	标志	
extract_month_from_date	标志	
extract_day_from_date	标志	
extract_hour_from_time	标志	
extract_minute_from_time	标志	
extract_second_from_time	标志	
exclude_low_quality_inputs	标志	
exclude_too_many_missing	标志	
maximum_percentage_missing	成员	
exclude_too_many_categories	标志	
maximum_number_categories	成员	
exclude_if_large_category	标志	
maximum_percentage_category	成员	
prepare_inputs_and_target	标志	

表 79: autodatapreprenode 属性 (继续)

autodatapreprenode 个属性	数据类型	属性描述
adjust_type_inputs	标志	
adjust_type_target	标志	
reorder_nominal_inputs	标志	
reorder_nominal_target	标志	
replace_outliers_inputs	标志	
replace_outliers_target	标志	
replace_missing_continuous_inputs	标志	
replace_missing_continuous_target	标志	
replace_missing_nominal_inputs	标志	
replace_missing_nominal_target	标志	
replace_missing_ordinal_inputs	标志	
replace_missing_ordinal_target	标志	
maximum_values_for_ordinal	成员	
minimum_values_for_continuous	成员	
outlier_cutoff_value	成员	
outlier_method	Replace Delete	
rescale_continuous_inputs	标志	
rescaling_method	MinMax ZScore	
min_max_minimum	成员	
min_max_maximum	成员	
z_score_final_mean	成员	
z_score_final_sd	成员	
rescale_continuous_target	标志	
target_final_mean	成员	
target_final_sd	成员	
transform_select_input_fields	标志	



表 79: autodatapreprenode 属性 (继续)

autodatapreprenode 个属性	数据类型	属性描述
maximize_association_with_target	标志	
p_value_for_merging	成员	
merge_ordinal_features	标志	
merge_nominal_features	标志	
minimum_cases_in_category	成员	
bin_continuous_fields	标志	
p_value_for_binning	成员	
perform_feature_selection	标志	
p_value_for_selection	成员	
perform_feature_construction	标志	
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	string	
minute_cyclical_name_extension	string	

表 79: autodatapreprenode 属性 (继续)

autodatapreprenode 个属性	数据类型	属性描述
second_cyclical_name_extension	string	

## astimeintervalsnode 属性



使用“时间间隔”节点可以指定时间间隔并派生用于估算或预测的新时间字段。支持全部范围的时间间隔，从秒到年。

表 80: astimeintervalsnode 属性

astimeintervalsnode 个属性	数据类型	属性描述
time_field	field	只能接受单个连续字段。此节点将该字段用作汇总键，以转换时间间隔。如果在此处使用了整数字段，那么会将其视为时间索引。
dimensions	[field1 field2 ... fieldn]	这些字段用于根据字段值来创建各个时间序列。
fields_to_aggregate	[field1 field2 ... fieldn]	在更改时间字段的时间段的过程中，这些字段将进行汇总。在离开此节点的数据中，将过滤掉所有未包括在此选取器中的字段。

## binningnode 属性



“分箱”节点根据一个或多个现有连续（数字范围）字段的值自动创建新的名义（集合）字段。例如，您可以将连续收入字段转换为一个包含各组收入（作为与均值之间的偏差）的新分类字段。一旦创建新字段分级后，即可根据割点创建“衍生”节点。

示例

```
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)
```

表 81: binningnode 属性

binningnode 个属性	数据类型	属性描述
fields	[field1 field2 ... fieldn]	要进行变换的连续（数值范围）字段。可以同时多个字段进行分级。

表 81: binningnode 属性 (继续)		
binningnode 个属性	数据类型	属性描述
method	FixedWidth EqualCount Rank SDev Optimal	用于为新字段分级（类别）确定分割点的方法。
recalculate_bins	Always IfNecessary	指定是重新计算分级且每次执行节点时将数据放置在相关分级中，还是仅将数据添加到现有分级和已添加的新分级中。
fixed_width_name_extension	string	缺省扩展名为 <i>_BIN</i> 。
fixed_width_add_as	Suffix Prefix	指定是将扩展名添加到字段名的末尾（后缀）还是开头（前缀）。缺省扩展名为 <i>income_BIN</i> 。
fixed_bin_method	Width Count	
fixed_bin_count	整数	指定用于确定新字段的固定宽度分级（类别）数的整数。
fixed_bin_width	real	用于计算分级宽度的值（整数或实数）。
equal_count_name_extension	string	缺省扩展名为 <i>_TILE</i> 。
equal_count_add_as	Suffix Prefix	指定用于使用标准 p-tile 法生成的字段名的扩展名（后缀或前缀）。缺省扩展名为 <i>_TILE</i> 加上 <i>N</i> ，其中 <i>N</i> 是分位数。
tile4	标志	生成四分位数分级，每个分级中包含 25% 的观测值。
tile5	标志	生成五分位数分级。
tile10	标志	生成十分位数分级。
tile20	标志	生成二十分位数分级。
tile100	标志	生成百分位数分级。
use_custom_tile	标志	
custom_tile_name_extension	string	缺省扩展名为 <i>_TILEN</i> 。
custom_tile_add_as	Suffix Prefix	
custom_tile	整数	

表 81: binningnode 属性 (继续)

binningnode 个属性	数据类型	属性描述
equal_count_method	RecordCount ValueSum	RecordCount 方法是每个分级分配相同数目的记录，而 ValueSum 方法是使分配记录后每个分级中值的总和相等。
tied_values_method	Next  Current  Random	指定要输入的分箱结值数据。
rank_order	Ascending  Descending	此属性包括 Ascending（最低值标记为 1）或 Descending（最高值标记为 1）。
rank_add_as	Suffix  Prefix	此选项适用于排序、分数排序和百分比排序。
rank	标志	
rank_name_extension	string	缺省扩展名为 <i>_RANK</i> 。
rank_fractional	标志	排序观测值，其中新字段的值是排序值除以非缺失观测值的权重和。分数排序值介于 0–1 之间。
rank_fractional_name_ extension	string	缺省扩展名为 <i>_F_RANK</i> 。
rank_pct	标志	每个排序值除以具有有效值的记录数再乘以 100。分数排序百分比值介于 1–100 之间。
rank_pct_name_extension	string	缺省扩展名为 <i>_P_RANK</i> 。
sdev_name_extension	string	
sdev_add_as	Suffix  Prefix	
sdev_count	One  Two  Three	
optimal_name_extension	string	缺省扩展名为 <i>_OPTIMAL</i> 。
optimal_add_as	Suffix  Prefix	
optimal_supervisor_field	field	作为监督字段选择的字段，为分级选择的字段与之相关。

表 81: binningnode 属性 (继续)		
binningnode 个属性	数据类型	属性描述
optimal_merge_bins	标志	指定将所有具有较小观测值计数的分级添加到更大的相邻分级。
optimal_small_bin_threshold	整数	
optimal_pre_bin	标志	表示要进行数据集的预分级。
optimal_max_bins	整数	指定上限以避免创建过大分级数。
optimal_lower_end_point	Inclusive	
	Exclusive	
optimal_first_bin	Unbounded	
	Bounded	
optimal_last_bin	Unbounded	
	Bounded	

## derivednode 属性



“派生”节点修改数据值或者根据一个或多个现有字段创建新字段。它创建类型为公式、标志、名义、状态、计数和条件的字段。

### 示例 1

```
# Create and configure a Flag Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Create and configure a Conditional Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" > @INDEX)")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

### 示例 2

此脚本假定存在两个分别名为 XPos 和 YPos 的数字列，这两个列分别代表一个点（例如事件发生位置）的 X 和 Y 坐标。此脚本创建“派生”节点，该节点根据特定坐标系中代表该点的 X 和 Y 坐标来计算地理空间列：

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
```

```
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

表 82: *derivenode* 属性

<b>derivenode</b> 个属性	数据类型	属性描述
new_name	<i>string</i>	新字段的名称。
mode	Single Multiple	指定单个或多个字段。
fields	列表	仅用于在 Multiple 模式下选择多个字段。
name_extension	<i>string</i>	为新字段名指定扩展名。
add_as	Suffix Prefix	将扩展名添加为字段名的前缀（开头）或后缀（末尾）。
result_type	Formula Flag Set State Count Conditional	可创建的六种类型的新字段。
formula_expr	<i>string</i>	用于在导出节点计算新字段值的表达式。
flag_expr	<i>string</i>	
flag_true	<i>string</i>	
flag_false	<i>string</i>	
set_default	<i>string</i>	
set_value_cond	<i>string</i>	提供与给定值相关联的条件的结构化属性。
state_on_val	<i>string</i>	指定满足 On 条件时新字段的值。
state_off_val	<i>string</i>	指定满足 Off 条件时新字段的值。
state_on_expression	<i>string</i>	
state_off_expression	<i>string</i>	
state_initial	On Off	为新字段的每个记录分配初始值 On 或 Off。可在满足每个条件时更改此值。

表 82: <i>derivenode</i> 属性 (继续)		
<b>derivenode</b> 个属性	数据类型	属性描述
count_initial_val	<i>string</i>	
count_inc_condition	<i>string</i>	
count_inc_expression	<i>string</i>	
count_reset_condition	<i>string</i>	
cond_if_cond	<i>string</i>	
cond_then_expr	<i>string</i>	
cond_else_expr	<i>string</i>	
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	此属性可用于定义与派生的字段相关联的测量。可以向 <b>setter</b> 函数传递字符串或者其中一个 <b>MeasureType</b> 值。getter 函数始终返回 <b>MeasureType</b> 值。
collection_measure	Range / MeasureType.RANGE  Flag / MeasureType.FLAG  Set / MeasureType.SET  OrderedSet / MeasureType.ORDERED_SET  Typeless / MeasureType.TYPELESS	对于集合字段（深度为 0 的列表），此属性定义与底层值相关联的测量类型。

表 82: *derivenode* 属性 (继续)

derivenode 个属性	数据类型	属性描述
geo_type	Point  MultiPoint  LineString  MultiLineString  Polygon  MultiPolygon	对于地理空间字段，此属性定义此字段所表示的地理空间对象的类型。这应该与这些值的列表深度一致。
has_coordinate_system	BOOLEAN	对于地理空间字段，此属性定义此字段是否具有坐标系
coordinate_system	string	对于地理空间字段，此属性定义此字段的坐标系。

## ensembledenode 属性



“整体”节点将两个或多个模型块组合在一起，以获得比可从任何一个模型实现的预测更准确的预测。

示例

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

表 83: *ensembledenode* 属性

ensembledenode 个属性	数据类型	属性描述
ensemble_target_field	field	为在整体中使用的所有模型指定目标字段。
filter_individual_model_output	标志	指定是否应抑制各个模型的评分结果。



表 83: *ensembledenode* 属性 (继续)

ensembledenode 个属性	数据类型	属性描述
flag_ensemble_method	Voting  ConfidenceWeightedVoting  RawPropensityWeightedVoting  AdjustedPropensityWeightedVoting  HighestConfidence  AverageRawPropensity  AverageAdjustedPropensity	指定用于确定整体评分的方法。仅当选定的目标是标志字段时，才会应用此设置。
set_ensemble_method	Voting  ConfidenceWeightedVoting  HighestConfidence	指定用于确定整体评分的方法。仅当选定的目标是名义字段时，才会应用此设置。
flag_voting_tie_selection	Random  HighestConfidence  RawPropensity  AdjustedPropensity	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是标志字段时，才会应用此设置。
set_voting_tie_selection	Random  HighestConfidence	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是名义字段时，才会应用此设置。
calculate_standard_error	标志	如果目标字段是连续的，那么缺省情况下将运行标准误差计算，以计算测量值或估算值与真实值之间的差，并显示这些估算值的匹配程度。

## fillernode 属性



“填充器”节点用于替换字段值并更改存储。您可以选择基于 CLEM 条件（例如 @BLANK(@FIELD)）的替换值。或者，也可以选择将所有空白值或空值替换为特定值。“填充”节点通常与“类型”节点一起使用以替换缺失值。

示例

```
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\"")
```

表 84: fillernode 属性

fillernode 个属性	数据类型	属性描述
fields	列表	数据集中其值将被检查并替换的字段。
replace_mode	Always Conditional Blank Null BlankAndNull	可以替换所有值、空白值或空值，也可以根据指定条件进行替换。
condition	string	
replace_with	string	

## filternode 属性



“过滤”节点用于过滤（废弃）字段、对字段进行重命名以及将字段从一个源节点映射到另一个节点。

示例：

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

**使用 default\_include 属性。** 请注意，设置 default\_include 属性的值不会自动包括或排除所有字段；它只确定针对当前选定字段的缺省行为。这在功能上等效于单击“过滤节点”对话框中的**缺省情况下包括字段**按钮。例如，假设运行以下脚本：

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

这将导致节点传递字段 Age 和 Sex，并废弃所有其他字段。运行先前的脚本后，现在假设您将以下行添加到该脚本中，以命名两个额外的字段：

```
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

这会将另外两个字段添加到过滤器中，以便总共传递四个字段（Age、Sex、BP、Na）。换言之，将 default\_include 的值重置为 False，不会自动重置所有字段。

此外，如果现在通过使用脚本或在“过滤节点”对话框中将 `default_include` 的值更改为 `True`，则会使此行为发生颠倒，即，将丢弃而非包括上面列出的四个字段。如有疑问，可使用“过滤节点”对话框中的控件进行实验，这将有助于理解此交互效应。

表 85: <i>filternode</i> 属性		
<b>filternode</b> 个属性	数据类型	属性描述
<code>default_include</code>	标志	用于指定缺省行为是传递还是过滤字段的键控属性：  注意，设置此属性不会自动包括或排除所有字段；它只确定缺省情况下是包括还是排除选定字段。有关其他注释请参见下面的示例。
<code>include</code>	标志	用于包括和删除字段的键控属性。
<code>new_name</code>	<i>string</i>	

## historynode 属性



“历史记录”节点创建新字段，这些字段包含先前记录中的字段的数据。“历史记录”节点最常用于顺序数据，如时间序列数据。在使用“历史记录”节点之前，您可能希望使用“排序”节点对数据进行排序。

示例

```
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")
```

表 86: <i>historynode</i> 属性		
<b>historynode</b> 个属性	数据类型	属性描述
<code>fields</code>	列表	需要其历史记录的字段。
<code>offset</code>	成员	指定要从中提取历史记录字段值的最新记录（当前记录之前的记录）。
<code>span</code>	成员	指定要从中提取值的以前记录的数目。
<code>unavailable</code>	Discard  Leave  Fill	处理不含历史记录值的记录时，通常参考没有以前的记录作为历史记录的前几个记录（位于数据集顶部）。
<code>fill_with</code>	String  Number	指定要用于无历史记录值可用的记录的值或字符串。

# partitionnode 属性



分区节点可生成分区字段，该字段可将数据分割为单独的子集以便在模型构建的训练、测试和验证阶段使用。

示例

```
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")
```

表 87: partitionnode 属性

partitionnode 个属性	数据类型	属性描述
new_name	string	由节点生成的分区字段的名称。
create_validation	标志	指定是否应创建验证分区。
training_size	整数	要分配给训练分区的记录所占的百分比 (0-100)。
testing_size	整数	要分配给测试分区的记录所占的百分比 (0-100)。
validation_size	整数	要分配给验证分区的记录所占的百分比 (0-100)。如果未创建验证分区，则忽略此属性。
training_label	string	训练分区的标签。
testing_label	string	测试分区的标签。
validation_label	string	验证分区的标签。如果未创建验证分区，则忽略此属性。
value_mode	System SystemAndLabel Label	指定用于表示数据中每个分区的值。例如，训练样本可以表示为系统整数 1、标签 Training 或二者的组合 1_Training。
set_random_seed	BOOLEAN	指定是否应使用用户指定的随机种子。
random_seed	整数	用户指定的随机种子值。如果要使用此值，set_random_seed 必须设置为 True。
enable_sql_generation	BOOLEAN	指定是否使用 SQL 回送以分配记录到分区。
unique_field		指定输入字段，用以确保以随机但可重复的方式将记录分配到分区。如果要使用此值，enable_sql_generation 必须设置为 True。

# reclassifynode 属性



“重新分类”节点将一组分类值转换为另一组值。对于压缩类别或为分析而进行的数据重新分组，重新分类非常有用。

示例

```
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"])
```

表 88: reclassifynode 属性		
reclassifynode 个属性	数据类型	属性描述
mode	Single Multiple	Single 对一个字段的类别进行重新分类。Multiple 将激活用于同时对多个字段进行变换的选项。
replace_field	标志	
field	string	仅在 Single 模式下使用。
new_name	string	仅在 Single 模式下使用。
fields	[field1 field2 ... fieldn]	仅在 Multiple 模式下使用。
name_extension	string	仅在 Multiple 模式下使用。
add_as	Suffix Prefix	仅在 Multiple 模式下使用。
reclassify	string	字段值的结构化属性。
use_default	标志	使用缺省值。
default	string	指定缺省值。
pick_list	[string string ... string]	允许用户导入已知新值的列表以填充表中的下拉列表。

# reordernode 属性



“字段重新排序器”节点定义用于显示下游字段的自然顺序。此顺序将影响字段在多个位置的显示方式，如表格、列表和字段选择器。在使用宽数据集以提高感兴趣字段的可见性时，此操作很有用。

示例

```
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"])
```

表 89: *reordernode* 属性

reordernode 个属性	数据类型	属性描述
mode	Custom Auto	可以自动对值进行排序，也可以指定自定义顺序。
sort_by	Name Type Storage	
ascending	标志	
start_fields	[field1 field2 ... fieldn]	新字段插入到这些字段之后。
end_fields	[field1 field2 ... fieldn]	新字段插入到这些字段之前。

## reprojectnode 属性



在 SPSS 建模器 中，表达式构建器空间函数、“空间-时间预测”(STP) 节点和“地图可视化”节点之类的项使用投影坐标系。使用“重新投影”节点可以更改所导入的任何使用了地理坐标系的数据的坐标系。

表 90: *reprojectnode* 属性

reprojectnode 个属性	数据类型	属性描述
reproject_fields	[field1 field2 ... fieldn]	列出所有要进行重新投影的字段。
reproject_type	Streamdefault Specify	选择如何对字段进行重新投影。
coordinate_system	string	要应用于字段的坐标系的名称。示例：  set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"

# restructurenode 属性



“重构”节点将名义字段或标志字段转换为一组字段，这组字段可以使用另一字段的值进行填充。例如，如果给定名为支付类型的字段，且值为信用、现金和借记，那么将创建三个新字段（信用、现金和 借记），其中每个字段都可能包含实际支付的值。

示例

```
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"])
```

表 91: restructurenode 属性		
restructurenode 个属性	数据类型	属性描述
fields_from	[category category category]  all	
include_field_name	标志	指示是否在重新结构化的字段名中使用字段名。
value_mode	OtherFields  Flags	表示用于为重新结构化字段指定值的模式。如果选择 OtherFields，必须指定要使用哪些字段（参阅下文）。如果选择 Flags，则值为数值标志。
value_fields	列表	如果 value_mode 是 OtherFields，则此属性是必需的。指定使用哪些字段作为值字段。

# rfmanalysisnode 属性



利用“近因、频数和货币 (RFM) 分析”节点，您能够通过检查客户最近一次从您那里购买的时间（近因）、购买的频率（频数）以及他们在所有交易中花费的金额（货币），以定量方式确定哪些客户可能是最佳客户。

示例

```
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])
```

表 92: rfmanalysisnode 属性		
rfmanalysisnode 个属性	数据类型	属性描述
recency	field	指定近因字段。它有可能是日期、时间戳记或简单的数值。
frequency	field	指定频率字段。

表 92: *rfmanalysisnode* 属性 (继续)

<b>rfmanalysisnode</b> 个属性	数据类型	属性描述
monetary	<i>field</i>	指定货币字段。
recency_bins	整数	指定要生成的近因分级数量。
recency_weight	成员	指定应用于近因数据的权重。缺省值为 100。
frequency_bins	整数	指定要生成的频率分级数量。
frequency_weight	成员	指定应用于频率数据的权重。缺省值为 10。
monetary_bins	整数	指定要生成的货币分级数量。
monetary_weight	成员	指定应用于货币数据的权重。缺省值为 1。
tied_values_method	Next Current	指定要输入的分级结值数据。
recalculate_bins	Always IfNecessary	
add_outliers	标志	只有当 <b>recalculate_bins</b> 设为 <b>IfNecessary</b> 时才可用。如果已设置，则将位于下限分级以下的记录添加到下限分级中，并且将最高分级以上的记录添加到最高分级中。
binned_field	Recency Frequency Monetary	
recency_thresholds	<i>value value</i>	只有当 <b>recalculate_bins</b> 设为 <b>Always</b> 时才可用。指定近因分级的下限阈值和上限阈值。一个分级的上限阈值用作下一个分级的下限阈值 例如，[10 30 60] 可定义两个分级，第一个分级的上限阈值和下限阈值分别为 10 和 30，第二个分级的两个阈值分别为 30 和 60。
frequency_thresholds	<i>value value</i>	只有当 <b>recalculate_bins</b> 设为 <b>Always</b> 时才可用。
monetary_thresholds	<i>value value</i>	只有当 <b>recalculate_bins</b> 设为 <b>Always</b> 时才可用。

## settoflagnode 属性



“设为标志”节点根据针对一个或多个名义字段定义的分类值派生多个标志字段。



示例

```
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"])
```

表 93: settoflagnode 属性		
settoflagnode 个属性	数据类型	属性描述
fields_from	[category category category]  all	
true_value	string	指定设置标志时节点所使用的真值。缺省值为 T。
false_value	string	指定设置标志时节点所使用的假值。缺省值为 F。
use_extension	标志	使用扩展名作为新标志字段的后缀或前缀。
extension	string	
add_as	Suffix  Prefix	指定所添加的扩展名是后缀还是前缀。
aggregate	标志	根据关键字段将记录分组。如果有任何记录被设置为真，则会启用组中的所有标志字段。
keys	列表	关键字段。

statisticstransformnode 属性



Statistics 转换节点针对 IBM SPSSModeler 中的数据源运行所选的 IBM SPSS Statistics 语法命令。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息，请参阅第 377 页的『statisticstransformnode 属性』。

timeintervalsnode 属性（不推荐）



注: 此节点在 SPSS 建模器 V18 中已不推荐使用，并且替换为新的“时间序列”节点。时间间隔节点指定区间，创建用于对时间序列数据进行建模的标签（如果需要）。如果值的间隔不是均匀的，那么此节点会根据需要填充值或将值集中起来以生成均匀的记录区间。

## 示例

```
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")
```

表 94: *timeintervals*node 属性

<b>timeintervals</b> node 个属性	数据类型	属性描述
interval_type	None  Periods  CyclicPeriods  Years  Quarters  Months  DaysPerWeek  DaysNonPeriodic  HoursPerDay  HoursNonPeriodic  MinutesPerDay  MinutesNonPeriodic  SecondsPerDay  SecondsNonPeriodic	
mode	Label  Create	指定是要连续标记记录还是要根据指定日期、时间戳或时间字段构建序列。
field	<i>field</i>	当根据数据构建序列时，指定表示每个记录的日期或时间的字段。

表 94: <i>timeintervalsnode</i> 属性 (继续)		
<b>timeintervalsnode</b> 个属性	数据类型	属性描述
period_start	整数	指定周期或循环周期的起始区间
cycle_start	整数	循环周期的起始周期。
year_start	整数	对于适用的区间类型，指第一个区间所属的年份。
quarter_start	整数	对于适用的区间类型，指第一个区间所属的季度。
month_start	January February March April May June July August September October November December	
day_start	整数	
hour_start	整数	
minute_start	整数	
second_start	整数	
periods_per_cycle	整数	对于循环周期，指每个周期中的期间数。
fiscal_year_begins	January February March April May June July August September October November December	对于季度区间，指定财政年度开始的月份。

表 94: *timeintervalsnode* 属性 (继续)

<b>timeintervalsnode</b> 个属性	数据类型	属性描述
week_begins_on	Sunday  Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  Sunday	对于周期性区间（一周中的天、一天中的小时、一天中的分钟和一天中的秒），指定一周开始的那一天。
day_begins_hour	整数	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的小时。可以与 <i>day_begins_minute</i> 和 <i>day_begins_second</i> 结合起来使用，以指定一个准确时间，例如 <b>8:05:01</b> 。请参见下面的使用示例。
day_begins_minute	整数	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的分钟（例如 <b>8:05</b> 中的 <b>5</b> ）。
day_begins_second	整数	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天开始的秒（例如 <b>8:05:17</b> 中的 <b>17</b> ）。
days_per_week	整数	对于周期性区间（一周中的天、一天中的小时、一天中的分钟和一天中的秒），指定一周中的天数。
hours_per_day	整数	对于周期性区间（一天中的小时、一天中的分钟和一天中的秒），指定一天中的小时数。

表 94: <i>timeintervalnode</i> 属性 (继续)		
<b>timeintervalnode</b> 个属性	数据类型	属性描述
interval_increment	1 2 3 4 5 6 10 15 20 30	对于一天中的分钟和一天中的秒，指定为每个记录增加的分钟数或秒数。
field_name_extension	<i>string</i>	
field_name_extension_as_prefix	标志	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	

表 94: *timeintervalsnode* 属性 (继续)

<b>timeintervalsnode</b> 个属性	数据类型	属性描述
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
aggregate	Mean  Sum  Mode  Min  Max  First  Last  TrueIfAnyTrue	指定字段的汇总方法。
pad	Blank  MeanOfRecentPoints  True  False	指定字段的填充方法。
agg_mode	All  Specify	指定是根据需要使用缺省函数汇总或填充所有字段，还是指定要使用的字段和函数。
agg_range_default	Mean  Sum  Mode  Min  Max	指定汇总连续字段时要使用的缺省函数。

表 94: <i>timeintervalsnode</i> 属性 (继续)		
<b>timeintervalsnode</b> 个属性	数据类型	属性描述
agg_set_default	Mode  First  Last	指定汇总名义字段时要使用的缺省函数。
agg_flag_default	TrueIfAnyTrue  Mode  First  Last	
pad_range_default	Blank  MeanOfRecentPoints	指定填充连续字段时要使用的缺省函数。
pad_set_default	Blank  MostRecentValue	
pad_flag_default	Blank  True  False	
max_records_to_create	整数	指定填充序列时要创建的最大记录数。
estimation_from_beginning	标志	
estimation_to_end	标志	
estimation_start_offset	整数	
estimation_num_holdouts	整数	
create_future_records	标志	
num_future_records	整数	
create_future_field	标志	
future_field_name	<i>string</i>	

## transposenode 属性



“转置”节点会交换行和列中的数据，以便记录成为字段，字段成为记录。

示例

```
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")
```

表 95: *transposenode* 属性

transposenode 个属性	数据类型	属性描述
transpose_method	<i>enum</i>	指定转置方法：正常 (normal)、CASE 到 VAR (casetovar) 或 VAR 到 CASE (vartocase)。
transposed_names	Prefix Read	“正常”转置方法的属性。可以根据指定前缀自动生成新字段名，也可以从数据的现有字段中读取新字段名。
prefix	<i>string</i>	“正常”转置方法的属性。
num_new_fields	整数	“正常”转置方法的属性。使用前缀时，指定要创建的新字段的最大数目。
read_from_field	<i>field</i>	“正常”转置方法的属性。从中读取名称的字段。此字段必须是一个实例化字段，否则执行节点时将出错。
max_num_fields	整数	“正常”转置方法的属性。当从某个字段中读取名称时，指定上限以避免创建过大的字段数。
transpose_type	Numeric String Custom	“正常”转置方法的属性。缺省情况下，只能转置连续（数值范围）字段，但也可以选择数字字段的自定义子集或转置所有字符串字段。
transpose_fields	列表	“正常”转置方法的属性。指定在使用 Custom 选项时要转置的字段。
id_field_name	<i>field</i>	“正常”转置方法的属性。
transpose_casetovar_id_fields	<i>field</i>	“CASE 到 VAR”(casetovar) 转置方法的属性。接受要用作索引字段的多个字段。  field1 ... fieldN
transpose_casetovar_columnfields	<i>field</i>	“CASE 到 VAR”(casetovar) 转置方法的属性。接受要用作列字段的多个字段。  field1 ... fieldN
transpose_casetovar_valuefields	<i>field</i>	“CASE 到 VAR”(casetovar) 转置方法的属性。接受要用作值字段的多个字段。  field1 ... fieldN



表 95: *transposenode* 属性 (继续)

transposenode 个属性	数据类型	属性描述
transpose_vartocase_id fields	<i>field</i>	“VAR 到 CASE”(vartocase) 转置方法的属性。接受要用作标识变量字段的多个字段。  field1 ... fieldN
transpose_vartocase_val fields	<i>field</i>	“VAR 到 CASE”(vartocase) 转置方法的属性。接受要用作值变量字段的多个字段。  field1 ... fieldN

## typenode 属性



“类型”节点指定字段元数据和属性。例如，您可以指定每个字段的测量级别（连续、名义、有序或标志）、设置用于处理缺失值和系统 Null 值的选项、设置用于建模的字段的角色、指定字段标签和值标签以及为字段指定值。

### 示例

```
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"]])
```

注意，某些情况下可能需要完全实例化类型节点才能使其他节点正常运行，例如，设为标志节点的 **fields from** 属性。可以只连接表节点并执行该节点以实例化这些字段：

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

表 96: *typenode* 属性

typenode 个属性	数据类型	属性描述
direction	Input  Target  Both  None  Partition  Split  Frequency  RecordID	字段角色的键控属性。  <b>注:</b> 现在不推荐使用值 In 和 Out。在将来的版本中可能取消对这些值的支持。
type	Range  Flag  Set  Typeless  Discrete  OrderedSet  Default	字段的测量级别（先前称为字段的“类型”）。将 type 设置为 Default 会清除所有 values 参数设置，并且如果 value_mode 的值为 Specify，那么它将重置为 Read。如果 value_mode 设置为 Pass 或 Read，那么设置 type 不会影响 value_mode。  <b>注:</b> 内部使用的数据类型与类型节点中的那些变量不同。对应关系如下所示：范围 -> 连续集合 -> 名义有序集合 -> 有序离散 -> 分类
storage	Unknown  String  Integer  Real  Time  Date  Timestamp	字段存储类型的只读键控属性。

表 96: <i>typenode</i> 属性 (继续)		
typenode 个属性	数据类型	属性描述
check	None  Nullify  Coerce  Discard  Warn  Abort	字段类型和范围检查的键控属性。
values	[value value]	对于连续型字段而言，第一个是最小值，后一个是最大值。对于名义字段，指定所有值。对标志字段而言，第一个值代表 <i>false</i> ，后一个值代表 <i>true</i> 。设置该属性将自动把 <i>value_mode</i> 属性设置为 <i>Specify</i> 。
value_mode	Read  Pass  Read+  Current  Specify	确定值的设置方式。注意，不能将此属性直接设置为 <i>Specify</i> ；要使用特定值，需设置 <i>values</i> 属性。
extend_values	标志	当 <i>value_mode</i> 设置为 <i>Read</i> 时将应用。设为 <i>T</i> 则将新读取的值添加到任意现有字段值。设置为 <i>F</i> 则丢弃现有值并添加新读取值。
enable_missing	标志	当设置为 <i>T</i> 时，则激活对字段缺失值的跟踪。
missing_values	[value value ...]	指定表示缺失数据的数据值。
range_missing	标志	指定是否为字段定义缺失值（空白）范围。
missing_lower	string	当 <i>range_missing</i> 为真时，指定缺失值范围的下限。
missing_upper	string	当 <i>range_missing</i> 为真时，指定缺失值范围的上限。
null_missing	标志	设置为 <i>T</i> 时，空（在软件中显示为 <i>\$null\$</i> 的未定义值）被视为缺失值。
whitespace_missing	标志	设置为 <i>T</i> 时，仅包含空白（空格、制表符和换行符）的值被视为缺失值。

表 96: *typenode* 属性 (继续)

<b>typenode 个属性</b>	<b>数据类型</b>	<b>属性描述</b>
description	string	为字段指定说明。
value_labels	[[Value LabelString] [ Value LabelString] ...]	用于为值对指定标签。
display_places	整数	为字段设置显示的小数位数（仅用于以 REAL 存储的字段）。值为 -1 将使用流缺省值。
export_places	整数	为字段设置导出时的小数位数（仅用于以 REAL 存储的字段）。值为 -1 将使用流缺省值。
decimal_separator	DEFAULT  PERIOD  COMMA	为字段设置十进制分隔符（仅用于以 REAL 存储的字段）。
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	为字段设置日期格式（仅用于以 DATE 或 TIMESTAMP 存储的字段）。
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	为字段设置时间格式（仅用于以 TIME 或 TIMESTAMP 存储的字段）。

表 96: <i>typenode</i> 属性 (继续)		
typenode 个属性	数据类型	属性描述
number_format	DEFAULT  STANDARD  SCIENTIFIC  CURRENCY	为字段设置数字显示格式。
standard_places	整数	为字段设置以标准格式显示时的小数位数。值为 -1 将使用流缺省值。请注意，现有的 <code>display_places</code> 通道也会更改此属性，但目前已不再使用。
scientific_places	整数	为字段设置以科学计数格式显示时的小数位数。值为 -1 将使用流缺省值。
currency_places	整数	为字段设置以货币格式显示时的小数位数。值为 -1 将使用流缺省值。
grouping_symbol	DEFAULT  NONE  LOCALE  PERIOD  COMMA  SPACE	为字段设置分组符号。
column_width	整数	为字段设置列宽度。值为 -1 表示将列宽度设置为 <code>Auto</code> 。
justify	AUTO  CENTER  LEFT  RIGHT	为字段设置列对齐格式。

表 96: *typenode* 属性 (继续)

<b>typenode 个属性</b>	<b>数据类型</b>	<b>属性描述</b>
<code>measure_type</code>	Range / <code>MeasureType.RANGE</code> Discrete / <code>MeasureType.DISCRETE</code> Flag / <code>MeasureType.FLAG</code> Set / <code>MeasureType.SET</code> OrderedSet / <code>MeasureType.ORDERED_SET</code> Typeless / <code>MeasureType.TYPELESS</code> Collection / <code>MeasureType.COLLECTION</code> Geospatial / <code>MeasureType.GEOSPATIAL</code>	这个键控属性与 <code>type</code> 的相似之处在于，它可以用于定义与字段相关联的测量。区别之处在于，在 Python 脚本编写中，还可以向 <code>setter</code> 函数传递其中一个 <code>MeasureType</code> 值，而 <code>getter</code> 将始终返回 <code>MeasureType</code> 值。
<code>collection_measure</code>	Range / <code>MeasureType.RANGE</code> Flag / <code>MeasureType.FLAG</code> Set / <code>MeasureType.SET</code> OrderedSet / <code>MeasureType.ORDERED_SET</code> Typeless / <code>MeasureType.TYPELESS</code>	对于集合字段（深度为 0 的列表），此键控属性定义与底层值相关联的测量类型。
<code>geo_type</code>	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	对于地理空间字段，此键控属性定义此字段所表示的地理空间对象的类型。这应该与这些值的列表深度一致。
<code>has_coordinate_system</code>	<code>BOOLEAN</code>	对于地理空间字段，此属性定义此字段是否具有坐标系
<code>coordinate_system</code>	<code>string</code>	对于地理空间字段，此键控属性定义此字段的坐标系。

表 96: <i>typenode</i> 属性 (继续)		
typenode 个属性	数据类型	属性描述
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	这个键控属性与 custom_storage 的相似之处在于，它可以用于定义字段的覆盖存储。区别之处在于，在 Python 脚本编写中，还可以向 setter 函数传递其中一个 StorageType 值，而 getter 将始终返回 StorageType 值。
custom_list_storage_type	String / MeasureType.STRING  Integer / MeasureType.INTEGER  Real / MeasureType.REAL  Time / MeasureType.TIME  Date / MeasureType.DATE  Timestamp / MeasureType.TIMESTAMP	对于列表字段，此键控属性指定底层值的存储类型。
custom_list_depth	整数	对于列表字段，此键控属性指定字段的深度。
max_list_length	整数	仅适用于测量级别为地理空间或集合的数据。通过指定列表可以包含的元素数目来设置列表的最大长度。
max_string_length	整数	仅适用于无类型数据，而且在生成 SQL 以创建表时使用。输入数据中最大字符串的值；这样会在表中生成一个足够大的列以包含该字符串。





# 第 12 章 图形节点属性

## 图形节点公共属性

本节介绍图形节点的可用属性，包括通用属性和每种节点类型特有的属性。

表 97: 通用图形节点属性		
通用图形节点属性	数据类型	属性描述
title	string	指定标题。 示例："This is a title."
caption	string	指定文字说明。 示例："This is a caption."
output_mode	Screen	指定是显示图形节点的输出还是将其写到文件中。
	File	
output_format	BMP	指定输出类型。 允许每个节点使用的确切输出类型是不同的。
	JPEG	
	PNG	
	HTML	
	output (.cou)	
full_filename	string	为从图形节点生成的输出指定目标路径和文件名。
use_graph_size	标志	控制是否使用下面的宽度和高度属性明确调整图形的大小。 只影响输出到屏幕的图形。 不适用于分布节点。
graph_width	成员	当 use_graph_size 为 True 时，以像素为单位设置图形宽度。
graph_height	成员	当 use_graph_size 为 True 时，以像素为单位设置图形高度。

### 关闭可选字段

通过将属性值设置为 " "（空字符串），可以将可选字段（如图的重叠字段）关闭，如下以示例所示：

```
plotnode.setPropertyValue("color_field", "")
```

### 指定颜色

使用十六进制字符串（从井号(#)开始），可指定标题、标注、背景和标签的颜色。例如，要将图形背景设置为天蓝色，请使用以下语句：

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

此处，前两位数 87 指定红色内容；中间两位数 CE 指定绿色内容；最后两位数 EB 指定蓝色内容。每个数字都可以取 0–9 或 A–F 范围内的值。这些值可以一起指定红-绿-蓝（或 RGB）颜色。

注: 以 RGB 形式指定颜色时, 可以使用用户界面中的字段选择器确定正确的颜色代码。只需将鼠标停留在颜色上面就可激活含所需信息的工具提示。

## collectionnode 属性



“收集”节点显示一个数字字段的值相对于另一个数字字段的值的分布。（它创建类似于直方图的图形。）图示说明值不断变化的变量或字段时, 它是有用的。使用 3-D 图形表示时, 还可以使用按分类显示分布的符号轴。

示例

```
node = stream.create("collection", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

表 98: collectionnode 属性

collectionnode 个属性	数据类型	属性描述
over_field	<i>field</i>	
over_label_auto	标志	
over_label	<i>string</i>	
collect_field	<i>field</i>	
collect_label_auto	标志	
collect_label	<i>string</i>	
three_D	标志	
by_field	<i>field</i>	
by_label_auto	标志	
by_label	<i>string</i>	

表 98: collectionnode 属性 (继续)

collectionnode 个属性	数据类型	属性描述
operation	Sum Mean Min Max SDev	
color_field	string	
panel_field	string	
animation_field	string	
range_mode	Automatic UserDefined	
range_min	成员	
range_max	成员	
bins	ByNumber ByWidth	
num_bins	成员	
bin_width	成员	
use_grid	标志	
graph_background	color	本节在开头处介绍了标准图形颜色。
page_background	color	本节在开头处介绍了标准图形颜色。

## distributionnode 属性



“分布”节点显示符号（分类）值（例如抵押类型或性别）的出现次数。通常，您可以使用“分布”节点来显示数据中的不平衡，然后可以在创建模型前使用“均衡”节点来纠正此类不平衡。

示例

```
node = stream.create("distribution", "My node")
# "Plot" tab
node.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

表 99: <i>distributionnode</i> 属性		
<b>distributionnode</b> 个属性	数据类型	属性描述
plot	SelectedFields Flags	
x_field	<i>field</i>	
color_field	<i>field</i>	重叠字段。
normalize	标志	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	标志	

## evaluationnode 属性



“评估”节点有助于评估和比较预测模型。评估图表显示模型预测特定结果的优劣程度。它根据预测值和预测置信度对记录进行排序。它将记录分成若干个相同大小的组（**分位数**），然后从高到底为每个分位数划分业务标准值。在散点图中，将以单独的线条显示多个模型。

示例

```
node = stream.create("evaluation", "My node")
# "Plot" tab
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

表 100: <i>evaluationnode</i> 属性		
<b>evaluationnode</b> 个属性	数据类型	属性描述
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	标志	

表 100: evaluationnode 属性 (继续)		
evaluationnode 个属性	数据类型	属性描述
field_detection_method	Metadata Name	
use_fixed_cost	标志	
cost_value	成员	
cost_field	string	
use_fixed_revenue	标志	
revenue_value	成员	
revenue_field	string	
use_fixed_weight	标志	
weight_value	成员	
weight_field	field	
n_tile	Quartiles  Quintiles  Deciles  Vingtiles  Percentiles  1000-tiles	
cumulative	标志	
style	Line  Point	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	

表 100: *evaluationnode* 属性 (继续)

<b>evaluationnode 个属性</b>	<b>数据类型</b>	<b>属性描述</b>
export_data	标志	
data_filename	<i>string</i>	
delimiter	<i>string</i>	
new_line	标志	
inc_field_names	标志	
inc_best_line	标志	
inc_business_rule	标志	
business_rule_condition	<i>string</i>	
plot_score_fields	标志	
score_fields	<i>[field1 ... fieldN]</i>	
target_field	<i>field</i>	
use_hit_condition	标志	
hit_condition	<i>string</i>	
use_score_expression	标志	
score_expression	<i>string</i>	
caption_auto	标志	

## graphboardnode 属性



“图形板”节点在单个节点中提供许多不同类型的图形。使用此节点，可以选择要探索的数据字段，然后从适用于选定数据的字段中选择一个图形。节点将自动过滤出适用于字段选项的所有图形类型。

**注:** 如果您设置对图形类型无效的属性（例如，为直方图指定 *y\_field*），该属性将被忽略。

**注:** 在 UI 中许多不同图形类型的“详细信息”选项卡上，有一个**摘要**字段；脚本编制当前不支持此字段。

示例

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

表 101: graphboardnode 属性		
graphboard 个属性	数据类型	属性描述
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues	标识图形类型。

表 101: graphboardnode 属性 (继续)

graphboard 个属性	数据类型	属性描述
	ChoroplethCounts	
	CoordinateMap	
	CoordinateChoroplethMeans	
	CoordinateChoroplethMedians	
	CoordinateChoroplethSums	
	CoordinateChoroplethValues	
	CoordinateChoroplethCounts	
	Dotplot	
	Heatmap	
	HexBinScatter	
	Histogram	
	Line	
	LineChartMap	
	LineOverlayMap	
	Parallel	
	Path	
	Pie	
	PieCountMap	
	PieCounts	
	PieMap	



表 101: graphboardnode 属性 (继续)		
graphboard 个属性	数据类型	属性描述
	PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	
x_field	field	为 x 轴指定自定义标签。只适用于标签。
y_field	field	为 y 轴指定自定义标签。只适用于标签。
z_field	field	在某些三维图中使用。
color_field	field	在热图中使用。
size_field	field	在气泡散点图中使用。
categories_field	field	
values_field	field	
rows_field	field	
columns_field	field	
fields	field	
start_longitude_field	field	与参考图中的箭头配合使用。
end_longitude_field	field	
start_latitude_field	field	
end_latitude_field	field	
data_key_field	field	用于各种图。
panelrow_field	string	
panelcol_field	string	
animation_field	string	
longitude_field	field	与地图中的坐标配合使用。
latitude_field	field	
map_color_field	field	

# histogramnode 属性



“直方图”节点显示数字字段的值的出现次数。此节点经常用来在进行数据操作和模型构建前探索数据。与“分布”节点相似，“直方图”节点经常用来揭示数据中的不平衡。

示例

```
node = stream.create("histogram", "My node")
# "Plot" tab
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

表 102: histogramnode 属性		
histogramnode 个属性	数据类型	属性描述
field	field	
color_field	field	
panel_field	field	
animation_field	field	
range_mode	Automatic UserDefined	
range_min	成员	
range_max	成员	
bins	ByNumber ByWidth	
num_bins	成员	
bin_width	成员	
normalize	标志	
separate_bands	标志	
x_label_auto	标志	
x_label	string	
y_label_auto	标志	
y_label	string	
use_grid	标志	

表 102: <i>histogramnode</i> 属性 (继续)		
histogramnode 个属性	数据类型	属性描述
graph_background	color	本节在开头处介绍了标准图形颜色。
page_background	color	本节在开头处介绍了标准图形颜色。
normal_curve	标志	指出是否应在输出中显示正态分布曲线。

# mapvisualization 属性



“地图可视化”节点可以接受多个输入连接，并在地图上将地理空间数据显示为一系列层。每个层都是单个地理空间字段；例如，底层可能是国家或地区的地图，在其之上可能存在一个道路层、一个河流层和一个城镇层。

表 103: <i>mapvisualization</i> 属性		
mapvisualization 个属性	数据类型	属性描述
tag	string	设置输入的标记名称。缺省标记是一个数字，基于输入连接到节点的顺序（第一个连接标记是 1，第二个连接标记是 2，依此类推）
layer_field	field	<p>选择将数据集中的哪个地理字段显示为地图上的层。缺省选择基于下列排序顺序：</p> <ul style="list-style-type: none"> <li>• 第一个 - 点</li> <li>• 线串</li> <li>• 多边形</li> <li>• 多点</li> <li>• MultiLinestring</li> <li>• 最后一个 - 多个多边形</li> </ul> <p>如果有两个字段具有相同测量类型，缺省情况下将选择按名称字母顺序排列的第一个字段。</p>
color_type	BOOLEAN	指定是对所有地理字段特征应用标准颜色，还是使用重叠字段，以根据数据集中另一个字段的值对特征应用颜色。可能的值是 <b>standard</b> 或 <b>overlay</b> 。缺省值为 <b>standard</b> 。
color	string	<p>如果为 <b>color_type</b> 选择了 <b>standard</b>，那么下拉列表将包含与用户选项“显示”选项卡上的图表类别颜色顺序相同的调色板。</p> <p>缺省值是图表类别颜色 1。</p>
color_field	field	如果为 <b>color_type</b> 选择了 <b>overlay</b> ，那么下拉菜单将包含与选作层的地理字段相同数据集中的所有字段。

表 103: mapvisualization 属性 (继续)

mapvisualization 个属性	数据类型	属性描述
symbol_type	BOOLEAN	指定是对所有地理字段记录应用标准符号，还是选择重叠符号，这样会基于数据集中另一字段的值更改点的符号图标。可能的值是 standard 或 overlay。缺省值为 standard。
symbol	string	如果为 symbol_type 选择了 standard，那么下拉列表将包含可以用于显示地图上的点的符号选择。
symbol_field	field	如果为 symbol_type 选择了 overlay，那么下拉菜单将包含与选作层的地理字段相同数据集中的所有名义、有序或分类字段。
size_type	BOOLEAN	指定是对所有地理字段记录应用标准大小，还是选择重叠大小，这样会基于数据集中另一字段的值更改符号图标的大小或线条粗细。可能的值是 standard 或 overlay。缺省值为 standard。
size	string	如果为 size_type 选择了 standard，那么对于 point 或 multipoint，下拉菜单将包含适用于所选符号的大小选择。对于 linestring 或 multilinestring，该下拉列表将包含线条粗细选择。
size_field	field	如果为 size_type 选择了 overlay，那么下拉菜单将包含与选作层的地理字段相同数据集中的所有字段。
transp_type	BOOLEAN	指定是对所有地理字段记录应用标准透明度，还是选择重叠透明度，这样会基于数据集中另一字段的值更改符号、线条或多边形的透明度。可能的值是 standard 或 overlay。缺省值为 standard。
transp	整数	<p>如果为 transp_type 选择了 standard，那么下拉菜单将包含透明度选择，从 0%（不透明）增加到 100%（透明），增量为 10%。设置地图上的点、线条或多边形的透明度。</p> <p>如果为 size_type 选择了 overlay，那么下拉菜单将包含与选作层的地理字段相同数据集中的所有字段。</p> <p>对于作为底层的 points、multipoints、linestrings 和 multilinestrings、polygons 和 multipolygons，缺省值为 0%。对于不作为底层的 polygons 和 multipolygons，缺省值为 50%（以避免遮挡这些多边形下方的图层）。</p>

表 103: mapvisualization 属性 (继续)		
mapvisualization 个属性	数据类型	属性描述
transp_field	field	如果为 transp_type 选择了 overlay, 那么下拉菜单将包含与选作层的地理字段相同数据集中的所有字段。
data_label_field	field	指定要用作地图上的数据标签的字段。例如, 如果对其应用该设置的层是多边形层, 那么数据标签可能是 name 字段 - 包含每个多边形的名称。因此在此选择 name 字段可能会导致在地图上显示这些名称。
use_hex_binning	BOOLEAN	启用十六进制分级并启用所有汇总下拉列表。缺省情况下, 该设置处于关闭状态。
color_aggregation 和 transp_aggregation	string	<p>如果为使用十六进制分级的点层选择了重叠字段, 那么该字段的所有值都必须针对该六边形内的所有点进行汇总。请为要应用于地图的任何重叠字段指定聚集函数。</p> <p>可用的汇总函数为:</p> <p>连续 (实数或整数存储) :</p> <ul style="list-style-type: none"> <li>• 总和</li> <li>• 平均值</li> <li>• 最小值</li> <li>• 最大值</li> <li>• 中位数</li> <li>• 第一个四分位</li> <li>• 第三个四分位</li> </ul> <p>连续 (时间、日期或时间戳记存储) :</p> <ul style="list-style-type: none"> <li>• 平均值</li> <li>• 最小值</li> <li>• 最大值</li> </ul> <p>名义/分类:</p> <ul style="list-style-type: none"> <li>• 众数</li> <li>• 最小值</li> <li>• 最大值</li> </ul> <p>标志:</p> <ul style="list-style-type: none"> <li>• 如果任意一项为 true 则为 true</li> <li>• 如果任意一项为 false 则为 false</li> </ul>
custom_storage	string	设置字段的总体存储类型。缺省值为 List。如果指定了 List, 那么将禁用以下 custom_value_storage 和 list_depth 控制。
custom_value_storage	string	设置列表中的元素的存储类型, 而不是设置为字段的整体存储类型。缺省值为 Real。

表 103: mapvisualization 属性 (继续)

mapvisualization 个属性	数据类型	属性描述
list_depth	整数	<p>设置列表字段的深度。所需深度取决于地理字段类型，遵循以下条件：</p> <ul style="list-style-type: none"> <li>点 - 0</li> <li>LineString - 1</li> <li>多边形 - 2</li> <li>多点 - 1</li> <li>MultiLineString - 2</li> <li>多个多边形 - 3</li> </ul> <p>您必须了解要重新转换为列表的地理空间字段的类型以及该类字段的所需深度。如果设置不正确，则不能使用该字段。</p> <p>缺省值为 0，最小值为 0，最大值为 10。</p>

## multiplotnode 属性



“多重散点图”节点创建在单个 X 字段上显示多个 Y 字段的散点图。Y 字段被绘制为彩色的线；每条线相当于“样式”设置为线且“X 模式”设置为排序的散点图节点。在探索多个变量随时间推移的变化情况时，多重散点图非常有用。

示例

```
node = stream.create("multiplot", "My node")
# "Plot" tab
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# "Overlay" section
node.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

表 104: multiplotnode 属性

multiplotnode 个属性	数据类型	属性描述
x_field	field	
y_fields	列表	
panel_field	field	
animation_field	field	
normalize	标志	
use_overlay_expr	标志	
overlay_expression	string	
records_limit	成员	

表 104: <i>multiplotnode</i> 属性 (继续)		
multiplotnode 个属性	数据类型	属性描述
if_over_limit	PlotBins  PlotSample  PlotAll	
x_label_auto	标志	
x_label	<i>string</i>	
y_label_auto	标志	
y_label	<i>string</i>	
use_grid	标志	
graph_background	<i>color</i>	本节在开头处介绍了标准图形颜色。
page_background	<i>color</i>	本节在开头处介绍了标准图形颜色。

# plotnode 属性



散点图节点可显示数字字段间的关系。可通过使用点（散点）或线创建散点图。

示例

```
node = stream.create("plot", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# "Output" tab
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/
plot_output.jpeg")
```

表 105: <i>plotnode</i> 属性		
plotnode 个属性	数据类型	属性描述
x_field	<i>field</i>	为 x 轴指定自定义标签。只适用于标签。
y_field	<i>field</i>	为 y 轴指定自定义标签。只适用于标签。
three_D	标志	为 y 轴指定自定义标签。只适用于 3 维图形中的标签。

表 105: *plotnode* 属性 (继续)

<b>plotnode</b> 个属性	数据类型	属性描述
<code>z_field</code>	<i>field</i>	
<code>color_field</code>	<i>field</i>	重叠字段。
<code>size_field</code>	<i>field</i>	
<code>shape_field</code>	<i>field</i>	
<code>panel_field</code>	<i>field</i>	指定用于为每个类别绘制单独图表的名义字段或标志字段。图表一起平铺在一个输出窗口中。
<code>animation_field</code>	<i>field</i>	指定以图说明数据值类别（通过使用动画创建一系列按顺序显示的图表来说明）时所使用的名义字段或标志字段。
<code>transp_field</code>	<i>field</i>	指定以图说明数据值类别（通过为每个类别使用不同级别的透明度来说明）时所使用的字段。不适用于线散点图。
<code>overlay_type</code>	None  Smoother  Function	指定是显示重叠函数还是 LOESS 平滑器。
<code>overlay_expression</code>	<i>string</i>	指定当 <code>overlay_type</code> 设置为 Function 时使用的表达式。
<code>style</code>	Point  Line	
<code>point_type</code>	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
<code>x_mode</code>	Sort  Overlay  AsRead	



表 105: <i>plotnode</i> 属性 (继续)		
plotnode 个属性	数据类型	属性描述
x_range_mode	Automatic	
	UserDefined	
x_range_min	成员	
x_range_max	成员	
y_range_mode	Automatic	
	UserDefined	
y_range_min	成员	
y_range_max	成员	
z_range_mode	Automatic	
	UserDefined	
z_range_min	成员	
z_range_max	成员	
jitter	标志	
records_limit	成员	
if_over_limit	PlotBins	
	PlotSample	
	PlotAll	
x_label_auto	标志	
x_label	<i>string</i>	
y_label_auto	标志	
y_label	<i>string</i>	
z_label_auto	标志	
z_label	<i>string</i>	
use_grid	标志	
graph_background	<i>color</i>	本节在开头处介绍了标准图形颜色。
page_background	<i>color</i>	本节在开头处介绍了标准图形颜色。
use_overlay_expr	标志	该属性已由 <i>overlay_type</i> 替代。

## timeplotnode 属性



“时间散点图”节点显示一组或多组时间序列数据。通常，您将首先使用“时间间隔”节点来创建 *TimeLabel* 字段，该字段将用于标注 *x* 轴。

示例

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Appearance settings
node.setPropertyValue("symbol_size", 2.0)
```

表 106: timeplotnode 属性

timeplotnode 个属性	数据类型	属性描述
plot_series	Series Models	
use_custom_x_field	标志	
x_field	field	
y_fields	列表	
panel	标志	
normalize	标志	
line	标志	
points	标志	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	标志	只有将 panel 设置为 True，才可将平滑其添加到图中。
use_records_limit	标志	
records_limit	整数	
symbol_size	成员	指定符号大小。
panel_layout	Horizontal Vertical	

## eplotnode 属性



E-Plot (Beta) 节点显示数字字段之间的关系。它与“绘图”节点类似，但是其选项不同，并且其输出使用特定于此节点的新图形界面。使用 **beta** 级节点可运用新图形功能。

表 107: eplotnode 属性

eplotnode 个属性	数据类型	属性描述
x_field	string	指定要在横轴 X 上显示的字段。
y_field	string	指定要在纵轴 Y 上显示的字段。
color_field	string	指定要用于输出中的颜色映射图重叠的字段（如果需要）。
size_field	string	指定要用于输出中的尺寸图重叠的字段（如果需要）。
shape_field	string	指定要用于输出中的形状图重叠的字段（如果需要）。
interested_fields	string	指定要在输出中包含的字段。
records_limit	整数	指定表示要在输出中绘制的最大记录数的数字。缺省值为 2000。
if_over_limit	BOOLEAN	指定是否使用 Sample 选项或 Use all data 选项（如果已超过 records_limit）。缺省值为 Sample，并且它对数据随机抽样，直至命中 records_limit 为止。如果指定 Use all data 以忽略 records_limit 并绘制所有数据点，请注意，这可能会显著降低性能。

## tsnnode 属性



t-分布随机邻域嵌入 (t-SNE) 是用于可视化高维数据的工具。其将数据点亲缘关系转换为可能性。此 t-SNE 节点在 SPSS 建模器 中使用 Python 进行实现并且需要 scikit-learn® Python 库。

表 108: tsnnode 属性

tsnnode 个属性	数据类型	属性描述
mode_type	string	指定 simple 或 expert 方式。
n_components	string	内嵌空间的维度（2D 或 3D）。指定 2 或 3。缺省值为 2。
method	string	指定 barnes_hut 或 exact。缺省值为 barnes_hut。
init	string	嵌套初始化。指定 random 或 pca。缺省值为 random。

表 108: tsnenode 属性 (继续)

tsnenode 个属性	数据类型	属性描述
target_field  从 V18.2.1.1 开始, 重命名为 target	string	目标字段名称。在输出图形将是颜色映射图。如果未指定目标字段, 那么图像将使用一种颜色。
perplexity	float	困惑度与其他各种学习算法中使用的最近邻元素数量相关。通常, 数据集越大, 需要的困惑度越大。请考虑选择 5 和 50 之间的值。缺省值为 30。
early_exaggeration	float	控制原始空间中的自然聚类在内嵌空间中的紧密程度以及两者之间的空间量。缺省值为 12.0。
learning_rate	float	缺省值为 200。
n_iter	整数	优化的最大迭代次数。至少设置为 250。缺省值为 1000。
angle	float	从一个点度量的远距离节点的角度大小。指定 0-1 范围内的值。缺省值为 0.5。
enable_random_seed	BOOLEAN	设置为 true 以后用 random_seed 参数。缺省值为 false。
random_seed	整数	要使用的随机数种子。缺省值为 None。
n_iter_without_progress	整数	不含进度的最大迭代次数。缺省值为 300。
min_grad_norm	string	如果梯度标准值低于此阈值, 那么优化将停止。缺省值为 1.0E-7。可能的值包括: <ul style="list-style-type: none"> <li>• 1.0E-1</li> <li>• 1.0E-2</li> <li>• 1.0E-3</li> <li>• 1.0E-4</li> <li>• 1.0E-5</li> <li>• 1.0E-6</li> <li>• 1.0E-7</li> <li>• 1.0E-8</li> </ul>
isGridSearch	BOOLEAN	设置为 true 以使用多个不同的困惑度执行 t-SNE。缺省值为 false。
output_Rename	BOOLEAN	如果想要提供定制名称, 那么指定 true, 或者指定 false 以自动命名输出。缺省值为 false。
output_to	string	指定 Screen 或 Output。缺省值为 Screen。
full_filename	string	指定输出文件名。
output_file_type	string	输出文件格式。指定 HTML 或 Output object。缺省值为 HTML。

# webnode 属性



Web 节点说明两个或两个以上符号（分类）字段的值之间的关系强度。此图使用不同粗细的线条来表示连接强度。例如，您可以使用 Web 节点来探索在电子商务站点上购买一组商品之间的关系。

示例

```
node = stream.create("web", "My node")
# "Plot" tab
node.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# "Options" tab
node.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

表 109: webnode 属性		
webnode 个属性	数据类型	属性描述
use_directed_web	标志	
fields	列表	
to_field	field	
from_fields	列表	
true_flags_only	标志	
line_values	Absolute  OverallPct  PctLarger  PctSmaller	
strong_links_heavier	标志	
num_links	ShowMaximum  ShowLinksAbove  ShowAll	
max_num_links	成员	

表 109: <i>webnode</i> 属性 (继续)		
<b>webnode</b> 个属性	数据类型	属性描述
links_above	成员	
discard_links_min	标志	
links_min_records	成员	
discard_links_max	标志	
links_max_records	成员	
weak_below	成员	
strong_above	成员	
link_size_continuous	标志	
web_display	Circular  Network  Directed  Grid	
graph_background	<i>color</i>	本节在开头处介绍了标准图形颜色。
symbol_size	成员	指定符号大小。

# 第 13 章 建模节点属性

## 公共建模节点属性

以下属性通用于某些或所有建模节点。所有例外情况均根据需要记录在各个建模节点的文档中。

表 110: 公共建模节点属性		
属性	值	属性描述
custom_fields	标志	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
target	field	根据模型类型指定一个目标字段或多个目标字段。
或者	或者	
targets	[field1 ... fieldN]	
inputs	[field1 ... fieldN]	模型所使用的输入或预测变量字段。
partition	field	
use_partitioned_data	标志	如果定义了分区字段，则此选项可确保仅训练分区的数据用于构建模型。
use_split_data	标志	
splits	[field1 ... fieldN]	指定一个或多个用于拆分建模的字段。仅在 use_split_data 设置为 True 时有效。
use_frequency	标志	特定模型所使用的权重和频率字段（参见每种模型类型的说明）。
frequency_field	field	
use_weight	标志	
weight_field	field	
use_model_name	标志	
model_name	string	新模型的定制名称。
mode	Simple	
	Expert	

## anomalydetectionnode 属性



Anomaly Detection 节点确定不符合“正常”数据格式的异常观测值（离群值）。通过此节点，即使离群值不符合任何先前已知的模式，即使您并不确定要查找的内容，也可以识别这些离群值。

## 示例

```
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)
```

表 111: anomalydetectionnode 属性

anomalydetectionnode 属性	值	属性描述
inputs	[field1 ... fieldN]	异常检测模型将根据指定的输入字段筛选记录。它们不使用目标字段。也不使用权重和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	指定用于确定将记录标记为异常的分界值的方法。
index_level	成员	指定用于标记异常的最小分界值。
percent_records	成员	根据训练数据中的记录百分比设置用于标记记录的阈值。
num_records	成员	根据训练数据中的记录数设置用于标记记录的阈值。
num_fields	整数	要对每条异常记录报告的字段数。
impute_missing_values	标志	
adjustment_coeff	成员	此值用于平衡计算距离时指定给连续字段和分类字段的相对权重。
peer_group_num_auto	标志	自动计算对等组数。
min_num_peer_groups	整数	指定 peer_group_num_auto 设置为 True 时使用的对等组的最小数目。
max_num_per_groups	整数	指定最大对等组数。
num_peer_groups	整数	指定 peer_group_num_auto 设置为 False 时使用的对等组数。
noise_level	成员	确定聚类期间离群值的处理方式。请指定介于 0 到 0.5 之间的值。
noise_ratio	成员	指定分配给组件的应该用于噪声缓存的内存量。请指定介于 0 到 0.5 之间的值。



# apriorinode 属性



“先验”节点从数据抽取一组规则，即抽取信息内容最多的规则。Apriori 节点提供五种选择规则的方法并使用复杂的索引模式来高效地处理大数据集。对于较大的问题，Apriori 训练的速度通常较快；它对可保留的规则数量没有任何限制，而且可处理最多带有 32 个前提条件的规则。“先验”要求输入和输出字段均为分类型字段，但因为它专为处理此类型数据而进行优化，因而处理速度快得多。

示例

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

表 112: apriorinode 属性

apriorinode 属性	值	属性描述
consequents	<i>field</i>	Apriori 模型使用“结果”和“条件”代替标准目标和输入字段。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
antecedents	<i>[field1 ... fieldN]</i>	
min_supp	成员	
min_conf	成员	
max_antecedents	成员	
true_flags	标志	
optimize	Speed Memory	
use_transactional_data	标志	值为 <b>true</b> 时，每个事务标识的评分与其他事务标识无关。如果要评分的数据太大而无法获得可接受的性能，我们建议将数据分开。

表 112: *apriorinode* 属性 (继续)

<b>apriorinode</b> 属性	值	属性描述
contiguous	标志	
id_field	<i>string</i>	
content_field	<i>string</i>	
mode	Simple Expert	
evaluation	RuleConfidence  DifferenceToPrior  ConfidenceRatio  InformationDifference  NormalizedChiSquare	
lower_bound	成员	
optimize	Speed  Memory	用于指定优化建模的方式是速度还是内存。

## associationrulesnode 属性



“关联规则”节点与 Apriori 节点类似；但是，与 Apriori 不同，“关联规则”节点能够处理列表数据。另外，“关联规则”节点可以与 IBM SPSS 分析服务器 配合使用，以处理大型数据以及利用更快的并行处理功能。

表 113: *associationrulesnode* 属性

<b>associationrulesnode</b> 个属性	数据类型	属性描述
predictions	<i>field</i>	此列表中的字段只能显示为规则的预测变量
conditions	<i>[field1...fieldN]</i>	此列表中的字段只能显示为规则的条件
max_rule_conditions	整数	单条规则中可以包含的最大条件数。最小值为 1，最大值为 9。
max_rule_predictions	整数	单条规则中可以包含的最大预测数。最小值为 1，最大值为 5。
max_num_rules	整数	在规则构建过程中可以考虑的最大规则数。最小值为 1，最大值为 10,000。

表 113: associationrulesnode 属性 (继续)

associationrulesnode 个属性	数据类型	属性描述
rule_criterion_top_n	Confidence  Rulesupport  Lift  Conditionsupport  Deployability	规则条件，此条件确定一个值，依据此值选择模型中的前“N”条规则。
true_flags	BOOLEAN	设置为 Y 确定规则构建期间仅考虑标志字段的 true 值。
rule_criterion	BOOLEAN	设置为 Y 确定模型构建期间使用规则条件值来排除规则。
min_confidence	成员	0.1 到 100 - 模型所产生规则的最小必需置信度水平的百分比值。如果模型产生了置信度水平小于此处指定的值的规则，那么将废弃该规则。
min_rule_support	成员	0.1 到 100 - 模型所产生规则的最小必需规则支持度的百分比值。如果模型产生了规则支持度水平小于指定值的规则，那么将废弃该规则。
min_condition_support	成员	0.1 到 100 - 模型所产生规则的最小必需条件支持度的百分比值。如果模型产生了条件支持度水平小于指定值的规则，那么将废弃该规则。
min_lift	整数	1 到 10 - 表示模型所产生规则的最小必需增益。如果模型产生了增益级别小于指定值的规则，那么将废弃该规则。
exclude_rules	BOOLEAN	用于选择您不希望模型在创建规则时使用的一系列相关字段。  示例：set :gsarsnode.exclude_rules = [[field1,field2, field3],[field4, field5]] - 其中由 [] 分隔的每个字段列表都是表中的一行。
num_bins	整数	设置用于对连续字段进行分级的自动分级数。最小值为 2，最大值为 10。
max_list_length	整数	应用于最大长度未知的任何列表字段。列表中此处所指定数字之前的元素将包含在模型构建中；任何后续元素都将被废弃。最小值为 1，最大值为 100。
output_confidence	BOOLEAN	
output_rule_support	BOOLEAN	
output_lift	BOOLEAN	

表 113: associationrulesnode 属性 (继续)

associationrulesnode 个属性	数据类型	属性描述
output_condition_support	BOOLEAN	
output_deployability	BOOLEAN	
rules_to_display	upto all	要在输出表中显示的最大规则数。
display_upto	整数	如果在 rules_to_display 中设置了 upto, 请设置要在输出表中显示的规则数。最小值为 1。
field_transformations	BOOLEAN	
records_summary	BOOLEAN	
rule_statistics	BOOLEAN	
most_frequent_values	BOOLEAN	
most_frequent_fields	BOOLEAN	
word_cloud	BOOLEAN	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	整数	最小值为 1, 最大值为 20
max_predictions	整数	可以应用于分数的每个输入的最大规则数。
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	选择用于确定规则强度的测量。
allow_repeats	BOOLEAN	确定是否将预测相同的规则包括在分数中。

表 113: associationrulesnode 属性 (继续)		
associationrulesnode 个属性	数据类型	属性描述
check_input	NoPredictions	
	Predictions	
	NoCheck	

## autoclassifiernode 属性



“自动分类器”节点用于创建和对比二元结果（是或否，流失或不流失等）的若干不同模型，使用户可以选择给定分析的最佳处理方法。由于支持多种建模算法，因此可以对用户希望使用的方法、每种方法的特定选项以及对比结果的标准进行选择。该节点根据指定的选项生成一组模型，并根据您指定的条件对最佳候选项进行排序。

示例

```
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)
```

表 114: autoclassifiernode 属性		
autoclassifiernode 属性	值	属性描述
target	field	对于标志目标，“自动分类器”节点要求单个目标字段以及一个或多个输入字段。也可以使用权重和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	整数	要包括在模型块中的模型数。指定整数必须为 1 到 100 之间。
calculate_variable_importance	标志	

表 114: autoclassifiernode 属性 (继续)

autoclassifiernode 属性	值	属性描述
enable_accuracy_limit	标志	
accuracy_limit	整数	介于 0 与 100 之间的整数。
enable_area_under_curve_limit	标志	
area_under_curve_limit	成员	介于 0.0 与 1.0 之间的实数。
enable_profit_limit	标志	
profit_limit	成员	大于 0 的整数。
enable_lift_limit	标志	
lift_limit	成员	这是大于 1.0 的实数。
enable_number_of_variables_limit	标志	
number_of_variables_limit	成员	大于 0 的整数。
use_fixed_cost	标志	
fixed_cost	成员	这是大于 0.0 的实数。
variable_cost	<i>field</i>	
use_fixed_revenue	标志	
fixed_revenue	成员	这是大于 0.0 的实数。
variable_revenue	<i>field</i>	
use_fixed_weight	标志	
fixed_weight	成员	这是大于 0.0 的实数。
variable_weight	<i>field</i>	
lift_percentile	成员	介于 0 与 100 之间的整数。
enable_model_build_time_limit	标志	
model_build_time_limit	成员	设置为分钟数的整数，用于限制构建各个模型所花费的时间。
enable_stop_after_time_limit	标志	
stop_after_time_limit	成员	设置为小时数的实数集，用于限制运行自动分类器的总耗时。
enable_stop_after_valid_model_produced	标志	
use_costs	标志	
<algorithm>	标志	允许或禁止使用特定算法。
<algorithm>.<property>	<i>string</i>	设置特定算法的属性值。有关更多信息，请参阅主题 <a href="#">第 197 页的『设置算法属性』</a> 。

# 设置算法属性

对于“自动分类器”、“自动数字”和“自动聚类”节点，可以使用通用格式来设置节点所使用的特定算法的属性：

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

例如：

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

用于自动分类器节点的算法名称有 cart、chaid、quest、c50、logreg、decisionlist、bayesnet、discriminant、svm 和 knn。

用于自动数值节点的算法名称有 cart、chaid、neuralnetwork、genlin、svm、regression、linear 和 knn。

自动聚类节点的算法名称有 twostep、k-means 和 kohonen。

属性名为各节点文档中记录的标准格式。

含有句点和其他标点符号的算法属性必须包括在半角单引号中，例如：

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

也可以为属性分配多个值，例如：

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

要启用或禁用特定算法：

```
node.setPropertyValue("chaid", True)
```

注: 如果某些算法选项在“自动分类器”节点中不可用，或者只能指定单个值而不是某个范围的值，则编写脚本时受到的限制与以标准方式访问节点时一样。

## autoclusternode 属性



“自动聚类”节点估算和比较识别具有类似特征记录组的聚类模型。节点工作方式与其他自动建模节点相同，使您在一次建模运行中即可试验多个选项组合。可使用基本度量对模型进行比较，尝试对聚类模型进行过滤，对其有用性进行排名，并提供基于特定字段重要性的度量。

示例

```
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)
```

表 115: autoclusternode 属性		
autoclusternode 属性	值	属性描述
evaluation	field	注: 仅限于“自动聚类”节点。标识要计算重要性值的字段。另外，可用于标识聚类区分此字段的值的良好程度，从而标识模型预测此字段的良好程度。

表 115: autoclusternode 属性 (继续)

autoclusternode 属性	值	属性描述
ranking_measure	Silhouette  Num_clusters  Size_smallest_cluster  Size_largest_cluster  Smallest_to_largest  Importance	
ranking_dataset	Training  Test	
summary_limit	整数	要在报告中列出的模型的数目。指定整数必须为 1 到 100 之间。
enable_silhouette_limit	标志	
silhouette_limit	整数	介于 0 与 100 之间的整数。
enable_number_less_limit	标志	
number_less_limit	成员	介于 0.0 与 1.0 之间的实数。
enable_number_greater_limit	标志	
number_greater_limit	成员	大于 0 的整数。
enable_smallest_cluster_limit	标志	
smallest_cluster_units	Percentage  Counts	
smallest_cluster_limit_percentage	成员	
smallest_cluster_limit_count	整数	大于 0 的整数。
enable_largest_cluster_limit	标志	
largest_cluster_units	Percentage  Counts	
largest_cluster_limit_percentage	成员	
largest_cluster_limit_count	整数	



表 115: autoclusternode 属性 (继续)		
autoclusternode 属性	值	属性描述
enable_smallest_largest_limit	标志	
smallest_largest_limit	成员	
enable_importance_limit	标志	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	成员	介于 0 与 100 之间的整数。
importance_limit_less_than	成员	介于 0 与 100 之间的整数。
<algorithm>	标志	允许或禁止使用特定算法。
<algorithm>.<property>	string	设置特定算法的属性值。有关更多信息，请参阅主题 第 197 页的『设置算法属性』。

## autonumericnode 属性



自动数字节点使用多种不同方法估计和对比模型的连续数字范围结果。此节点和自动分类器节点的工作方式相同，因此可以选择要使用和要在单个建模传递中使用多个选项组合进行测试的算法。受支持的算法包括神经网络、C&R 树、CHAID、线性回归、广义线性回归以及支持向量机 (SVM)。可以根据相关度、相对误差或使用的变量数来比较模型。

### 示例

```
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)
```

表 116: autonumericnode 属性		
autonumericnode 属性	值	属性描述
custom_fields	标志	如果为 True，将使用定制字段设置代替类型节点设置。
target	field	“自动数字”节点要求单个目标字段以及一个或多个输入字段。也可以使用权重和频率字段。有关更多信息，请参阅主题 第 189 页的『公共建模节点属性』。
inputs	[field1 ... field2]	
partition	field	

表 116: autonumericnode 属性 (继续)

autonumericnode 属性	值	属性描述
use_frequency	标志	
frequency_field	<i>field</i>	
use_weight	标志	
weight_field	<i>field</i>	
use_partitioned_data	标志	如果定义了分区字段，那么仅将培训数据用于模型构建。
ranking_measure	Correlation	
	NumberOfFields	
ranking_dataset	Test	
	Training	
number_of_models	整数	要包括在模型块中的模型数。指定整数必须为 1 到 100 之间。
calculate_variable_importance	标志	
enable_correlation_limit	标志	
correlation_limit	整数	
enable_number_of_fields_limit	标志	
number_of_fields_limit	整数	
enable_relative_error_limit	标志	
relative_error_limit	整数	
enable_model_build_time_limit	标志	
model_build_time_limit	整数	
enable_stop_after_time_limit	标志	
stop_after_time_limit	整数	
stop_if_valid_model	标志	
<algorithm>	标志	允许或禁止使用特定算法。
<algorithm>.<property>	<i>string</i>	设置特定算法的属性值。有关更多信息，请参阅主题 <a href="#">第 197 页的『设置算法属性』</a> 。

# bayesnetnode 属性



通过贝叶斯网络节点，你可以利用对真实世界认知的判断力并结合所观察和记录的证据来构建概率模型。该节点侧重于主要用于分类的树增强朴素贝叶斯 (TAN) 和马尔可夫毯网络。

示例

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

表 117: bayesnetnode 属性

bayesnetnode 属性	值	属性描述
inputs	[field1 ... fieldN]	贝叶斯网络模型使用一个目标字段以及一个或多个输入字段。连续字段将自动进行分箱。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue_training_existing_model	标志	
structure_type	TAN MarkovBlanket	选择要在构建贝叶斯网络时使用的结构。
use_feature_selection	标志	
parameter_learning_method	Likelihood Bayes	指定用于预测父节点的值已知的节点之间的条件概率表的方法。
mode	Expert Simple	
missing_values	标志	
all_probabilities	标志	
independence	Likelihood Pearson	指定用于确定两个变量上的成对观测值是否相互独立的方法。
significance_level	成员	指定用于确定独立性的分界值。
maximal_conditioning_set	成员	设置用于独立性测试的条件变量的最大数。

表 117: bayesnetnode 属性 (继续)

bayesnetnode 属性	值	属性描述
inputs_always_selected	[field1 ... fieldN]	指定构建贝叶斯网络时始终使用的数据集中的字段。  注: 目标字段始终处于选中状态。
maximum_number_inputs	成员	指定构建贝叶斯网络时使用的输入字段的最大数。
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## buildr 属性



“R 构建”节点使您能够输入定制 R 脚本，以执行 IBM SPSSModeler 中部署的模型构建和模型评分。

示例

```
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)""")
```

表 118: buildr 属性

buildr 属性	值	属性描述
build_syntax	string	这是用于进行模型构建的 R 脚本语法。
score_syntax	string	这是用于进行模型评分的 R 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。

表 118: <i>buildr</i> 属性 (继续)		
buildr 属性	值	属性描述
output_html	标志	此选项用于在 R 模型块中的选项卡上显示图形。
output_text	标志	此选项用于将 R 控制台文本输出写至 R 模型块中的选项卡。

## c50node 属性



C5.0 节点构建决策树或规则集。该模型的工作原理是根据在每个级别提供最大信息收获的字段分割样本。目标字段必须为分类字段。允许进行多次多于两个子组的分割。

示例

```
node = stream.create("c50", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tab
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

表 119: <i>c50node</i> 属性		
c50node 属性	值	属性描述
target	<i>field</i>	C50 模型使用单个目标字段以及一个或多个输入字段。还可以指定权重字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
output_type	DecisionTree RuleSet	
group_symbolics	标志	
use_boost	标志	
boost_num_trials	成员	
use_xval	标志	
xval_num_folds	成员	
mode	Simple Expert	

表 119: c50node 属性 (继续)

c50node 属性	值	属性描述
favor	Accuracy Generality	首选准确性或通用性。
expected_noise	成员	
min_child_records	成员	
pruning_severity	成员	
use_costs	标志	
costs	结构化	这是结构化属性。
use_winning	标志	
use_global_pruning	标志	缺省为“启用 (True) 。
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## carmanode 属性



CARMA 模型在不要求用户指定输入或目标字段的情况下从数据抽取一组规则。与 Apriori 相反，CARMA 节点提供规则支持的构建设置（支持前项和后项），而不仅仅是前项支持。这就意味着生成的规则可以用于更多应用程序，例如用于查找产品或服务（前项）的列表，这些产品或服务的后项为想在节日期间促销的商品。

### 示例

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
node.setPropertyValue("vary_support", True)
```

```
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

表 120: carmanode 属性

carmanode 属性	值	属性描述
inputs	[field1 ... fieldn]	CARMA 模型使用输入字段列表，而不使用目标字段。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
id_field	field	用作模型构建标识字段的字段。
contiguous	标志	用于指定标识字段中的标识是否连续。
use_transactional_data	标志	
content_field	field	
min_supp	数字（百分比）	与规则支持相关，而不是与条件支持相关。缺省值为 20%。
min_conf	数字（百分比）	缺省值为 20%。
max_size	成员	缺省值为 10。
mode	Simple Expert	缺省值为 Simple。
exclude_multiple	标志	排除具有多结果的规则。缺省值为 False。
use_pruning	标志	缺省值为 False。
pruning_value	成员	缺省值为 500。
vary_support	标志	
estimated_transactions	整数	
rules_without_antecedents	标志	

## cartnode 属性



分类和回归 (C&R) 树节点生成可用于预测或分类未来观测值的决策树。该方法通过在每个步骤最大限度降低不纯度，使用递归分区来将训练记录分割为组。如果树中某个节点中 100% 的观测值都属于目标字段的一个特定类别，那么该节点将被认定为“纯洁”。目标和输入字段可以是数字范围或分类（名义、有序或标志）；所有拆分都是二进制的（只有两个子组）。

示例

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "" "Grow Node Index 0 Children 1 2")
```

```

Grow Node Index 2 Children 3 4"")
# "Build Options" tab, "Basics" panel
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# "Model Options" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")

```

表 121: *cartnode* 属性

cartnode 属性	值	属性描述
target	<i>field</i>	C&R 树模型需要单个目标字段以及一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue_training_existing_model	标志	
objective	Standard Boosting Bagging psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single InteractiveBuilder	
use_tree_directives	标志	
tree_directives	<i>string</i>	指定生成树的指令。指令可放在三重引号 (""") 中，以防止丢失新行或引号。请注意，指令可能对数据或建模选项的细小更改高度敏感，并且可能无法通用于其他数据集。
use_max_depth	Default Custom	
max_depth	整数	最大树深度，从 0 到 1000。仅在 use_max_depth = Custom 时使用。
prune_tree	标志	修剪树，以避免过度拟合。
use_std_err	标志	使用最大风险差（标准误差）。
std_err_multiplier	成员	最大差值。



表 121: cartnode 属性 (继续)		
cartnode 属性	值	属性描述
max_surrogates	成员	最大代用项。
use_percentage	标志	
min_parent_records_pc	成员	
min_child_records_pc	成员	
min_parent_records_abs	成员	
min_child_records_abs	成员	
use_costs	标志	
costs	结构化	结构化属性。
priors	Data Equal Custom	
custom_priors	结构化	结构化属性。
adjust_priors	标志	
trails	成员	用于推进或组装的组件模型数。
set_ensemble_method	Voting HighestProbability HighestMeanProbabilit y	分类目标的缺省合并规则。
range_ensemble_method	Mean Median	连续目标的缺省合并规则。
large_boost	标志	对非常大的数据集应用推进。
min_impurity	成员	
impurity_measure	Gini Twoing Ordered	
train_pct	成员	防止过度拟合集合。
set_random_seed	标志	复制结果选项。
seed	成员	
calculate_variable_importance	标志	
calculate_raw_propensities	标志	

表 121: cartnode 属性 (继续)		
cartnode 属性	值	属性描述
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## chaidnode 属性



CHAID 使用卡方统计来生成决策树，以确定最佳的分割。CHAID 与 C&R 树和 QUEST 节点不同，它可以生成非二元树，这意味着有些分割将有多于两个的分支。目标和输入字段可以是数字范围（连续）或分类。穷举 CHAID 是 CHAID 的修正版，它可以更彻底地检查所有可能的拆分，但计算时间较长。

### 示例

```

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)

```

表 122: chaidnode 属性		
chaidnode 属性	值	属性描述
target	<i>field</i>	CHAID 模型需要一个目标以及一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue_training_existing_model	标志	

表 122: chaidnode 属性 (继续)		
chaidnode 属性	值	属性描述
objective	Standard  Boosting  Bagging  psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single  InteractiveBuilder	
use_tree_directives	标志	
tree_directives	string	
method	Chaid  ExhaustiveChaid	
use_max_depth	Default  Custom	
max_depth	整数	最大树深度，从 0 到 1000。仅在 use_max_depth = Custom 时使用。
use_percentage	标志	
min_parent_records_pc	成员	
min_child_records_pc	成员	
min_parent_records_abs	成员	
min_child_records_abs	成员	
use_costs	标志	
costs	结构化	结构化属性。
trails	成员	用于推进或组装的组件模型数。
set_ensemble_method	Voting  HighestProbability  HighestMeanProbabilit y	分类目标的缺省合并规则。
range_ensemble_method	Mean  Median	连续目标的缺省合并规则。
large_boost	标志	对非常大的数据集应用推进。
split_alpha	成员	分割的显著性水平。
merge_alpha	成员	合并的显著性水平。

表 122: chaidnode 属性 (继续)

chaidnode 属性	值	属性描述
bonferroni_adjustment	标志	使用 Bonferroni 法调整显著性值。
split_merged_categories	标志	允许对合并的类别进行再分割。
chi_square	Pearson LR	这是用于计算卡方统计的方法： Pearson 或似然比
epsilon	成员	期望单元格频率的最小变化值。
max_iterations	成员	收敛的最大迭代次数。
set_random_seed	整数	
seed	成员	
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	整数	

## coxregnode 属性



使用 Cox 回归节点，您可以在已有的检查记录中建立时间事件的生存模型。对于输入变量的给定值，该模型会生成一个生存函数，用来预测在给定时间 (*t*) 发生相关事件的概率。

示例

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

表 123: coxregnode 属性

coxregnode 属性	值	属性描述
survival_time	<i>field</i>	Cox 回归模型需要一个包含生存时间的字段。
target	<i>field</i>	Cox 回归模型需要一个目标字段以及一个或多个输入字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。

表 123: <i>coxregnode</i> 属性 (继续)		
coxregnode 属性	值	属性描述
method	Enter	
	Stepwise	
	BackwardsStepwise	
groups	<i>field</i>	
model_type	MainEffects	
	Custom	
custom_terms	["BP*Sex" "BP*Age"]	
mode	Expert	
	Simple	
max_iterations	成员	
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	

表 123: *coxregnode* 属性 (继续)

<b>coxregnode</b> 属性	值	属性描述
<code>l_converge</code>	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
<code>removal_criterion</code>	LR Wald Conditional	
<code>probability_entry</code>	成员	
<code>probability_removal</code>	成员	
<code>output_display</code>	EachStep LastStep	
<code>ci_enable</code>	标志	
<code>ci_value</code>	90 95 99	
<code>correlation</code>	标志	
<code>display_baseline</code>	标志	
<code>survival</code>	标志	
<code>hazard</code>	标志	
<code>log_minus_log</code>	标志	
<code>one_minus_survival</code>	标志	
<code>separate_line</code>	<i>field</i>	
<code>value</code>	数字或字符串	如果未对某个字段指定值，那么将对 该字段使用缺省选项“Mean”。

# decisionlistnode 属性



决策列表节点可标识子组或段，显示与总体相关的给定二元结果的似然度的高低。例如，您或许在寻找那些最不可能流失的客户或最有可能对某个商业活动作出积极响应的客户。通过定制段和并排预览备选模型来比较结果，您可以将自己的业务知识体现在模型中。决策列表模型由一组规则构成，其中每个规则具备一个条件和一个结果。规则依顺序应用，相匹配的第一个规则将决定结果。

示例

```
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)
```

表 124: decisionlistnode 属性		
decisionlistnode 属性	值	属性描述
target	field	决策列表模型使用一个目标以及一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	与查找段相关；其中 Up 相当于“高概率”，而 Down 相当于“低概率”。
target_value	string	如果未指定，则标志采用真值。
max_rules	整数	除余数之外的最大段数。
min_group_size	整数	最小段大小。
min_group_size_pct	成员	最小段大小（以百分比表示）。
confidence_level	成员	为了使输入字段符合添加到段定义的条件，输入字段必须提高的响应似然值（提升）的最小阈值。
max_segments_per_rule	整数	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	成员	
max_models_per_cycle	整数	列表的搜索宽度。
max_rules_per_cycle	整数	段规则的搜索宽度。
segment_growth	成员	

表 124: *decisionlistnode* 属性 (继续)

<b>decisionlistnode</b> 属性	值	属性描述
include_missing	标志	
final_results_only	标志	
reuse_fields	标志	允许重复使用属性（出现在规则中的输入字段）。
max_alternatives	整数	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## discriminantnode 属性



判别分析提出比 Logistic 回归更加严格的假设，但是在满足这些假设时可成为 Logistic 回归的有价值替代方案或补充。

示例

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

表 125: *discriminantnode* 属性

<b>discriminantnode</b> 属性	值	属性描述
target	<i>field</i>	判别模型需要一个目标字段以及一个或多个输入字段。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	



表 125: discriminantnode 属性 (继续)		
<b>discriminantnode 属性</b>	<b>值</b>	<b>属性描述</b>
means	标志	“高级输出”对话框中的统计选项。
univariate_anovas	标志	
box_m	标志	
within_group_covariance	标志	
within_groups_correlation	标志	
separate_groups_covariance	标志	
total_covariance	标志	
fishers	标志	
unstandardized	标志	
casewise_results	标志	“高级输出”对话框中的分类选项。
limit_to_first	成员	缺省值为 10。
summary_table	标志	
leave_one_classification	标志	
combined_groups	标志	
separate_groups_covariance	标志	矩阵选项类协方差。
territorial_map	标志	
combined_groups	标志	散点图选项 联合组。
separate_groups	标志	散点图选项 独立组。
summary_of_steps	标志	
F_pairwise	标志	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	成员	
criteria	UseValue UseProbability	
F_value_entry	成员	缺省值是 3.84。
F_value_removal	成员	缺省值是 2.71。

表 125: *discriminantnode* 属性 (继续)

<b>discriminantnode</b> 属性	值	属性描述
probability_entry	成员	缺省值是 0.05。
probability_removal	成员	缺省值是 0.10。
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## extensionmodelnode 属性



通过“扩展模型”节点，可以运行 R 或 Python for Spark 脚本来构建和评分结果。

### Python for Spark 示例

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

build_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTree

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]

target = "Drug"
predictors = ["Age", "BP", "Sex", "Cholesterol", "Na", "K"]

def metaMap(row,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(set([row[col]]))
        else:
            meta.append((row[col],row[col]))
        col += 1
    return meta

def metaReduce(meta1,meta2,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(meta1[col].union(meta2[col]))
        else:
```

```

        meta.append((min(meta1[col][0],meta2[col][0]),max(meta1[col][1],meta2[col][1])))
    col += 1
    return meta

metadata = df.rdd.map(lambda row: metaMap(row,schema)).reduce(lambda x,y:metaReduce(x,y,schema))

def setToList(v):
    if isinstance(v,set):
        return list(v)
    return v

metadata = map(lambda x: setToList(x), metadata)
print metadata

lookup = {}
for i in range(0,len(schema)):
    lookup[schema[i][0]] = i

def row2LabeledPoint(dm,lookup,target,predictors,row):
    target_index = lookup[target]
    tval = dm[target_index].index(row[target_index])
    pvals = []
    for predictor in predictors:
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            pval = dm[predictor_index].index(row[predictor_index])
        else:
            pval = row[predictor_index]
        pvals.append(pval)
    return LabeledPoint(tval,DenseVector(pvals))

# count number of target classes
predictorClassCount = len(metadata[lookup[target]])

# define function to extract categorical predictor information from datamodel
def getCategoricalFeatureInfo(dm,lookup,predictors):
    info = {}
    for i in range(0,len(predictors)):
        predictor = predictors[i]
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            info[i] = len(dm[predictor_index])
    return info

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))

treeModel = DecisionTree.trainClassifier(
    lps,
    numClasses=predictorClassCount,
    categoricalFeaturesInfo=getCategoricalFeatureInfo(metadata, lookup, predictors),
    impurity='gini',
    maxDepth=5,
    maxBins=100)

_outputPath = cxt.createTemporaryFolder()
treeModel.save(cxt.getSparkContext(), _outputPath)
cxt.setModelContentFromPath("TreeModel", _outputPath)
cxt.setModelContentFromString("model.dm",json.dumps(metadata), mimeType="application/json")\
    .setModelContentFromString("model.structure",treeModel.toDebugString())

"""

node.setPropertyValue("python_build_syntax", build_script)

```

## R 示例

```

#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <-
lm(modelerData$Na~modelerData$K,modelerData)
modelerDataModel
modelerModel
""")

```

表 126: extensionmodelnode 属性

extensionmodelnode 属性	值	属性描述
syntax_type	<i>R</i> <i>Python</i>	指定运行哪个脚本 - R 还是 Python（R 是缺省值）。
r_build_syntax	<i>string</i>	这是用于进行模型构建的 R 脚本语法。
r_score_syntax	<i>string</i>	这是用于进行模型评分的 R 脚本语法。
python_build_syntax	<i>string</i>	这是用于进行模型构建的 Python 脚本语法。
python_score_syntax	<i>string</i>	这是用于进行模型评分的 Python 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
output_html	标志	此选项用于在 R 模型块中的选项卡上显示图形。
output_text	标志	此选项用于将 R 控制台文本输出写至 R 模型块中的选项卡。

## factornode 属性



“PCA/因子”节点提供用于降低数据复杂程度的强大数据降维技术。主成份分析（PCA）可找出输入字段的线性组合，该组合最好地捕获了整个字段集合中的方差，且组合中的各个成分相互正交（相互垂直）。因子分析则尝试识别底层因素，这些因素说明了观测的字段集合内的相关性模式。对于这两种方法，其共同的目标是找到可对原始字段集合中的信息进行有效总结的少量派生字段。

示例

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
```

```

node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" section
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)

```

表 127: *factornode* 属性

factornode 属性	值	属性描述
inputs	[ <i>field1</i> ... <i>fieldN</i> ]	PCA/因子模型使用输入字段的列表，但不使用目标。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
method	PC  ULS  GLS  ML  PAF  Alpha  Image	
mode	Simple  Expert	
max_iterations	成员	
complete_records	标志	
matrix	Correlation  Covariance	
extract_factors	ByEigenvalues  ByFactors	
min_eigenvalue	成员	
max_factor	成员	

表 127: *factornode* 属性 (继续)

factornode 属性	值	属性描述
rotation	None  Varimax  DirectOblimin  Equamax  Quartimax  Promax	
delta	成员	如果选择 DirectOblimin 作为旋转数据的类型，则可以指定 delta 的值。  如果未指定一个值，则将使用 delta 的缺省值。
kappa	成员	如果选择 Promax 作为旋转数据的类型，则可以指定 kappa 的值。  如果未指定一个值，则将使用 kappa 的缺省值。
sort_values	标志	
hide_values	标志	
hide_below	成员	

## featureselectionnode 属性



“特征选择”节点根据一组条件（例如缺失值百分比）筛选要移除的输入字段，然后，相对于指定目标对余下的输入的重要性进行排序。例如，假如某个给定数据集有上千个潜在输入，那么哪些输入最有可能用于对患者结果进行建模呢？

### 示例

```
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)
```

有关创建和应用特征选择模型的更详细示例，请参阅["创建和应用特征选择模型"](#)。

表 128: <i>featureselectionnode</i> 属性		
<b>featureselectionnode</b> 属性	值	属性描述
target	<i>field</i>	特征选择模型相对于指定的目标对预测变量进行排名。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
screen_single_category	标志	如果为 True，则将筛选相对于记录总数而言同个类别中具有过多记录的字段。
max_single_category	成员	指定 screen_single_category 为 True 时使用的阈值。
screen_missing_values	标志	如果为 True，则将筛选具有过多缺失值的字段，字段数表示为记录总数的百分比。
max_missing_values	成员	
screen_num_categories	标志	如果为 True，则将筛选相对于记录总数而言具有过多类别的字段。
max_num_categories	成员	
screen_std_dev	标志	如果为 True，则将筛选标准差小于或等于指定最小值的字段。
min_std_dev	成员	
screen_coeff_of_var	标志	如果为 True，则将筛选方差系数小于或等于指定最小值的字段。
min_coeff_of_var	成员	
criteria	Pearson  Likelihood  CramersV  Lambda	对照分类目标对分类预测变量进行排序时，指定用作确定重要性值的根据的度量。
unimportant_below	成员	指定用于将变量排序为“重要”、“边际”或“不重要”的 <i>p</i> 阈值。接受从 0.0 到 1.0 的值。
important_above	成员	接受从 0.0 到 1.0 的值。
unimportant_label	<i>string</i>	指定“不重要”排序的标签。
marginal_label	<i>string</i>	
important_label	<i>string</i>	
selection_mode	ImportanceLevel  ImportanceValue  TopN	

表 128: *featureselectionnode* 属性 (继续)

<b>featureselectionnode</b> 属性	值	属性描述
select_important	标志	在 <i>selection_mode</i> 设置为 <i>ImportanceLevel</i> 时, 指定是否选择“重要”字段。
select_marginal	标志	在 <i>selection_mode</i> 设置为 <i>ImportanceLevel</i> 时, 指定是否选择“边际”字段。
select_unimportant	标志	在 <i>selection_mode</i> 设置为 <i>ImportanceLevel</i> 时, 指定是否选择“不重要”字段。
importance_value	成员	在 <i>selection_mode</i> 设置为 <i>ImportanceValue</i> 时, 指定要使用的分界值。接受从 0 到 100 的值。
top_n	整数	在 <i>selection_mode</i> 设置为 <i>TopN</i> 时, 指定要使用的分界值。接受从 0 到 1000 的值。

## genlinnode 属性



广义线性模型对广义线性模型进行了扩展, 这样因变量通过指定的关联函数与因子和协变量线性相关。而且, 该模型还允许因变量为非正态分布。它涵盖了大量统计模型的功能, 包括线性回归、逻辑回归、计数数据的对数线性模型和区间删失生存模型。

### 示例

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

表 129: *genlinnode* 属性

<b>genlinnode</b> 属性	值	属性描述
target	<i>field</i>	广义线性模型要求单个目标字段（必须是一个集合或标志），以及一个或多个输入字段。还可以指定权重字段。有关更多信息, 请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
use_weight	标志	
weight_field	<i>field</i>	字段类型只有连续。
target_represents_trials	标志	
trials_type	Variable FixedValue	
trials_field	<i>field</i>	字段类型有连续、标志或有序。
trials_number	成员	缺省值为 10。



表 129: <i>genlinnode</i> 属性 (继续)		
<b>genlinnode</b> 属性	值	属性描述
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	<i>field</i>	字段类型只有连续。
offset_value	成员	必须是实数。
base_category	Last First	
include_intercept	标志	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS：逆高斯。 NEGBIN：负二项式。
negbin_para_type	Specify Estimate	
negbin_parameter	成员	缺省值为 1。必须包含一个非负实数。
tweedie_parameter	成员	

表 129: *genlinnode* 属性 (继续)

<b>genlinnode 属性</b>	<b>值</b>	<b>属性描述</b>
link_function	IDENTITY	CLOGLOG：互补双对数。
	CLOGLOG	LOGC：对数补数。
	LOG	NEGBIN：负二项式。
	LOGC	NLOGLOG：负双对数。
	LOGIT	CUMCAUCHIT：累积 Cauchit。
	NEGBIN	CUMCLOGLOG：累积互补双对数。
	NLOGLOG	CUMLOGIT：累积分对数。
	ODDSPower	CUMNLOGLOG：累积负双对数。
	PROBIT	CUMPROBIT：累积概率。
	POWER	
	CUMCAUCHIT	
	CUMCLOGLOG	
	CUMLOGIT	
	CUMNLOGLOG	
	CUMPROBIT	
power	成员	值必须是非零实数。
method	Hybrid	
	Fisher	
	NewtonRaphson	
max_fisher_iterations	成员	缺省值为 1；只允许正整数。
scale_method	MaxLikelihoodEstimate	
	Deviance	
	PearsonChiSquare	
	FixedValue	
scale_value	成员	缺省值为 1；必须大于 0。

表 129: genlinnode 属性 (继续)		
genlinnode 属性	值	属性描述
covariance_matrix	ModelEstimator	
	RobustEstimator	
max_iterations	成员	缺省值为 100；仅允许使用非负整数。
max_step_halving	成员	缺省值为 5；仅允许使用正整数。
check_separation	标志	
start_iteration	成员	缺省值为 20；仅允许使用正整数。
estimates_change	标志	
estimates_change_min	成员	缺省值为 1E-006；仅允许使用正数。
estimates_change_type	Absolute	
	Relative	
loglikelihood_change	标志	
loglikelihood_change_min	成员	仅允许使用整数。
loglikelihood_change_type	Absolute	
	Relative	
hessian_convergence	标志	
hessian_convergence_min	成员	仅允许使用整数。
hessian_convergence_type	Absolute	
	Relative	
case_summary	标志	
contrast_matrices	标志	
descriptive_statistics	标志	
estimable_functions	标志	
model_info	标志	
iteration_history	标志	
goodness_of_fit	标志	
print_interval	成员	缺省值为 1；必须是正整数。
model_summary	标志	
lagrange_multiplier	标志	
parameter_estimates	标志	
include_exponential	标志	
covariance_estimates	标志	
correlation_estimates	标志	

表 129: *genlinnode* 属性 (继续)

<b>genlinnode 属性</b>	<b>值</b>	<b>属性描述</b>
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	成员	缺省值是 0.0001。
confidence_interval	成员	缺省值是 95。
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

# glmmnode 属性



广义线性混合模型 (GLMM) 扩展了线性模型，使得目标可以有非正态分布，通过指定的连接函数与因子和协变量线性相关，并且观测值可能相关。广义线性混合模型涵盖了各种模型，从简单线性回归模型到非正态纵向模型数据的复杂多级模型。

表 130: glmmnode 属性		
glmmnode 属性	值	属性描述
residual_subject_spec	结构化	这是指定的分类字段的值组合，此组合唯一地定义数据集中的主体。
repeated_measures	结构化	这些字段用于标识重复的观测值。
residual_group_spec	[field1 ... fieldN]	这是用于定义重复效应协方差参数的独立集合的字段。
residual_covariance_type	Diagonal  AR1  ARMA11  COMPOUND_SYMMETRY  IDENTITY  TOEPLITZ  UNSTRUCTURED  VARIANCE_COMPONENTS	指定残值的协方差结构。
custom_target	标志	指明是使用在上游节点定义的目标 (false) 还是由 target_field 指定的定制目标 (true)。
target_field	field	要用作目标的字段（如果 custom_target 为 true）。
use_trials	标志	指示目标响应是一组试验中发生的众多事件时，是否使用用于指定试验数的附加字段或值。缺省值为 false。
use_field_or_value	Field  Value	指示是使用字段（缺省）还是值来指定试验数。
trials_field	field	此字段用于指定试验数。
trials_value	整数	此值用于指定试验数。如果指定此属性，那么最小值为 1。
use_custom_target_reference	标志	指示将定制参考类别用于分类目标。缺省值为 false。

表 130: <i>glmmnode</i> 属性 (继续)		
<b>glmmnode 属性</b>	<b>值</b>	<b>属性描述</b>
target_reference_value	<i>string</i>	要使用的参考类别（如果 use_custom_target_reference 为 true）。
dist_link_combination	Nominal  Logit  GammaLog  BinomialLogit  PoissonLog  BinomialProbit  NegbinLog  BinomialLogC  Custom	目标值的分布的公共模型。选择 Custom 可以指定 target_distribution 所提供的列表中的分布。
target_distribution	Normal  Binomial  Multinomial  Gamma  Inverse  NegativeBinomial  Poisson	当 dist_link_combination 为 Custom 时目标值的分布。

表 130: *glmmnode* 属性 (继续)

<b>glmmnode 属性</b>	<b>值</b>	<b>属性描述</b>
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	关联函数，用于将目标值关联到预测变量。 如果 target_distribution 为 Binomial，那么您可以使用所列出的任何关联函数。 如果 target_distribution 为 Multinomial 之外的任何值，那么您可以使用 CLOGLOG, CAUCHIT, LOGIT, NLOGLOG 或 PROBIT。 如果 target_distribution 为 Binomial 或 Multinomial 之外的任何值，那么您可以使用 IDENTITY、LOG 或 POWER。
link_function_param	成员	要使用的关联函数参数值。仅当 normal_link_function 或 link_function_type 为 POWER 时才适用。
use_predefined_inputs	标志	指示固定效应字段是将上游定义为输入字段的字段 (true) 还是来自 fixed_effects_list 的字段 (false)。缺省值 false。
fixed_effects_list	结构化	如果 use_predefined_inputs 为 false，那么指定将输入字段用作固定效应字段。
use_intercept	标志	如果为 true (缺省)，那么在模型中包括截距。
random_effects_list	结构化	作为随机效应指定的字段列表。
regression_weight_field	field	这是用作分析权重字段的字段。
use_offset	None offset_value offset_field	指示如何指定偏移。值 None 表示不使用偏移。
offset_value	成员	use_offset 设置为 offset_value 时使用的偏移值。
offset_field	field	use_offset 设置为 offset_field 时用于偏移值的字段。

表 130: glmmnode 属性 (继续)

glmmnode 属性	值	属性描述
target_category_order	Ascending Descending Data	分类目标的排序顺序。值 Data 指定使用数据中的排序顺序。缺省值为 Ascending。
inputs_category_order	Ascending Descending Data	分类预测变量的排序顺序。值 Data 指定使用数据中的排序顺序。缺省值为 Ascending。
max_iterations	整数	此算法要执行的最大迭代次数。非负整数；缺省值为 100。
confidence_level	整数	这是用于计算模型系数的区间估计值的置信度级别。非负整数；最大值为 100，缺省值为 95。
degrees_of_freedom_method	Fixed Varied	指定如何计算自由度以进行显著性检验。
test_fixed_effects_coefficients	Model Robust	这是用于计算参数估计协方差矩阵的方法。
use_p_converge	标志	用于参数收敛的选项。
p_converge	成员	空白或任何正值。
p_converge_type	Absolute Relative	
use_l_converge	标志	用于对数似然收敛的选项。
l_converge	成员	空白或任何正值。
l_converge_type	Absolute Relative	
use_h_converge	标志	用于 Hessian 收敛的选项。
h_converge	成员	空白或任何正值。
h_converge_type	Absolute Relative	
max_fisher_steps	整数	
singularity_tolerance	成员	
use_model_name	标志	指示是为模型指定定制名称 (true) 还是使用系统生成的名称 (false)。缺省值为 false。



表 130: glmmnode 属性 (继续)		
glmmnode 属性	值	属性描述
model_name	string	如果 use_model_name 为 true, 那么指定使用的模型名称。
confidence	onProbability onIncrease	计算评分置信度值的基础：最高预测概率或者最高与次高预测概率之差。
score_category_probabilities	标志	如果为 true, 则为分类目标生成预测概率。缺省值为 false。
max_categories	整数	如果 score_category_probabilities 为 true, 那么指定保存最大类别数。
score_propensity	标志	如果为 true, 则为标记目标字段生成倾向评分, 指示字段结果为“true”的可能性。
emeans	structure	对于固定效应列表中的每个分类字段, 指定是否生成估计边际均值。
covariance_list	structure	对于固定效应列表中的每个连续字段, 指定计算估计边际均值时是使用均值还是自定义值。
mean_scale	Original Transformed	指定是根据目标的原始尺度 (缺省) 还是根据关联函数转换来计算估计边际均值。
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	对多个对比执行假设检验时使用的调整方法。

## gle 属性



GLE 扩展了线性模型, 以便目标可以有非正态分布, 通过指定的连接函数与因子和协变量线性相关, 并且观测值可能相关。广义线性混合模型涵盖了各种模型, 从简单线性回归模型到非正态纵向模型数据的复杂多级模型。

表 131: gle 属性		
gle 属性	值	属性描述
custom_target	标志	指明是使用在上游节点定义的目标 (false) 还是由 target_field 指定的定制目标 (true)。
target_field	field	要用作目标的字段 (如果 custom_target 为 true)。
use_trials	标志	指示目标响应是一组试验中发生的众多事件时, 是否使用用于指定试验数的附加字段或值。缺省值为 false。

表 131: *gle* 属性 (继续)

gle 属性	值	属性描述
use_trials_field_or_value	Field Value	指示是使用字段（缺省）还是值来指定试验数。
trials_field	<i>field</i>	此字段用于指定试验数。
trials_value	整数	此值用于指定试验数。 如果指定此属性，那么最小值为 1。
use_custom_target_reference	标志	指示将定制参考类别用于分类目标。 缺省值为 <b>false</b> 。
target_reference_value	<i>string</i>	要使用的参考类别（如果 <b>use_custom_target_reference</b> 为 <b>true</b> ）。
dist_link_combination	NormalIdentity  GammaLog  PoissonLog  NegbinLog  TweedieIdentity  NominalLogit  BinomialLogit  BinomialProbit  BinomialLogC  CUSTOM	目标值的分布的公共模型。  选择 CUSTOM 可以根据 <b>target_distribution</b> 所提供的列表指定分布。

表 131: <i>gle</i> 属性 (继续)		
gle 属性	值	属性描述
target_distribution	Normal	当 dist_link_combination 为 Custom 时目标值的分布。
	Binomial	
	Multinomial	
	Gamma	
	INVERSE_GAUSS	
	NEG_BINOMIAL	
	Poisson	
	TWEEDIE	
	UNKNOWN	

表 131: gle 属性 (继续)

gle 属性	值	属性描述
link_function_type	UNKNOWN	这是用于使目标值与预测变量相关的关联函数。如果 target_distribution 为 Binomial, 那么您可以使用：
	IDENTITY	
	LOG	
	LOGIT	IDENTITY
	PROBIT	LOG
	COMPL_LOG_LOG	LOGIT
	POWER	PROBIT
	LOG_COMPL	COMPL_LOG_LOG
	NEG_LOG_LOG	POWER
	ODDS_POWER	LOG_COMPL
	NEG_BINOMIAL	NEG_LOG_LOG
	GEN_LOGIT	ODDS_POWER
	CUMUL_LOGIT	如果 target_distribution 为 NEG_BINOMIAL, 那么您可以使用：
	CUMUL_PROBIT	NEG_BINOMIAL.
	CUMUL_COMPL_LOG_LOG	如果 target_distribution 为 UNKNOWN, 那么您可以使用：
	CUMUL_NEG_LOG_LOG	GEN_LOGIT
	CUMUL_CAUCHIT	CUMUL_LOGIT
		CUMUL_PROBIT
		CUMUL_COMPL_LOG_LOG
		CUMUL_NEG_LOG_LOG
		CUMUL_CAUCHIT
link_function_param	成员	要使用的 Tweedie 参数值。仅当 normal_link_function 或 link_function_type 为 POWER 时才适用。

表 131: gle 属性 (继续)		
gle 属性	值	属性描述
tweedie_param	成员	要使用的关联函数参数值。仅适用于 dist_link_combination 设置为 TweedieIdentity 或 link_function_type 为 TWEEDIE 的情况。
use_predefined_inputs	标志	指示模型效应字段是将上游定义为输入字段的字段 (true) 还是来自 fixed_effects_list 的字段 (false)。
model_effects_list	结构化	如果 use_predefined_inputs 为 false, 请指定要用作模型效应字段的输入字段。
use_intercept	标志	如果为 true (缺省), 那么在模型中包括截距。
regression_weight_field	field	这是用作分析权重字段的字段。
use_offset	None Value Variable	指示如何指定偏移。值 None 表示不使用偏移。
offset_value	成员	use_offset 设置为 offset_value 时使用的偏移值。
offset_field	field	use_offset 设置为 offset_field 时用于偏移值的字段。
target_category_order	Ascending Descending	分类目标的排序顺序。缺省值为 Ascending。
inputs_category_order	Ascending Descending	分类预测变量的排序顺序。缺省值为 Ascending。
max_iterations	整数	此算法要执行的最大迭代次数。非负整数; 缺省值为 100。
confidence_level	成员	这是用于计算模型系数的区间估计值的置信度级别。非负整数; 最大值为 100, 缺省值为 95。
test_fixed_effects_coef fecients	Model Robust	这是用于计算参数估计协方差矩阵的方法。
detect_outliers	标志	值为 true 时, 算法将查找除多项式分布以外的所有分布的影响离群值。
conduct_trend_analysis	标志	值为 true 时, 此算法会对散点图执行趋势分析。

表 131: gle 属性 (继续)

gle 属性	值	属性描述
estimation_method	FISHER_SCORING  NEWTON_RAPHSON  HYBRID	指定极大似然估计算法。
max_fisher_iterations	整数	最大迭代次数（如果使用 FISHER_SCORING estimation_method）。最小值为 0，最大值为 20。
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	指定要用于估计尺度参数的方法。
scale_value	成员	仅当 scale_parameter_method 设置为 Fixed 时可用。
negative_binomial_method	MLE FIXED	指定要用于估计负二项式辅助参数的方法。
negative_binomial_value	成员	仅当 negative_binomial_method 设置为 Fixed 时可用。
non_neg_least_squares	标志	是否执行非负最小平方。缺省值为 false。
use_p_converge	标志	用于参数收敛的选项。
p_converge	成员	空白或任何正值。
p_converge_type	标志	True 表示绝对，False 表示相对
use_l_converge	标志	用于对数似然收敛的选项。
l_converge	成员	空白或任何正值。
l_converge_type	标志	True 表示绝对，False 表示相对
use_h_converge	标志	用于 Hessian 收敛的选项。
h_converge	成员	空白或任何正值。
h_converge_type	标志	True 表示绝对，False 表示相对
max_iterations	整数	此算法要执行的最大迭代次数。非负整数；缺省值为 100。
sing_tolerance	整数	
use_model_selection	标志	启用参数阈值和模型选择方法控件。
method	LASSO ELASTIC_NET FORWARD_STEPWISE RIDGE	确定所使用的模型选择方法（如果使用 Ridge，那么确定正则化方法）。

表 131: gle 属性 (继续)		
gle 属性	值	属性描述
detect_two_way_interactions	标志	<p>值为 True 时, 模型将自动检测输入字段之间的双向交互。</p> <p>仅当模型为仅主效应 (即, 用户尚未创建任何高阶效应时) 并且所选 method 为向前步进、Lasso 或弹性网络时, 才应该启用此控件。</p>
automatic_penalty_params	标志	<p>仅当模型选择 method 为 Lasso 或弹性网络时才可用。</p> <p>使用此函数可以输入与 Lasso 或弹性网络变量选择方法关联的罚参数。</p> <p>如果值为 True, 那么将使用缺省值。如果值为 False, 那么将启用罚参数, 并且可以输入定制值。</p>
lasso_penalty_param	成员	仅当模型选择 method 为 Lasso 或弹性网络并且 automatic_penalty_params 为 False 时才可用。为 Lasso 指定罚参数值。
elastic_net_penalty_param1	成员	仅当模型选择 method 为 Lasso 或弹性网络并且 automatic_penalty_params 为 False 时才可用。为弹性网络参数 1 指定罚参数值。
elastic_net_penalty_param2	成员	仅当模型选择 method 为 Lasso 或弹性网络并且 automatic_penalty_params 为 False 时才可用。为弹性网络参数 2 指定罚参数值。
probability_entry	成员	仅当所选 method 为向前步进时才可用。针对效应包含, 指定 f 统计标准的显著性水平。
probability_removal	成员	仅当所选 method 为向前步进时才可用。针对效应删除, 指定 f 统计标准的显著性水平。
use_max_effects	标志	<p>仅当所选 method 为向前步进时才可用。</p> <p>启用 max_effects 控件。</p> <p>当值为 False 时, 所包含的缺省效应数应等于提供给模型的总效应数减截距。</p>
max_effects	整数	指定使用向前步进构建方法时的最大效应数。
use_max_steps	标志	<p>启用 max_steps 控件。</p> <p>当值为 False 时, 缺省步骤数应等于提供给模型的效应数的三倍减去截距。</p>
max_steps	整数	指定使用向前步进构建 method 时要执行的最大步骤数。
use_model_name	标志	指示是为模型指定定制名称 (true) 还是使用系统生成的名称 (false)。缺省值为 false。
model_name	string	如果 use_model_name 为 true, 那么指定使用的模型名称。

表 131: gle 属性 (继续)		
gle 属性	值	属性描述
usePI	标志	如果值为 <code>true</code> ，那么将计算预测变量重要性。

## kmeansnode 属性



K-Means 节点将数据集聚类到不同分组（或聚类）。此方法将定义固定的聚类数量，将记录迭代分配给聚类，以及调整聚类中心，直到进一步优化无法再改进模型。k-means 节点作为一种非监督学习机制，它并不试图预测结果，而是揭示隐含在输入字段集中的模式。

### 示例

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K", "Age"])
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

表 132: kmeansnode 属性		
kmeansnode 属性	值	属性描述
inputs	[ <i>field1 ... fieldN</i> ]	K-means 模型在一系列输入字段上执行聚类分析，但并不使用目标字段。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
num_clusters	成员	
gen_distance	标志	
cluster_label	String Number	
label_prefix	<i>string</i>	
mode	Simple Expert	



表 132: *kmeansnode* 属性 (继续)

<b>kmeansnode</b> 属性	值	属性描述
stop_on	Default	
	Custom	
max_iterations	成员	
tolerance	成员	
encoding_value	成员	
optimize	Speed	用于指定优化建模的方式是速度还是内存。
	Memory	

## kmeansasnode 属性



K-Means 是最常用的聚类算法之一。它将数据点聚集成多个预定义聚类。SPSS 建模器中的 K-Means-AS 节点使用 Spark 进行实现。有关 K-Means 算法的详细信息，请参阅 <https://spark.apache.org/docs/3.5.4/ml-clustering.html#clustering>。请注意，K-Means-AS 节点自动对分类变量执行独热编码。

表 133: *kmeansasnode* 属性

<b>kmeansasnode</b> 属性	值	属性描述
roleUse	<i>string</i>	指定 predefined 表示使用预定义角色，指定 custom 表示使用定制字段分配。缺省值为 predefined。
autoModel	<i>BOOLEAN</i>	指定 true 表示将缺省名称 (\$S-prediction) 用于新生成的评分字段，指定 false 表示使用定制名称。缺省值为 true。
features	<i>field</i>	用于输入的字段名称列表（如果 roleUse 属性设置为 custom）。
name	<i>string</i>	新生成的评分字段的名称（如果 autoModel 属性设置为 false）。
clustersNum	整数	要创建的聚类数目。缺省值为 5。
initMode	<i>string</i>	初始化算法。可能的值为 k-means   或 random。缺省值为 k-means  。
initSteps	整数	初始化步骤数（如果 initMode 设置为 k-means  ）。缺省值为 2。
advancedSettings	<i>BOOLEAN</i>	指定 true 表示让下列四个属性可用。缺省值为 false。
maxIteration	整数	聚类的最大迭代次数。缺省值为 20。
tolerance	<i>string</i>	停止迭代的容差。可能的设置为 1.0E-1、1.0E-2、...、1.0E-6。缺省值为 1.0E-4。

表 133: *kmeansasnode* 属性 (继续)

<b>kmeansasnode</b> 属性	值	属性描述
setSeed	<i>BOOLEAN</i>	指定 <b>true</b> 表示使用定制随机种子。缺省值为 <b>false</b> 。
randomSeed	整数	定制随机种子（如果 <b>setSeed</b> 属性为 <b>true</b> ）。

## knnnode 属性



The *k*-最近相邻元素 (KNN) 节点将新的观测值关联到预测变量空间中与其最邻近的 *k* 个对象的类别或值（其中 *k* 为整数）。类似观测值相互靠近，而不同观测值相互远离。

示例

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

表 134: *knnnode* 属性

<b>knnnode</b> 属性	值	属性描述
analysis	PredictTarget	
	IdentifyNeighbors	
objective	Balance	
	Speed	
	Accuracy	
	Custom	
normalize_ranges	标志	
use_case_labels	标志	此复选框用于启用下一个选项。
case_labels_field	<i>field</i>	
identify_focal_cases	标志	此复选框用于启用下一个选项。
focal_cases_field	<i>field</i>	
automatic_k_selection	标志	
fixed_k	整数	只有当 <b>automatic_k_selectio</b> 为 <b>False</b> 时才启用。
minimum_k	整数	只有当 <b>automatic_k_selectio</b> 为 <b>True</b> 时才启用。

表 134: knnnode 属性 (继续)		
knnnode 属性	值	属性描述
maximum_k	整数	
distance_computation	Euclidean CityBlock	
weight_by_importance	标志	
range_predictions	Mean Median	
perform_feature_selection	标志	
forced_entry_inputs	[field1 ... fieldN]	
stop_on_error_ratio	标志	
number_to_select	整数	
minimum_change	成员	
validation_fold_assign_by_field	标志	
number_of_folds	整数	只有当 validation_fold_assign_by_field 为 False 时才启用
set_random_seed	标志	
random_seed	成员	
folds_field	field	只有当 validation_fold_assign_by_field 为 True 时才启用
all_probabilities	标志	
save_distances	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## kohonenode 属性



Kohonen 节点会生成一种神经网络，此神经网络可用于将数据集聚类到各个差异组。此网络训练完成后，相似的记录应在输出映射中紧密地聚集，差异大的记录则应彼此远离。您可以通过查看模型块中每个单元所捕获观测值的数量来找出规模较大的单元。这将让您对聚类的相应数量有所估计。

## 示例

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

表 135: kohonennode 属性

kohonennode 属性	值	属性描述
inputs	[field1 ... fieldN]	Kohonen 模型使用输入字段的列表，但不使用目标。不使用频率和权重字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue	标志	
show_feedback	标志	
stop_on	Default Time	
time	成员	
optimize	Speed Memory	用于指定优化建模的方式是速度还是内存。
cluster_label	标志	
mode	Simple Expert	
width	成员	
length	成员	
decay_style	Linear Exponential	
phase1_neighborhood	成员	
phase1_eta	成员	

表 135: kohonennode 属性 (继续)		
kohonennode 属性	值	属性描述
phase1_cycles	成员	
phase2_neighborhood	成员	
phase2_eta	成员	
phase2_cycles	成员	

# linearnode 属性



线性回归模型根据目标与一个或多个预测变量之间的线性关系来预测连续目标。

示例

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

表 136: linearnode 属性		
linearnode 属性	值	属性描述
target	<i>field</i>	指定单个目标字段。
inputs	<i>[field1 ... fieldN]</i>	模型使用的预测变量字段。
continue_training_existing_model	标志	
objective	Standard  Bagging  Boosting  psm	psm 用于非常大的数据集，同时需要 Server 连接。
use_auto_data_preparation	标志	
confidence_level	成员	
model_selection	ForwardStepwise  BestSubsets  None	

表 136: *linearnode* 属性 (继续)

<b>linearnode</b> 属性	值	属性描述
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	成员	
probability_removal	成员	
use_max_effects	标志	
max_effects	成员	
use_max_steps	标志	
max_steps	成员	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuouss	Mean Median	
component_models_n	成员	
use_random_seed	标志	
random_seed	成员	
use_custom_model_name	标志	
custom_model_name	<i>string</i>	
use_custom_name	标志	
custom_name	<i>string</i>	
tooltip	<i>string</i>	
keywords	<i>string</i>	
annotation	<i>string</i>	

## linearnode 属性



线性回归模型根据目标与一个或多个预测变量之间的线性关系来预测连续目标。

表 137: <i>linearasnode</i> 属性		
<b>linearasnode</b> 属性	值	属性描述
target	<i>field</i>	指定单个目标字段。
inputs	[ <i>field1 ... fieldN</i> ]	模型使用的预测变量字段。
weight_field	<i>field</i>	模型使用的分析字段。
custom_fields	标志	缺省值为 TRUE。
intercept	标志	缺省值为 TRUE。
detect_2way_interaction	标志	是否考虑双向交互。 缺省值为 TRUE。
cin	成员	用于计算模型系数的估算的置信区间。请指定大于 0 且小于 100 的值。 缺省值为 95。
factor_order	ascending descending	分类预测变量的排序顺序。 缺省值为 ascending。
var_select_method	ForwardStepwise BestSubsets none	要使用的模型选择方法。 缺省值为 ForwardStepwise。
criteria_for_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	用于确定应向模型中添加效应还是应从模型中移除效应的统计信息。 缺省值为 AdjustedRSquare。
pin	成员	将向模型中添加具有小于此指定 pin 阈值的最小 p 值的效应。 缺省值为 0.05。
pout	成员	将移除模型中具有大于此指定 pout 阈值的 p 值的任何效应。 缺省值为 0.10。
use_custom_max_effects	标志	是否在最终模型中使用最大效应数。 缺省值为 FALSE。
max_effects	成员	要在最终模型中使用的最大效应数。 缺省值为 1。
use_custom_max_steps	标志	是否使用最大步骤数。 缺省值为 FALSE。
max_steps	成员	分步算法停止之前的最大步骤数。 缺省值为 1。

表 137: linearasnode 属性 (继续)

linearasnode 属性	值	属性描述
criteria_for_best_subsets	AICC  AdjustedRSquare  ASE	要使用的条件方式。缺省值为 AdjustedRSquare。

## logregnode 属性



Logistic 回归是一种统计方法，它可根据输入字段的值对记录进行分类。它与线性回归类似，但采用分类目标字段而不是数字范围。

### 多项式示例

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" options
node.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
node.setPropertyValue("max_terms", 10)
```



```
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

## 二项式示例

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

表 138: logregnode 属性

logregnode 属性	值	属性描述
target	field	Logistic 回归模型需要一个目标字段以及一个或多个输入字段。不使用频率和权重字段。有关更多信息，请参阅主题第 189 页的『公共建模节点属性』。
logistic_procedure	Binomial Multinomial	
include_constant	标志	
mode	Simple Expert	

表 138: logregnode 属性 (继续)

logregnode 属性	值	属性描述
method	Enter  Stepwise  Forwards  Backwards  BackwardsStepwise	
binomial_method	Enter  Forwards  Backwards	
model_type	MainEffects  FullFactorial  Custom	将 FullFactorial 指定为模型类型时，即使指定了步进方法，步进方法也不会运行。而是使用 Enter 方法。  如果将模型类型设置为 Custom，但未指定定制字段，则将构建主效应模型。
custom_terms	[[BP Sex]][BP][Age]]	
multinomial_base_category	string	指定如何确定参考类别。
binomial_categorical_input	string	
binomial_input_contrast	Indicator  Simple  Difference  Helmert  Repeated  Polynomial  Deviation	分类输入的键控属性，用于指定如何确定对比。
binomial_input_category	First  Last	这是分类输入的键控属性，用于指定如何确定参考类别。

表 138: logregnode 属性 (继续)		
logregnode 属性	值	属性描述
scale	None  UserDefined  Pearson  Deviance	
scale_value	成员	
all_probabilities	标志	
tolerance	1.0E-5  1.0E-6  1.0E-7  1.0E-8  1.0E-9  1.0E-10	
min_terms	成员	
use_max_terms	标志	
max_terms	成员	
entry_criterion	Score  LR	
removal_criterion	LR  Wald	
probability_entry	成员	
probability_removal	成员	
binomial_probability_entry	成员	
binomial_probability_removal	成员	
requirements	HierarchyDiscrete HierarchyAll  Containment  None	
max_iterations	成员	

表 138: logregnode 属性 (继续)

logregnode 属性	值	属性描述
max_steps	成员	
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	成员	
iteration_history	标志	
history_steps	成员	
summary	标志	
likelihood_ratio	标志	
asymptotic_correlation	标志	
goodness_fit	标志	
parameters	标志	
confidence_interval	成员	
asymptotic_covariance	标志	
classification_table	标志	
stepwise_summary	标志	
info_criteria	标志	
monotonicity_measures	标志	
binomial_output_display	at_each_step at_last_step	

表 138: logregnode 属性 (继续)		
logregnode 属性	值	属性描述
binomial_goodness_of_fit	标志	
binomial_parameters	标志	
binomial_iteration_history	标志	
binomial_classification_plots	标志	
binomial_ci_enable	标志	
binomial_ci	成员	
binomial_residual	outliers all	
binomial_residual_enable	标志	
binomial_outlier_threshold	成员	
binomial_classification_cutoff	成员	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	标志	
calculate_raw_propensities	标志	

## lsvmnode 属性



通过线性支持向量机 (LSVM) 节点，您可以将数据分为两组，而无需过度拟合。LSVM 是线性的，并且可以与大量数据集配合使用，例如包含大量记录的数据集。

表 139: lsvmnode 属性		
lsvmnode 属性	值	属性描述
intercept	标志	在模型中包括截距。缺省值为 True。
target_order	Ascending Descending	指定分类目标的排序顺序。对于连续目标，将忽略此项。缺省值为 Ascending。

表 139: lsvmnode 属性 (继续)

lsvmnode 属性	值	属性描述
precision	成员	仅当目标字段的测量级别为 Continuous 时才使用。指定与回归损失的敏感性相关的参数。最小值为 0，而没有最大值。缺省值为 0.1。
exclude_missing_values	标志	设置为 True 时，如果任何单个值缺失，那么将排除记录。缺省值为 False。
penalty_function	L1 L2	指定所使用的罚函数的类型。缺省值为 L2。
lambda	成员	罚（规则化）参数。
calculate_variable_importance	标志	对于生成相应重要性测量的模型，此选项将显示一张图表来说明评估模型时每个预测变量的相对重要性。请注意，对于某些模型，计算预测变量重要性（特别在处理大型数据集时）可能需要花费较长时间，因此缺省情况下，预测变量重要性对某些模型处于关闭状态。变量重要性不适用于决策列表模型。

## neuralnetnode 属性

**要点:** 在此发行版中提供了具有增强功能的新版本的神经网络建模节点，并将在下一节 (*neuralnetwork*) 中进行介绍。尽管您仍然可以使用先前版本来构建模型并对其评分，但我们建议您将脚本更新为使用新版本。这里保留了先前版本的详细信息以供您参考。

示例

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

表 140: <i>neuralnetnode</i> 属性		
neuralnetnode 属性	值	属性描述
targets	[ <i>field1</i> ... <i>fieldN</i> ]	“神经网络”节点需要一个或多个目标字段以及一个或多个输入字段。将忽略频率和权重字段。有关更多信息，请参阅主题 第 189 页的『公共建模节点属性』。
method	Quick  Dynamic  Multiple  Prune  ExhaustivePrune  RBFN	
prevent_overtrain	标志	
train_pct	成员	
set_random_seed	标志	
random_seed	成员	
mode	Simple  Expert	
stop_on	Default  Accuracy  Cycles  Time	停止方式。
accuracy	成员	停止准确性。
cycles	成员	训练周期数。
time	成员	训练时间（分钟数）。
continue	标志	
show_feedback	标志	
binary_encode	标志	
use_last_model	标志	
gen_logfile	标志	
logfile_name	<i>string</i>	
alpha	成员	

表 140: *neuralnetnode* 属性 (继续)

<b>neuralnetnode</b> 属性	值	属性描述
initial_eta	成员	
high_eta	成员	
low_eta	成员	
eta_decay_cycles	成员	
hid_layers	One Two Three	
hl_units_one	成员	
hl_units_two	成员	
hl_units_three	成员	
persistence	成员	
m_topologies	<i>string</i>	
m_non_pyramids	标志	
m_persistence	成员	
p_hid_layers	One Two Three	
p_hl_units_one	成员	
p_hl_units_two	成员	
p_hl_units_three	成员	
p_persistence	成员	
p_hid_rate	成员	
p_hid_pers	成员	
p_inp_rate	成员	
p_inp_pers	成员	
p_overall_pers	成员	
r_persistence	成员	
r_num_clusters	成员	
r_eta_auto	标志	
r_alpha	成员	
r_eta	成员	



表 140: <i>neuralnetnode</i> 属性 (继续)		
neuralnetnode 属性	值	属性描述
optimize	Speed Memory	用于指定优化建模的方式是速度还是内存。
calculate_variable_importance	标志	注：此属性取代了先前版本中使用的 <i>sensitivity_analysis</i> 属性。仍然支持旧属性，但建议使用 <i>calculate_variable_importance</i> 。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## neuralnetworknode 属性



神经网络节点使用的模型是对人类大脑处理信息的方式简化了的模型。此模型通过模拟大量类似于神经元的抽象形式的互连简单处理单元而运行。神经网络是功能强大的一般函数估计器，只需要最少的统计或数学知识就可以对其进行训练或应用。

示例

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

表 141: <i>neuralnetworknode</i> 属性		
neuralnetworknode 属性	值	属性描述
targets	[field1 ... fieldN]	指定目标字段。
inputs	[field1 ... fieldN]	模型使用的预测变量字段。
splits	[field1 ... fieldN]	指定一个或多个用于拆分建模的字段。
use_partition	标志	如果定义了分区字段，则此选项可确保仅训练分区的数据用于构建模型。
continue	标志	继续训练现有模型。
objective	Standard Bagging Boosting psm	psm 用于非常大的数据集，同时需要 Server 连接。

表 141: *neuralnetworknode* 属性 (继续)

<b>neuralnetworknode</b> 属性	值	属性描述
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	标志	
first_layer_units	成员	
second_layer_units	成员	
use_max_time	标志	
max_time	成员	
use_max_cycles	标志	
max_cycles	成员	
use_min_accuracy	标志	
min_accuracy	成员	
combining_rule_categorical	Voting HighestProbability HighestMeanProbabilit y	
combining_rule_continuou s	Mean Median	
component_models_n	成员	
overfit_prevention_pct	成员	
use_random_seed	标志	
random_seed	成员	
missing_values	listwiseDeletion missingValueImputatio n	
use_model_name	BOOLEAN	
model_name	string	
confidence	onProbability onIncrease	
score_category_probabili ties	标志	
max_categories	成员	
score_propensity	标志	

表 141: <i>neuralnetworknode</i> 属性 (继续)		
neuralnetworknode 属性	值	属性描述
use_custom_name	标志	
custom_name	string	
tooltip	string	
keywords	string	
annotation	string	

## questnode 属性



QUEST 节点可提供用于构建决策树的二元分类法，此方法的设计目的是减少大型 C&R 树分析所需的处理时间，同时也减少在分类树方法中发现的趋势以便支持允许有多个分割的输入。输入字段可以是数字范围（连续），但目标字段必须是分类。所有分割都是二元的。

示例

```
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)
```

表 142: <i>questnode</i> 属性		
questnode 属性	值	属性描述
target	field	QUEST 模型需要一个目标字段以及一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue_training_existing_model	标志	
objective	Standard Boosting Bagging psm	psm 用于非常大的数据集，同时需要 Server 连接。
model_output_type	Single InteractiveBuilder	

表 142: questnode 属性 (继续)

questnode 属性	值	属性描述
use_tree_directives	标志	
tree_directives	string	
use_max_depth	Default Custom	
max_depth	整数	最大树深度，从 0 到 1000。仅在 use_max_depth = Custom 时使用。
prune_tree	标志	修剪树，以避免过度拟合。
use_std_err	标志	使用最大风险差（标准误差）。
std_err_multiplier	成员	最大差值。
max_surrogates	成员	最大代用项。
use_percentage	标志	
min_parent_records_pc	成员	
min_child_records_pc	成员	
min_parent_records_abs	成员	
min_child_records_abs	成员	
use_costs	标志	
costs	结构化	结构化属性。
priors	Data Equal Custom	
custom_priors	结构化	结构化属性。
adjust_priors	标志	
trails	成员	用于推进或组装的组件模型数。
set_ensemble_method	Voting HighestProbability HighestMeanProbabilit y	分类目标的缺省合并规则。
range_ensemble_method	Mean Median	连续目标的缺省合并规则。
large_boost	标志	对非常大的数据集应用推进。
split_alpha	成员	分割的显著性水平。
train_pct	成员	防止过度拟合集合。

表 142: <i>questnode</i> 属性 (继续)		
<b>questnode</b> 属性	值	属性描述
set_random_seed	标志	复制结果选项。
seed	成员	
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test Validation	

## randomtrees 属性



“随机树”节点与现有 C&RT 节点相似；但是，“随机树”节点旨在处理大数据以创建单个树，并在 SPSS 建模器 V17 中添加的输出查看器中显示产生的模型。“随机树”节点将生成您可以对未来观测值进行预测或分类的决策树。通过在每个步骤最大限度降低不纯度，此方法使用递归分区将训练记录分割为多个段。如果树中某个节点的全部观测值都属于目标字段的一个特定类别，那么系统会将该节点视为纯洁。目标和输入字段可以是数字范围或分类（名义、有序或标志）；所有分割均为二元分割（即仅分割为两个子组）。

表 143: <i>randomtrees</i> 属性		
<b>randomtrees</b> 属性	值	属性描述
target	<i>field</i>	在“随机树”节点中，模型需要单个目标字段以及一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
number_of_models	整数	确定要在整体建模过程中构建的模型数。
use_number_of_predictors	标志	确定是否使用 <code>number_of_predictors</code> 。
number_of_predictors	整数	指定在构建拆分模型时要使用的预测变量数。
use_stop_rule_for_accuracy	标志	确定在准确性无法提高时是否停止模型构建。
sample_size	成员	减小此值可以提高处理超大数据集时的性能。

表 143: *randomtrees* 属性 (继续)

randomtrees 属性	值	属性描述
handle_imbalanced_data	标志	如果模型的目标是特定的标志结果，并且所需结果与非所需结果的比率非常小，那么数据是不平衡数据并且模型所执行的 Bootstrap 采样可能会影响模型精确性。启用不平衡数据处理，以便模型将捕获所需结果中的更大比例部分并生成更强大的模型。
use_weighted_sampling	标志	为 <i>False</i> 时，每个节点的变量将以同一概率随机进行选择。为 <i>True</i> 时，会对变量进行相应地加权和选择。
max_node_number	整数	允许各个树中存在的最大节点数。如果下一次分割时将超过此数字，那么树增长将停止。
max_depth	整数	树增长停止之前的最大树深度。
min_child_node_size	整数	确定拆分父节点之后允许子节点中存在的最小记录数。如果子节点包含的记录数少于此处指定的数目，那么不会拆分父节点
use_costs	标志	
costs	结构化	结构化属性。格式是由 3 个值组成的列表：实际值、预测值和成本（如果预测错误）。例如：  <code>tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])</code>
default_cost_increase	none  linear  square  custom	注: 仅对有序目标启用。  设置成本矩阵中的缺省值。
max_pct_missing	整数	如果任何输入中缺失值所占百分比超过此处指定的值，那么将排除此输入。最小值为 0，最大值为 100。
exclude_single_cat_pct	整数	如果某个类别值表示的记录百分比高于此处指定的值，那么将从模型构建中排除整个字段。最小值为 1，最大值为 99。
max_category_number	整数	如果字段中类别数超过了此值，那么将从模型构建中排除此字段。最小值为 2。
min_field_variation	成员	如果某个连续字段的变异系数小于此值，那么将从模型构建中排除此字段。

表 143: *randomtrees* 属性 (继续)

randomtrees 属性	值	属性描述
num_bins	整数	仅当数据由连续输入组成时才使用。设置要用于输入的等频分级数；选项为：2、4、5、10、20、25、50 或 100。
topN	整数	指定要报告的规则数量。缺省值为 50，最小值为 1，最大值为 1000。

## regressionnode 属性



线性回归是一种常用的统计技术，用于汇总数据并通过拟合直线或曲面来进行预测，从而最大限度地减少预测输出值与实际输出值之间的差异。

注: 在未来的发行版中，“回归”节点将替换为“线性”节点。我们建议您从现在开始使用线性模型进行线性回归。

示例

```
node = stream.create("regression", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
node.setPropertyValue("weight_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

表 144: regressionnode 属性

regressionnode 属性	值	属性描述
target	field	回归模型需要一个目标字段以及一个或多个输入字段。还可以指定权重字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
method	Enter Stepwise Backwards Forwards	
include_constant	标志	
use_weight	标志	
weight_field	field	
mode	Simple Expert	
complete_records	标志	
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	请使用双引号将自变量括起来。
stepping_method	useP useF	useP : 使用 F 的概率 useF : 使用 F 值



表 144: regressionnode 属性 (继续)

regressionnode 属性	值	属性描述
probability_entry	成员	
probability_removal	成员	
F_value_entry	成员	
F_value_removal	成员	
selection_criteria	标志	
confidence_interval	标志	
covariance_matrix	标志	
collinearity_diagnostics	标志	
regression_coefficients	标志	
exclude_fields	标志	
durbin_watson	标志	
model_fit	标志	
r_squared_change	标志	
p_correlations	标志	
descriptives	标志	
calculate_variable_importance	标志	

## sequencenode 属性



序列节点可发现连续数据或与时间有关的数据中的关联规则。序列是一系列可能会以可预测顺序发生的项目集合。例如，一个购买了剃刀和须后水的顾客可能在下次购物时购买剃须膏。序列节点基于 CARMA 关联规则算法，该算法使用一个有效的两次传递方法查找序列。

示例

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
```

```
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

表 145: sequencenode 属性

sequencenode 属性	值	属性描述
id_field	<i>field</i>	要创建序列规则集，您需要指定一个标识字段以及一个可选的时间字段，以及一个或多个内容字段。不使用权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
time_field	<i>field</i>	
use_time_field	标志	
content_fields	[ <i>field1 ... fieldN</i> ]	
contiguous	标志	
min_supp	成员	
min_conf	成员	
max_size	成员	
max_predictions	成员	
mode	Simple Expert	
use_max_duration	标志	
max_duration	成员	
use_gaps	标志	
min_item_gap	成员	
max_item_gap	成员	
use_pruning	标志	
pruning_value	成员	
set_mem_sequences	标志	
mem_sequences	整数	

## slrmnode 属性



自学响应模型（SLRM）节点可用于构建一个包含单个新观测值或少量新观测值的模型，通过此模型，无需使用全部数据对模型进行重新训练即可对模型进行重新评估。

示例

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

表 146: slrmnode 属性

slrmnode 属性	值	属性描述
target	field	目标字段必须是名义字段或标志字段。还可以指定频率字段。有关更多信息, 请参阅主题 第 189 页的『公共建模节点属性』。
target_response	field	类型必须是标志。
continue_training_existing_model	标志	
target_field_values	标志	Use all : 使用来自源的全部值。  Specify : 选择所需的值。
target_field_values_specify	[field1 ... fieldN]	
include_model_assessment	标志	
model_assessment_random_seed	成员	必须是实数。
model_assessment_sample_size	成员	必须是实数。
model_assessment_iterations	成员	迭代数。
display_model_evaluation	标志	
max_predictions	成员	
randomization	成员	
scoring_random_seed	成员	
sort	Ascending  Descending	指定先显示评分最高还是最低的报价。
model_reliability	标志	
calculate_variable_importance	标志	

## statisticsmodelnode 属性



Statistics 模型节点使您能够通过运行生成 PMML 的 IBM SPSS Statistics 过程分析和处理数据。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息，请参阅第 378 页的『statisticsmodelnode 属性』。

## stpnode 属性



空间-时间预测 (STP) 节点使用包含位置数据、预测输入字段（预测变量）、时间字段和目标字段的数据。每个位置在数据中都有许多行，这些行表示每个预测变量在每个测量时间的值。分析数据后，可以使用该数据来预测分析中使用的形状数据内任意位置处的目标值。

表 147: stpnode 属性

stpnode 个属性	数据类型	属性描述
字段选项卡		
target	field	此为目标的字段。
location	field	模型的位置字段。 只允许使用地理空间字段。
location_label	field	这是要在输出中使用的分类字段，用于标注 location 中选择的位置。
time_field	field	模型的时间字段。 只允许使用具有连续测量的字段，并且存储类型必须为时间、日期、时间戳或整数。
inputs	[field1 ... fieldN]	输入字段的列表。
时间间隔选项卡		
interval_type_timestamp	Years Quarters Months Weeks Days Hours Minutes Seconds	

表 147: stpnode 属性 (继续)		
stpnode 个属性	数据类型	属性描述
interval_type_date	Years  Quarters  Months  Weeks  Days	
interval_type_time	Hours  Minutes  Seconds	限制创建 STP 用于计算的时间索引时考虑的每周天数。
interval_type_integer	Periods  (仅限时间索引字段, 整数存储)	数据集所要转换为的时间间隔。可用的选择取决于选择作为模型 time_field 的字段的存储类型。
period_start	整数	
start_month	January  February  March  April  May  June  July  August  September  October  November  December	模型将开始建立索引的月份, 例如, 如果设置为 March, 但是数据集中的第一条记录为 January, 那么模型将跳过前两条记录, 并从三月开始建立索引。

表 147: *stpnode* 属性 (继续)

stpnode 个属性	数据类型	属性描述
week_begins_on	Sunday  Monday  Tuesday  Wednesday  Thursday  Friday  Saturday	STP 根据数据创建的时间索引的起点。
days_per_week	整数	最小值为 1，最大值为 7，增量为 1。
hours_per_day	整数	模型在某天中占用的小时数。如果此项设置为 10，那么模型将从 day_begins_at 时间开始建立索引并持续 10 个小时，然后跳至与 day_begins_at 值匹配的下一个值等等。
day_begins_at	00:00  01:00  02:00  03:00  ...  23:00	设置模型开始建立索引的起始小时值。

表 147: <i>stpnode</i> 属性 (继续)		
stpnode 个属性	数据类型	属性描述
interval_increment	1 2 3 4 5 6 10 12 15 20 30	此增量设置针对分钟或秒。此项决定了模型根据数据创建索引的位置。因此，对于增量 30 和时间间隔类型 <b>seconds</b> ，模型将每隔 30 秒从数据创建索引。
data_matches_interval	BOOLEAN	<p>如果设置为 N，那么从数据到常规 <b>interval_type</b> 的转换在构建模型前进行。</p> <p>如果数据已具有正确的格式，并且 <b>interval_type</b> 以及任何相关设置与数据匹配，那么将此属性设置为 Y 可以阻止转换或汇总数据。</p> <p>将此属性设置为 Y 将禁用所有汇总控件。</p>
agg_range_default	Sum Mean Min Max Median 1stQuartile 3rdQuartile	此项决定用于连续字段的缺省汇总方法。未明确包含在定制汇总中的所有连续字段将使用此处指定的方法进行汇总。

表 147: stpnode 属性 (继续)

stpnode 个属性	数据类型	属性描述
custom_agg	<p>[[field, aggregation method], [ ] ..]</p> <p>演示：</p> <pre>[[ 'x5'   'FirstQuartile' ] [ 'x4'   'Sum' ]]</pre>	<p>结构化属性：</p> <p>脚本参数：custom_agg</p> <p>例如：</p> <pre>set :stpnode.custom_agg = [   [field1 function]   [field2 function] ]</pre> <p>其中 function 是要与该字段配合使用的汇总函数。</p>
基本选项卡		
include_intercept	标志	
max_autoregressive_lag	整数	最小值 1，最大值 5，增量为 1。这是预测所需的先前记录数。因此，如果设置为 5，那么将使用先前的 5 条记录创建新预测。此处根据构建数据指定的记录数将合并到模型中，因此用户无需在对模型进行评分时再次提供数据。
estimation_method	Parametric  Nonparametric	用于对空间协方差矩阵进行建模的方法。
parametric_model	Gaussian  Exponential  PoweredExponential	Parametric 空间协方差模型的阶参数。
exponential_power	成员	PoweredExponential 模型的幂级别。最小值为 1，最大值为 2。
高级选项卡		
max_missing_values	整数	模型中允许具有缺失值的记录所占的最大百分比。
significance	成员	模型构建中的假设检验的显著性水平。指定 STP 模型估算中所有检验（包括两项拟合优度检验、效应 F 检验及系数 T 检验）的显著性值。
输出选项卡		



表 147: stpnode 属性 (继续)

stpnode 个属性	数据类型	属性描述
model_specifications	标志	
temporal_summary	标志	
location_summary	标志	决定“位置摘要”表是否包括在模型输出中。
model_quality	标志	
test_mean_structure	标志	
mean_structure_coefficients	标志	
autoregressive_coefficients	标志	
test_decay_space	标志	
parametric_spatial_covariance	标志	
correlations_heat_map	标志	
correlations_map	标志	
location_clusters	标志	
similarity_threshold	成员	这是一个阈值，达到此阈值后，认为输出聚类足够相似，应合并为单个聚类。
max_number_clusters	整数	可以包括在模型输出中的聚类数目的上限。
<b>模型选项选项卡</b>		
use_model_name	标志	
model_name	string	
uncertainty_factor	成员	最小值为 0，最大值为 100。决定应用于未来预测的不确定性（错误）增大。它是预测的上限和下限。

## svmnode 属性



使用支持向量机 (SVM) 节点，可以将数据分为两组，而无需过度拟合。SVM 可以与宽数据集配合使用，例如那些含有大量输入字段的数据集。

示例

```
node = stream.create("svm", "My node")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

表 148: svmnode 属性

svmnode 属性	值	属性描述
all_probabilities	标志	
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (default) 1.0E-4 1.0E-5 1.0E-6	确定何时停止优化算法。
regularization	成员	也称为 C 参数。
precision	成员	仅当目标字段的测量级别为 Continuous 时才使用。
kernel	RBF (缺省)  Polynomial  Sigmoid  Linear	用于变换的内核函数的类型。
rbf_gamma	成员	仅在 kernel 为 RBF 时使用。
gamma	成员	仅在 kernel 为 Polynomial 或 Sigmoid 时使用。
bias	成员	
degree	成员	仅在 kernel 为 Polynomial 时使用。
calculate_variable_importance	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	
adjusted_propensity_partition	Test  Validation	



时间因果建模尝试发现时间序列数据中的关键因果关系。在时间因果建模中，您指定一组目标序列，并指定这些目标的一组候选输入。然后，此过程为每个目标构建自回归时间序列模型，并且仅包括那些与目标的因果关系最为重要的输入。

表 149: tcmnode 属性		
tcmnode 属性	值	属性描述
custom_fields	BOOLEAN	
dimensionlist	[dimension1 ... dimensionN]	
data_struct	Multiple	
	Single	
metric_fields	字段	
both_target_and_input	[f1 ... fN]	
targets	[f1 ... fN]	
candidate_inputs	[f1 ... fN]	
forced_inputs	[f1 ... fN]	
use_timestamp	Timestamp	
	Period	

表 149: tcmnode 属性 (继续)

tcmnode 属性	值	属性描述
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	<i>string</i>	
period_start_value	整数	
num_days_per_week	整数	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	整数	
start_hour_of_day	整数	
timestamp_increments	整数	

表 149: <i>tcmnode</i> 属性 (继续)		
<b>tcmnode</b> 属性	值	属性描述
cyclic_increments	整数	
cyclic_periods	列表	
output_interval	None  Year  Quarter  Month  Week  Day  Hour  Minute  Second	
is_same_interval	Same  Notsame	
cross_hour	<i>BOOLEAN</i>	
aggregate_and_distribute	列表	
aggregate_default	Mean  Sum  Mode  Min  Max	
distribute_default	Mean  Sum	

表 149: tcmnode 属性 (继续)

tcmnode 属性	值	属性描述
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend None	
k_mean_param	整数	
k_median_param	整数	
missing_value_threshold	整数	
conf_level	整数	
max_num_predictor	整数	
max_lag	整数	
epsilon	成员	
threshold	整数	
is_re_est	BOOLEAN	
num_targets	整数	
percent_targets	整数	
fields_display	列表	
series_display	列表	
network_graph_for_target	BOOLEAN	
sign_level_for_target	成员	
fit_and_outlier_for_target	BOOLEAN	
sum_and_para_for_target	BOOLEAN	
impact_diag_for_target	BOOLEAN	

表 149: tcmnode 属性 (继续)		
tcmnode 属性	值	属性描述
impact_diag_type_for_target	Effect Cause Both	
impact_diag_level_for_target	整数	
series_plot_for_target	BOOLEAN	
res_plot_for_target	BOOLEAN	
top_input_for_target	BOOLEAN	
forecast_table_for_target	BOOLEAN	
same_as_for_target	BOOLEAN	
network_graph_for_series	BOOLEAN	
sign_level_for_series	成员	
fit_and_outlier_for_series	BOOLEAN	
sum_and_para_for_series	BOOLEAN	
impact_diagram_for_series	BOOLEAN	
impact_diagram_type_for_series	Effect Cause Both	
impact_diagram_level_for_series	整数	
series_plot_for_series	BOOLEAN	
residual_plot_for_series	BOOLEAN	
forecast_table_for_series	BOOLEAN	
outlier_root_cause_analysis	BOOLEAN	
causal_levels	整数	
outlier_table	Interactive Pivot Both	
rmisp_error	BOOLEAN	

表 149: tcmnode 属性 (继续)		
tcmnode 属性	值	属性描述
bic	BOOLEAN	
r_square	BOOLEAN	
outliers_over_time	BOOLEAN	
series_transormation	BOOLEAN	
use_estimation_period	BOOLEAN	
estimation_period	Times Observation	
observations	列表	
observations_type	Latest Earliest	
observations_num	整数	
observations_exclude	整数	
extend_records_into_future	BOOLEAN	
forecastperiods	整数	
max_num_distinct_values	整数	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	BOOLEAN	
aic	BOOLEAN	
rmse	BOOLEAN	

## ts 属性



时间序列节点估计时间序列数据的指数平滑模型、单变量自回归整合移动平均值 (ARIMA) 模型和多变量 ARIMA (即变换函数) 模型，并生成未来性能的预测数据。此“时间序列”节点类似于 SPSS 建模器 V18 中不推荐使用的先前“时间序列”节点。但是，此较新“时间序列”节点旨在利用 IBM SPSS 分析服务器的能力来处理大数据，并在 SPSS 建模器 V17 中添加的输出查看器中显示生成的模型。



表 150: ts 属性		
ts 属性	值	属性描述
targets	<i>field</i>	“时间序列”节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率和权重字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
candidate_inputs	[ <i>field1</i> ... <i>fieldN</i> ]	模型所使用的输入或预测变量字段。
use_period	标志	
date_time_field	<i>field</i>	
input_interval	None  Unknown  Year  Quarter  Month  Week  Day  Hour  Hour_nonperiod  Minute  Minute_nonperiod  Second  Second_nonperiod	
period_field	<i>field</i>	
period_start_value	整数	
num_days_per_week	整数	

表 150: ts 属性 (继续)		
ts 属性	值	属性描述
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	整数	
start_hour_of_day	整数	
timestamp_increments	整数	
cyclic_increments	整数	
cyclic_periods	列表	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	标志	
cross_hour	标志	
aggregate_and_distribute	列表	

表 150: ts 属性 (继续)		
ts 属性	值	属性描述
aggregate_default	Mean	
	Sum	
	Mode	
	Min	
	Max	
distribute_default	Mean	
	Sum	
group_default	Mean	
	Sum	
	Mode	
	Min	
	Max	
missing_imput	Linear_interp	
	Series_mean	
	K_mean	
	K_median	
	Linear_trend	
k_span_points	整数	
use_estimation_period	标志	
estimation_period	Observations	
	Times	
date_estimation	列表	仅在使用 date_time_field 时可用
period_estimation	列表	仅在使用 use_period 时可用
observations_type	Latest	
	Earliest	

表 150: ts 属性 (继续)		
ts 属性	值	属性描述
observations_num	整数	
observations_exclude	整数	
method	ExpertModeler  Exsmooth  Arima	
expert_modeler_method	ExpertModeler  Exsmooth  Arima	
consider_seasonal	标志	
detect_outliers	标志	
expert_outlier_additive	标志	
expert_outlier_level_shift	标志	
expert_outlier_innovational	标志	
expert_outlier_level_shift	标志	
expert_outlier_transient	标志	
expert_outlier_seasonal_additive	标志	
expert_outlier_local_trend	标志	
expert_outlier_additive_patch	标志	
consider_newesmodels	标志	

表 150: ts 属性 (继续)		
ts 属性	值	属性描述
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	指定“指数平滑”方法。缺省值为 Simple。

表 150: ts 属性 (继续)

ts 属性	值	属性描述
futureValue_type_method	Compute  specify	<p>如果使用 Compute, 那么系统计算每个预测变量的预测期的“未来值”。</p> <p>对于每个预测变量, 您可以从函数列表中进行选择 (空白、最近点的均数、最近值), 或者使用 specify 以手动输入值。要指定单个字段和属性, 请使用 extend_metric_values 属性。例如:</p> <pre>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', ''}]</pre>
exsmooth_transformation_type	None  SquareRoot  NaturalLog	
arma.p	整数	
arma.d	整数	
arma.q	整数	
arma.sp	整数	
arma.sd	整数	
arma.sq	整数	
arma_transformation_type	None  SquareRoot  NaturalLog	
arma_include_constant	标志	
tf_arma.p.fieldname	整数	用于转换函数。
tf_arma.d.fieldname	整数	用于转换函数。
tf_arma.q.fieldname	整数	用于转换函数。
tf_arma.sp.fieldname	整数	用于转换函数。
tf_arma.sd.fieldname	整数	用于转换函数。

表 150: ts 属性 (继续)		
ts 属性	值	属性描述
tf_arma.sql. <i>fieldname</i>	整数	用于转换函数。
tf_arma.delay. <i>fieldname</i>	整数	用于转换函数。
tf_arma.transformation_type. <i>fieldname</i>	None  SquareRoot  NaturalLog	用于转换函数。
arma_detect_outliers	标志	
arma_outlier_additive	标志	
arma_outlier_level_shift	标志	
arma_outlier_innovational	标志	
arma_outlier_transient	标志	
arma_outlier_seasonal_additive	标志	
arma_outlier_local_trend	标志	
arma_outlier_additive_patch	标志	
max_lags	整数	
cal_PI	标志	
conf_limit_pct	<i>real</i>	
events	字段	
continue	标志	
scoring_model_only	标志	用于包含大量（数万）时间序列的模型。
forecastperiods	整数	
extend_records_into_future	标志	
extend_metric_values	字段	允许您为预测变量提供未来值。
conf_limits	标志	
noise_res	标志	
max_models_output	整数	用于控制输出中显示的模型的数量。缺省值为 10。如果构建的模型总数超过此值，那么模型不会显示在输出。模型仍可用于评分。

# timeseriesnode 属性（不推荐）



注: 此原始“时间序列”节点在 SPSS 建模器 V18 中已不推荐使用，并且替换为旨在利用 IBM SPSS 分析服务器的能力并处理大数据的新的“时间序列”节点。

时间序列节点估计时间序列数据的指数平滑模型、单变量自回归整合移动平均值 (ARIMA) 模型和多变量 ARIMA（即变换函数）模型，并生成未来性能的预测数据。在时间序列节点之前必须有时间间隔节点。

示例

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

表 151: timeseriesnode 属性		
timeseriesnode 属性	值	属性描述
targets	field	“时间序列”节点可以预测一个或多个目标，可以选择使用一个或多个输入字段作为预测变量。不使用频率和权重字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
continue	标志	
method	ExpertModeler  Exsmooth  Arima  Reuse	
expert_modeler_method	标志	
consider_seasonal	标志	
detect_outliers	标志	
expert_outlier_additive	标志	
expert_outlier_level_shift	标志	
expert_outlier_innovational	标志	
expert_outlier_level_shift	标志	
expert_outlier_transient	标志	
expert_outlier_seasonal_additive	标志	
expert_outlier_local_trend	标志	
expert_outlier_additive_patch	标志	



表 151: <i>timeseriesnode</i> 属性 (继续)		
<b>timeseriesnode</b> 属性	值	属性描述
exsmooth_model_type	Simple  HoltsLinearTrend  BrownsLinearTrend  DampedTrend  SimpleSeasonal  WintersAdditive  WintersMultiplicative	
exsmooth_transformation_type	None  SquareRoot  NaturalLog	
arma_p	整数	
arma_d	整数	
arma_q	整数	
arma_sp	整数	
arma_sd	整数	
arma_sq	整数	
arma_transformation_type	None  SquareRoot  NaturalLog	
arma_include_constant	标志	
tf_arma_p.fieldname	整数	用于转换函数。
tf_arma_d.fieldname	整数	用于转换函数。
tf_arma_q.fieldname	整数	用于转换函数。
tf_arma_sp.fieldname	整数	用于转换函数。
tf_arma_sd.fieldname	整数	用于转换函数。
tf_arma_sq.fieldname	整数	用于转换函数。
tf_arma_delay.fieldname	整数	用于转换函数。

表 151: *timeseriesnode* 属性 (继续)

<b>timeseriesnode</b> 属性	值	属性描述
tf_arma_transformation_type. <i>fieldname</i>	None  SquareRoot  NaturalLog	用于转换函数。
arma_detect_outlier_mode	None  Automatic	
arma_outlier_additive	标志	
arma_outlier_level_shift	标志	
arma_outlier_innovational	标志	
arma_outlier_transient	标志	
arma_outlier_seasonal_additive	标志	
arma_outlier_local_trend	标志	
arma_outlier_additive_patch	标志	
conf_limit_pct	<i>real</i>	
max_lags	整数	
events	字段	
scoring_model_only	标志	用于包含大量（数万）时间序列的模型。

## treeas 属性



树 AS 节点类似于现有的 CHAID 节点；但是，“树 AS”节点旨在处理大量数据以创建单个树，并在 SPSS 建模器 V17 中添加的输出查看器中显示生成的模型。此节点通过使用卡方统计 (CHAID) 来识别最优拆分，从而生成决策树。对 CHAID 的这一使用可生成非二元树，意味着某些拆分将具有两个以上的分支。目标和输入字段可以是数字范围（连续）或分类。Exhaustive CHAID 是 CHAID 的修正版，它对所有分割进行更彻底的检查，但计算时间比较长。

表 152: *treeas* 属性

<b>treeas</b> 属性	值	属性描述
target	<i>field</i>	在 Tree-AS 节点中，CHAID 模型需要单个目标和一个或多个输入字段。还可以指定频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
method	chaid  exhaustive_chaid	
max_depth	整数	最大树深度（从 0 到 20）。缺省值为 5。

表 152: *treeas* 属性 (继续)

<b>treeas 属性</b>	<b>值</b>	<b>属性描述</b>
num_bins	整数	仅当数据由连续输入组成时才使用。设置要用于输入的等频分级数；选项为：2、4、5、10、20、25、50 或 100。
record_threshold	整数	记录数，在达到此数量的情况下在构建树时模型将从使用 p 值切换为效应大小。缺省值为 1,000,000；请按增量 10,000 将此增大或减小。
split_alpha	成员	分割的显著性水平。该值必须介于 0.01 和 0.99 之间。
merge_alpha	成员	合并的显著性水平。该值必须介于 0.01 和 0.99 之间。
bonferroni_adjustment	标志	使用 Bonferroni 法调整显著性值。
effect_size_threshold_count	成员	设置在使用连续目标拆分节点和合并类别时的效应大小阈值。该值必须介于 0.01 和 0.99 之间。
effect_size_threshold_cat	成员	设置使用分类目标拆分节点和合并类别时的效应大小阈值。该值必须介于 0.01 和 0.99 之间。
split_merged_categories	标志	允许对合并的类别进行再分割。
grouping_sig_level	成员	用于确定如何形成节点组或如何识别异常节点。
chi_square	pearson likelihood_ratio	这是用于计算卡方统计的方法：Pearson 或似然比
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	成员	缺省值为 2。最小值为 1，最大值为 100，增量为 1。父级分支值必须高于子级分支。
min_child_records_pc	成员	缺省值为 1。最小值为 1，最大值为 100，增量为 1。
min_parent_records_abs	成员	缺省值为 100。最小值为 1，最大值为 100，增量为 1。父级分支值必须高于子级分支。
min_child_records_abs	成员	缺省值为 50。最小值为 1，最大值为 100，增量为 1。
epsilon	成员	期望单元格频率的最小变化值。
max_iterations	成员	收敛的最大迭代次数。
use_costs	标志	

表 152: *treeas* 属性 (继续)

treeas 属性	值	属性描述
costs	结构化	结构化属性。格式是由 3 个值组成的列表：实际值、预测值和成本（如果预测错误）。例如：  tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none  linear  square  custom	注: 仅对有序目标启用。  设置成本矩阵中的缺省值。
calculate_conf	标志	
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。

## twostepnode 属性



TwoStep 节点使用二阶聚类方法。第一步完成简单数据处理，以便将原始输入数据压缩为可管理的子聚类集合。第二步使用层级聚类方法将子聚类一步一步合并为更大的聚类。TwoStep 具有一个优点，就是能够为训练数据自动估计最佳聚类数。它可以高效处理混合的字段类型和大型的数据集。

### 示例

```
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)
```

表 153: *twostepnode* 属性

twostepnode 属性	值	属性描述
inputs	[field1 ... fieldN]	TwoStep 模型使用输入字段的列表，但不使用目标。不识别权重字段和频率字段。有关更多信息，请参阅主题 <a href="#">第 189 页的『公共建模节点属性』</a> 。
standardize	标志	

表 153: <i>twostepnode</i> 属性 (继续)		
<b>twostepnode</b> 属性	值	属性描述
exclude_outliers	标志	
percentage	成员	
cluster_num_auto	标志	
min_num_clusters	成员	
max_num_clusters	成员	
num_clusters	成员	
cluster_label	String Number	
label_prefix	<i>string</i>	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

## twostepAS 属性



“二阶聚类”是一个探索工具，用于揭示数据集中原本不明显的自然分组（即聚类）。此过程使用的算法有多个不错的特征使其有别于传统聚类技术，例如处理分类和连续变量、聚类数目的自动选择以及可扩展性。

表 154: <i>twostepAS</i> 属性		
<b>twostepAS</b> 属性	值	属性描述
inputs	$[f_1 \dots f_N]$	TwoStepAS 模型使用一系列输入字段，但不使用目标字段。不识别权重字段和频率字段。
use_predefined_roles	布尔值	缺省值为 True
use_custom_field_assignments	布尔值	缺省值为 False
cluster_num_auto	布尔值	缺省值为 True
min_num_clusters	integer	Default=2
max_num_clusters	integer	Default=15
num_clusters	integer	Default=5
clustering_criterion	AIC BIC	

表 154: twostepAS 属性 (继续)

twostepAS 属性	值	属性描述
automatic_clustering_method	use_clustering_criterion_setting  Distance_jump  Minimum  Maximum	
feature_importance_method	use_clustering_criterion_setting  effect_size	
use_random_seed	布尔值	
random_seed	integer	
distance_measure	Euclidean  Loglikelihood	
include_outlier_clusters	布尔值	缺省值为 True
num_cases_in_feature_tree_leaf_is_less_than	integer	Default=10
top_perc_outliers	integer	Default=5
initial_dist_change_threshold	integer	Default=0
leaf_node_maximum_branches	integer	Default=8
non_leaf_node_maximum_branches	integer	Default=8
max_tree_depth	integer	Default=3
adjustment_weight_on_measurement_level	integer	Default=6
memory_allocation_mb	数字	Default=512
delayed_split	布尔值	缺省值为 True
fields_to_standardize	[f1 ... fN]	
adaptive_feature_selection	布尔值	缺省值为 True
featureMisPercent	integer	Default=70
coefRange	数字	Default=0.05
percCasesSingleCategory	integer	Default=95
numCases	integer	Default=24
include_model_specifications	布尔值	缺省值为 True
include_record_summary	布尔值	缺省值为 True
include_field_transformations	布尔值	缺省值为 True

表 154: twostepAS 属性 (继续)		
twostepAS 属性	值	属性描述
excluded_inputs	布尔值	缺省值为 True
evaluate_model_quality	布尔值	缺省值为 True
show_feature_importance_bar chart	布尔值	缺省值为 True
show_feature_importance_ word_cloud	布尔值	缺省值为 True
show_outlier_clusters interactive_table_and_chart	布尔值	缺省值为 True
show_outlier_clusters_pivot_table	布尔值	缺省值为 True
across_cluster_feature_importance	布尔值	缺省值为 True
across_cluster_profiles_pivot_table	布尔值	缺省值为 True
withinprofiles	布尔值	缺省值为 True
cluster_distances	布尔值	缺省值为 True
cluster_label	String	
	Number	
label_prefix	String	





# 第 14 章 模型块节点属性

模型块节点具有与其他节点相同的通用属性。有关更多信息，请参阅主题 [第 63 页的『通用节点属性』](#)。

## applyanomalydetectionnode 属性

异常检测建模节点可用于生成异常检测模型块。该模型块的脚本编制名称为 *applyanomalydetectionnode*。有关编写建模节点本身脚本的更多信息，请参阅 [第 189 页的『anomalydetectionnode 属性』](#)

表 155: applyanomalydetectionnode 属性		
applyanomalydetectionnode 属性	值	属性描述
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	确定创建哪些输出进行评分。
num_fields	整数	要报告的字段。
discard_records	标志	指示是否从输出中废弃记录。
discard_anomalous_records	标志	指示废弃异常记录还是废弃非异常记录。缺省值为 <b>off</b> ，表示将废弃非异常记录。否则，如果值为 <b>on</b> ，那么将废弃异常记录。仅当启用了 <b>discard_records</b> 属性时，才会启用此属性。

## applyapriorinode 属性

可使用 Apriori 建模节点生成 Apriori 模型块。该模型块的脚本名称为 *applyapriorinode*。有关编写建模节点本身脚本的更多信息，请参阅 [第 191 页的『apriorinode 属性』](#)

表 156: applyapriorinode 属性		
applyapriorinode 属性	值	属性描述
max_predictions	数字（整数）	
ignore_unmatached	标志	
allow_repeats	标志	
check_basket	NoPredictions Predictions NoCheck	

表 156: *applyapriorinode* 属性 (继续)

<b>applyapriorinode</b> 属性	值	属性描述
criterion	Confidence  Support  RuleSupport  Lift  Deployability	

## applyassociationrulesnode 属性

“关联规则”建模节点可以用于生成关联规则模型块。该模型块的脚本名称为 *applyassociationrulesnode*。有关编写建模节点自身脚本的更多信息，请参阅第 192 页的『*associationrulesnode* 属性』。

表 157: *applyassociationrulesnode* 属性

<b>applyassociationrulesnode</b> 个属性	数据类型	属性描述
max_predictions	整数	可以应用于分数的每个输入的最大规则数。
criterion	Confidence  Rulesupport  Lift  Conditionsupport  Deployability	选择用于确定规则强度的测量。
allow_repeats	BOOLEAN	确定是否将预测相同的规则包括在分数中。
check_input	NoPredictions  Predictions  NoCheck	

## applyautoclassifiernode 属性

“自动分类器”建模节点可用于生成“自动分类器”模型块。此模型块的脚本名称为 *applyautoclassifiernode*。有关编写建模节点自身脚本的详细信息，请参阅第 195 页的『*autoclassifiernode* 属性』。

表 158: *applyautoclassifiernode* 属性

<b>applyautoclassifiernode</b> 属性	值	属性描述
flag_ensemble_method	Voting  EvaluationWeightedVoting  ConfidenceWeightedVoting  RawPropensityWeightedVoting  HighestConfidence  AverageRawPropensity	指定用于确定整体评分的方法。仅当选定的目标是标志字段时，才会应用此设置。
flag_evaluation_selection	Accuracy  AUC_ROC	此选项仅适用于标志目标，以决定选择哪个评估测量进行评估加权投票。
filter_individual_model_output	标志	指定是否应抑制各个模型的评分结果。
is_ensemble_update	标志	启用连续自动机器学习方式，以将新的组件模型添加到现有自动模型集而不是替换现有自动模型，并使用新提供的数据重新评估现有组件模型的测量。
is_auto_ensemble_weights_reevaluation	标志	启用自动模型权重重新评估。
use_accumulated_factor	标志	累计因子用于计算累计测量。
accumulated_factor	<i>number (double)</i>	最大值为 0.99，最小值为 0.85。
use_accumulated_reducing	标志	基于模型刷新期间的累计限制来执行模型缩减。
accumulated_reducing_limit	<i>number (double)</i>	最大值为 0.7，最小值为 0.1。
use_accumulated_weighted_evaluation	标志	累计评估测量用于在选择评估加权投票方法作为整体方法时的投票。
flag_voting_tie_selection	Random  HighestConfidence  RawPropensity	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是标志字段时，才会应用此设置。

表 158: *applyautoclassifiernode* 属性 (继续)

<b>applyautoclassifiernode</b> 属性	值	属性描述
set_ensemble_method	Voting  EvaluationWeightedVoting  ConfidenceWeightedVoting  HighestConfidence	指定用于确定整体评分的方法。仅当选定的目标为集合字段时，才会应用此设置。
set_voting_tie_selection	Random  HighestConfidence	如果已选定投票方法，那么指定解决结的方法。仅当选定的目标是名义字段时，才会应用此设置。

## applyautoclusternode 属性

您可以使用“自动聚类”建模节点来生成“自动聚类”模型块。该模型块的脚本名称为 *applyautoclusternode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 197 页的『[autoclusternode 属性](#)』。

## applyautonumericnode 属性

您可以使用“自动数字”建模节点来生成“自动数字”模型块。此模型块的脚本名称为 *applyautonumericnode*。有关编写建模节点自身脚本的详细信息，请参阅第 199 页的『[autonumericnode 属性](#)』。

表 159: *applyautonumericnode* 属性

<b>applyautonumericnode</b> 属性	值	属性描述
calculate_standard_error	标志	

## applybayesnetnode 属性

您可以使用“贝叶斯网络”建模节点来生成“贝叶斯网络”模型块。该模型块的脚本名称为 *applybayesnetnode*。有关编写建模节点自身脚本的详细信息，请参阅第 201 页的『[bayesnetnode 属性](#)』。

表 160: *applybayesnetnode* 属性

<b>applybayesnetnode</b> 属性	值	属性描述
all_probabilities	标志	
raw_propensity	标志	
adjusted_propensity	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applyc50node 属性

可使用 C5.0 建模节点生成 C5.0 模型块。该模型块的脚本名称为 *applyc50node*。有关编写建模节点自身脚本的详细信息，请参阅第 203 页的『c50node 属性』。

表 161: <i>applyc50node</i> 属性		
applyc50node 属性	值	属性描述
sql_generate	udf  Never  NoMissingValues	用于设置规则集执行期间的 SQL 生成选项。缺省值为 udf。
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applycarmanode 属性

可使用 CARMA 建模节点生成 CARMA 模型块。该模型块的脚本名称为 *applycarmanode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 204 页的『carmanode 属性』。

## applycartnode 属性

可使用 C&R 树建模节点生成 C&R 树模型块。该模型块的脚本名称为 *applycartnode*。有关编写建模节点自身脚本的详细信息，请参阅第 205 页的『cartnode 属性』。

表 162: <i>applycartnode</i> 属性		
applycartnode 属性	值	属性描述
enable_sql_generation	Never  MissingValues  NoMissingValues	用于设置规则集执行期间的 SQL 生成选项。
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applychaidnode 属性

可使用 CHAID 建模节点生成 CHAID 模型块。该模型块的脚本名称为 *applychaidnode*。有关编写建模节点自身脚本的详细信息，请参阅第 208 页的『chaidnode 属性』。

表 163: <i>applychaidnode</i> 属性		
<b>applychaidnode</b> 属性	值	属性描述
enable_sql_generation	Never MissingValues	用于设置规则集执行期间的 SQL 生成选项。
calculate_conf	标志	
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applycoxregnode 属性

可使用 Cox 建模节点生成 Cox 模型块。该模型块的脚本名称为 *applycoxregnode*。有关编写建模节点自身脚本的详细信息，请参阅第 210 页的『*coxregnode* 属性』。

表 164: <i>applycoxregnode</i> 属性		
<b>applycoxregnode</b> 属性	值	属性描述
future_time_as	Intervals Fields	
time_interval	成员	
num_future_times	整数	
time_field	<i>field</i>	
past_survival_time	<i>field</i>	
all_probabilities	标志	
cumulative_hazard	标志	

## applydecisionlistnode 属性

决策列表建模节点可用于生成决策列表模型块。该模型块的脚本名称为 *applydecisionlistnode*。有关编写建模节点自身脚本的详细信息，请参阅第 213 页的『*decisionlistnode* 属性』。

表 165: <i>applydecisionlistnode</i> 属性		
<b>applydecisionlistnode</b> 属性	值	属性描述
enable_sql_generation	标志	值为真时，IBM SPSSModeler 会尝试将决策列表模型回推到 SQL。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

# applydiscriminantnode 属性

判别建模节点可用于生成判别模型块。该模型块的脚本名称为 *applydiscriminantnode*。有关编写建模节点自身脚本的详细信息，请参阅第 214 页的『discriminantnode 属性』。

表 166: <i>applydiscriminantnode</i> 属性		
applydiscriminantnode 属性	值	属性描述
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

# applyextension 属性



“扩展模型”节点可以用于生成“扩展”模型块。此模型块的脚本编写名称为 *applyextension*。有关编写建模节点自身脚本的更多信息，请参阅第 216 页的『extensionmodelnode 属性』。

## Python for Spark 示例

```
#### script example for Python for Spark
applyModel = stream.findByType("extension_apply", None)

score_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTreeModel
from pyspark.sql.types import StringType, StructField

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()

    _modelPath = cxt.getModelContentToPath("TreeModel")
    metadata = json.loads(cxt.getModelContentToString("model.dm"))

    schema = df.dtypes[:]
    target = "Drug"
    predictors = ["Age", "BP", "Sex", "Cholesterol", "Na", "K"]

    lookup = {}
    for i in range(0, len(schema)):
        lookup[schema[i][0]] = i

    def row2LabeledPoint(dm, lookup, target, predictors, row):
        target_index = lookup[target]
        tval = dm[target_index].index(row[target_index])
        pvals = []
        for predictor in predictors:
            predictor_index = lookup[predictor]
            if isinstance(dm[predictor_index], list):
                pval = row[predictor_index] in dm[predictor_index] and
dm[predictor_index].index(row[predictor_index]) or -1
            else:
                pval = row[predictor_index]
            pvals.append(pval)
```

```

        return LabeledPoint(tval, DenseVector(pvals))

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
# score the model, produces an RDD containing just double values
predictions = treeModel.predict(lps.map(lambda lp: lp.features))

def addPrediction(x,dm,lookup,target):
    result = []
    for _idx in range(0, len(x[0])):
        result.append(x[0][_idx])
    result.append(dm[lookup[target]][int(x[1])])
    return result

_schema = cxt.getSparkInputSchema()
_schema.fields.append(StructField("Prediction", StringType(), nullable=True))
rdd2 = df.rdd.zip(predictions).map(lambda x:addPrediction(x, metadata, lookup, target))
outDF = cxt.getSparkSQLContext().createDataFrame(rdd2, _schema)

cxt.setSparkOutputData(outDF)
"""
applyModel.setPropertyValue("python_syntax", score_script)

```

## R 示例

```

##### 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)""")

```

表 167: *applyextension* 属性

applyextension 属性	值	属性说明
r_syntax	string	这是用于进行模型评分的 R 脚本语法。
python_syntax	string	用于模型评分的 Python 脚本编写语法。
use_batch_size	标志	允许使用批处理。
batch_size	整数	指定要包括在每个批次的数据记录数。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。

## applyfactornode 属性

“主成分分析/因子”建模节点可用于生成“主成分分析/因子”模型块。该模型块的脚本名称为 *applyfactornode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 218 页的『[factornode 属性](#)』。



## applyfeatureselectionnode 属性

“特征选择”建模节点可用于生成“特征选择”模型块。该模型块的脚本名称为 *applyfeatureselectionnode*。有关编写建模节点自身脚本的详细信息，请参阅第 220 页的『*featureselectionnode* 属性』。

表 168: <i>applyfeatureselectionnode</i> 属性		
<b>applyfeatureselectionnode</b> 属性	值	属性描述
selected_ranked_fields		指定在模型浏览器中检查哪些已排序字段。
selected_screened_fields		指定在模型浏览器中检查哪些已筛选字段。

## applygeneralizedlinearnode 属性

“广义线性 (genlin)”建模节点可用于生成广义线性模型块。该模型块的脚本名称为 *applygeneralizedlinearnode*。有关编写建模节点自身脚本的详细信息，请参阅第 222 页的『*genlinnode* 属性』。

表 169: <i>applygeneralizedlinearnode</i> 属性		
<b>applygeneralizedlinearnode</b> 属性	值	属性描述
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applyglmnode 属性

GLMM 建模节点可用于生成 GLMM 模型块。该模型块的脚本名称为 *applyglmnode*。有关编写建模节点自身脚本的详细信息，请参阅第 227 页的『*glmnode* 属性』。

表 170: <i>applyglmnode</i> 属性		
<b>applyglmnode</b> 属性	值	属性描述
confidence	onProbability onIncrease	计算评分置信度值的基础：最高预测概率或者最高与次高预测概率之差。
score_category_probabilities	标志	如果设置为 True，那么会为分类目标生成预测概率。为每个类别创建一个字段。缺省值为 False。
max_categories	整数	要预测概率的最大类别数目。仅当 <i>score_category_probabilities</i> 为 True 时才使用。
score_propensity	标志	如果设置为 True，那么为具有标志目标的模型生成原始倾向评分（“True”结果的可能性）。如果分区处于有效，则模型还会根据测试分区产生调整后的倾向评分。缺省值为 False。

表 170: *applyglmnode* 属性 (继续)

<b>applyglmnode 属性</b>	<b>值</b>	<b>属性描述</b>
enable_sql_generation	udf  native	用于在流执行期间设置 SQL 生成选项。这些选项回送到数据库并使用 SPSS® Modeler Server 评分适配器（如果已连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内部评分。  缺省值为 udf。

## applygle 属性

可以将 GLE 建模节点用于生成 GLE 模型块。此模型块的脚本编制名称为 *applygle*。有关编写建模节点自身脚本的更多信息，请参阅第 231 页的『gle 属性』。

表 171: *applygle* 属性

<b>applygle 属性</b>	<b>值</b>	<b>属性描述</b>
enable_sql_generation	udf  native	用于在流执行期间设置 SQL 生成选项。选择后推到数据库并使用 SPSSModeler 服务器 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS 建模器 内评分。

## applygmm 属性

高斯混合节点可用于生成高斯混合模型块。此模型块的脚本编写名称为 *applygmm*。下表中的属性在 V18.2.1.1 和更高版本中可用。有关编写建模节点自身脚本的更多信息，请参阅第 381 页的『gmm 属性』。

表 172: *applygmm* 属性

<b>applygmm 个属性</b>	<b>数据类型</b>	<b>属性描述</b>
centers		
item_count		
total		
dimension		
components		
partition		

## applykmeansnode 属性

K-Means 建模节点可用于生成 K-Means 模型块。该模型块的脚本名称为 *applykmeansnode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 238 页的『kmeansnode 属性』。

## applyknnnode 属性

KNN 建模节点可用于生成 KNN 模型块。此模型块的脚本名称是 *applyknnnode*。有关编写建模节点自身脚本的详细信息，请参阅第 240 页的『knnnode 属性』。

表 173: <i>applyknnnode</i> 属性		
<b>applyknnnode</b> 属性	值	属性描述
all_probabilities	标志	
save_distances	标志	

## applykohonennode 属性

Kohonen 建模节点可用于生成 Kohonen 模型块。该模型块的脚本名称为 *applykohonennode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 203 页的『*c50node* 属性』。

## applylinearnode 属性

可使用线性建模节点生成线性模型块。该模型块的脚本名称为 *applylinearnode*。有关编写建模节点自身脚本的详细信息，请参阅第 243 页的『*linearnode* 属性』。

表 174: <i>applylinearnode</i> 属性		
<b>linear</b> 属性	值	属性描述
use_custom_name	标志	
custom_name	string	
enable_sql_generation	udf  native  puresql	用于在流执行期间设置 SQL 生成选项。这些选项回送到数据库并使用 SPSS® Modeler Server 评分适配器（如果已连接到安装了评分适配器的数据库）进行评分，在 SPSS Modeler 内部评分，或者回送到数据库并使用 SQL 评分。  缺省值为 udf。

## applylinearasnode 属性

Linear-AS 建模节点可用于生成线性 AS 模型块。此模型块的脚本编制名称为 *applylinearasnode*。有关编写建模节点自身脚本的详细信息，请参阅第 244 页的『*linearasnode* 属性』。

表 175: <i>applylinearasnode</i> 属性		
<b>applylinearasnode</b> 属性	值	属性描述
enable_sql_generation	udf  native	缺省值为 udf。

## applylogregnode 属性

您可以使用“Logistic 回归模型”建模节点来生成“Logistic 回归模型”模型块。该模型块的脚本名称为 *applylogregnode*。有关编写建模节点自身脚本的详细信息，请参阅第 246 页的『*logregnode* 属性』。

表 176: <i>applylogregnode</i> 属性		
<b>applylogregnode</b> 属性	值	属性描述
calculate_raw_propensiti es	标志	

表 176: *applylogregnode* 属性 (继续)

<b>applylogregnode</b> 属性	值	属性描述
calculate_conf	标志	
enable_sql_generation	标志	

## applysvmnode 属性

可以使用 LSVM 建模节点来生成 LSVM 模型块。此模型块的脚本编制名称为 *applysvmnode*。有关编写建模节点自身脚本的更多信息，请参阅第 251 页的『*svmnode* 属性』。

表 177: *applysvmnode* 属性

<b>applysvmnode</b> 属性	值	属性描述
calculate_raw_propensities	标志	指定是否计算原始倾向评分。
enable_sql_generation	udf native	指定使用评分适配器（如果已安装）进行评分、在进行中评分还是在数据库外部进行评分。

## applyneuralnetnode 属性

“神经网络”建模节点可用于生成“神经网络”模型块。该模型块的脚本名称为 *applyneuralnetnode*。有关编写建模节点自身脚本的详细信息，请参阅第 252 页的『*neuralnetnode* 属性』。

**注意：**在此发行版中提供了具有增强功能的新版本的神经网络模型块，并将在下一节 (*applyneuralnetwork*) 中进行介绍。尽管先前版本仍然可用，但我们建议您更新脚本以使用新的版本。此处保留了先前版本的详细信息以供参考，但会在将来的发行版中不再支持。

表 178: *applyneuralnetnode* 属性

<b>applyneuralnetnode</b> 属性	值	属性描述
calculate_conf	标志	启用 SQL 生成时可用；此属性将置信度的计算包括在生成的树中。
enable_sql_generation	标志	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applyneuralnetworknode 属性

可使用神经网络建模节点生成神经网络模型块。该模型块的脚本名称为 *applyneuralnetworknode*。有关对建模节点自身编写脚本的更多信息，请参阅 *neuralnetworknode* 属性。

表 179: *applyneuralnetworknode* 属性

<b>applyneuralnetworknode</b> 属性	值	属性描述
use_custom_name	标志	

表 179: <i>applyneuralnetworknode</i> 属性 (继续)		
<b>applyneuralnetworknode</b> 属性	值	属性描述
custom_name	string	
confidence	onProbability onIncrease	
score_category_probabilities	标志	
max_categories	成员	
score_propensity	标志	
enable_sql_generation	udf  native  puresql	用于在流执行期间设置 SQL 生成选项。这些选项回送到数据库并使用 SPSS® Modeler Server 评分适配器（如果已连接到安装了评分适配器的数据库）进行评分，在 SPSS Modeler 内部评分，或者回送到数据库并使用 SQL 评分。  缺省值为 udf。

## applyocsvmnode 属性

单类 SVM 节点可用于生成单类 SVM 模型块。此模型块的脚本编写名称为 *applyocsvmnode*。此模型块没有其他属性。有关编写建模节点自身脚本的更多信息，请参阅第 385 页的『ocsvmnode 属性』。

## applyquestnode 属性

可使用 QUEST 建模节点生成 QUEST 模型块。该模型块的脚本名称为 *applyquestnode*。有关编写建模节点自身脚本的详细信息，请参阅第 257 页的『questnode 属性』。

表 180: <i>applyquestnode</i> 属性		
<b>applyquestnode</b> 属性	值	属性描述
enable_sql_generation	Never  MissingValues  NoMissingValues	用于设置规则集执行期间的 SQL 生成选项。
calculate_conf	标志	
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applyr 属性

“R 构建”建模节点可用于生成 R 模型块。此模型块的脚本编制名称为 *applyr*。有关编写建模节点自身脚本的详细信息，请参阅第 202 页的『*buildr* 属性』。

表 181: <i>applyr</i> 属性		
applyr 属性	值	属性说明
score_syntax	string	这是用于进行模型评分的 R 脚本语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
use_batch_size	标志	启用批处理
batch_size	整数	指定要包括在每个批次的数据记录数

## applyrandomtrees 属性

可以使用“随机树建模”节点来生成随机树模型块。此模型块的脚本编制名称为 *applyrandomtrees*。有关编写建模节点自身脚本的更多信息，请参阅第 259 页的『*randomtrees* 属性』。

表 182: <i>applyrandomtrees</i> 属性		
applyrandomtrees 属性	值	属性描述
calculate_conf	标志	此属性将置信度计算包含在生成的树中。
enable_sql_generation	udf  native	用于在流执行期间设置 SQL 生成选项。选择后推到数据库并使用 SPSSModeler 服务器 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS 建模器 内评分。

## asapplyregressionnode 属性

“线性回归”建模节点可用于生成“线性回归”模型块。该模型块的脚本名称为 *applyregressionnode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 261 页的『*regressionnode* 属性』。

## applyselflearningnode 属性

“自学响应模型 (SLRM)”建模节点可用于生成 SLRM 模型块。该模型块的脚本名称为 *applyselflearningnode*。有关编写建模节点自身脚本的详细信息，请参阅第 264 页的『*slrmnode* 属性』。

表 183: <i>applyselflearningnode</i> 属性		
<b>applyselflearningnode</b> 属性	值	属性描述
max_predictions	成员	
randomization	成员	
scoring_random_seed	成员	
sort	ascending descending	指定先显示评分最高还是最低的报价。
model_reliability	标志	将“设置”选项卡中的模型可靠性选项考虑在内。

## applysequencenode 属性

“序列”建模节点可用于生成“序列”模型块。该模型块的脚本名称为 *applysequencenode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 263 页的『[sequencenode 属性](#)』。

## applysvmnode 属性

您可以使用 SVM 建模节点来生成 SVM 模型块。该模型块的脚本名称为 *applysvmnode*。有关编写建模节点自身脚本的详细信息，请参阅第 271 页的『[svmnode 属性](#)』。

表 184: <i>applysvmnode</i> 属性		
<b>applysvmnode</b> 属性	值	属性描述
all_probabilities	标志	
calculate_raw_propensities	标志	
calculate_adjusted_propensities	标志	

## applystpnode 属性

STP 建模节点可以用于生成相关联的模型块，该模型块在输出查看器中显示模型输出。该模型块的脚本名称为 *applystpnode*。有关编写建模节点自身脚本的更多信息，请参阅第 266 页的『[stpnode 属性](#)』。

表 185: <i>applystpnode</i> 属性		
<b>applystpnode</b> 个属性	数据类型	属性描述
uncertainty_factor	BOOLEAN	最小值为 0，最大值为 100。

## applytcmnode 属性

时间因果建模 (TCM) 建模节点可用于生成 TCM 模型块。此模型块的脚本编制名称为 *applytcmnode*。有关编写建模节点自身脚本的更多信息，请参阅第 273 页的『[tcmnode 属性](#)』。

表 186: <i>applytcmnode</i> 属性		
<b>applytcmnode</b> 属性	值	属性描述
ext_future	BOOLEAN	
ext_future_num	整数	

表 186: <i>applytcmnode</i> 属性 (继续)		
<b>applytcmnode</b> 属性	值	属性描述
noise_res	BOOLEAN	
conf_limits	BOOLEAN	
target_fields	列表	
target_series	列表	

## applyts 属性

“时间序列”建模节点可用于生成“时间序列”模型块。此模型块的脚本编制名称为 *applyts*。有关编写建模节点自身脚本的更多信息，请参阅第 278 页的『*ts* 属性』。

表 187: <i>applyts</i> 属性		
<b>applyts</b> 属性	值	属性描述
extend_records_into_future	BOOLEAN	
ext_future_num	整数	
compute_future_values_input	BOOLEAN	
forecastperiods	整数	
noise_res	BOOLEAN	
conf_limits	BOOLEAN	
target_fields	列表	
target_series	列表	
includeTargets	<i>field</i>	

## applytimeseriesnode 属性（不推荐）

“时间序列”建模节点可用于生成“时间序列”模型块。该模型块的脚本名称为 *applytimeseriesnode*。有关编写建模节点自身脚本的详细信息，请参阅第 286 页的『*timeseriesnode* 属性（不推荐）』。

表 188: <i>applytimeseriesnode</i> 属性		
<b>applytimeseriesnode</b> 属性	值	属性描述
calculate_conf	标志	
calculate_residuals	标志	

## applytreeas 属性

Tree-AS 建模节点可用于生成 Tree-AS 模型块。此模型块的脚本编制名称为 *applytreeas*。有关编写建模节点自身脚本的更多信息，请参阅第 288 页的『*treeas* 属性』。

表 189: <i>applytreeas</i> 属性		
<b>applytreeas</b> 属性	值	属性描述
calculate_conf	标志	此属性将置信度计算包含在生成的树中。



表 189: <i>applytreeas</i> 属性 (继续)		
<b>applytreeas 属性</b>	值	属性描述
display_rule_id	标志	在评分输出中添加一个字段，表示每个记录分配到的终端节点的标识。
enable_sql_generation	udf  native	用于在流执行期间设置 SQL 生成选项。选择后推到数据库并使用 SPSSModeler 服务器 评分适配器（如果连接到安装了评分适配器的数据库）进行评分，或者在 SPSS 建模器 内评分。

## applytwostepnode 属性

“二阶”建模节点可用于生成“二阶”模型块。该模型块的脚本名称为 *applytwostepnode*。此模型块没有其他属性。有关编写建模节点自身脚本的详细信息，请参阅第 290 页的『*twostepnode* 属性』。

## applytwostepAS 属性

二阶 AS 建模节点可用于生成二阶 AS 模型块。此模型块的脚本编制名称为 *applytwostepAS*。有关编写建模节点自身脚本的详细信息，请参阅第 291 页的『*twostepAS* 属性』。

表 190: <i>applytwostepAS</i> 属性		
<b>applytwostepAS 属性</b>	值	属性描述
enable_sql_generation	udf  native	用于在流执行期间设置 SQL 生成选项。这些选项回送到数据库并使用 SPSS® Modeler Server 评分适配器（如果已连接到安装了评分适配器的数据库）进行评分，或者在 SPSS Modeler 内部评分。  缺省值为 udf。

## applyxgboosttreenode 属性

XGBoost Tree 节点可用于生成 XGBoost Tree 模型块。此模型块的脚本编写名称为 *applyxgboosttreenode*。下表中的属性在 18.2.1.1 中新增。有关编写建模节点自身脚本的更多信息，请参阅第 392 页的『*xgboosttreenode* 属性』。

表 191: <i>applyxgboosttreenode</i> 属性		
<b>applyxgboosttreenode 个属性</b>	数据类型	属性描述
use_model_name		
model_name		

## applyxgboostlinearnode 属性

XGBoost Linear 节点可用于生成 XGBoost Linear 模型块。此模型块的脚本编写名称为 *applyxgboostlinearnode*。此模型块没有其他属性。有关编写建模节点自身脚本的更多信息，请参阅第 391 页的『*xgboostlinearnode* 属性』。

# hdbscannugget 属性

HDBSCAN 节点可用于生成 HDBSCAN 模型块。此模型块的脚本编写名称为 `hdbscannugget`。此模型块没有其他属性。有关编写建模节点自身脚本的更多信息，请参阅第 381 页的『[hdbscannode 属性](#)』。

# kdeapply 属性

可使用 KDE 建模节点来生成 KDE 模型块。此模型块的脚本编写名称为 `kdeapply`。有关编写建模节点自身脚本的信息，请参阅第 383 页的『[kdemodel 属性](#)』。

表 192: <i>kdeapply</i> 属性		
kdeapply 个属性	数据类型	属性描述
outLogDensity  从 V18.2.1.1 开始，重命名为 out_log_density	BOOLEAN	指定 True 或 False 以在输出中包含或排除日志密度值。缺省值为 False。

# 第 15 章 数据库建模节点属性

IBM SPSSModeler 支持与数据库供应商提供的数据挖掘和建模工具相集成，包括 Microsoft SQL Server Analysis Services、Oracle Data Mining、和 IBM Netezza 分析。您可以使用 IBM SPSSModeler 应用程序自有的数据库算法来构建模型并对模型进行评分。还可以使用本节介绍的属性通过编写脚本来构建和操纵数据库模型。

例如，下面这段脚本说明了如何使用 IBM SPSSModeler 脚本编写界面来创建 Microsoft 决策树模型：

```
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([])
```

## Microsoft 建模的节点属性

### Microsoft 建模节点属性

#### 通用属性

Microsoft 数据库建模节点的公共属性如下所示。

表 193: 公共 Microsoft 节点属性		
公共 Microsoft 节点属性	值	属性说明
analysis_database_name	string	Analysis Services 数据库的名称。
analysis_server_name	string	Analysis Services 主机的名称。
use_transactional_data	标志	指定输入数据是采用表格式还是事务格式。
inputs	列表	表格数据的输入字段。
target	field	预测字段（不适用于“MS 聚类”或“序列聚类”节点）。
unique_field	field	关键字段。
msas_parameters	结构化	算法参数。有关更多信息，请参阅主题 <a href="#">第 315 页的『算法参数』</a> 。

表 193: 公共 Microsoft 节点属性 (继续)		
公共 Microsoft 节点属性	值	属性说明
with_drillthrough	标志	“进行深入钻取”选项。

## MS 决策树

没有为 `mstreenode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 聚类

没有为 `msclusternode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 关联规则

以下特定属性可用于类型为 `msassocnode` 的节点：

表 194: <code>msassocnode</code> 属性		
<code>msassocnode</code> 属性	值	属性说明
<code>id_field</code>	<i>field</i>	标识数据中的每项事务。
<code>trans_inputs</code>	列表	事务数据的输入字段。
<code>transactional_target</code>	<i>field</i>	预测字段（事务数据）。

## MS 朴素贝叶斯

没有为 `msbayesnode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 线性回归

没有为 `msregressionnode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 神经网络

没有为 `msneuralnetworknode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS Logistic 回归

没有为 `mslogisticnode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 时间序列

没有为 `mstimeseriesnode` 类型的节点定义具体属性。请参阅本节开头的公共 Microsoft 属性。

## MS 序列聚类

以下特定属性可用于类型为 `mssequenceclusternode` 的节点：

表 195: <code>mssequenceclusternode</code> 属性		
<code>mssequenceclusternode</code> 属性	值	属性说明
<code>id_field</code>	<i>field</i>	标识数据中的每项事务。
<code>input_fields</code>	列表	事务数据的输入字段。
<code>sequence_field</code>	<i>field</i>	序列标识。

表 195: mssequenceclusternode 属性 (继续)		
mssequenceclusternode 属性	值	属性说明
target_field	field	预测字段（表格数据）。

## 算法参数

每种 Microsoft 数据库模型类型都具有可以使用 msas\_parameters 属性来设置的特定参数，例如：

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters",
[["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

这些参数源自 SQL Server。要查看每个节点的相关参数，请执行如下操作：

1. 将数据库源节点放入画布中。
2. 打开该数据库源节点。
3. 从**数据源**下拉列表选择一个有效源。
4. 从**表名称**列表选择一个有效表。
5. 单击**确定**以关闭该数据库源节点。
6. 附加要列出其属性的 Microsoft 数据库建模节点。
7. 打开该数据库建模节点。
8. 选择 **专家** 选项卡。

此时会显示该节点的可用 msas\_parameters 属性。

## Microsoft 模型块属性

使用 Microsoft 数据库建模节点创建的模型块具有下列属性。

### MS 决策树

表 196: MS 决策树属性		
appliedstreenode 属性	值	描述
analysis_database_name	string	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	string	Analysis 服务器主机的名称。
datasource	string	SQL Server ODBC 数据源名称 (DSN) 的名称。
sql_generate	标志	启用 SQL 生成。
	udf	

## MS 线性回归

表 197: MS 线性回归属性		
<b>applymsregressionnode</b> 属性	值	描述
analysis_database_name	string	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	string	Analysis 服务器主机的名称。

## MS 神经网络

表 198: MS 神经网络属性		
<b>applymsneuralnetworknode</b> 属性	值	描述
analysis_database_name	string	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	string	Analysis 服务器主机的名称。

## MS Logistic 回归

表 199: MS Logistic 回归属性		
<b>applymslogisticnode</b> 属性	值	描述
analysis_database_name	string	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	string	Analysis 服务器主机的名称。

## MS 时间序列

表 200: MS 时间序列属性		
<b>applymstimeseriesnode</b> 属性	值	描述
analysis_database_name	string	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	string	Analysis 服务器主机的名称。
start_from	new_prediction  historical_prediction	指定是进行未来预测还是历史预测。
new_step	成员	定义未来预测的开始时间段。

表 200: MS 时间序列属性 (继续)		
<b>applymstimeseriesnode</b> 属性	值	描述
historical_step	成员	定义历史预测的开始时间段。
end_step	成员	定义预测结束时间段。

## MS 序列聚类

表 201: MS 序列聚类属性		
<b>applymssequenceclusternode</b> 属性	值	描述
analysis_database_name	<i>string</i>	可以直接在流中对此节点进行评分。  此属性用于标识 Analysis Services 数据库的名称。
analysis_server_name	<i>string</i>	Analysis 服务器主机的名称。

## Oracle 建模的节点属性

### Oracle 建模节点属性

Oracle 数据库建模节点的公共属性如下所示。

表 202: 公共 Oracle 节点属性		
公共 <b>Oracle</b> 节点属性	值	属性说明
target	<i>field</i>	
inputs	字段的列表	
partition	<i>field</i>	此域用于将数据分区为不同的样本，以用于模型构建的训练、检验和验证阶段。
datasource		
username		
password		
epassword		
use_model_name	标志	
model_name	<i>string</i>	新模型的定制名称。
use_partitioned_data	标志	如果定义了分区字段，则此选项可确保仅训练分区的数据用于构建模型。
unique_field	<i>field</i>	
auto_data_prep	标志	启用或禁用 Oracle 自动数据准备功能（仅适用于 11g 数据库）。
costs	结构化	格式如下的结构化属性：  [[drugA drugB 1.5] [drugA drugC 2.1]]，其中 [] 中的自变量是实际预测成本。

表 202: 公共 Oracle 节点属性 (继续)		
公共 Oracle 节点属性	值	属性说明
mode	Simple	如在各个节点属性中注释的那样，如果设置为 Simple，会导致忽略某些属性。
	Expert	
use_prediction_probability	标志	
prediction_probability	string	
use_prediction_set	标志	

## Oracle 朴素贝叶斯

类型为 oranbnode 的节点的可用属性如下所示：

表 203: oranbnode 属性		
oranbnode 属性	值	属性说明
singleton_threshold	成员	0.0–1.0.*
pairwise_threshold	成员	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	结构化	格式如下的结构化属性：  <pre>set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]</pre>

\* 如果 mode 设置为 Simple，则忽略属性。

## Oracle Adaptive Bayes

类型为 oraabnnode 的节点的可用属性如下所示：

表 204: oraabnnode 属性		
oraabnnode 属性	值	属性说明
model_type	SingleFeature	
	MultiFeature	
	NaiveBayes	
use_execution_time_limit	标志	*
execution_time_limit	整数	值必须大于 0。*
max_naive_bayes_predictors	整数	值必须大于 0。*
max_predictors	整数	值必须大于 0。*



表 204: oraabnnode 属性 (继续)		
oraabnnode 属性	值	属性说明
priors	Data Equal Custom	
custom_priors	结构化	格式如下的结构化属性：  <pre>set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]</pre>

\* 如果 mode 设置为 Simple，则忽略属性。

## Oracle 支持向量机

类型为 orasvmnode 的节点的可用属性如下所示：

表 205: orasvmnode 属性		
orasvmnode 属性	值	属性说明
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	整数	仅适用于高斯内核。值必须大于 0。*
convergence_tolerance	成员	值必须大于 0。*
use_standard_deviation	标志	仅适用与高斯内核。*
standard_deviation	成员	值必须大于 0。*
use_epsilon	标志	仅适用于回归模型。*
epsilon	成员	值必须大于 0。*
use_complexity_factor	标志	*
complexity_factor	成员	*
use_outlier_rate	标志	仅适用于单类变体。*

表 205: orasvmnode 属性 (继续)		
orasvmnode 属性	值	属性说明
outlier_rate	成员	仅适用于单类变体。0.0–1.0.*
weights	Data  Equal  Custom	
custom_weights	结构化	格式如下的结构化属性：  <pre>set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]</pre>

\* 如果 mode 设置为 Simple，则忽略属性。

## Oracle 广义线性模型

类型为 oraglmnode 的节点的可用属性如下所示：

表 206: oraglmnode 属性		
oraglmnode 属性	值	属性说明
normalization_method	zscore  minmax  none	
missing_value_handling	ReplaceWithMean  UseCompleteRecords	
use_row_weights	标志	*
row_weights_field	<i>field</i>	*
save_row_diagnostics	标志	*
row_diagnostics_table	<i>string</i>	*
coefficient_confidence	成员	*
use_reference_category	标志	*
reference_category	<i>string</i>	*
ridge_regression	Auto  Off  On	*
parameter_value	成员	*

表 206: oraglmnode 属性 (继续)		
oraglmnode 属性	值	属性说明
vif_for_ridge	标志	*

\* 如果 mode 设置为 Simple, 则忽略属性。

## Oracle 决策树

类型为 oradecisiontreenode 的节点的可用属性如下所示：

表 207: oradecisiontreenode 属性		
oradecisiontreenode 属性	值	属性说明
use_costs	标志	
impurity_metric	Entropy Gini	
term_max_depth	整数	2–20.*
term_minpct_node	成员	0.0–10.0.*
term_minpct_split	成员	0.0–20.0.*
term_minrec_node	整数	值必须大于 0。 *
term_minrec_split	整数	值必须大于 0。 *
display_rule_ids	标志	*

\* 如果 mode 设置为 Simple, 则忽略属性。

## Oracle O-Cluster

类型为 oraoclusternode 的节点的可用属性如下所示：

表 208: oraoclusternode 属性		
oraoclusternode 属性	值	属性说明
max_num_clusters	整数	值必须大于 0。
max_buffer	整数	值必须大于 0。 *
sensitivity	成员	0.0–1.0.*

\* 如果 mode 设置为 Simple, 则忽略属性。

## Oracle KMeans

类型为 orakmeansnode 的节点的可用属性如下所示：

表 209: orakmeansnode 属性		
orakmeansnode 属性	值	属性说明
num_clusters	整数	值必须大于 0。

表 209: orakmeansnode 属性 (继续)		
orakmeansnode 属性	值	属性说明
normalization_method	zscore	
	minmax	
	none	
distance_function	Euclidean	
	Cosine	
iterations	整数	0–20.*
conv_tolerance	成员	0.0–0.5.*
split_criterion	Variance	缺省值为 Variance。*
	Size	
num_bins	整数	值必须大于 0。*
block_growth	整数	1–5.*
min_pct_attr_support	成员	0.0–1.0.*

\* 如果 mode 设置为 Simple，则忽略属性。

## Oracle NMF

类型为 oranmfnode 的节点的可用属性如下所示：

表 210: oranmfnode 属性		
oranmfnode 属性	值	属性说明
normalization_method	minmax	
	none	
use_num_features	标志	*
num_features	整数	0-1. 缺省值由算法根据数据估计得出。*
random_seed	成员	*
num_iterations	整数	0–500.*
conv_tolerance	成员	0.0–0.5.*
display_all_features	标志	*

\* 如果 mode 设置为 Simple，则忽略属性。

## Oracle Apriori

类型为 oraapriorinode 的节点的可用属性如下所示：

表 211: oraapriorinode 属性		
oraapriorinode 属性	值	属性说明
content_field	<i>field</i>	
id_field	<i>field</i>	
max_rule_length	整数	2–20。
min_confidence	成员	0.0–1.0.
min_support	成员	0.0–1.0.
use_transactional_data	标志	

## Oracle 最小描述长度 (MDL)

没有为类型为 oramdlnode 的节点定义具体属性。请参阅本章节开头部分的通用 Oracle 属性。

## Oracle 属性重要性 (AI)

类型为 oraainode 的节点的可用属性如下所示：

表 212: oraainode 属性		
oraainode 属性	值	属性说明
custom_fields	标志	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
selection_mode	ImportanceLevel  ImportanceValue  TopN	
select_important	标志	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择“重要”字段。
important_label	<i>string</i>	指定“重要”排序的标签。
select_marginal	标志	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择“边际”字段。
marginal_label	<i>string</i>	指定“边际”排序的标签。
important_above	成员	0.0–1.0.
select_unimportant	标志	在 selection_mode 设置为 ImportanceLevel 时，指定是否选择“不重要”字段。
unimportant_label	<i>string</i>	指定“不重要”排序的标签。
unimportant_below	成员	0.0–1.0.

表 212: <i>oraainode</i> 属性 (继续)		
<b>oraainode</b> 属性	值	属性说明
importance_value	成员	在 <i>selection_mode</i> 设置为 <i>ImportanceValue</i> 时，指定要使用的分界值。接受从 0 到 100 的值。
top_n	成员	在 <i>selection_mode</i> 设置为 <i>TopN</i> 时，指定要使用的分界值。接受从 0 到 1000 的值。

## Oracle 模型块属性

使用 Oracle 模型创建的模型块具有下列属性。

### Oracle 朴素贝叶斯

没有为 *applyoranbnnode* 类型的节点定义具体属性。

### Oracle Adaptive Bayes

没有为 *applyoraabnnode* 类型的节点定义具体属性。

### Oracle 支持向量机

没有为 *applyorasvmnode* 类型的节点定义具体属性。

### Oracle 决策树

类型为 *applyoradecisiontreenode* 的节点的可用属性如下所示：

表 213: <i>applyoradecisiontreenode</i> 属性		
<b>applyoradecisiontreenode</b> 属性	值	属性说明
use_costs	标志	
display_rule_ids	标志	

### Oracle O-Cluster

没有为 *applyoraoclusternode* 类型的节点定义具体属性。

### Oracle KMeans

没有为 *applyorakmeansnode* 类型的节点定义具体属性。

### Oracle NMF

下列属性用于 *applyoranmfnnode* 类型的节点：

表 214: <i>applyoranmfnnode</i> 属性		
<b>applyoranmfnnode</b> 属性	值	属性说明
display_all_features	标志	

### Oracle Apriori

该模型块不能应用于脚本。

## Oracle MDL

该模型块不能应用于脚本。

## IBM Netezza 分析 建模节点属性

### Netezza 建模节点属性

IBM Netezza 数据库建模节点的公共属性如下所示。

表 215: 公共 Netezza 节点属性		
公共 Netezza 节点属性	值	属性说明
custom_fields	标志	如果为 true，则允许您为当前节点指定目标、输入和其他字段。如果为 false，则使用来自上游类型节点的当前设置。
inputs	[field1 ... fieldN]	模型所使用的输入或预测变量字段。
target	field	目标字段（连续或分类）。
record_id	field	要用作唯一记录标识的字段。
use_upstream_connection	标志	如果为 true（缺省），那么连接详细信息在上游节点中指定。在指定了 move_data_to_connection 时不使用。
move_data_connection	标志	如果为 true，则将数据移动到由 connection 指定的数据库。在指定了 use_upstream_connection 时不使用。
connection	结构化	<p>这是用于存储模型的 Netezza 数据库的连接字符串。格式如下的结构化属性：</p> <pre>['odbc' '&lt;dsn&gt;' '&lt;username&gt;' '&lt;psw&gt;' '&lt;catname&gt;' '&lt;conn_attribs&gt;' [true false]]</pre> <p>其中：</p> <p>&lt;dsn&gt; 是数据源名称</p> <p>&lt;username&gt; 和 &lt;psw&gt; 是数据库的用户名和密码</p> <p>&lt;catname&gt; 是目录名称</p> <p>&lt;conn_attribs&gt; 是连接属性</p> <p>true   false 指示是否需要密码。</p>
table_name	string	这是用于存储模型的数据库表的名称。
use_model_name	标志	如果为 true，使用由 model_name 指定的名称作为模型名称，否则采用系统创建的模型名称。
model_name	string	新模型的定制名称。
include_input_fields	标志	如果为 true，向下游传递所有输入字段，否则仅传递模型产生的 record_id 和字段。

## Netezza 决策树

类型为 `netezadectreenode` 的节点的可用属性如下所示：

表 216: netezadectreenode 属性		
netezadectreenode 属性	值	属性说明
impurity_measure	Entropy Gini	对杂质的测量，用于评估树的最佳拆分位置。
max_tree_depth	整数	树可以增长到的最大级别数。缺省值为 62（最大可能值）。
min_improvement_splits	成员	进行分割前必须满足的最低杂质改进。缺省值为 0.01。
min_instances_split	整数	可以进行分割前余下的最小未分割记录数。缺省值为 2（最小可能值）。
weights	结构化	各个类的相对权重。格式如下的结构化属性：  <pre>set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]]</pre> 缺省情况是所有类的权重均为 1。
pruning_measure	Acc wAcc	缺省值为 Acc（准确性）。如果要在应用修剪时将类权重考虑在内，可使用 wAcc（加权精确度）替代。
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	缺省情况下，使用 allTrainingData 来估计模型精确度。使用 partitionTrainingData 来指定要使用训练数据的百分比，或 useOtherTable 来使用源自指定数据库表的训练数据集。
perc_training_data	成员	如果 prune_tree_options 设置为 partitionTrainingData，则指定用于训练的数据所占的百分比。
prune_seed	整数	在 prune_tree_options 设置为 partitionTrainingData 时，用于重复分析结果的随机种子，缺省值是 1。
pruning_table	string	这是用于估计模型精确度的单独修剪数据集的表名称。
compute_probabilities	标志	如果为 true，那么将生成置信度级别（概率）字段以及预测字段。



## Netezza K-Means

类型为 `netezzakmeansnode` 的节点的可用属性如下所示：

表 217: <code>netezzakmeansnode</code> 属性		
<b>netezzakmeansnode</b> 属性	值	属性说明
<code>distance_measure</code>	Euclidean Manhattan Canberra maximum	这是用于对数据点之间的距离进行测量的方法。
<code>num_clusters</code>	整数	要创建的聚类数；缺省值为 3。
<code>max_iterations</code>	整数	算法迭代次数，模型训练在此之后停止；缺省值为 5。
<code>rand_seed</code>	整数	这是用于复制分析结果的随机种子；缺省值为 12345。

## Netezza 贝叶斯网络

类型为 `netezزابayesnode` 的节点的可用属性如下所示：

表 218: <code>netezزابayesnode</code> 属性		
<b>netezزابayesnode</b> 属性	值	属性说明
<code>base_index</code>	整数	对第一个输入字段指定的数字标识，用于进行内部管理；缺省值为 777。
<code>sample_size</code>	整数	属性数目非常大时的采样大小；缺省值为 10,000。
<code>display_additional_information</code>	标志	如果为 <code>true</code> ，则在消息对话框中显示额外的进度信息。
<code>type_of_prediction</code>	best neighbors nn-neighbors	要使用的预测算法类型： <code>best</code> （最相关的相邻值）、 <code>neighbors</code> （相邻值的加权预测）或 <code>nn-neighbors</code> （非空相邻值）。

## Netezza 朴素贝叶斯

类型为 `netezzanaiवेbayesnode` 的节点的可用属性如下所示：

表 219: <code>netezzanaiवेbayesnode</code> 属性		
<b>netezzanaiवेbayesnode</b> 属性	值	属性说明
<code>compute_probabilities</code>	标志	如果为 <code>true</code> ，那么将生成置信度级别（概率）字段以及预测字段。
<code>use_m_estimation</code>	标志	如果为 <code>true</code> ，则使用 <code>m-estimation</code> 技术以避免估算期间的零概率。

## Netezza KNN

类型为 `netezzaknnnode` 的节点的可用属性如下所示：

表 220: <code>netezzaknnnode</code> 属性		
netezzaknnnode 属性	值	属性说明
<code>weights</code>	结构化	这是用于对各个类指定权重的结构化属性。 示例：  <code>set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]</code>
<code>distance_measure</code>	Euclidean Manhattan Canberra Maximum	这是用于对数据点之间的距离进行测量的方法。
<code>num_nearest_neighbors</code>	整数	特定观测值的最近邻元素数；缺省值为 3。
<code>standardize_measurements</code>	标志	如果为 <code>true</code> ，那么在计算距离值之前，对连续输入字段的测量值进行标准化。
<code>use_coresets</code>	标志	如果为 <code>true</code> ，则对大型数据集使用核心集采样以提高计算速度。

## Netezza 分裂式聚类

类型为 `netez zadivclusternode` 的节点的可用属性如下所示：

表 221: <code>netez zadivclusternode</code> 属性		
netez zadivclusternode 属性	值	属性说明
<code>distance_measure</code>	Euclidean Manhattan Canberra Maximum	这是用于对数据点之间的距离进行测量的方法。
<code>max_iterations</code>	整数	在模型训练停止前执行的最大算法迭代次数；缺省值为 5。
<code>max_tree_depth</code>	整数	可以将数据集拆分为的最大级别数；缺省值为 3。
<code>rand_seed</code>	整数	随机种子，用于复制分析；缺省值为 12345。
<code>min_instances_split</code>	整数	可以拆分的最小记录数，缺省值为 5。
<code>level</code>	整数	要将记录评分到的层次结构级别；缺省值为 -1。

## Netezza PCA

类型为 `netez zapcanode` 的节点的可用属性如下所示：

表 222: <i>netezzapcanode</i> 属性		
<b>netezzapcanode 属性</b>	值	属性说明
center_data	标志	如果为 true（缺省值），那么先执行数据集中（也称为“均值消去法”），然后再执行分析。
perform_data_scaling	标志	如果为 true，那么在分析前执行数据换算。这样做可以减低以不同单位测量不同变量时的分析任意性。
force_eigensolve	标志	如果为 true，则使用不太准确但较快的方法来查找主成份。
pc_number	整数	要将数据集精简到的主成份数；缺省值为 1。

## Netezza 回归树

类型为 *netezzaregtreenode* 的节点的可用属性如下所示：

表 223: <i>netezzaregtreenode</i> 属性		
<b>netezzaregtreenode 属性</b>	值	属性说明
max_tree_depth	整数	树在根节点下可以增长到的最大级别数；缺省值为 10。
split_evaluation_measure	Variance	类杂质测量，用于评估分割树的最佳位置，缺省值（当前唯一选项）是 Variance。
min_improvement_splits	成员	在树中进行新拆分前要将杂质减少到的最小数量。
min_instances_split	整数	可以拆分的最小记录数。
pruning_measure	mse r2 pearson spearman	要使用的修剪方法
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	缺省情况下，使用 allTrainingData 来估计模型精确度。使用 partitionTrainingData 来指定要使用训练数据的百分比，或 useOtherTable 来使用源自指定数据库表的训练数据集。
perc_training_data	成员	如果 prune_tree_options 设置为 PercTrainingData，则指定用于训练的数据所占的百分比。
prune_seed	整数	在 prune_tree_options 设置为 PercTrainingData 时，用于重复分析结果的随机种子，缺省值是 1。
pruning_table	string	这是用于估计模型精确度的单独修剪数据集的表名称。

表 223: netezzaregtreenode 属性 (继续)		
netezzaregtreenode 属性	值	属性说明
compute_probabilities	标志	如果为 true，则指定应该包括在输出中的指定类的方差。

## Netezza 线性回归

类型为 netezzalineressionnode 的节点的可用属性如下所示：

表 224: netezzalineressionnode 属性		
netezzalineressionnode 属性	值	属性说明
use_svd	标志	如果为 true，则使用“奇异值分解”矩阵代替原始矩阵，以便提高速度和数字准确性。
include_intercept	标志	如果为 true（缺省值），那么提高解的整体准确性。
calculate_model_diagnostics	标志	如果为 true，则对模型计算诊断信息。

## Netezza 时间序列

类型为 netezzatimeseriesnode 的节点的可用属性如下所示：

表 225: netezzatimeseriesnode 属性		
netezzatimeseriesnode 属性	值	属性说明
time_points	<i>field</i>	此输入字段包含时间序列的日期值或时间值。
time_series_ids	<i>field</i>	此输入字段包含时间序列标识；在输入包含多个时间序列时使用。
model_table	<i>field</i>	这是用于存储 Netezza 时间序列模型的数据库表。
description_table	<i>field</i>	这是包含时间序列名称和描述的输入表的名称。
seasonal_adjustment_table	<i>field</i>	这是一个输出表的名称，该表用于存储指数平滑或季节性趋势分解算法所计算的按季度调整值。
algorithm_name	SpectralAnalysis 或 spectral  ExponentialSmoothing 或 esmoothing  ARIMA  SeasonalTrendDecomposition 或 std	这是用于时间序列模型的算法。

表 225: <i>netezzatimeseriesnode</i> 属性 (继续)		
<b>netezzatimeseriesnode</b> 属性	值	属性说明
trend_name	N  A  DA  M  DM	指数平滑的趋势类型：  N - 无  A - 加性  DA - 衰减加性  M - 乘性  DM - 衰减乘性
seasonality_type	N  A  M	指数平滑的季节性类型：  N - 无  A - 加性  M - 乘性
interpolation_method	linear  cubicspline  exponentialspline	要使用的插值方法。
timerange_setting	SD  SP	用于设置要使用的时间范围：  SD - 由系统确定（使用时间序列数据的完整范围）  SP - 用户通过 <code>earliest_time</code> 和 <code>latest_time</code> 指定

表 225: <i>netezzatimeseriesnode</i> 属性 (继续)		
<b>netezzatimeseriesnode</b> 属性	值	属性说明
earliest_time	整数	开始值和结束值（如果 <code>timerange_setting</code> 为 SP）。  格式应遵循 <code>time_points</code> 值。  例如，如果 <code>time_points</code> 字段包含日期，那么此值也应该是日期。  示例：  <pre>set NZ_DT1.timerange_setting = 'SP'  set NZ_DT1.earliest_time = '1921-01-01'  set NZ_DT1.latest_time = '2121-01-01'</pre>
latest_time	<i>date</i>	
	时间	
	<i>timestamp</i>	

表 225: netezzatimeseriesnode 属性 (继续)

netezzatimeseriesnode 属性	值	属性说明
arima_setting	SD  SP	<p>用于设置 ARIMA 算法（仅当 algorithm_name 设置为 ARIMA 时才使用）：</p> <p>SD - 由系统确定</p> <p>SP - 由用户指定</p> <p>如果 arima_setting = SP, 请使用下列参数来设置季节性值和非季节性值。示例（仅非季节性）：</p> <pre>set NZ_DT1.algorithm_name = 'arima'</pre> <pre>set NZ_DT1.arima_setting = 'SP'</pre> <pre>set NZ_DT1.p_symbol = 'lesseq'</pre> <pre>set NZ_DT1.p = '4'</pre> <pre>set NZ_DT1.d_symbol = 'lesseq'</pre> <pre>set NZ_DT1.d = '2'</pre> <pre>set NZ_DT1.q_symbol = 'lesseq'</pre> <pre>set NZ_DT1.q = '4'</pre>
p_symbol	less  eq  lesseq	<p>ARIMA - 参数 p、d、q、sp、sd 和 sq 的运算符：</p> <p>less - 小于</p> <p>eq - 等于</p> <p>lesseq - 小于或等于</p>
d_symbol		
q_symbol		
sp_symbol		
sd_symbol		
sq_symbol		
p	整数	ARIMA - 自动关联的非季节性程度。
q	整数	ARIMA - 非季节性派生值。
d	整数	ARIMA - 模型中的移动平均值移动平均值阶的非季节性数目。

表 225: *netezzatimeseriesnode* 属性 (继续)

netezzatimeseriesnode 属性	值	属性说明
sp	整数	ARIMA - 自动关联的季节性程度。
sq	整数	ARIMA - 季节性派生值。
sd	整数	ARIMA - 模型中的移动平均值移动平均值阶的季节性数目。
advanced_setting	SD  SP	<p>确定如何处理高级设置：</p> <p>SD - 由系统确定</p> <p>SP - 由用户通过 period、units_period 和 forecast_setting 指定。</p> <p>示例：</p> <pre>set NZ_DT1.advanced_setting = 'SP'</pre> <pre>set NZ_DT1.period = 5</pre> <pre>set NZ_DT1.units_period = 'd'</pre>
period	整数	季节周期的长度，与 units_period 一起指定。不适用于谱分析。
units_period	ms  s  min  h  d  wk  q  y	<p>period 的表示单位：</p> <p>ms - 毫秒</p> <p>s - 秒</p> <p>min - 分钟</p> <p>h - 小时</p> <p>d - 日</p> <p>wk - 星期</p> <p>q - 季度</p> <p>y - 年</p> <p>例如，对于每周时间序列，请对 period 使用 1，并对 units_period 使用 wk。</p>



表 225: <i>netezzatimeseriesnode</i> 属性 (继续)		
netezzatimeseriesnode 属性	值	属性说明
forecast_setting	forecasthorizon	指定如何进行预测。
	forecasttimes	
forecast_horizon	整数	如果 forecast_setting = forecasthorizon, 那么指定预测结束点值。  格式应遵循 time_points 值。  例如, 如果 time_points 字段包含日期, 那么此值也应该是日期。
	date	
	时间	
	timestamp	
forecast_times	整数	如果 forecast_setting = forecasttimes, 那么指定用于进行预测的值。  格式应遵循 time_points 值。  例如, 如果 time_points 字段包含日期, 那么此值也应该是日期。
	date	
	时间	
	timestamp	
include_history	标志	指示是否将历史值包括在输出中。
include_interpolated_values	标志	指示是否将内插值包括在输出中。如果 include_history 为 false, 则不适用。

## Netezza 广义线性

类型为 netezzaglmnode 的节点的可用属性如下所示：

表 226: <i>netezza GLM</i> 属性		
netezzaglmnode 属性	值	属性说明
dist_family	bernoulli	分布类型；缺省值为 bernoulli。
	gaussian	
	poisson	
	negativebinomial	
	wald	
	gamma	
dist_params	成员	要使用的分布参数值。仅当 distribution 为 Negativebinomial 时才适用。

表 226: netezza GLM 属性 (继续)		
netezzaglmnode 属性	值	属性说明
trials	整数	仅当 distribution 为 Binomial 时才适用。当目标响应为发生在一组试验中的事件数时, target 字段包含事件数, trials 字段包含试验数。
model_table	field	这是用于存储 Netezza 广义线性模型的数据库表。
maxit	整数	算法应执行的最大迭代次数;缺省值为 20。
eps	成员	指定最大误差值(以科学记数法表示), 达到此值后, 算法应停止查找最佳匹配模型。缺省值为 -3, 这表示 1E-3, 即 0.001。
tol	成员	设置数值(用科学表示法), 低于此值的所有误差均被视为 0 值。缺省值为 -7, 表示误差值若低于 1E-7 (或 0.0000001), 则被视为不显著。

表 226: netezza GLM 属性 (继续)		
netezzaglmnode 属性	值	属性说明
link_func	identity  inverse  invnegative  invsquare  sqrt  power  oddspower  log  clog  loglog  cloglog  logit  probit  gaussit  cauchit  canbinom  cangeom  cannegbinom	要使用的联接函数；缺省值为 logit。
link_params	成员	要使用的关联函数参数值。仅当 link_function 为 power 或 oddspower 时才适用。

表 226: netezza GLM 属性 (继续)		
netezzaglmnode 属性	值	属性说明
interaction	[[[colnames1],[levels1]], [[colnames2],[levels2]], ..., [[colnamesN],[levelsN]],]	指定字段之间的交互。colnames 是输入字段的列表，而 level 对于每个字段始终为 0。  示例：  <pre>[[["K", "BP", "Sex", "K"], [0, 0, 0, 0]], [["Age", "Na"], [0, 0]]]</pre>
intercept	标志	如果为 true，则在模型中包括截距。

## Netezza 模型块属性

Netezza 数据库模型块的公共属性如下所示。

表 227: 公共 Netezza 模型块属性		
通用 Netezza 模型块属性	值	属性说明
connection	string	这是用于存储模型的 Netezza 数据库的连接字符串。
table_name	string	这是用于存储模型的数据库表的名称。

其他模型块属性与相应建模节点的属性相同。

模型块的脚本名称如下所示。

表 228: Netezza 模型块的脚本名称	
模型块	脚本名称
决策树	applynetezadectreenode
K-Means	applynetezzakmeansnode
贝叶斯网络	applynetez zabayesnode
朴素贝叶斯	applynetez zanaivebayesnode
KNN	applynetez zaknnnode
分裂式聚类	applynetez zadivclusternode
PCA	applynetez zapcanode
回归树	applynetez zaregtreenode
线性回归	applynetez zalineregressionnode
时间序列	applynetez zatimeseriesnode
广义线性	applynetez zaglmnode

# 第 16 章 输出节点属性

输出节点的属性与其他“类型”节点的属性略有不同。输出节点属性不是指特定的节点选项，而是存储对输出对象的引用。这在从表中提取值并将其设置为流参数时非常有用。

本节说明输出节点的可用的脚本编制属性。

## analysisnode 属性



“分析”节点评估预测模型生成准确预测的能力。“分析”节点执行一个或多个模型块的预测值和实际值之间的各种比较。它们还可以将预测模型进行相互比较。

示例

```
node = stream.create("analysis", "My node")
# "Analysis" tab
node.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# "Define User Measure..."
node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# "Output" tab
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

表 229: analysisnode 属性		
analysisnode 个属性	数据类型	属性描述
output_mode	Screen  File	用于指定从输出节点中生成的输出的目标位置。
use_output_name	标志	指定是否使用自定义的输出名。
output_name	string	如果 use_output_name 为真，则指定使用的名称。
output_format	Text (.txt)  HTML (.html)  Output (.cou)	用于指定输出类型。
by_fields	列表	
full_filename	string	如果是磁盘、数据或 HTML 输出，则此属性指输出文件的名称。

表 229: *analysisnode* 属性 (继续)

<b>analysisnode</b> 个属性	数据类型	属性描述
coincidence	标志	
performance	标志	
evaluation_binary	标志	
confidence	标志	
threshold	成员	
improve_accuracy	成员	
field_detection_method	Metadata Name	确定预测字段与原始目标字段匹配的方式。请指定 Metadata 或 Name。
inc_user_measure	标志	
user_if	<i>expr</i>	
user_then	<i>expr</i>	
user_else	<i>expr</i>	
user_compute	[Mean Sum Min Max SDev]	

## dataauditnode 属性



“数据审核”节点提供有关数据的全面概览，包括每个字段的汇总统计、直方图和分布以及有关离群值、缺失值和极值的信息。结果显示在易于读取的矩阵中，可进行排序并且可用于生成完整大小的图形和数据准备节点。

示例

```

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")

```

表 230: *dataauditnode* 属性

<b>dataauditnode</b> 个属性	数据类型	属性描述
custom_fields	标志	
fields	[ <i>field1</i> ... <i>fieldN</i> ]	
overlay	<i>field</i>	

表 230: dataauditnode 属性 (继续)		
dataauditnode 个属性	数据类型	属性描述
display_graphs	标志	用于打开或关闭输出矩阵中图形的显示。
basic_stats	标志	
advanced_stats	标志	
median_stats	标志	
calculate	Count Breakdown	用于计算缺失值。选择两种计算方法中的一种、两种，或均不选择。
outlier_detection_method	std iqr	用于指定离群值和极值的检测方法。
outlier_detection_std_outlier	成员	如果 outlier_detection_method 是 std, 那么指定用于定义离群值的数值。
outlier_detection_std_extreme	成员	如果 outlier_detection_method 是 std, 那么指定用于定义极值的数值。
outlier_detection_iqr_outlier	成员	如果 outlier_detection_method 是 iqr, 那么指定用于定义离群值的数值。
outlier_detection_iqr_extreme	成员	如果 outlier_detection_method 是 iqr, 那么指定用于定义极值的数值。
use_output_name	标志	指定是否使用自定义的输出名。
output_name	string	如果 use_output_name 为真, 则指定使用的名称。
output_mode	Screen File	用于指定从输出节点中生成的输出的目标位置。
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	用于指定输出类型。
paginate_output	标志	当 output_format 是 HTML 时, 使输出分页。

表 230: dataauditnode 属性 (继续)

dataauditnode 个属性	数据类型	属性描述
lines_per_page	成员	与 paginate_output 一起使用时，指定每个输出页中的行数。
full_filename	string	

## extensionoutputnode 属性



通过使用您自己的定制 R 或 Python for Spark 脚本，可以使用“扩展输出”节点来分析模型评分的数据和结果。分析的输出可以是文本，也可以是图形。输出将添加到管理器窗格的输出选项卡中；另外，可以将输出重定向到文件。

### Python for Spark 示例

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

python_script = """
import json
import spss.pyspark.runtime

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]
print df
"""

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

### R 示例

```
##### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

表 231: extensionoutputnode 属性

extensionoutputnode 个属性	数据类型	属性描述
syntax_type	R  Python	指定运行哪个脚本 - R 还是 Python（R 是缺省值）。
r_syntax	string	这是用于进行模型评分的 R 脚本语法。
python_syntax	string	用于模型评分的 Python 脚本编写语法。
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。



表 231: <i>extensionoutputnode</i> 属性 (继续)		
<b>extensionoutputnode</b> 个属性	数据类型	属性描述
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。
output_to	Screen File	指定输出类型 (Screen 或 File) 。
output_type	Graph Text	指定是产生图形还是文本输出。
full_filename	<i>string</i>	用于所生成输出的文件名。
graph_file_type	HTML COU	输出文件的文件类型 (.html 或 .cou) 。
text_file_type	HTML TEXT COU	指定文本输出的文件类型 (.html、.txt 或 .cou) 。

## kdeexport 属性



Kernel Density Estimation (KDE)<sup>®</sup> 使用 Ball Tree 或 KD Tree 算法以进行效率查询，并结合无监督学习、特征工程和数据建模等概念。基于相邻元素的方法（例如，KDE）是最流行且最有用的一些密度估算方法。SPSS 建模器 中的 KDE 建模和 KDE 模拟节点公开 KDE 库的核心特征和常用参数。节点使用 Python 进行实现。

表 232: <i>kdeexport</i> 属性		
<b>kdeexport</b> 个属性	数据类型	属性描述
bandwidth	<i>double</i>	缺省值为 1。
kernel	<i>string</i>	要使用的内核：gaussian 或 tophat。缺省值为 gaussian。
algorithm	<i>string</i>	要使用的树算法：kd_tree、ball_tree 或 auto。缺省值为 auto。

表 232: *kdeexport* 属性 (继续)

<b>kdeexport</b> 个属性	数据类型	属性描述
metric	<i>string</i>	在计算距离时要使用的度量。对于 <i>kd_tree</i> 算法, 从以下项进行选择: Euclidean、Chebyshev、Cityblock、Minkowski、Manhattan、Infinity、P、L2 或 L1。对于 <i>ball_tree</i> 算法, 从以下项进行选择: Euclidian、Braycurtis、Chebyshev、Canberra、Cityblock、Dice、Hamming、Infinity、Jaccard、L1、L2、Minkowski、Matching、Manhattan、P、Rogersanimoto、Russellrao、Sokalmichener、Sokalsneath 或 Kulsinski。缺省值为 Euclidean。
atol	<i>float</i>	期望的结果的绝对容差。较大的容差通常将导致更快执行。缺省值为 0.0。
rtol	<i>float</i>	期望的结果的相对容差。较大的容差通常将导致更快执行。缺省值为 1E-8。
breadthFirst	<i>BOOLEAN</i>	设置为 True 以使用广度优先方法。设置为 False 以使用深度优先方法。缺省值为 True。
LeafSize	整数	底层树的叶大小。缺省值为 40。更改此值可能显著影响性能。
pValue	<i>double</i>	指定将 Minkowski 用于度量时要使用的“P 值”。缺省值为 1.5。

## matrixnode 属性



“矩阵”节点创建一个表, 用于显示字段之间的关系。最常用于显示两个符号字段之间的关系, 但也可以显示标志字段或数字字段之间的关系。

示例

```
node = stream.create("matrix", "My node")
# "Settings" tab
node.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
# "Appearance" tab
node.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# "Output" tab
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
```

```
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

表 233: *matrixnode* 属性

matrixnode 个属性	数据类型	属性描述
fields	Selected  Flags  Numerics	
row	field	
column	field	
include_missing_values	标志	指定在行和列输出中是否包含用户缺失值（空白）和系统缺失值（空值）。
cell_contents	CrossTabs  Function	
function_field	string	
function	Sum  Mean  Min  Max  SDev	
sort_mode	Unsorted  Ascending  Descending	
highlight_top	成员	如果非零，则为真。
highlight_bottom	成员	如果非零，则为真。

表 233: *matrixnode* 属性 (继续)

<b>matrixnode</b> 个属性	数据类型	属性描述
display	[Counts Expected Residuals RowPct ColumnPct TotalPct]	
include_totals	标志	
use_output_name	标志	指定是否使用自定义的输出名。
output_name	<i>string</i>	如果 <code>use_output_name</code> 为真，则指定使用的名称。
output_mode	Screen File	用于指定从输出节点中生成的输出的目标位置。
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	用于指定输出类型。Formatted 和 Delimited 格式都可使用修饰符 <code>transposed</code> ，此符号可转置表中的行和列。
paginate_output	标志	当 <code>output_format</code> 是 HTML 时，使输出分页。
lines_per_page	成员	与 <code>paginate_output</code> 一起使用时，指定每个输出页中的行数。
full_filename	<i>string</i>	

## meansnode 属性



“平均值”节点在独立组之间或相关字段对之间进行平均值比较，以检验是否存在显著差别。例如，您可以比较开展促销前后的平均收入，或者将来自未接受促销客户的收入与接受促销客户的收入进行比较。

示例

```
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")
```

表 234: *meansnode* 属性

meansnode 个属性	数据类型	属性描述
means_mode	BetweenGroups BetweenFields	指定要在数据上执行的均值统计量的类型。
test_fields	[field1 ... fieldn]	指定当 means_mode 设置为 BetweenGroups 时的检验字段。
grouping_field	field	指定分组字段。
paired_fields	[[field1 field2] [field3 field4] ...]	指定当 means_mode 设置为 BetweenFields 时要使用的字段对。
label_correlations	标志	指定在输出中是否显示相关标签。仅在当 means_mode 设置为 BetweenFields 时才应用此设置。
correlation_mode	Probability Absolute	指定用概率还是绝对值标注相关。
weak_label	string	
medium_label	string	
strong_label	string	
weak_below_probability	成员	当 correlation_mode 设置为 Probability 时, 指定弱相关的分界值。这必须是 0 到 1 之间的一个值, 例如 0.90。
strong_above_probability	成员	强相关的分界值。
weak_below_absolute	成员	当 correlation_mode 设置为 Absolute 时, 指定弱相关的分界值。这必须是 0 到 1 之间的一个值, 例如 0.90。
strong_above_absolute	成员	强相关的分界值。
unimportant_label	string	
marginal_label	string	
important_label	string	
unimportant_below	成员	低字段重要性的分界值。这必须是 0 到 1 之间的一个值, 例如 0.90。
important_above	成员	

表 234: *meansnode* 属性 (继续)

<b>meansnode</b> 个属性	数据类型	属性描述
use_output_name	标志	指定是否使用自定义的输出名。
output_name	<i>string</i>	使用的名称。
output_mode	Screen File	指定从输出节点中生成的输出的目标位置。
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	指定输出类型。
full_filename	<i>string</i>	
output_view	Simple Advanced	指定在输出中显示简单视图还是高级视图。

## reportnode 属性



“报告”节点可创建格式化报告，其中包含固定文本、数据及得自数据的其他表达式。您可以使用文本模板指定报告的格式，以定义固定文本和数据输出构造。通过在模板中使用 HTML 标记以及在“输出”选项卡上设置选项，可以提供定制文本格式。您可以通过在模板中使用 CLEM 表达式来包含数据值和其他条件输出。

### 示例

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

表 235: *reportnode* 属性

<b>reportnode</b> 个属性	数据类型	属性描述
output_mode	Screen File	用于指定从输出节点中生成的输出的目标位置。
output_format	HTML (.html) Text (.txt) Output (.cou)	用于指定文件输出的类型。

表 235: reportnode 属性 (继续)

reportnode 个属性	数据类型	属性描述
format	Auto Custom	用于选择自动设置输出格式还是使用模板中包含的 HTML 设置输出格式。要使用模板中的 HTML 格式设置, 请指定 Custom。
use_output_name	标志	指定是否使用自定义的输出名。
output_name	string	如果 use_output_name 为真, 则指定使用的名称。
text	string	
full_filename	string	
highlights	标志	
title	string	
lines_per_page	成员	

## rouputnode 属性



通过使用您自己的定制 R 脚本, 可以使用“R 输出”节点来分析模型评分的数据和结果。 分析的输出可以是文本, 也可以是图形。 输出将添加到管理器窗格的输出选项卡中; 另外, 可以将输出重定向到文件。

表 236: rouputnode 属性

rouputnode 个属性	数据类型	属性描述
syntax	string	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	标志	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	标志	
output_name	Auto Custom	
custom_name	string	
output_to	Screen File	
output_type	Graph Text	
full_filename	string	

表 236: routputnode 属性 (继续)

routputnode 个属性	数据类型	属性描述
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

## setglobalsnode 属性



设置全局量节点扫描数据并计算可在 CLEM 表达式中使用的汇总值。例如，您可以使用此节点来计算名为年龄的字段统计信息，然后通过插入函数 @GLOBAL\_MEAN(age)，在 CLEM 表达式中使用年龄的总体平均值。

示例

```
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)
```

表 237: setglobalsnode 属性

setglobalsnode 个属性	数据类型	属性描述
globals	[Sum Mean Min Max SDev]	结构属性，在其中，必须使用下面的语法引用要设置的字段：  node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	标志	
show_preview	标志	

## simevalnode 属性



“模拟评估”节点对指定的预测目标字段进行评估，并显示有关该目标字段的分布和相关性信息。

表 238: simevalnode 属性

simevalnode 个属性	数据类型	属性描述
target	field	
iteration	field	



表 238: <i>simevalnode</i> 属性 (继续)		
<b>simevalnode</b> 个属性	数据类型	属性描述
presorted_by_iteration	BOOLEAN	
max_iterations	成员	
tornado_fields	[field1...fieldN]	
plot_pdf	BOOLEAN	
plot_cdf	BOOLEAN	
show_ref_mean	BOOLEAN	
show_ref_median	BOOLEAN	
show_ref_sigma	BOOLEAN	
num_ref_sigma	成员	
show_ref_pct	BOOLEAN	
ref_pct_bottom	成员	
ref_pct_top	成员	
show_ref_custom	BOOLEAN	
ref_custom_values	[number1...numberN]	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	BOOLEAN	
pct_table	Quartiles Intervals Custom	
pct_intervals_num	成员	
pct_custom_values	[number1...numberN]	

## simfitnode 属性



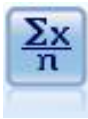
“模拟拟合”节点检查每个字段中数据的统计分布，并生成（或更新）“模拟生成”节点，同时将最佳拟合分布分配给每个字段。然后，可以使用“模拟生成”节点来生成模拟数据。

表 239: <i>simfitnode</i> 属性		
<b>simfitnode</b> 个属性	数据类型	属性描述
build	Node XMLExport Both	

表 239: *simfitnode* 属性 (继续)

<b>simfitnode</b> 个属性	数据类型	属性描述
use_source_node_name	<i>BOOLEAN</i>	
source_node_name	<i>string</i>	正在生成或更新的源节点的定制名称。
use_cases	All LimitFirstN	
use_case_limit	整数	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	整数	
parameter_xml_filename	<i>string</i>	
generate_parameter_import	<i>BOOLEAN</i>	

## statisticsnode 属性



“统计”节点提供有关数字字段的基本汇总信息。它计算各个字段的汇总统计以及字段间的相关性。

示例

```
node = stream.create("statistics", "My node")
# "Settings" tab
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
node.setPropertyValue("strong_label", "upper quartile")
# "Output" tab
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

表 240: *statisticsnode* 属性

<b>statisticsnode</b> 个属性	数据类型	属性描述
use_output_name	标志	指定是否使用自定义的输出名。
output_name	<i>string</i>	如果 use_output_name 为真，则指定使用的名称。
output_mode	Screen  File	用于指定从输出节点中生成的输出的目标位置。

表 240: *statisticsnode* 属性 (继续)

<b>statisticsnode</b> 个属性	数据类型	属性描述
output_format	Text (.txt)  HTML (.html)  Output (.cou)	用于指定输出类型。
full_filename	string	
examine	列表	
correlate	列表	
statistics	[count mean sum min max range variance sdev semean median mode]	
correlation_mode	Probability  Absolute	指定用概率还是绝对值标注相关。
label_correlations	标志	
weak_label	string	
medium_label	string	
strong_label	string	
weak_below_probability	成员	当 correlation_mode 设置为 Probability 时, 指定弱相关的分界值。这必须是 0 到 1 之间的一个值, 例如 0.90。
strong_above_probability	成员	强相关的分界值。
weak_below_absolute	成员	当 correlation_mode 设置为 Absolute 时, 指定弱相关的分界值。这必须是 0 到 1 之间的一个值, 例如 0.90。
strong_above_absolute	成员	强相关的分界值。

## statisticsoutputnode 属性



Statistics 输出节点可调用 IBM SPSS Statistics 过程以分析您的 IBM SPSSModeler 数据。可以访问许多不同的 IBM SPSS Statistics 分析过程。此节点需要 IBM SPSS Statistics 的许可副本。

有关此节点属性的信息, 请参阅第 378 页的『[statisticsoutputnode 属性](#)』。

# tablenode 属性



“表”节点以表格式显示数据，这些数据还可以写入到文件中。每当您需要检查数据值或者采用可轻松阅读的格式导出这些数据值时，此节点非常有用。

示例

```
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)
```

表 241: tablenode 属性

tablenode 个属性	数据类型	属性描述
full_filename	string	如果是磁盘、数据或 HTML 输出，则此属性指输出文件的名称。
use_output_name	标志	指定是否使用自定义的输出名。
output_name	string	如果 use_output_name 为真，则指定使用的名称。
output_mode	Screen File	用于指定从输出节点中生成的输出的目标位置。
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	用于指定输出类型。
transpose_data	标志	导出前转置数据，使行表示字段，列表示记录。
paginate_output	标志	当 output_format 是 HTML 时，使输出分页。
lines_per_page	成员	与 paginate_output 一起使用时，指定每个输出页中的行数。
highlight_expr	string	
output	string	只读属性，可保留对由节点构建的最后一个表的引用。
value_labels	[[Value LabelString] [Value LabelString] ...]	用于为值对指定标签。

表 241: <i>tablenode</i> 属性 (继续)		
tablenode 个属性	数据类型	属性描述
display_places	整数	为字段设置显示的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时将使用流缺省值。
export_places	整数	为字段设置导出的小数位数（仅用于以 REAL 存储的字段）。值为 -1 时将使用流缺省值。
decimal_separator	DEFAULT  PERIOD  COMMA	为字段设置十进制分隔符（仅用于以 REAL 存储的字段）。
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	为字段设置日期格式（仅用于以 DATE 或 TIMESTAMP 存储的字段）。

表 241: <i>tablenode</i> 属性 (继续)		
tablenode 个属性	数据类型	属性描述
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	为字段设置时间格式（仅用于以 TIME 或 TIMESTAMP 存储的字段）。
column_width	整数	为字段设置列宽度。值为 -1 表示将列宽度设置为 Auto。
justify	AUTO CENTER LEFT RIGHT	为字段设置列对齐格式。

## transformnode 属性



“变换”节点允许您先选择并直观预览变换的结果，然后再将其应用于所选字段。

示例

```
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)
```

表 242: *transformnode* 属性

transformnode 个属性	数据类型	属性描述
fields	[ <i>field1</i> ... <i>fieldn</i> ]	要在变换中使用的字段。
formula	All  Select	表示应计算所有变换还是选定的变换。
formula_inverse	标志	表示是否应使用逆变换。
formula_inverse_offset	数字	表示公式中要使用的数据偏移量。除非用户指定，否则默认情况下设置为 0。
formula_log_n	标志	表示是否应使用 $\log_n$ 变换。
formula_log_n_offset	数字	
formula_log_10	标志	表示是否应使用 $\log_{10}$ 变换。
formula_log_10_offset	数字	
formula_exponential	标志	表示是否应使用指数 ( $e^x$ ) 变换。
formula_square_root	标志	表示是否应使用平方根变换。
use_output_name	标志	指定是否使用自定义的输出名。
output_name	<i>string</i>	如果 <i>use_output_name</i> 为真，则指定使用的名称。
output_mode	Screen  File	用于指定从输出节点中生成的输出的目标位置。
output_format	HTML (.html)  Output (.cou)	用于指定输出类型。
paginate_output	标志	当 <i>output_format</i> 是 HTML 时，使输出分页。
lines_per_page	成员	与 <i>paginate_output</i> 一起使用时，指定每个输出页中的行数。
full_filename	<i>string</i>	表示要在文件输出中使用的文件名。





# 第 17 章 导出节点属性

## 通用导出节点属性

以下属性通用于所有导出节点。

表 243: 公共导出节点属性		
属性	值	属性描述
publish_path	string	输入供发布的图像和参数文件使用的 rootname 名称。
publish_metadata	标志	指定是否生成介绍图像的输入和输出以及它们的数据模型的元数据文件。
publish_use_parameters	标志	指定是否在 *.par 文件中包含流参数。
publish_parameters	字符串列表	指定要包括的参数。
execute_mode	export_data	指定是执行节点而不发布流，还是在执行节点时自动发布流。
	publish	

## asexport 属性

您可以使用 分析服务器 导出在 Hadoop 分布式文件系统 (HDFS) 上运行流。

### 示例

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

表 244: asexport 属性		
asexport 个属性	数据类型	属性描述
data_source	string	数据源名称。
export_mode	string	指定是将导出的数据 <b>append</b> 到现有数据源，还是 <b>overwrite</b> 现有数据源。
use_default_as	BOOLEAN	如果设置为 True，那么将使用服务器 options.cfg 文件中配置的缺省 分析服务器 连接。如果设置为 False，请使用此节点的连接。

表 244: *asexport* 属性 (继续)

asexport 个属性	数据类型	属性描述
connection	["string", "string", "string", "string", "string", "string", "string", "string", "string", "string"]	这是包含 分析服务器 连接详细信息的列表属性。格式为：["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"], 其中： is_secure_connect: 指示是否使用安全连接，并且是 true 还是 false。 use-kerberos-auth: 指示 是否使用 Kerberos 认证，其值为 true 或 false。 enable-kerberos-debug: 指示是否使用 Kerberos 认证的调试模式，其值为 true 或 false。

## cognosexportnode 属性



IBM Cognos 导出节点以 Cognos 数据库可以读取的格式导出数据。

对于此节点，必须定义 Cognos 连接和 ODBC 连接。

### Cognos 连接

Cognos 连接的属性如下。

表 245: *cognosexportnode* 属性

cognosexportnode 个属性	数据类型	属性描述
cognos_connection	["string","flag","string","string","string"]	<p>列表属性，其中包含 Cognos 服务器的连接详细信息。格式为：["Cognos_server_URL", login_mode, "namespace", "username", "password"]</p> <p>其中：</p> <p>Cognos_server_URL 是包含源的 Cognos 服务器的 URL。</p> <p>login_mode 指示是否使用匿名登录，并且为 true 或 false；如果设置为 true，那么应将以下字段设置为 ""。</p> <p>namespace 指定用于登录服务器的安全认证提供程序。</p> <p>username 和 password 是用于登录 Cognos 服务器的用户名和密码。</p> <p>您还可以使用以下方式替代 login_mode：</p> <ul style="list-style-type: none"> <li>• anonymousMode。例如： ['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]</li> <li>• credentialMode。例如： ['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]</li> </ul>
		<ul style="list-style-type: none"> <li>• storedCredentialMode。例如：['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]</li> </ul> <p>其中， stored_credential_name 是存储库中 Cognos 凭证的名称。</p>

表 245: cognosexportnode 属性 (继续)

cognosexportnode 个属性	数据类型	属性描述
cognos_package_name	string	数据所要导出到的 Cognos 数据包的路径和名称，例如：  /Public Folders/MyPackage
cognos_datasource	string	
cognos_export_mode	Publish  ExportFile	
cognos_filename	string	

## ODBC 连接

ODBC 连接属性与下一节中列出的 databaseexportnode 属性相同，只不过 datasource 属性无效。

## databaseexportnode 属性



“数据库导出”节点将数据写入符合 ODBC 标准的关系数据源。为了写入 ODBC 数据源，该数据源必须存在，并且您必须对其具有写入许可权。

示例

```
'''
Assumes a datasource named "MyDatasource" has been configured
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByName("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tab
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialog
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialog
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP
INDEX <index-name>
ON <table-name> (<index-columns>")
```

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id", "region"]])
```

表 246: databaseexportnode 属性

databaseexportnode 个属性	数据类型	属性描述
datasource	string	
username	string	
password	string	
epassword	string	在执行过程中，此槽为只读。要生成加密密码，请使用“工具”菜单中的“密码工具”。有关更多信息，请参阅主题 第 45 页的『生成经过编码的密码』。
table_name	string	
write_mode	Create Append Merge	
map	string	用于将流字段名称映射到数据库列名称（仅在 write_mode 为 Merge 的情况下才有效）。  对于合并，所有字段必须经过映射才能导出。数据库中不存在的字段名称将添加为新列。
key_fields	列表	指定用于键的流字段；map 属性显示此项在数据库中的对应内容。
join	Database Add	
drop_existing_table	标志	
delete_existing_rows	标志	
default_string_size	整数	
type		用于设置模式类型的结构属性。
generate_import	标志	
use_custom_create_table_command	标志	使用 custom_create_table 通道修改标准 CREATE TABLE SQL 命令。
custom_create_table_command	string	指定字符串命令代替标准 CREATE TABLE SQL 命令使用。
use_batch	标志	下列属性是数据库批量加载的高级选项。Use_batch 为 true 时将关闭向数据库逐行提交的功能。

表 246: *databaseexportnode* 属性 (继续)

<b>databaseexportnode</b> 个属性	数据类型	属性描述
batch_size	成员	指定在提交到内存前发送到数据库的记录数。
bulk_loading	Off  ODBC  External	指定批量加载的类型。下面列出了 ODBC 和 External 的其他选项。
not_logged	标志	
odbc_binding	Row  Column	指定通过 ODBC 批量加载时使用逐行绑定或逐列绑定。
loader_delimit_mode	Tab  Space  Other	对于通过外部程序的批量加载，指定定界符的类型。选择 Other 连同  loader_other_delimiter  属性以指定定界符，例如逗号 (,)。
loader_other_delimiter	string	
specify_data_file	标志	标志为 True 时可激活下面的 data_file 属性，在该属性中可以指定批量装入到数据库时写入的文件名和路径。
data_file	string	
specify_loader_program	标志	标志为 True 时可激活下面的 loader_program 属性，在该属性中可以指定外部装入程序脚本或程序的名称和位置。
loader_program	string	
gen_logfile	标志	标志为 True 时可激活下面的 logfile_name，在该属性中可以指定服务器上的文件的名称以生成错误日志。
logfile_name	string	
check_table_size	标志	标志为 True 时允许进行表检查以确保数据库表大小的增加与从 IBM SPSSModeler 导出的行数相符。
loader_options	string	指定加载程序的其他参数，例如 -comment 和 -specialdir。
export_db_primarykey	标志	指定给定字段是否是主关键字。
use_custom_create_index_command	标志	如果标志为 true，则为所有索引启用自定义 SQL。

表 246: <i>databaseexportnode</i> 属性 (继续)		
<b>databaseexportnode</b> 个属性	数据类型	属性描述
custom_create_index_command	<i>string</i>	当已启用自定义 SQL 时, 指定用于创建索引的 SQL 命令。(创建特定索引时, 该值可被覆盖, 如下所示。)
indexes.INDEXNAME.fields		必要时创建指定的索引并列出了将要包含在该索引中的字段名。
INDEXNAME "use_custom_create_index_command"	标志	用于启用或禁用特定索引的定制 SQL。请参阅下表之后的示例。
INDEXNAME "custom_create_index_command"	<i>string</i>	指定用于指定索引的自定义 SQL。请参阅下表之后的示例。
indexes.INDEXNAME.remove	标志	如果为 <b>True</b> , 那么将从索引集中移除指定的索引。
table_space	<i>string</i>	指定将创建表空间。
use_partition	标志	指定将创建分布散列字段。
partition_field	<i>string</i>	指定分布散列字段的内容。

注: 对于某些数据库, 您可以指定针对使用压缩的导出创建数据库表 (例如, SQL 中 CREATE TABLE MYTABLE (...) COMPRESS YES; 的等效项)。为了支持此功能, 提供了属性 *use\_compression* 和 *compression\_mode*, 如下所示。

表 247: 使用了压缩功能的 <i>databaseexportnode</i> 属性		
<b>databaseexportnode</b> 个属性	数据类型	属性描述
use_compression	<i>BOOLEAN</i>	如果设置为 <b>True</b> , 那么将以导出时进行压缩的方式创建表。

表 247: 使用了压缩功能的 *databaseexportnode* 属性 (继续)

databaseexportnode 个属性	数据类型	属性描述
compression_mode	Row	设置 SQL Server 数据库的压缩级别。
	Page	
	Default	设置 Oracle 数据库的压缩级别。 请注意, 值 OLTP、Query_High、Query_LowArchive_High 和 Archive_Low 至少需要 Oracle 11gR2。
	Direct_Load_Operations	
	All_Operations	
	Basic	
	OLTP	
	Query_High	
	Query_Low	
	Archive_High	
	Archive_Low	

显示如何针对特定索引更改 CREATE INDEX 命令的示例：

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["use_custom_create_index_command",
True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["custom_create_index_command",
"CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

或者, 也可以通过散列表完成此操作：

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", [{"fields":["id",
"region"],
"use_custom_create_index_command":True,
"custom_create_index_command":"CREATE INDEX <index-name> ON
<table-name> <(index-columns)>"}])
```

## datacollectionexportnode 属性



数据收集 导出节点以 数据收集 市场调查软件使用的格式输出数据。要使用此节点, 必须安装 数据收集 Data Library。

示例

```
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)
```

表 248: datacollectionexportnode 属性		
datacollectionexportnode 个属性	数据类型	属性描述
metadata_file	string	要导出的元数据文件的名称。
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	标志	指定导出的 .mdd 文件是否应包括 数据收集 系统变量。
casedata_file	string	要导出观测数据的 .sav 文件的名称。
generate_import	标志	

## excelexportnode 属性



Excel 导出节点在 Microsoft Excel 中输出数据。xlsx 文件格式。 您还可选择在执行完此节点后自动启动 Excel 并打开导出的文件。

示例

```
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)
```

表 249: excelexportnode 属性		
excelexportnode 个属性	数据类型	属性描述
full_filename	string	
excel_file_type	Excel2007	
export_mode	Create Append	
inc_field_names	标志	指定字段名是否可以包含在工作表的第一行中。
start_cell	string	指定导出的开始单元格。
worksheet_name	string	要写入的工作表的名称。

表 249: excelexportnode 属性 (继续)		
excelexportnode 个属性	数据类型	属性描述
launch_application	标志	指定是否应在结果文件上调用 Excel。请注意，必须在“帮助应用程序”对话框（“工具”菜单 > “帮助应用程序”）中指定启动 Excel 的路径。
generate_import	标志	指定是否应生成将读取已导出数据文件的 Excel 导入节点。

## extensionexportnode 属性



通过“扩展导出”节点，您可以运行 R 或 Python for Spark 脚本来导出数据。

### Python for Spark 示例

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

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

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
print df.dtypes[:]
_newDF = df.select("Age", "Drug")
print _newDF.dtypes[:]

df.select("Age", "Drug").write.save("c:/data/ageAndDrug.json", format="json")
"""

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

### R 示例

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

表 250: extensionexportnode 属性		
extensionexportnode 个属性	数据类型	属性描述
syntax_type	R Python	指定运行哪个脚本 - R 还是 Python（R 是缺省值）。
r_syntax	string	要运行的 R 脚本编写语法。
python_syntax	string	要运行的 Python 脚本编写语法。

表 250: <i>extensionexportnode</i> 属性 (继续)		
<b>extensionexportnode</b> 个属性	数据类型	属性描述
convert_flags	StringsAndDoubles LogicalValues	此选项用于转换标志字段。
convert_missing	标志	用于将缺失值转换为 R NA 值的选项。
convert_datetime	标志	此选项用于将具有日期或日期时间格式的变量转换为 R 日期/时间格式。
convert_datetime_class	POSIXct POSIXlt	这些选项用于指定要将日期或日期时间格式的变量转换为什么格式。

## jsonexportnode 属性



JSON 导出节点以 JSON 格式输出数据。

表 251: <i>jsonexportnode</i> 属性		
<b>jsonexportnode</b> 个属性	数据类型	属性描述
full_filename	string	完整文件名（包括路径）。
string_format	records 值	指定 JSON 字符串格式。缺省值为 records。
generate_import	标志	指定是否应生成将读取已导出数据文件的“JSON 导入”节点。缺省值为 False。

## outputfilenode 属性



“平面文件导出”节点将数据输出到定界文本文件中。它对于导出可由其他分析或电子表格软件读取的数据非常有用。

示例

```
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)
```

表 252: *outputfilenode* 属性

outputfilenode 个属性	数据类型	属性描述
full_filename	string	输出文件的名称。
write_mode	Overwrite Append	
inc_field_names	标志	
use_newline_after_records	标志	
delimit_mode	Comma Tab Space Other	
other_delimiter	字符	
quote_mode	None Single Double Other	
other_quote	标志	
generate_import	标志	
encoding	StreamDefault SystemDefault "UTF-8"	

## sasexportnode 属性



“SAS 导出”节点以 SAS 格式输出数据，以便将该数据读入 SAS 或者与 SAS 兼容的软件包。提供三种 SAS 文件格式：SAS for Windows/OS2、SAS for UNIX 或 SAS V7/8。

示例

```
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)
```

表 253: sasexportnode 属性		
sasexportnode 个属性	数据类型	属性描述
format	Windows  UNIX  SAS7  SAS8	可变属性标签字段。
full_filename	string	
export_names	NamesAndLabels  NamesAsLabels	用于将字段名从 IBM SPSSModeler 的导出中映射到 IBM SPSS Statistics 或 SAS 的变量名中。
generate_import	标志	

statisticsexportnode 属性



Statistics 导出节点以 IBM SPSS Statistics .sav 或 .zsav 格式输出数据。 .sav 或 .zsav 文件可以由 IBM SPSS Statistics Base 和其他产品读取。 这也是用于 IBM SPSSModeler 中的高速缓存文件的格式。

有关此节点属性的信息，请参阅第 379 页的『statisticsexportnode 属性』。

tm1odataexport 节点属性



IBM Cognos TM1 导出节点以 Cognos TM1 数据库可以读取的格式导出数据。

表 254: tm1odataexport 节点属性		
tm1odataexport 节点属性	数据类型	属性描述
credential_type	inputCredential 或 storedCredential	用于指示凭证类型。
input_credential	列表	当 credential_type 为 inputCredential 时；指定域、用户名和密码。
stored_credential_name	string	在 credential_type 为 storedCredential 时，指定 C&DS 服务器上凭证的名称。
selected_cube	field	要将数据导出到的多维数据集的名称。 TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")例如：

表 254: tm1odataexport 节点属性 (继续)

tm1odataexport 节点属性	数据类型	属性描述
spss_field_to_tm1_element_mapping	列表	<p>要映射到的 TM1 元素必须是所选多维数据集视图的列维度的组成部分。格式为：[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]</p> <p>存在 2 个用于描述映射信息的列表。将叶元素映射到维度对应于以下示例 2：</p> <p>示例 1：第一个列表：([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) 用于 TM1 维度映射信息。</p> <p>每一个 3 值列表指示维度映射信息。第三个布尔值用于指示是否选择了维度元素。例如： "[Field_1, Dimension_1, False]" 表示 Field_1 映射到 Dimension_1； "[Element_1, Dimension_2, True]" 表示针对 Dimension_2 选择 Element_1。</p> <p>示例 2：第二个列表：([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) 用于 TM1 测量维度元素映射信息。</p> <p>每一个 3 值列表指示测量元素映射信息。第三个布尔值用于指示需要创建新元素。 "[Field_2, ExistMeasureElement, False]" 表示 Field_2 映射到 ExistMeasureElement； "[Field_3, NewMeasureElement, True]" 表示 NewMeasureElement 需要是在 selected_measure 中选择的策略维度，而 Field_3 映射到它。</p>
selected_measure	string	<p>指定测量维度。</p> <p>示例： setProperty("selected_measure", "Measures")</p>
connection_type	AdminServer TM1Server	指示连接类型。缺省值为 AdminServer。
admin_host	string	REST API 的主机名的 URL。如果 connection_type 是 AdminServer，则此属性是必需的。
server_name	string	从 admin_host 中选择的 TM1 服务器的名称。如果 connection_type 是 AdminServer，则此属性是必需的。
server_url	string	TM1 服务器 REST API 的 URL。如果 connection_type 是 TM1Server，则此属性是必需的。

# tm1export 节点属性（不推荐）



IBM Cognos TM1 导出节点以 Cognos TM1 数据库可以读取的格式导出数据。

注: 该节点在 Modeler 18.0 中不推荐使用。替换节点脚本名称是 *tm1odataexport*。

表 255: <i>tm1export</i> 节点属性		
tm1export 节点属性	数据类型	属性描述
pm_host	string	注: 仅适用于 V16.0 和 V17.0。  主机名。 TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm') 例如：
tm1_connection	["field","field", ...,"field"]	注: 仅适用于 V16.0 和 V17.0。  列表属性，其中包含 TM1 服务器的连接详细信息。格式为：[ "TM1_Server_Name", "tm1_username", "tm1_password"]  TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"]) 例如：
selected_cube	field	要将数据导出到的多维数据集的名称。 TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan") 例如：

表 255: *tm1export* 节点属性 (继续)

tm1export 节点属性	数据类型	属性描述
spssfield_tm1element_mapping	列表	<p>要映射到的 TM1 元素必须是所选多维数据集视图的列维度的组成部分。格式为：[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]</p> <p>有 2 个列表用于描述映射信息。将叶元素映射到维度对应于以下第 2 个示例：</p> <p>示例 1：第一个列表：([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) 用于 TM1 维度映射信息。</p> <p>每一个 3 值列表指示维度映射信息。第三个布尔值用于指示是否选择了维度元素。例如："[Field_1, Dimension_1, False]" 表示 Field_1 映射到 Dimension_1；"[Element_1, Dimension_2, True]" 表示针对 Dimension_2 选择 Element_1。</p> <p>示例 2：第二个列表：([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) 用于 TM1 测量维度元素映射信息。</p> <p>每一个 3 值列表指示测量元素映射信息。第三个布尔值用于指示需要创建新元素。"[Field_2, ExistMeasureElement, False]" 表示 Field_2 映射到 ExistMeasureElement；"[Field_3, NewMeasureElement, True]" 表示 NewMeasureElement 需要是在 selected_measure 中选择的策略维度，而 Field_3 映射到它。</p>
selected_measure	string	<p>指定测量维度。</p> <p>示例： setProperty("selected_measure", "Measures")</p>



# xmlexportnode 属性



“XML 导出”节点将数据以 XML 格式输出到文件。还可选择创建 XML 源节点，以将导出的数据读取回到流中。

示例

```
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"}])
```

表 256: xmlexportnode 属性

xmlexportnode 个属性	数据类型	属性描述
full_filename	string	(必需) XML 导出文件的完整路径和文件名。
use_xml_schema	标志	指定是否使用 XML 模式 (XSD 或 DTD 文件) 控制导出数据的结构。
full_schema_filename	string	要使用的 XSD 或 DTD 文件的完整路径和文件名。如果 use_xml_schema 设为 true，则为必需。
generate_import	标志	生成将已导出数据文件读回到流中的 XML 源节点。
records	string	表示记录边界的 XPath 表达式。
map	string	将字段名映射到 XML 结构。



# 第 18 章 IBM SPSS Statistics 节点属性

## statisticsimportnode 属性



Statistics 文件节点从 IBM SPSS Statistics 使用的 .sav 或 .zsav 文件格式以及保存在 IBM SPSSModeler 中的缓存文件（也使用同一格式）读取数据。

示例

```
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)
```

表 257: statisticsimportnode 属性

statisticsimportnode 个属性	数据类型	属性描述
full_filename	string	完整文件名（包括路径）。
password	string	密码。password 参数必须在 file_encrypted 参数之前设置。
file_encrypted	标志	指示文件是否受密码保护。
import_names	NamesAndLabels LabelsAsNames	处理变量名和标签的方法。
import_data	DataAndLabels LabelsAsData	处理值和标签的方法。
use_field_format_for_storage	BOOLEAN	指定导入时是否使用 IBM SPSS Statistics 字段格式信息。

## statistictransformnode 属性



Statistics 转换节点针对 IBM SPSSModeler 中的数据源运行所选的 IBM SPSS Statistics 语法命令。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
stream = modeler.script.stream()
statistictransformnode = stream.createAt("statistictransform",
"Transform", 200, 200)
statistictransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na +
K.")
statistictransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed
Drugs")
statistictransformnode.setPropertyValue("check_before_saving", True)
```

表 258: *statisticstransformnode* 属性

<b>statisticstransformnode</b> 个属性	数据类型	属性描述
syntax	<i>string</i>	
check_before_saving	标志	保存输入项之前验证已输入的语法。如果语法无效，则会显示一条错误消息。
default_include	标志	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。
include	标志	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。
new_name	<i>string</i>	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。

## statisticsmodelnode 属性



Statistics 模型节点使您能够通过运行生成 PMML 的 IBM SPSS Statistics 过程分析和处理数据。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
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")
```

<b>statisticsmodelnode</b> 个属性	数据类型	属性描述
syntax	<i>string</i>	
default_include	标志	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。
include	标志	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。
new_name	<i>string</i>	有关更多信息，请参阅主题 <a href="#">第 144 页的『filternode 属性』</a> 。

## statisticsoutputnode 属性



Statistics 输出节点可调用 IBM SPSS Statistics 过程以分析您的 IBM SPSSModeler 数据。可以访问许多不同的 IBM SPSS Statistics 分析过程。此节点需要 IBM SPSS Statistics 的许可副本。

示例

```
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")

```

表 259: *statisticsoutputnode* 属性

<b>statisticsoutputnode</b> 个属性	数据类型	属性描述
mode	Dialog Syntax	选择“IBM SPSS Statistics 对话框”选项或语法编辑器
syntax	<i>string</i>	
use_output_name	标志	
output_name	<i>string</i>	
output_mode	Screen File	
full_filename	<i>string</i>	
file_type	HTML SPV SPW	

## statisticsexportnode 属性



Statistics 导出节点以 IBM SPSS Statistics *.sav* 或 *.zsav* 格式输出数据。 *.sav* 或 *.zsav* 文件可以由 IBM SPSS Statistics Base 和其他产品读取。这也是用于 IBM SPSSModeler 中的高速缓存文件的格式。

示例

```

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)

```

表 260: *statisticsexportnode* 属性

<b>statisticsexportnode</b> 个属性	数据类型	属性描述
full_filename	<i>string</i>	

表 260: <i>statisticsexportnode</i> 属性 (继续)		
<b>statisticsexportnode</b> 个属性	数据类型	属性描述
file_type	sav	以 <i>sav</i> 或 <i>zsav</i> 格式保存文件。例如：
	zsav	<code>statisticsexportnode.setPropertyValue("file_type", "sav")</code>
encrypt_file	标志	指示文件是否受密码保护。
password	<i>string</i>	密码。
launch_application	标志	
export_names	NamesAndLabels	用于将字段名从 IBM SPSSModeler 的导出中映射到 IBM SPSS Statistics 或 SAS 的变量名中。
	NamesAsLabels	
generate_import	标志	

# 第 19 章 Python 节点属性

## gmm 属性



Gaussian Mixture<sup>®</sup> 模型是一个概率模型，其假定从有限数量的高斯分布和未知参数混合中生成所有数据点。可以将混合模型认为是广义 K-Means 聚类以包含有关数据的协方差结构以及潜伏高斯分布的中心的信息。SPSS 建模器 中的 Gaussian Mixture 节点公开 Gaussian Mixture 库的核心特征和常用参数。此节点使用 Python 进行实现。

表 261: gmm 属性		
gmm 个属性	数据类型	属性描述
use_partition	BOOLEAN	设置为 True 或 False 以指定是否使用分区数据。缺省值为 False。
covariance_type	string	指定 Full、Tied、Diag 或 Spherical 以设置协方差类型。
number_component	整数	指定用于混合组件数量的整数。最小值为 1。缺省值为 2。
component_lable	BOOLEAN	指定 True 以将集群标签设置为字符串，或者指定 False 以将集群标签设置为数字。缺省值为 False。
label_prefix	string	如果使用字符串集群标签，那么可指定前缀。
enable_random_seed	BOOLEAN	如果想要使用随机种子，那么指定 True。缺省值为 False。
random_seed	整数	如果使用随机种子，那么指定要用于生成随机样本的整数。
tol	double	指定收敛阈值。缺省值为 0.0001。
max_iter	整数	指定要执行的最大迭代次数。缺省值为 100。
init_params	string	设置要使用的初始化参数。选项为 Kmeans 或 Random。
warm_start	BOOLEAN	指定 True 以使用最新拟合的解作为下一个拟合调用的初始化。缺省值为 False。

## hdbscannode 属性



Hierarchical Density-Based Spatial Clustering (HDBSCAN)<sup>®</sup> 使用非监督学习来查找数据集的聚类或密集区域。SPSS 建模器 中的 HDBSCAN 节点公开 HDBSCAN 库的核心特征和常用参数。此节点以 Python 实现，当您一开始不了解数据集的分组时，可以使用此节点将数据集聚类为不同的组。

表 262: hdbscannode 属性		
hdbscannode 个属性	数据类型	属性描述
inputs	field	用于聚类的输入字段。

表 262: *hdbscannode* 属性 (继续)

<b>hdbscannode</b> 个属性	数据类型	属性描述
useHPO	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> 可启用或禁用基于 Rbfopt 的超参数优化 (HPO)，该优化可自动发现最佳参数组合，使模型在样本上实现期望或更低的误差率。缺省值为 <b>false</b> 。
min_cluster_size	整数	聚类的最小大小。请指定整数。缺省值为 5。
min_samples	整数	将一个点视为核心点之前，近邻中的样本数。请指定整数。如果设置为 0，那么将使用 min_cluster_size。缺省值为 0。
algorithm	<i>string</i>	指定所要使用的算法： <b>best</b> 、 <b>generic</b> 、 <b>prims_kdtree</b> 、 <b>prims_balltree</b> 、 <b>boruvka_kdtree</b> 或 <b>boruvka_balltree</b> 。缺省值为 <b>best</b> 。
metric	<i>string</i>	指定计算特征数组中实例之间的距离时所要使用的指标： <b>euclidean</b> 、 <b>cityblock</b> 、 <b>L1</b> 、 <b>L2</b> 、 <b>manhattan</b> 、 <b>braycurtis</b> 、 <b>canberra</b> 、 <b>chebyshev</b> 、 <b>correlation</b> 、 <b>minkowski</b> 或 <b>squeclidean</b> 。缺省值为 <b>euclidean</b> 。
useStringLabel	<i>BOOLEAN</i>	指定 <b>true</b> 表示使用字符串聚类标签，指定 <b>false</b> 表示使用数字聚类标签。缺省值为 <b>false</b> 。
stringLabelPrefix	<i>string</i>	如果 useStringLabel 参数设置为 <b>true</b> ，请为字符串标签前缀指定值。缺省前缀为 <b>cluster</b> 。
approx_min_span_tree	<i>BOOLEAN</i>	指定 <b>true</b> 表示接受近似最小生成树，指定 <b>false</b> 表示您希望牺牲速度换取正确性。缺省值为 <b>true</b> 。
cluster_selection_method	<i>string</i>	指定要使用何种方法从密集树中选择聚类： <b>eom</b> 或 <b>leaf</b> 。缺省值为 <b>eom</b> （“质量过剩”算法）。
allow_single_cluster	<i>BOOLEAN</i>	指定 <b>true</b> 表示允许单聚类结果。缺省值为 <b>false</b> 。
p_value	<i>double</i>	如果要使用 <b>minkowski</b> 用于度量，请指定要使用的 <b>p value</b> 。缺省值为 1.5。
leaf_size	整数	使用空间树算法（ <b>boruvka_kdtree</b> 或 <b>boruvka_balltree</b> ）时，请指定树叶节点中的点数。缺省值为 40。
outputValidity	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> ，以控制是否将“有效性指标”图表包括在模型输出中。
outputCondensed	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> ，以控制是否将“密集树”图表包括在模型输出中。
outputSingleLinkage	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> ，以控制是否将“单联结树”图表包括在模型输出中。



表 262: *hdbscannode* 属性 (继续)

<b>hdbscannode</b> 个属性	数据类型	属性描述
outputMinSpan	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> ，以控制是否将“最小生成树”图表包括在模型输出中。
is_split		在 V18.2.1.1 中新增。

## kdemodel 属性



Kernel Density Estimation (KDE)<sup>®</sup> 使用 Ball Tree 或 KD Tree 算法以进行效率查询，并结合无监督学习、特征工程和数据建模等概念。基于相邻元素的方法（例如，KDE）是最流行且最有用的一些密度估算方法。SPSS 建模器 中的 KDE 建模和 KDE 模拟节点公开 KDE 库的核心特征和常用参数。节点使用 Python 进行实现。

表 263: *kdemodel* 属性

<b>kdemodel</b> 个属性	数据类型	属性描述
bandwidth	<i>double</i>	缺省值为 1。
kernel	<i>string</i>	要使用的内核：gaussian、tophat、epanechnikov、exponential、linear 或 cosine。缺省值为 gaussian。
algorithm	<i>string</i>	要使用的树算法：kd_tree、ball_tree 或 auto。缺省值为 auto。
metric	<i>string</i>	在计算距离时要使用的度量。对于 kd_tree 算法，从以下项进行选择：Euclidean、Chebyshev、Cityblock、Minkowski、Manhattan、Infinity、P、L2 或 L1。对于 ball_tree 算法，从以下项进行选择：Euclidian、Braycurtis、Chebyshev、Canberra、Cityblock、Dice、Hamming、Infinity、Jaccard、L1、L2、Minkowski、Matching、Manhattan、P、Rogersanimoto、Russellrao、Sokalmichener、Sokalsneath 或 Kulsinski。缺省值为 Euclidean。
atol	<i>float</i>	期望的结果的绝对容差。较大的容差通常将导致更快执行。缺省值为 0.0。
rtol	<i>float</i>	期望的结果的相对容差。较大的容差通常将导致更快执行。缺省值为 1E-8。
breadthFirst 从 V18.2.1.1 开始，重命名为 breadth_first	<i>BOOLEAN</i>	设置为 <b>True</b> 以使用广度优先方法。设置为 <b>False</b> 以使用深度优先方法。缺省值为 <b>True</b> 。
LeafSize 从 V18.2.1.1 开始，重命名为 leaf_size	整数	底层树的叶大小。缺省值为 40。更改此值可能显著影响性能。

表 263: *kdemodel* 属性 (继续)

<b>kdemodel</b> 个属性	数据类型	属性描述
pValue	<i>double</i>	指定将 Minkowski 用于度量时要使用的“P 值”。缺省值为 1.5。
custom_name		
default_node_name		
use_HPO		

## kdeexport 属性



Kernel Density Estimation (KDE)<sup>®</sup> 使用 Ball Tree 或 KD Tree 算法以进行效率查询，并结合无监督学习、特征工程和数据建模等概念。基于相邻元素的方法（例如，KDE）是最流行且最有用的一些密度估算方法。SPSS 建模器中的 KDE 建模和 KDE 模拟节点公开 KDE 库的核心特征和常用参数。节点使用 Python 进行实现。

表 264: *kdeexport* 属性

<b>kdeexport</b> 个属性	数据类型	属性描述
bandwidth	<i>double</i>	缺省值为 1。
kernel	<i>string</i>	要使用的内核：gaussian 或 tophat。缺省值为 gaussian。
algorithm	<i>string</i>	要使用的树算法：kd_tree、ball_tree 或 auto。缺省值为 auto。
metric	<i>string</i>	在计算距离时要使用的度量。对于 kd_tree 算法，从以下项进行选择：Euclidean、Chebyshev、Cityblock、Minkowski、Manhattan、Infinity、P、L2 或 L1。对于 ball_tree 算法，从以下项进行选择：Euclidian、Braycurtis、Chebyshev、Canberra、Cityblock、Dice、Hamming、Infinity、Jaccard、L1、L2、Minkowski、Matching、Manhattan、P、Rogersanimoto、Russellrao、Sokalmichener、Sokalsneath 或 Kulsinski。缺省值为 Euclidean。
atol	<i>float</i>	期望的结果的绝对容差。较大的容差通常将导致更快执行。缺省值为 0.0。
rtol	<i>float</i>	期望的结果的相对容差。较大的容差通常将导致更快执行。缺省值为 1E-8。
breadthFirst	<i>BOOLEAN</i>	设置为 True 以使用广度优先方法。设置为 False 以使用深度优先方法。缺省值为 True。
LeafSize	整数	底层树的叶大小。缺省值为 40。更改此值可能显著影响性能。
pValue	<i>double</i>	指定将 Minkowski 用于度量时要使用的“P 值”。缺省值为 1.5。

# gmm 属性



Gaussian Mixture<sup>®</sup> 模型是一个概率模型，其假定从有限数量的高斯分布和未知参数混合中生成所有数据点。可以将混合模型认为是广义 K-Means 聚类以包含有关数据的协方差结构以及潜伏高斯分布的中心的的信息。SPSS 建模器 中的 Gaussian Mixture 节点公开 Gaussian Mixture 库的核心特征和常用参数。此节点使用 Python 进行实现。

表 265: gmm 属性		
gmm 个属性	数据类型	属性描述
use_partition	BOOLEAN	设置为 True 或 False 以指定是否使用分区数据。缺省值为 False。
covariance_type	string	指定 Full、Tied、Diag 或 Spherical 以设置协方差类型。
number_component	整数	指定用于混合组件数量的整数。最小值为 1。缺省值为 2。
component_lable	BOOLEAN	指定 True 以将集群标签设置为字符串，或者指定 False 以将集群标签设置为数字。缺省值为 False。
label_prefix	string	如果使用字符串集群标签，那么可指定前缀。
enable_random_seed	BOOLEAN	如果想要使用随机种子，那么指定 True。缺省值为 False。
random_seed	整数	如果使用随机种子，那么指定要用于生成随机样本的整数。
tol	double	指定收敛阈值。缺省值为 0.0001。
max_iter	整数	指定要执行的最大迭代次数。缺省值为 100。
init_params	string	设置要使用的初始化参数。选项为 Kmeans 或 Random。
warm_start	BOOLEAN	指定 True 以使用最新拟合的解作为下一个拟合调用的初始化。缺省值为 False。

# ocsvmnode 属性



单类 SVM 节点使用无监督学习算法。此节点可用于新内容检测。它将检测指定样本集的软边界，以便按是否属于该集合对新点进行分类。SPSS 建模器 中的这个单类 SVM 建模节点是在 Python 中实现的，并且需要使用 scikit-learn<sup>®</sup> Python 库。

表 266: ocsvmnode 属性		
ocsvmnode 个属性	数据类型	属性描述
role_use 从 V18.2.1.1 开始，重命名为 custom_fields	string	指定 predefined 以使用预定义角色，或者指定 custom 以使用定制字段分配。缺省为 predefined。
splits	field	这是用于分隔的字段名称的列表。

表 266: ocsvmnode 属性 (继续)

ocsvmnode 个属性	数据类型	属性描述
use_partition	BOOLEAN	请指定 true 或 false。缺省值为 true。如果设置为 true, 那么在构建模型时, 将仅使用训练数据。
mode_type	string	这是模式。可能的值为 simple 或 expert。如果指定 simple, 那么将禁用“专家”选项卡上的所有参数。
stopping_criteria	string	这是科学记数法字符串。可能的值为 1.0E-1、1.0E-2、1.0E-3、1.0E-4、1.0E-5 或 1.0E-6。缺省值为 1.0E-3。
precision	float	回归精度 (nu)。训练错误和支持向量的尾数边界。请指定大于 0 并小于或等于 1.0 的数字。缺省值为 0.1。
kernel	string	要用在算法中的内核类型。可能的值为 linear、poly、rbf、sigmoid 或 precomputed。缺省值为 rbf。
enable_gamma	BOOLEAN	启用 gamma 参数。请指定 true 或 false。缺省值为 true。
gamma	float	仅针对内核 rbf、poly 和 sigmoid 启用此参数。如果将 enable_gamma 参数设置为 false, 那么此参数将设置为 auto。如果设置为 true, 那么缺省值为 0.1。
coef0	float	这是内核函数中的独立项。仅针对 poly 内核和 sigmoid 内核启用此参数。缺省值为 0.0。
degree	整数	多项式内核函数的次数。仅针对 poly 内核启用此参数。请指定任何整数。缺省值为 3。
shrinking	BOOLEAN	用于指定是否要使用缩小启发式选项。请指定 true 或 false。缺省值为 false。
enable_cache_size	BOOLEAN	启用 cache_size 参数。请指定 true 或 false。缺省值为 false。
cache_size	float	这是内核缓存的大小 (MB)。缺省值为 200。
pc_type	string	这是并行坐标图形的类型。可能的选项为 independent 或 general。
lines_amount	整数	这是要包含在图形上的最大行数。请指定介于 1 和 1000 之间的整数。
lines_fields_custom	BOOLEAN	用于启用 lines_fields 参数, 您可以通过此参数指定要显示在图形输出中的定制字段。如果设置为 false, 那么将显示所有字段。如果设置为 true, 那么仅显示使用 lines_fields 参数指定的字段。为了提高性能, 最多将显示 20 个字段。

表 266: ocsvmnode 属性 (继续)

ocsvmnode 个属性	数据类型	属性描述
lines_fields	field	这是要作为纵轴包含在图形上的字段名称的列表。
enable_graphic	BOOLEAN	请指定 true 或 false。启用图形输出（如果想要节约时间和减少流文件大小，请禁用此选项）。
enable_hpo	BOOLEAN	指定 true 或 false 以启用或禁用 HPO 选项。如果设置为 true，那么将应用 Rbfopt 以自动查找“最佳”单类 SVM 模型，这可达到用户使用以下 target_objval 参数定义的目标对象值。
target_objval	float	我们想要实现的目标函数值（基于样本的模型的错误率）（例如，未知最佳值）。如果最佳值未知，那么将此参数设置为相应的值（例如，0.01）。
max_iterations	整数	尝试模型的最大迭代次数。缺省值为 1000。
max_evaluations	整数	尝试模型的函数求值的最大次数，其中焦点是速度准确性。缺省值为 300。

## rfnode 属性



随机森林节点使用将树模型作为基本模型的组装算法的高级实现。SPSS 建模器 中的此“随机森林”建模节点是在 Python 中实现的，并且需要 scikit-learn® Python 库。

表 267: rfnode 属性

rfnode 个属性	数据类型	属性描述
role_use	string	指定 predefined 以使用预定义角色，或者指定 custom 以使用定制字段分配。缺省为 predefined。
inputs	field	这是输入的字段名称。
splits	field	这是用于分隔的字段名称的列表。
n_estimators	整数	要构建的树数量。缺省值为 10。
specify_max_depth	BOOLEAN	指定定制最大深度。如果为 false，那么将展开节点直至所有叶片均纯净或者直至所有叶片包含的样本书小于 min_samples_split。缺省值为 false。
max_depth	整数	树的最大深度。缺省值为 10。
min_samples_leaf	整数	最小叶节点大小。缺省值为 1。

表 267: *rfnode* 属性 (继续)

<b>rfnode</b> 个属性	数据类型	属性描述
max_features	string	<p>在查找最佳分割时要考虑的特征数目。</p> <ul style="list-style-type: none"> <li>• 如果为 <code>auto</code>, 那么为 <code>max_features=sqrt(n_features)</code> (对于分类器) 和 <code>max_features=sqrt(n_features)</code> (对于回归)。</li> <li>• 如果为 <code>sqrt</code>, 那么为 <code>max_features=sqrt(n_features)</code>。</li> <li>• 如果为 <code>log2</code>, 那么为 <code>max_features=log2(n_features)</code>。</li> </ul> <p>缺省值为 <code>auto</code>。</p>
bootstrap	BOOLEAN	在构建树时, 使用 Bootstrap 样本。缺省值为 <code>true</code> 。
oob_score	BOOLEAN	使用袋外样本来估算泛化关系准确性。缺省值为 <code>false</code> 。
extreme	BOOLEAN	使用仅限随机树。缺省值为 <code>false</code> 。
use_random_seed	BOOLEAN	指定此可复制结果。缺省值为 <code>false</code> 。
random_seed	整数	这是构建树时要使用的随机数种子。请指定任何整数。
cache_size	float	这是内核缓存的大小 (MB)。缺省值为 200。
enable_random_seed	BOOLEAN	启用 <code>random_seed</code> 参数。指定 <code>true</code> 或 <code>false</code> 。缺省值为 <code>false</code> 。
enable_hpo	BOOLEAN	指定 <code>true</code> 或 <code>false</code> 以启用或禁用 HPO 选项。如果设置为 <code>true</code> , 那么将应用 <code>Rbfopt</code> 以自动确定“最佳”随机森林模型, 这可达到用户使用以下 <code>target_objval</code> 参数定义的目标对象值。
target_objval	float	您想要实现的目标函数值 (基于样本的模型的错误率) (例如, 未知最佳值)。如果最佳值未知, 那么将此参数设置为相应的值 (例如, 0.01)。
max_iterations	整数	尝试模型的最大迭代次数。缺省值为 1000。
max_evaluations	整数	尝试模型的函数求值的最大次数, 其中焦点是速度准确性。缺省值为 300。

## smotenode 属性



合成少数类过采样技术 (Synthetic Minority Over-sampling Technique, SMOTE) 节点提供了用于处理不平衡数据集的过采样算法。它提供了用于均衡数据的高级方法。SPSS 建模器中的 SMOTE 流程节点在 Python 中实现, 并且需要 `imbalanced-learn`® Python 库。

表 268: smotenode 属性		
smotenode 个属性	数据类型	属性描述
target_field  从 V18.2.1.1 开始, 重命名为 target	field	目标字段。
sample_ratio	string	用于启用定制比率值。两个选项分别为“自动”(sample_ratio_auto) 和“设置比率”(sample_ratio_manual)。
sample_ratio_value	float	此比率是少数类中的样本数与多数类中的样本数之比。它必须大于 0 并小于或等于 1。缺省值为 auto。
enable_random_seed	BOOLEAN	如果设置为 true, 那么将启用 random_seed 属性。
random_seed	整数	这是由随机数字生成器使用的种子。
k_neighbours	整数	这是要用于构建合成样本的最近邻居的数量。缺省值为 5。
m_neighbours	整数	这是要用于确定是否少数样本处于危险状态的最近邻居的数量。只有在 SMOTE 算法类型为 borderline1 和 borderline2 时, 才启用此选项。缺省值为 10。
algorithm_kind  从 V18.2.1.1 开始, 重命名为 algorithm	string	SMOTE 算法的类型: regular、borderline1 或 borderline2。
usepartition  从 V18.2.1.1 开始, 重命名为 use_partition	BOOLEAN	如果设置为 true, 那么仅将训练数据用于模型构建。缺省值为 true。

## tsnenode 属性



t-分布随机邻域嵌入 (t-SNE) 是用于可视化高维数据的工具。其将数据点亲缘关系转换为可能性。此 t-SNE 节点在 SPSS 建模器 中使用 Python 进行实现并且需要 scikit-learn® Python 库。

表 269: tsnenode 属性		
tsnenode 个属性	数据类型	属性描述
mode_type	string	指定 simple 或 expert 方式。
n_components	string	内嵌空间的维度 (2D 或 3D) 。指定 2 或 3。缺省值为 2。
method	string	指定 barnes_hut 或 exact。缺省值为 barnes_hut。

表 269: tsnode 属性 (继续)

tsnode 个属性	数据类型	属性描述
init	string	嵌套初始化。指定 random 或 pca。缺省值为 random。
target_field 从 V18.2.1.1 开始, 重命名为 target	string	目标字段名称。在输出图形将是颜色映射图。如果未指定目标字段, 那么图像将使用一种颜色。
perplexity	float	困惑度与其他各种学习算法中使用的最近邻元素数量相关。通常, 数据集越大, 需要的困惑度越大。请考虑选择 5 和 50 之间的值。缺省值为 30。
early_exaggeration	float	控制原始空间中的自然聚类在内嵌空间中的紧密程度以及两者之间的空间量。缺省值为 12.0。
learning_rate	float	缺省值为 200。
n_iter	整数	优化的最大迭代次数。至少设置为 250。缺省值为 1000。
angle	float	从一个点度量的远距离节点的角度大小。指定 0-1 范围内的值。缺省值为 0.5。
enable_random_seed	BOOLEAN	设置为 true 以启用 random_seed 参数。缺省值为 false。
random_seed	整数	要使用的随机数种子。缺省值为 None。
n_iter_without_progress	整数	不含进度的最大迭代次数。缺省值为 300。
min_grad_norm	string	如果梯度标准值低于此阈值, 那么优化将停止。缺省值为 1.0E-7。可能的值包括: <ul style="list-style-type: none"> <li>• 1.0E-1</li> <li>• 1.0E-2</li> <li>• 1.0E-3</li> <li>• 1.0E-4</li> <li>• 1.0E-5</li> <li>• 1.0E-6</li> <li>• 1.0E-7</li> <li>• 1.0E-8</li> </ul>
isGridSearch	BOOLEAN	设置为 true 以使用多个不同的困惑度执行 t-SNE。缺省值为 false。
output_Rename	BOOLEAN	如果想要提供定制名称, 那么指定 true, 或者指定 false 以自动命名输出。缺省值为 false。
output_to	string	指定 Screen 或 Output。缺省值为 Screen。
full_filename	string	指定输出文件名。



表 269: tsnode 属性 (继续)		
tsnode 个属性	数据类型	属性描述
output_file_type	string	输出文件格式。指定 HTML 或 Output object。缺省值为 HTML。

# xgboostlinearnode 属性



XGBoost Linear<sup>®</sup> 是将线性模型用作基本模型的梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。SPSS 建模器 中的 XGBoost Linear 节点使用 Python 进行实现。

表 270: xgboostlinearnode 属性		
xgboostlinearnode 个属性	数据类型	属性描述
TargetField  从 V18.2.1.1 开始，重命名为 target	field	
InputFields  从 V18.2.1.1 开始，重命名为 inputs	field	
alpha	double	这是 alpha 线性提升系数参数。请指定任何数字，0 或更大值。缺省值为 0。
lambda	double	这是 lambda 线性提升系数参数。请指定任何数字，0 或更大值。缺省值为 1。
lambdaBias	double	这是 lambda 偏差线性提升系数参数。请指定任何数。缺省值为 0。
numBoostRound  从 V18.2.1.1 开始，重命名为 num_boost_round	整数	模型构建的提升舍入次数值。请指定介于 1 和 1000 之间的值。缺省值为 10。
objectiveType	string	学习任务的目标类型。可能的值为 reg:linear、reg:logistic、reg:gamma、reg:tweedie、count:poisson、rank:pairwise、binary:logistic 或 multi。请注意，对于标志目标，只能使用 binary:logistic 或 multi。如果使用了 multi，那么评分结果将显示 multi:softmax 和 multi:softprob XGBoost 目标类型。
random_seed	整数	随机数种子。任何介于 0 和 9999999 之间的数字。缺省值为 0。

表 270: *xgboostlinearnode* 属性 (继续)

<b>xgboostlinearnode</b> 个属性	数据类型	属性描述
useHPO	<i>BOOLEAN</i>	指定 <b>true</b> 或 <b>false</b> 以后用或禁用 HPO 选项。如果设置为 <b>true</b> ，那么将应用 <b>Rbfopt</b> 以自动查找“最佳”单类 SVM 模型，这可达到用户使用 <b>target_objval</b> 参数定义的目标对象值。

## xgboosttreenode 属性



XGBoost Tree® 是将树模型用作基本模型的梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。XGBoost Tree 具有很高的灵活性，并提供了很多对于大多数用户来说过于复杂的参数，因此 SPSS 建模器中的 XGBoost Tree 节点仅显示了核心功能和常用参数。此节点使用 Python 进行实现。

表 271: *xgboosttreenode* 属性

<b>xgboosttreenode</b> 个属性	数据类型	属性描述
TargetField  从 V18.2.1.1 开始，重命名为 <b>target</b>	<i>field</i>	目标字段。
InputFields  从 V18.2.1.1 开始，重命名为 <b>inputs</b>	<i>field</i>	输入字段。
treeMethod  从 V18.2.1.1 开始，重命名为 <b>tree_method</b>	<i>string</i>	模型构建的树方法。可能的值为 <b>auto</b> 、 <b>exact</b> 或 <b>approx</b> 。缺省值为 <b>auto</b> 。
numBoostRound  从 V18.2.1.1 开始，重命名为 <b>num_boost_round</b>	整数	模型构建的提升舍入次数值。请指定介于 1 和 1000 之间的值。缺省值为 10。
maxDepth  从 V18.2.1.1 开始，重命名为 <b>max_depth</b>	整数	树增长的最大深度。请指定值 1 或更高值。缺省值为 6。
minChildWeight  从 V18.2.1.1 开始，重命名为 <b>min_child_weight</b>	<i>double</i>	树增长的最小子代权重。请指定值 0 或更高值。缺省值为 1。
maxDeltaStep  从 V18.2.1.1 开始，重命名为 <b>max_delta_step</b>	<i>double</i>	树增长的最大变化量步骤。请指定值 0 或更高值。缺省值为 0。

表 271: xgboosttreenode 属性 (继续)

xgboosttreenode 个属性	数据类型	属性描述
objectiveType  从 V18.2.1.1 开始, 重命名为 objective_type	string	学习任务的目标类型。可能的值为 reg:linear、reg:logistic、reg:gamma、reg:tweedie、count:poisson、rank:pairwise、binary:logistic 或 multi。请注意, 对于标志目标, 只能使用 binary:logistic 或 multi。如果使用了 multi, 那么评分结果将显示 multi:softmax 和 multi:softprob XGBoost 目标类型。
earlyStopping  从 V18.2.1.1 开始, 重命名为 early_stopping	BOOLEAN	是否使用提前停止功能。缺省值为 False。
earlyStoppingRounds  从 V18.2.1.1 开始, 重命名为 early_stopping_rounds	整数	验证错误需要至少在每个提前停止舍入处降低才能继续培训。缺省值为 10。
evaluationDataRatio  从 V18.2.1.1 开始, 重命名为 evaluation_data_ratio	double	用于验证错误的输入数据的比率。缺省值为 0.3。
random_seed	整数	随机数种子。任何介于 0 和 9999999 之间的数字。缺省值为 0。
sampleSize  从 V18.2.1.1 开始, 重命名为 sample_size	double	用于控制过度拟合的子样本。请指定介于 0.1 和 1.0 之间的值。缺省值为 0.1。
eta	double	用于控制过度拟合的 eta。请指定介于 0 和 1 之间的值。缺省值为 0.3。
gamma	double	用于控制过度拟合的伽玛。请指定任何数字, 0 或更大值。缺省值为 6。
colsSampleRatio  从 V18.2.1.1 开始, 重命名为 col_sample_ratio	double	用于控制过度拟合的列样本 (按树列出)。请指定介于 0.01 和 1 之间的值。缺省值为 1。
colsSampleLevel  从 V18.2.1.1 开始, 重命名为 col_sample_level	double	用于控制过度拟合的列样本 (按级别列出)。请指定介于 0.01 和 1 之间的值。缺省值为 1。
lambda	double	用于控制过度拟合的 lambda。请指定任何数字, 0 或更大值。缺省值为 1。
alpha	double	用于控制过度拟合的 alpha。请指定任何数字, 0 或更大值。缺省值为 0。

表 271: <i>xgboosttreenode</i> 属性 (继续)		
<b>xgboosttreenode</b> 个属性	数据类型	属性描述
scalePosWeight  从 V18.2.1.1 开始, 重命名为 scale_pos_weight	<i>double</i>	用于处理不平衡数据集的刻度位置权重。 缺省值为 1。
use_HP0  针对 V18.2.1.1 新增		

# 第 20 章 Spark 节点属性

## isotonicasnode 属性



保序回归属于回归算法系列。SPSS 建模器 中的 Isotonic-AS 节点使用 Spark 进行实现。有关保序回归算法的详细信息，请参阅 <https://spark.apache.org/docs/3.5.4/mllib-isotonic-regression.html#isotonic-regression>。

表 272: isotonicasnode 属性		
isotonicasnode 个属性	数据类型	属性描述
label	string	此属性是为其计算了保序回归的因变量。
features	string	此属性是因变量。
weightCol	string	权重表示测量值的数字。缺省值为 1。
isotonic	BOOLEAN	此属性指示类型为 isotonic 还是 antitonic。
featureIndex	整数	如果 featuresCol 是矢量列，那么此属性表示功能的索引。缺省值为 0。

## kmeansasnode 属性



K-Means 是最常用的聚类算法之一。它将数据点聚集成多个预定义聚类。SPSS 建模器 中的 K-Means-AS 节点使用 Spark 进行实现。有关 K-Means 算法的详细信息，请参阅 <https://spark.apache.org/docs/3.5.4/ml-clustering.html#clustering>。请注意，K-Means-AS 节点自动对分类变量执行独热编码。

表 273: kmeansasnode 属性		
kmeansasnode 属性	值	属性描述
roleUse	string	指定 predefined 表示使用预定义角色，指定 custom 表示使用定制字段分配。缺省值为 predefined。
autoModel	BOOLEAN	指定 true 表示将缺省名称 (\$S-prediction) 用于新生成的评分字段，指定 false 表示使用定制名称。缺省值为 true。
features	field	用于输入的字段名称列表（如果 roleUse 属性设置为 custom）。
name	string	新生成的评分字段的名称（如果 autoModel 属性设置为 false）。
clustersNum	整数	要创建的聚类数目。缺省值为 5。
initMode	string	初始化算法。可能的值为 k-means   或 random。缺省值为 k-means  。

表 273: *kmeansasnode* 属性 (继续)

<b>kmeansasnode</b> 属性	值	属性描述
initSteps	整数	初始化步骤数（如果 <code>initMode</code> 设置为 <code>k-means  </code> ）。缺省值为 2。
advancedSettings	<i>BOOLEAN</i>	指定 <code>true</code> 表示让下列四个属性可用。缺省值为 <code>false</code> 。
maxIteration	整数	聚类的最大迭代次数。缺省值为 20。
tolerance	<i>string</i>	停止迭代的容差。可能的设置为 <code>1.0E-1</code> 、 <code>1.0E-2</code> 、...、 <code>1.0E-6</code> 。缺省值为 <code>1.0E-4</code> 。
setSeed	<i>BOOLEAN</i>	指定 <code>true</code> 表示使用定制随机种子。缺省值为 <code>false</code> 。
randomSeed	整数	定制随机种子（如果 <code>setSeed</code> 属性为 <code>true</code> ）。

## multilayerperceptronnode 属性



多层感知器是基于前馈人工神经网络的分类器，其由多个层组成。每一层都完全连接到网络中的下一层。SPSS 建模器 中的 MultiLayerPerceptron-AS 节点使用 Spark 进行实现。有关多层感知器分类器 (MLPC) 的详细信息，请参阅 <https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier>。

表 274: *multilayerperceptronnode* 属性

<b>multilayerperceptronnode</b> 个属性	数据类型	属性描述
features	<i>field</i>	用作预测输入的一个或多个字段。
label	<i>field</i>	用作预测目标的字段。
layers[0]	整数	要包含的感知器层数。缺省值为 1。
layers[1...<latest-1>]	整数	隐藏层数。缺省值为 1。
layers[<latest>]	整数	输出层数。缺省值为 1。
seed	整数	定制随机种子。
maxiter	整数	要执行的最大迭代次数。缺省值为 10。

## xgboostasnode 属性



XGBoost 是梯度提升算法的高级实现。提升算法以迭代方式学习弱分类器，然后将它们添加到最终的强分类器中。XGBoost 具有很高的灵活性，并提供了很多对于大多数用户来说过于复杂的参数，因此 SPSS 建模器 中的 XGBoost-AS 节点仅显示了核心功能和常用参数。在 Spark 中实现 XGBoost-AS 节点。

表 275: *xgboostasnode* 属性

<b>xgboostasnode</b> 个属性	数据类型	属性描述
target_field	<i>field</i>	目标的字段名称的列表。

表 275: xgboostasnode 属性 (继续)

xgboostasnode 个属性	数据类型	属性描述
input_fields	<i>field</i>	输入的字段名称的列表。
nWorkers	整数	用于训练 XGBoost 模型的工作程序的数量。缺省值为 1。
numThreadPerTask	整数	每个工作程序使用的线程数。缺省值为 1。
useExternalMemory	<i>BOOLEAN</i>	是否使用外部内存作为缓存。缺省值为 <b>false</b> 。
boosterType	<i>string</i>	要使用的提升机类型。可用选项为 <i>gbtree</i> 、 <i>gblinear</i> 或 <i>dart</i> 。缺省值为 <i>gbtree</i> 。
numBoostRound	整数	用于提升的舍入次数。请指定值 0 或更高值。缺省值为 10。
scalePosWeight	<i>double</i>	用于控制正权重和负权重的平衡。缺省值为 1。
randomseed	整数	这是由随机数字生成器使用的种子。缺省值为 0。
objectiveType	<i>string</i>	学习目标。可能的值为 <i>reg:linear</i> 、 <i>reg:logistic</i> 、 <i>reg:gamma</i> 、 <i>reg:tweedie</i> 、 <i>rank:pairwise</i> 、 <i>binary:logistic</i> 或 <i>multi</i> 。请注意，对于标志目标，只能使用 <i>binary:logistic</i> 或 <i>multi</i> 。如果使用了 <i>multi</i> ，那么评分结果将显示 <i>multi:softmax</i> 和 <i>multi:softprob</i> XGBoost 目标类型。缺省值为 <i>reg:linear</i> 。
evalMetric	<i>string</i>	用于验证数据的评估度量。根据目标，将分配缺省度量。可能的值为 <i>rmse</i> 、 <i>mae</i> 、 <i>logloss</i> 、 <i>error</i> 、 <i>merror</i> 、 <i>mlogloss</i> 、 <i>auc</i> 、 <i>ndcg</i> 、 <i>map</i> 或 <i>gamma-deviance</i> 。缺省值为 <i>rmse</i> 。
lambda	<i>double</i>	这是有关权重的 L2 规则化术语。增大此值将使模型更保守。请指定任何数字，0 或更大值。缺省值为 1。
alpha	<i>double</i>	有关权重的 L1 规则化术语。增大此值将使模型更保守。请指定任何数字，0 或更大值。缺省值为 0。
lambdaBias	<i>double</i>	这是有关基本选项的 L2 规则化术语。如果使用 <i>gblinear</i> 提升机类型，那么此 <i>lambda</i> 偏差线性提升机参数可用。请指定任何数字，0 或更大值。缺省值为 0。

表 275: xgboostasnode 属性 (继续)

<b>xgboostasnode</b> 个属性	数据类型	属性描述
treeMethod	<i>string</i>	如果使用 gbtree 或 dart 提升机类型, 那么用于树生成的此树方法参数 (以及随后的其他树参数) 可用。它指定要使用的 XGBoost Tree 结构算法。可用选项为 auto、exact 或 approx。缺省值为 auto。
maxDepth	整数	树的最大深度。请指定值 2 或更高值。缺省值为 6。
minChildWeight	<i>double</i>	子代中需要的实例权重 (hessian) 的最小总和。请指定值 0 或更高值。缺省值为 1。
maxDeltaStep	<i>double</i>	允许用于每棵树的权重估算的最大增量幅度。请指定值 0 或更高值。缺省值为 0。
sampleSize	<i>double</i>	子样本表示训练实例的比率。请指定介于 0.1 和 1.0 之间的值。缺省值为 1.0。
eta	<i>double</i>	这是更新步骤期间用于防止过度拟合的步长收缩。请指定介于 0 和 1 之间的值。缺省值为 0.3。
gamma	<i>double</i>	这是对树的某个叶节点进行进一步分区所需的最小损失减小。请指定任何数字, 0 或更大值。缺省值为 6。
colsSampleRatio	<i>double</i>	构造每棵树时列的子样本比率。请指定介于 0.01 和 1 之间的值。缺省值为 1。
colsSampleLevel	<i>double</i>	每个级别中每个分割的列的子样本比率。请指定介于 0.01 和 1 之间的值。缺省值为 1。
normalizeType	<i>string</i>	如果使用 dart 提升机类型, 那么此 dart 参数和以下三个 dart 参数可用。此参数设置标准化算法。请指定 tree 或 forest。缺省值为 tree。
sampleType	<i>string</i>	采样算法类型。请指定 uniform 或 weighted。缺省值为 uniform。
rateDrop	<i>double</i>	dart 提升机参数的丢弃率。请指定介于 0.0 和 1.0 之间的值。缺省值为 0.0。
skipDrop	<i>double</i>	表示跳过丢弃的概率的 dart 提升机参数。请指定介于 0.0 和 1.0 之间的值。缺省值为 0.0。



# 第 21 章 超节点属性

下表中说明了超节点的特定属性。 注意公共节点属性也可应用于超节点。

表 276: 终端超节点属性		
属性名称	属性类型/值列表	属性描述
execute_method	Script	
	Normal	
script	string	

## 超节点参数

可使用通用格式在脚本中创建或设置超节点参数：

```
mySuperNode.setParameterValue("minvalue", 30)
```

您可以使用以下内容检索参数值：

```
value mySuperNode.getParameterValue("minvalue")
```

## 查找现有超节点

您可以使用 `findByType()` 函数在流中查找超节点：

```
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)
```

## 设置已封装节点的属性

可以通过访问某个超节点中的子图来设置封装在该超节点中特定节点的属性。例如，假设有一个源超节点，其中封装有“变量文件”节点以读取数据。可以通过访问子图并查找相关节点来传递要读取的文件的名称（使用 `full_filename` 属性指定），如下所示：

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

## 创建超节点

如果您要从头开始创建超节点及其内容，那么可以通过创建超节点、访问子图并创建所需节点来以类似方式完成此操作。还必须确保超节点图中的节点也链接到输入 - 和/或输出连接器节点。例如，如果您要创建过程超节点：

```
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)
```



# 附录 A 节点名引用

此部分提供 IBM SPSS Modeler 中节点的脚本编制名称引用。

## 模型块名称

模型块（也称为生成的模型）可以按类型进行引用，就好像节点和输出对象一样。下表列出模型对象的引用名称。

请注意，这些名称专用于引用模型选用板（位于 IBM SPSS Modeler 窗口的右上角）中的模型块。要引用已经添加到流中进行评分的模型节点，那么使用另外一套以 `apply...` 为前缀的名称。

注：在通常情况下，建议按名称和类型来引用模型，以避免引起混淆。

表 277: 模型块名称（建模选用板）	
模型名称	模型
anomalydetection	异常
Apriori	Apriori
autoclassifier	自动分类器
autocluster	自动聚类
autonumeric	自动数字
bayesnet	贝叶斯网络
c50	C5.0
carma	Carma
cart	C&R 树
chaid	CHAID
coxreg	Cox 回归
decisionlist	决策列表
discriminant	判别
因子	PCA/因子
featureselection	特征选择
genlin	广义线性回归
glmm	GLMM
kmeans	K-Means
knn	k-最近相邻元素
kohonen	Kohonen
线性	线性
logreg	Logistic 回归
neuralnetwork	神经网络
quest	QUEST

表 277: 模型块名称（建模选用板）（继续）	
模型名称	模型
回归	线性回归
序列	序列
slrm	自学响应模型
statisticsmodel	IBM SPSS Statistics 模型
SVM	支持向量机
timeseries	时间序列
twostep	TwoStep

表 278: 模型块名称（数据库建模选用板）	
模型名称	模型
db2imcluster	IBM ISW 聚类
db2imlog	IBM ISW Logistic 回归
db2imnb	IBM ISW 朴素贝叶斯
db2imreg	IBM ISW 回归
db2imtree	IBM ISW 决策树
msassoc	MS 关联规则
msbayes	MS 朴素贝叶斯
mscluster	MS 聚类
mslogistic	MS Logistic 回归
msneuralnetwork	MS 神经网络
msregression	MS 线性回归
mssequencecluster	MS 序列聚类
mstimeseries	MS 时间序列
mstree	MS 决策树
netezababes	Netezza 贝叶斯网络
netezadectree	Netezza 决策树
netezadivcluster	Netezza 分裂式聚类
netezaglm	Netezza 广义线性
netezakmeans	Netezza K-Means
netezaknn	Netezza KNN
netezalineregression	Netezza 线性回归
netezanaivebayes	Netezza 朴素贝叶斯
netezapca	Netezza PCA
netezaregtree	Netezza 回归树
netezatimeseries	Netezza 时间序列

表 278: 模型块名称（数据库建模选用板）(继续)	
模型名称	模型
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Oracle 决策树
oraglm	Oracle GLM
orakmeans	Oracle <i>k</i> -Means
oranb	Oracle 朴素贝叶斯
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

## 避免重复的模型名称

使用脚本对生成的模型进行操作时，务必注意：允许重复的模型名称可能会导致歧义引用。为了避免这种情况的发生，最好在编制脚本时要求对生成的模型使用唯一的名称。

要为重复模型名称设置选项：

1. 从菜单中选择：  
    **工具 > 用户选项**
2. 单击**通知**选项卡。
3. 选择**替换原有模型**以限制生成的模型的重复命名。

存在不明确的模型引用时，脚本执行行为在 SPSS Modeler 与 IBM SPSS Collaboration and Deployment Services 之间可能有所不同。SPSS Modeler 客户机提供了“替换先前模型”选项，此选项将自动替换同名的模型（例如，脚本通过循环执行迭代，以便每次都生成不同的模型）。但是，在 IBM SPSS Collaboration and Deployment Services 中运行同一脚本时，此选项不可用。通过将每次迭代中生成的模型重命名以避免对模型进行不明确的引用，或者通过在循环结束前清除当前模型（例如，添加 `clear generated palette` 语句），可以避免这种情况。

## 输出类型名称

下表列出了所有的输出对象类型和创建它们的节点。

表 279: 输出对象类型以及创建这些类型的节点	
输出对象类型	节点
analysisoutput	分析
collectionoutput	集合
dataauditoutput	数据审核
distributionoutput	分布
evaluationoutput	评估
histogramoutput	直方图
matrixoutput	矩阵
meansoutput	平均值

表 279: 输出对象类型以及创建这些类型的节点 (继续)	
输出对象类型	节点
multiplotoutput	多重散点图
plotoutput	统计图
qualityoutput	质量
reportdocumentoutput	此对象类型不属于节点；它是工程报告创建的输出
reportoutput	报告
statisticsprocedureoutput	统计信息 输出
statisticsoutput	统计信息
tableoutput	表
timeplotoutput	时间散点图
weboutput	Web

# 附录 B 从旧脚本编制迁移到 Python 脚本编制

## 旧脚本迁移概述

本节提供 IBM SPSS Modeler 中 Python 脚本编制与旧脚本编制之间的差异摘要，并提供有关如何将旧脚本迁移为 Python 脚本的信息。在本节中，您将找到标准 SPSS Modeler 旧命令和等效的 Python 命令的列表。

## 一般差异

旧脚本编制的设计在很大程度上借鉴了操作系统命令脚本。尽管包含一些块结构（例如 `if...then...else...endif` 和 `for...endfor`），但旧脚本编制面向行，并且缩进通常没有意义。

在 Python 脚本编制中，缩进有意义，并且属于同一逻辑块的行必须在同一级别进行缩进。

**注：**复制和粘贴 Python 代码时，请务必小心操作。在编辑器中，使用 `tab` 缩进的行可能与使用空格缩进的行看起来一样。但是，Python 脚本将生成错误，这是因为未将这些行视作缩进相同的行。

## 脚本编制上下文

脚本编制上下文定义了将在其中执行脚本的环境，例如，用于执行脚本的流或超节点。例如，在旧脚本编制中，上下文是隐式的，这意味着假定流脚本中的所有节点引用都包含在执行该脚本的流中。

在 Python 脚本编制中，脚本编制上下文通过 `modeler.script` 模块以显式方式提供。例如，Python 流脚本可以使用以下代码访问执行该脚本的流：

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

然后，可以通过返回的对象来调用与流相关的函数。

## 命令与函数

旧脚本编制面向命令。这意味着脚本的每一行通常以后跟参数进行运行的命令开始，例如：

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python 使用通常通过定义函数的对象（模块、类或对象）所调用的函数，例如：

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type")
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

## 文字和注释

IBM SPSS Modeler 中一些常用的文字和注释命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 280: 文字和注释的旧脚本编制到 Python 脚本编制的映射

旧脚本编制	Python 脚本编制
整数，例如 4	相同
浮点数，例如 0.003	相同

表 280: 文字和注释的旧脚本编制到 <i>Python</i> 脚本编制的映射 (继续)	
旧脚本编制	Python 脚本编制
单个加引号的字符串, 例如 'Hello'	相同  注: 包含非 ASCII 字符的字符串字面值必须以 u 作为前缀, 以确保它们表示为 Unicode。
双引号字符串, 例如 "Hello again"	相同  注: 包含非 ASCII 字符的字符串字面值必须以 u 作为前缀, 以确保它们表示为 Unicode。
长字符串, 例如 <pre>"""This is a string that spans multiple lines"""</pre>	相同
列表, 例如 [1 2 3]	[1, 2, 3]
变量引用, 例如 set x = 3	x = 3
行继续符 (\), 例如 <pre>set x = [1 2 \ 3 4]</pre>	<pre>x = [ 1, 2,\ 3, 4]</pre>
块注释, 例如 <pre>/* This is a long comment over a line. */</pre>	<pre>""" This is a long comment over a line. """</pre>
行注释, 例如 set x = 3 # make x 3	x = 3 # make x 3
undef	None
true	True
false	False

## 运算符

IBM SPSS Modeler 中一些常用的运算符命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本, 以便在 IBM SPSS Modeler 17 中使用。

表 281: 运算符的旧脚本编制到 <i>Python</i> 脚本编制的映射	
旧脚本编制	Python 脚本编制
<pre>NUM1 + NUM2 LIST + ITEM LIST1 + LIST2</pre>	<pre>NUM1 + NUM2 LIST.append(ITEM) LIST1.extend(LIST2)</pre>
<pre>NUM1 - NUM2 LIST - ITEM</pre>	<pre>NUM1 - NUM2 LIST.remove(ITEM)</pre>
NUM1 * NUM2	NUM1 * NUM2
NUM1 / NUM2	NUM1 / NUM2



表 281: 运算符的旧脚本编制到 Python 脚本编制的映射 (继续)

旧脚本编制	Python 脚本编制
<code>=</code> <code>==</code>	<code>==</code>
<code>/=</code> <code>/==</code>	<code>!=</code>
<code>X ** Y</code>	<code>X ** Y</code>
<code>X &lt; Y</code> <code>X &lt;= Y</code> <code>X &gt; Y</code> <code>X &gt;= Y</code>	<code>X &lt; Y</code> <code>X &lt;= Y</code> <code>X &gt; Y</code> <code>X &gt;= Y</code>
<code>X div Y</code> <code>X rem Y</code> <code>X mod Y</code>	<code>X // Y</code> <code>X % Y</code> <code>X % Y</code>
<code>and</code> <code>or</code> <code>not (EXPR)</code>	<code>and</code> <code>or</code> <code>not EXPR</code>

## 条件语句和循环

IBM SPSS Modeler 中一些常用的条件和循环命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 282: 条件语句和循环的旧脚本编制到 Python 脚本编制的映射

旧脚本编制	Python 脚本编制
<code>for VAR from INT1 to INT2</code> <code>...</code> <code>endfor</code>	<code>for VAR in range(INT1, INT2):</code> <code>...</code> 或者 <code>VAR = INT1</code> <code>while VAR &lt;= INT2:</code> <code>...</code> <code>VAR += 1</code>
<code>for VAR in LIST</code> <code>...</code> <code>endfor</code>	<code>for VAR in LIST:</code> <code>...</code>
<code>for VAR in_fields_to NODE</code> <code>...</code> <code>endfor</code>	<code>for VAR in NODE.getInputDataModel():</code> <code>...</code>
<code>for VAR in_fields_at NODE</code> <code>...</code> <code>endfor</code>	<code>for VAR in NODE.getOutputDataModel():</code> <code>...</code>

表 282: 条件语句和循环的旧脚本编制到 <i>Python</i> 脚本编制的映射 (继续)	
旧脚本编制	Python 脚本编制
<pre> if...then ... elseif...then ... else ... endif </pre>	<pre> if ...: ... elif ...: ... else: ... </pre>
<pre> with TYPE OBJECT ... endwith </pre>	无等效项
<pre> var VAR1 </pre>	不需要变量声明

## 变量

在旧脚本编制中，引用变量之前对变量进行了声明，例如：

```

var mynode
set mynode = create typenode at 96 96

```

在 Python 脚本编制中，变量在首次引用时进行创建，例如：

```

mynode = stream.createAt("type", "Type", 96, 96)

```

在旧脚本编制中，必须使用 ^ 运算符显式除去对变量的引用，例如：

```

var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input

```

与大多对数脚本编制语言一样，在 Python 脚本编制中，这不是必需操作，例如：

```

mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction", "Age", "Input")

```

## 节点、输出和模型类型

在旧脚本编制中，各种对象类型（节点、输出和模型）通常向对象类型追加了类型。例如，“派生”节点具有 `derivenode` 类型：

```

set feature_name_node = create derivenode at 96 96

```

Python 中的 IBM SPSS Modeler API 未包含 `node` 后缀，因此“派生”节点具有 `derive` 类型，例如：

```

feature_name_node = stream.createAt("derive", "Feature", 96, 96)

```

旧脚本编制与 Python 脚本编制的类型名称中的唯一差异在于缺少类型后缀。

## 属性名

在旧脚本编制和 Python 脚本编制中，属性名相同。例如，在这两种脚本编制环境中，变量文件节点中用于定义文件位置的属性为 `full_filename`。

## 节点引用

许多旧脚本使用隐式搜索来查找和访问要修改的节点。例如，下列命令用于在当前流中搜索带有“类型”标签的“类型”节点，然后将“年龄”字段的方向（或建模角色）设置为输入，并将“药品”字段设置为目标（也就是要预测的值）：

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

在 Python 脚本编制中，必须先显式查找节点对象，然后再调用用于设置属性值的函数，例如：

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

注：在本例中，“Target”必须包含在字符串引号中。

另外，Python 脚本可使用 `modeler.api` 软件包中的 `ModelingRole` 枚举。

虽然 Python 脚本编制版本可能更为繁琐，但它能够实现更佳的运行时性能，这是因为通常仅执行一次搜索节点。在旧脚本编制示例中，将针对各个命令搜索节点。

另外，还支持按标识查找节点（可以在节点对话框的“注释”选项卡中查看节点标识）。例如，在旧脚本编制中：

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

以下脚本显示 Python 脚本编制中的同一示例：

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

## 获取并设置属性

旧脚本编制使用 `set` 命令来指定值。`set` 命令后跟的词汇可以是属性定义。以下脚本显示了两种可能的用于设置属性的脚本格式：

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

在 Python 脚本编制中，通过使用函数 `setProperty()` 和 `setKeyedPropertyValue()`，可以实现同一结果，例如：

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

在旧脚本编制中，可以使用 `get` 命令来实现访问属性值，例如：

```
var n v
set n = get node :filternode
set v = ^n.name
```

在 Python 脚本编制中，通过使用函数 `getPropertyValue()`，可以实现同一结果，例如：

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

在旧脚本编制中，`create` 命令用于创建新节点，例如：

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

在 Python 脚本编制中，流具有多种创建节点的方法，例如：

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

在旧脚本编制中，`connect` 命令用于创建节点之间的链接，例如：

```
connect ^agg to ^select
```

在 Python 脚本编制中，`link` 方法用于创建节点之间的链接，例如：

```
stream.link(agg, select)
```

在旧脚本编制中，`disconnect` 命令用于除去节点之间的链接，例如：

```
disconnect ^agg from ^select
```

在 Python 脚本编制中，`unlink` 方法用于除去节点之间的链接，例如：

```
stream.unlink(agg, select)
```

在旧脚本编制中，`position` 命令用于将节点放在流画布上或其他节点之间，例如：

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

在 Python 脚本编制中，通过使用两种不同的方法（`setXYPosition` 和 `setPositionBetween`），可以实现同一结果。例如：

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

## 节点操作

IBM SPSS Modeler 中一些常用的节点操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 283: 节点操作的旧脚本编制到 Python 脚本编制的映射	
旧脚本编制	Python 脚本编制
create <i>nodespec</i> at x y	<pre>stream.create(type, name) stream.createAt(type, name, x, y) stream.createBetween(type, name, preNode, postNode) stream.createModelApplier(model, name)</pre>
connect <i>fromNode</i> to <i>toNode</i>	<pre>stream.link(fromNode, toNode)</pre>
delete <i>node</i>	<pre>stream.delete(node)</pre>
disable <i>node</i>	<pre>stream.setEnabled(node, False)</pre>
enable <i>node</i>	<pre>stream.setEnabled(node, True)</pre>

表 283: 节点操作的旧脚本编制到 Python 脚本编制的映射 (继续)	
旧脚本编制	Python 脚本编制
disconnect <i>fromNode</i> from <i>toNode</i>	<pre>stream.unlink(<i>fromNode</i>, <i>toNode</i>) stream.disconnect(<i>node</i>)</pre>
duplicate <i>node</i>	<pre>node.duplicate()</pre>
execute <i>node</i>	<pre>stream.runSelected(<i>nodes</i>, <i>results</i>) stream.runAll(<i>results</i>)</pre>
flush <i>node</i>	<pre>node.flushCache()</pre>
position <i>node</i> at x y	<pre>node.setXYPosition(x, y)</pre>
position <i>node</i> between <i>node1</i> and <i>node2</i>	<pre>node.setPositionBetween(<i>node1</i>, <i>node2</i>)</pre>
rename <i>node</i> as <i>name</i>	<pre>node.setLabel(<i>name</i>)</pre>

## 循环

在旧脚本编制中，主要支持下列两种循环选项：

- 计数循环，在此循环中，下标变量在两个整数范围之间进行移动。
- 序列循环，此循环对值序列进行遍历，以便将当前值绑定到循环变量。

以下脚本是旧脚本编制中的计数循环示例：

```
for i from 1 to 10
    println ^i
endfor
```

以下脚本是旧脚本编制中的序列循环示例：

```
var items
set items = [a b c d]

for i in items
    println ^i
endfor
```

另外，还可以使用其他类型的循环：

- 对模型选用板中的模型或输出选用板中的输出执行迭代。
- 对传入或传出节点的字段执行迭代。

Python 脚本编制还支持其他类型的循环。以下脚本是 Python 脚本编制中的计数循环示例：

```
i = 1
while i <= 10:
    print i
    i += 1
```

以下脚本是 Python 脚本编制中的序列循环示例：

```
items = ["a", "b", "c", "d"]
for i in items:
    print i
```

序列循环非常灵活，并且在与 IBM SPSS Modeler API 方法结合后，此循环可支持多个脚本编制用例。以下示例显示如何使用 Python 脚本编制中的序列循环对传出节点的字段执行迭代：

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
    print column.getColumnName()
```

## 执行流

执行流的过程中，生成的模型或输出对象将添加到其中一个对象管理器中。在旧脚本编制中，脚本必须在对象管理器中找到构建对象，或者从生成输出的节点访问最新生成的输出。

在 Python 中，执行流的过程有所不同：执行生成的任何模型或输出将以传递到执行函数的列表形式返回。这使得可以更加轻松地访问流执行结果。

旧脚本编制支持下列三种流执行命令：

- `execute_all`，用于执行流中的所有可执行终端节点。
- `execute_script`，用于执行流脚本（与脚本执行的设置无关）。
- `execute node`，用于执行指定的节点。

Python 脚本编制支持一组类似的函数：

- `stream.runAll(results-list)`，用于执行流中的所有可执行终端节点。
- `stream.runScript(results-list)`，用于执行流脚本（与脚本执行的设置无关）。
- `stream.runSelected(node-array, results-list)`，用于按节点的提供顺序执行指定的一组节点。
- `node.run(results-list)`，用于执行指定的节点。

在旧脚本中，可以使用带有可选整数代码的 `exit` 命令来终止流执行，例如：

```
exit 1
```

在 Python 脚本编制中，使用以下脚本可实现同一结果：

```
modeler.script.exit(1)
```

## 通过文件系统和存储库访问对象

在旧脚本编制中，您可以使用 `open` 命令打开现有流、模型或输出对象，例如：

```
var s
set s = open stream "c:/my streams/modeling.str"
```

在 Python 脚本编制中，存在一个可从会话进行访问的 `TaskRunner` 类，并且这个类可用于执行类似的任务，例如：

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

在旧脚本编制中，要保存对象，您可以使用 `save` 命令，例如：

```
save stream s as "c:/my streams/new_modeling.str"
```

等效的 Python 脚本方法将使用 `TaskRunner` 类，例如：

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

基于 IBM SPSS Collaboration and Deployment Services Repository 的操作在旧脚本编制中通过 `retrieve` 和 `store` 命令受支持，例如：

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

在 Python 脚本编制中，可以通过与会话关联的存储库对象来访问等效功能，例如：

```
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)
```

注: 存储库访问要求使用有效存储库连接对会话进行了配置。

流操作

IBM SPSS Modeler 中一些常用的流操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 284: 流操作的旧脚本编制到 Python 脚本编制的映射	
旧脚本编制	Python 脚本编制
create stream <i>DEFAULT_FILENAME</i>	<code>taskrunner.createStream(name, autoConnect, autoManage)</code>
close stream	<code>stream.close()</code>
clear stream	<code>stream.clear()</code>
get stream <i>stream</i>	无等效项
load stream <i>path</i>	无等效项
open stream <i>path</i>	<code>taskrunner.openStreamFromFile(path, autoManage)</code>
save <i>stream</i> as <i>path</i>	<code>taskrunner.saveStreamToFile(stream, path)</code>
retreive stream <i>path</i>	<code>repository.retreiveStream(path, version, label, autoManage)</code>
store <i>stream</i> as <i>path</i>	<code>repository.storeStream(stream, path, label)</code>

模型操作

IBM SPSS Modeler 中一些常用的模型操作命令在 Python 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 Python 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 285: 模型操作的旧脚本编制到 Python 脚本编制的映射	
旧脚本编制	Python 脚本编制
open model <i>path</i>	<code>taskrunner.openModelFromFile(path, autoManage)</code>
save <i>model</i> as <i>path</i>	<code>taskrunner.saveModelToFile(model, path)</code>
retrieve model <i>path</i>	<code>repository.retrieveModel(path, version, label, autoManage)</code>

表 285: 模型操作的旧脚本编制到 <i>Python</i> 脚本编制的映射 (继续)	
旧脚本编制	Python 脚本编制
<code>store model as path</code>	<code>repository.storeModel(model, path, label)</code>

## 文档输出操作

IBM SPSS Modeler 中一些常用的文档输出操作命令在 *Python* 脚本编制中具有等效命令。这可以帮助您将现有 SPSS Modeler 旧脚本转换为 *Python* 脚本，以便在 IBM SPSS Modeler 17 中使用。

表 286: 文档输出操作的旧脚本编制到 <i>Python</i> 脚本编制的映射	
旧脚本编制	Python 脚本编制
<code>open output path</code>	<code>taskrunner.openDocumentFromFile(path, autoManage)</code>
<code>save output as path</code>	<code>taskrunner.saveDocumentToFile(output, path)</code>
<code>retrieve output path</code>	<code>repository.retrieveDocument(path, version, label, autoManage)</code>
<code>store output as path</code>	<code>repository.storeDocument(output, path, label)</code>

## 旧脚本编制与 *Python* 脚本编制之间的其他差异

旧脚本提供对处理 IBM SPSS Modeler 工程的支持。Python 脚本编制当前不提供此支持。

旧脚本编制提供了某种装入状态对象（流和模型的组合）的支持。IBM SPSS Modeler 8.0 后的版本不推荐使用状态对象。Python 脚本编制不支持状态对象。

Python 脚本编制提供了下列附加功能，旧脚本编制中未提供这些功能：

- 类和函数定义
- 错误处理
- 更复杂的输入/输出支持
- 外部模块和第三方模块



# 声明

---

本信息是为在美国提供的产品和服务编写的。该材料可以以其他语言从 IBM 获取。然而，您可能需要拥有产品或产品版本的该语言副本才能进行访问。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并不意味着授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往：

*IBM Director of Licensing*  
*IBM Corporation*  
*North Castle Drive, MD-NC119*  
*Armonk, NY 10504-1785*  
美国

有关双字节 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

*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“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些管辖区域在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

*IBM Director of Licensing*  
*IBM Corporation*  
*North Castle Drive, MD-NC119*  
*Armonk, NY 10504-1785*  
美国

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

所引用的性能数据和客户示例仅作参考用途。实际的性能结果可能会因特定的配置和运营条件而异。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若与实际个人或业务企业相似，纯属巧合。

## 商标

---

IBM、IBM 徽标和 `ibm.com` 是 International Business Machines Corp.，在全球许多司法管辖区的注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。IBM 商标的最新列表可从 Web 上的“Copyright and trademark information”处获得，网址为：[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)。

Adobe、Adobe 徽标、PostScript 和 PostScript 徽标是 Adobe Systems Incorporated 在美国和/或其他国家或地区的注册商标或商标。

Intel、Intel 徽标、Intel Inside、Intel Inside 徽标、Intel Centrino、Intel Centrino 徽标、Celeron、Intel Xeon、Intel SpeedStep、Itanium 和 Pentium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其子公司的商标或注册商标。

## 产品文档的条款和条件

---

只有遵守以下条款和条件才会授予使用这些出版物的许可权。

### 适用性

这些条款和条件是有关 IBM Web 站点使用的任何条款的补充。

### 个人使用

您可以为了个人使用而非商业性使用复制这些出版物，但前提是保留所有专有权声明。未经 IBM 的明确许可，您不得分发、显示这些出版物或其中部分出版物，也不得制作其演绎作品。

### 商业性使用

您仅可在贵公司内部复制、分发和显示这些出版物，但前提是保留所有专有权声明。未经 IBM 的明确许可，您不得制作这些出版物的演绎作品，也不得在贵公司外部复制、分发或显示这些出版物或其部分出版物。

### 权利

除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论明示的还是暗含的。

只要 IBM 认为这些出版物的使用会损害其利益或者 IBM 判定未正确遵守上述指示信息，IBM 将有权撤销此处授予的许可权。

您不得下载、出口或再出口这些信息，除非完全遵守所有适用的法律和法规（包括所有的美国出口法律和法规）。

IBM 对这些出版物的内容不作任何保证。本出版物以“按现状”的基础提供，不附有任何形式的（无论是明示的，还是暗含的）保证，包括但不限于暗含的有关适销性、非侵权以及适用于某特定用途的保证。

# Index

## Special Characters

“IBM SPSS Statistics 导出”节点

属性 [379](#)

“R 变换”节点

属性 [117](#)

“R 构建”节点

节点脚本编制属性 [202](#)

“TWC 导入”源节点

属性 [97](#)

“关联规则”节点

属性 [192](#)

“关联规则”节点块

属性 [296](#)

“分布”节点

属性 [169](#)

“分析”节点

属性 [339](#)

“区分”节点

属性 [111](#)

“匿名化”节点

属性 [131](#)

“变量文件”节点

属性 [99](#)

“合并”节点

属性 [114](#)

“地图可视化”节点

属性 [177](#)

“地理空间”源节点

属性 [92](#)

“均值”节点

属性 [346](#)

“平衡”节点

属性 [106](#)

“扩展变换”节点

属性 [113](#)

“扩展导入”节点

属性 [86](#)

“扩展导出”节点

属性 [368](#)

“扩展模型”节点

节点脚本编制属性 [216](#)

“扩展输出”节点

属性 [342](#)

“数据审核”节点

属性 [340](#)

“样本”节点

属性 [118](#)

“模拟拟合”节点

属性 [351](#)

“模拟生成”节点

属性 [93](#)

“模拟评估”节点

属性 [350](#)

“汇总”节点

属性 [105](#)

“派生”节点

“派生”节点 (*continued*)

属性 [139](#)

“流式时间序列”模型

节点脚本编制属性 [122](#)

“流式时间序列”节点

属性 [128](#)

“直方图”节点

属性 [176](#)

“矩阵”节点

属性 [344](#)

“空间-时间预测”节点

属性 [266](#)

“空间时间限制”节点

属性 [109, 121](#)

“空间时间限制”节点属性 [109](#)

“设为标志”节点

属性 [150](#)

“转置”节点

属性 [157](#)

“追加”节点

属性 [105](#)

“重新投影”节点

属性 [148](#)

两步模型

节点脚本编制属性 [290, 311](#)

中断脚本 [9](#)

主成分分析/因子模型

节点脚本编制属性 [218, 302](#)

二阶 AS 模型

节点脚本编制属性 [291, 311](#)

代码块 [17](#)

传递参数 [17](#)

修改流 [28, 30](#)

最近邻元素模型

节点脚本编制属性 [240](#)

决策列表模型

节点脚本编制属性 [213, 300](#)

函数

对象引用 [405](#)

循环 [407](#)

文字 [405](#)

文档输出操作 [414](#)

条件语句 [407](#)

模型操作 [413](#)

注释 [405](#)

流操作 [413](#)

节点操作 [410](#)

运算符 [406](#)

分区节点

属性 [146](#)

分级节点

属性 [136](#)

列表 [14](#)

创建类 [21](#)

创建节点 [28–30](#)

判别模型

节点脚本编制属性 [214, 301](#)

- 加密密码
  - 添加至脚本 [45](#)
- 单类 SVM 节点
  - 属性 [385](#)
- 历史记录节点
  - 属性 [145](#)
- 参数
  - SuperNodes [399](#)
  - 脚本编制 [13](#)
- 变换节点
  - 属性 [356](#)
- 变量
  - 脚本编制 [13](#)
- 命令行
  - 参数 [57](#)
  - 参数列表 [56, 58, 59](#)
  - 多个参数 [60](#)
  - 脚本编制 [46](#)
  - 运行 IBM SPSS Modeler [55](#)
- 固定文件节点
  - 属性 [88](#)
- 图形板节点
  - 属性 [172](#)
- 图形节点
  - 脚本编制属性 [167](#)
- 图表 [25](#)
- 坐标系重新投影
  - 属性 [148](#)
- 块
  - 节点脚本编制属性 [295](#)
- 填充节点
  - 属性 [143](#)
- 备注 [16](#)
- 多重散点图节点
  - 属性 [180](#)
- 多重集合命令 [61](#)
- 字段
  - 在脚本中关闭 [167](#)
- 字段名
  - 更改大小写 [43](#)
- 字段重排节点
  - 属性 [147](#)
- 字符串
  - 更改大小写 [43](#)
- 字符串函数 [43](#)
- 安全性
  - 加密密码 [45, 58](#)
- 定义属性 [22](#)
- 定义方法 [22](#)
- 定义类 [21](#)
- 定向 Web 节点
  - 属性 [187](#)
- 密码
  - 已编码 [58](#)
  - 添加至脚本 [45](#)
- 导出节点
  - 节点脚本编制属性 [359](#)
- 属性
  - SuperNodes [399](#)
  - 数据库建模节点 [313](#)
  - 流 [65](#)
  - 脚本编制 [61–63, 189, 295, 359](#)
  - 过滤节点 [61](#)
  - 通用脚本编写 [63](#)
- 已生成的模型
  - 脚本编写名称 [401, 403](#)
- 平面文件节点
  - 属性 [369](#)
- 广义线性模型
  - 节点脚本编制属性 [222, 303](#)
- 序列模型
  - 节点脚本编制属性 [263, 309](#)
- 建模节点
  - 节点脚本编制属性 [189](#)
- 异常检测模型
  - 节点脚本编制属性 [189, 295](#)
- 引用节点
  - 查找节点 [26](#)
  - 设置属性 [27](#)
- 循环
  - 在脚本中使用 [43](#)
- 执行流 [25](#)
- 执行脚本 [9](#)
- 执行顺序
  - 用脚本更改 [43](#)
- 报告节点
  - 属性 [348](#)
- 排序节点
  - 属性 [120](#)
- 支持向量机模型
  - 节点脚本编制属性 [271, 309](#)
- 收集节点
  - 属性 [168](#)
- 散点图节点
  - 属性 [181](#)
- 数学方法 [18](#)
- 数据库导出节点
  - 属性 [362](#)
- 数据库建模 [313](#)
- 数据库节点
  - 属性 [79](#)
- 整体节点
  - 属性 [142](#)
- 时间因果模型
  - 节点脚本编制属性 [273](#)
- 时间序列模型
  - 节点脚本编制属性 [278, 286, 310](#)
- 时间散点图节点
  - 属性 [183](#)
- 时间间隔节点
  - 属性 [151](#)
- 服务器
  - 命令行自变量 [58](#)
- 查找节点 [26](#)
- 标识 [17](#)
- 模型
  - 脚本编写名称 [401, 403](#)
- 模型块
  - 脚本编写名称 [401, 403](#)
  - 节点脚本编制属性 [295](#)
- 模型对象
  - 脚本编写名称 [401, 403](#)
- 流
  - 修改 [28](#)
  - 多重集合命令 [61](#)
  - 属性 [65](#)
  - 循环 [5, 6](#)
  - 执行 [25](#)

## 流 (continued)

- 条件执行 [5, 8](#)
- 脚本编制 [1, 25](#)
- 流中的循环 [5, 6](#)
- 流执行顺序
  - 用脚本更改 [43](#)
- 流的条件执行 [5, 8](#)
- 添加属性 [21](#)
- 源节点
  - 属性 [69](#)
- 特征选择模型
  - 应用 [4](#)
  - 脚本编制 [4](#)
  - 节点脚本编制属性 [220, 303](#)
- 独立脚本 [1, 3, 25](#)
- 生成的关键字 [46](#)
- 用户输入节点
  - 属性 [98](#)
- 示例 [18](#)
- 神经网络
  - 节点脚本编制属性 [255, 306](#)
- 神经网络模型
  - 节点脚本编制属性 [252, 306](#)
- 类型节点
  - 属性 [159](#)
- 系统
  - 命令行自变量 [56](#)
- 线性 AS 模型
  - 节点脚本编制属性 [244, 305](#)
- 线性回归模型
  - 节点脚本编制属性 [261, 308](#)
- 线性属性 [243](#)
- 线性支持向量机模型
  - 节点脚本编制属性 [251, 306](#)
- 线性模型
  - 节点脚本编制属性 [243, 305](#)
- 结构化属性 [61](#)
- 统计量节点
  - 属性 [352](#)
- 继承 [22](#)
- 脚本
  - 从文本文件导入 [1](#)
  - 保存 [1](#)
  - 循环 [5, 6](#)
  - 条件执行 [5, 8](#)
  - 迭代关键字 [6](#)
  - 迭代变量 [7](#)
  - 选择字段 [8](#)
- 脚本编写 API
  - 会话参数 [38](#)
  - 元数据 [34](#)
  - 全局值 [41](#)
  - 处理错误 [37](#)
  - 多个流 [41](#)
  - 搜索 [33](#)
  - 流参数 [38](#)
  - 独立脚本 [41](#)
  - 示例 [33](#)
  - 简介 [33](#)
  - 获取目录 [33](#)
  - 访问已生成的对象 [36](#)
  - 超节点参数 [38](#)
- 脚本编制
  - Python 脚本编制 [405–407, 410, 413, 414](#)

## 脚本编制 (continued)

- SuperNode 流 [25](#)
- 上下文 [25](#)
- 与早期版本的兼容性 [46](#)
- 中断 [9](#)
- 从命令行 [46](#)
- 可见循环 [5, 6](#)
- 图形节点 [167](#)
- 图表 [25](#)
- 在超节点中 [4](#)
- 所用缩写 [62](#)
- 执行 [9](#)
- 旧脚本编制 [405–407, 410, 413, 414](#)
- 条件执行 [5, 8](#)
- 概述 [1, 13](#)
- 流 [1, 25](#)
- 流执行顺序 [43](#)
- 特征选择模型 [4](#)
- 独立脚本 [1, 25](#)
- 用户界面 [1, 3, 4](#)
- 语法 [13–18, 20–22](#)
- 超节点脚本 [1, 25](#)
- 输出节点 [339](#)
- 迭代关键字 [6](#)
- 迭代变量 [7](#)
- 选择字段 [8](#)
- 通用属性 [63](#)
- 错误检查 [46](#)
- 自动分类器模型
  - 节点脚本编制属性 [296](#)
- 自动分类器节点
  - 节点脚本编制属性 [195](#)
- 自动数值模型
  - 节点脚本编制属性 [199](#)
- 自动数字模型
  - 节点脚本编制属性 [298](#)
- 自动数据准备
  - 属性 [132](#)
- 自动聚类模型
  - 节点脚本编制属性 [298](#)
- 自动聚类节点
  - 节点脚本编制属性 [197](#)
- 自变量
  - IBM SPSS Analytic Server 存储库连接 [59](#)
  - IBM SPSS Collaboration and Deployment Services Repository 连接 [59](#)
  - 命令文件 [60](#)
  - 服务器连接 [58](#)
  - 系统 [56](#)
- 自学响应模型
  - 节点脚本编制属性 [264, 308](#)
- 节点
  - 信息 [31](#)
  - 删除 [30](#)
  - 取消链接节点 [29](#)
  - 名称引用 [401](#)
  - 在脚本中循环 [43](#)
  - 导入 [30](#)
  - 替换 [30](#)
  - 链接节点 [29](#)
- 节点脚本编制属性
  - 导出节点 [359](#)
  - 建模节点 [189](#)
  - 模型块 [295](#)

- 表内容模型 [47](#)
- 表节点
  - 属性 [354](#)
- 设置全局量节点
  - 属性 [350](#)
- 设置属性 [27](#)
- 访问流执行结果
  - JSON 内容模型 [50](#)
  - XML 内容模型 [48](#)
  - 表内容模型 [47](#)
- 评估节点
  - 属性 [170](#)
- 语句 [16](#)
- 贝叶斯网络模型
  - 节点脚本编制属性 [201](#), [298](#)
- 超节点 [61](#)
- 输出对象
  - 脚本编写名称 [403](#)
- 输出节点
  - 脚本编制属性 [339](#)
- 迁移
  - 一般差异 [405](#)
  - 其他 [414](#)
  - 函数 [405](#)
  - 变量 [408](#)
  - 命令 [405](#)
  - 存储库 [412](#)
  - 属性名 [408](#)
  - 循环 [411](#)
  - 执行流 [412](#)
  - 文件系统 [412](#)
  - 概述 [405](#)
  - 模型类型 [408](#)
  - 清除流, 输出, 和模型管理器 [31](#)
  - 编辑流 [410](#)
  - 脚本编制上下文 [405](#)
  - 节点引用 [409](#)
  - 节点类型 [408](#)
  - 获取属性 [409](#)
  - 设置属性 [409](#)
  - 访问对象 [412](#)
  - 输出类型 [408](#)
- 过滤节点
  - 属性 [144](#)
- 运算 [13](#)
- 迭代关键字
  - 脚本中的循环 [6](#)
- 迭代变量
  - 脚本中的循环 [7](#)
- 选择节点
  - 属性 [120](#)
- 通道参数 [4](#), [61](#), [63](#)
- 遍历节点 [30](#)
- 重排节点
  - 属性 [147](#)
- 重新分类节点
  - 属性 [147](#)
- 重新结构化节点
  - 属性 [149](#)
- 错误检查
  - 脚本编制 [46](#)
- 随机树模型
  - 节点脚本编制属性 [259](#), [308](#)
- 随机森林节点

- 随机森林节点 (*continued*)
  - 属性 [387](#)
- 隐藏变量 [22](#)
- 非 ASCII 字符 [20](#)
- 面向对象 [21](#)
- 高斯混合节点
  - 属性 [381](#), [385](#)

## A

- aggreatenode 属性 [105](#)
- analysisnode 属性 [339](#)
- Analytic Server 源节点
  - 属性 [76](#)
- anomalydetectionnode 属性 [189](#)
- anonymizenode 属性 [131](#)
- appendnode 属性 [105](#)
- applyanomalydetectionnode 属性 [295](#)
- applyapriorinode 属性 [295](#)
- applyassociationrulesnode 属性 [296](#)
- applyautoclassifiernode 属性 [296](#)
- applyautoclusternode 属性 [298](#)
- applyautonumericnode 属性 [298](#)
- applybayesnetnode 属性 [298](#)
- applyc50node 属性 [299](#)
- applycarmanode 属性 [299](#)
- applycartnode 属性 [299](#)
- applychaidnode 属性 [299](#)
- applycoxregnode 属性 [300](#)
- applydecisionlistnode 属性 [300](#)
- applydiscriminantnode 属性 [301](#)
- applyextension 属性 [301](#)
- applyfactornode 属性 [302](#)
- applyfeatureselectionnode 属性 [303](#)
- applygeneralizedlinearnode 属性 [303](#)
- applygle 属性 [304](#)
- applyglmnode 属性 [303](#)
- applygmm 属性 [304](#)
- applykmeansnode 属性 [304](#)
- applyknnnode 属性 [304](#)
- applykohonennode 属性 [305](#)
- applylinearnode 属性 [305](#)
- applylinearnode 属性 [305](#)
- applylogregnode 属性 [305](#)
- applylsvmnode 属性 [306](#)
- applymslogisticnode 属性 [315](#)
- applymsneuralnetworknode 属性 [315](#)
- applymsregressionnode 属性 [315](#)
- applymssequenceclusternode 属性 [315](#)
- applymstimeseriesnode 属性 [315](#)
- applymstreenode 属性 [315](#)
- applynetezabayesnode 属性 [338](#)
- applynetezadectreenode 属性 [338](#)
- applynetezadivclusternode 属性 [338](#)
- applynetezakmeansnode 属性 [338](#)
- applynetezaknnnode 属性 [338](#)
- applynetezalineregressionnode 属性 [338](#)
- applynetezanaivebayesnode 属性 [338](#)
- applynetezapcanode 属性 [338](#)
- applynetezaregtreenode 属性 [338](#)
- applyneuralnetnode 属性 [306](#)
- applyneuralnetworknode 属性 [306](#)
- applyocsvm 属性 [307](#)
- applyoraabnnode 属性 [324](#)



applyoradecisiontreenode 属性 [324](#)  
applyorakmeansnode 属性 [324](#)  
applyoranbnode 属性 [324](#)  
applyoranmfnode 属性 [324](#)  
applyoraoclusternode 属性 [324](#)  
applyorasvmnode 属性 [324](#)  
applyquestnode 属性 [307](#)  
applyr 属性 [308](#)  
applyrandomtrees 属性 [308](#)  
applyregressionnode 属性 [308](#)  
applyselflearningnode 属性 [308](#)  
applysequencenode 属性 [309](#)  
applystpnode 属性 [309](#)  
applysvmnode 属性 [309](#)  
applytcmnode 属性 [309](#)  
applytimeseriesnode 属性 [310](#)  
applytreeas 属性 [310](#)  
applyts 属性 [310](#)  
applytwestepAS 属性 [311](#)  
applytwestepnode 属性 [311](#)  
applyxgboostlinearnode 属性 [311](#)  
applyxgboosttreenode 属性 [311](#)  
Apriori 模型  
    节点脚本编制属性 [191](#), [295](#)  
apriorinode 属性 [191](#)  
AS“时间间隔”节点  
    属性 [136](#)  
asexport 属性 [359](#)  
asimport 属性 [76](#)  
associationrulesnode 属性 [192](#)  
astimeintervalsnode 属性 [136](#)  
autoclassifiernode 属性 [195](#)  
autoclusternode 属性 [197](#)  
autodataprepnode 属性 [132](#)  
autonumericnode 属性 [199](#)

## B

balancenode 属性 [106](#)  
bayesnet 属性 [201](#)  
binningnode 属性 [136](#)  
buildr 属性 [202](#)

## C

C&R 树模型  
    节点脚本编制属性 [205](#), [299](#)  
C5.0 模型  
    节点脚本编制属性 [203](#), [299](#)  
c50node 属性 [203](#)  
CARMA 模型  
    节点脚本编制属性 [204](#), [299](#)  
carmanode 属性 [204](#)  
cartnode 属性 [205](#)  
CHAID 模型  
    节点脚本编制属性 [208](#), [299](#)  
chaidnode 属性 [208](#)  
clear generated palette 命令 [46](#)  
CLEM  
    脚本编制 [1](#)  
cognosimport 节点属性 [77](#)  
collectionnode 属性 [168](#)  
Cox 回归模型

Cox 回归模型 (*continued*)  
    节点脚本编制属性 [210](#), [300](#)  
coxregnode 属性 [210](#)  
CPLEX Optimization 节点  
    属性 [107](#)  
cplexoptnode 属性 [107](#)

## D

Data Collection 导出节点  
    属性 [366](#)  
Data Collection 源节点  
    属性 [81](#)  
dataauditnode 属性 [340](#)  
databaseexportnode 属性 [362](#)  
databasenode 属性 [79](#)  
datacollectionexportnode 属性 [366](#)  
datacollectionimportnode 属性 [81](#)  
decisionlist 属性 [213](#)  
derive\_stbnode  
    属性 [109](#)  
derivenode 属性 [139](#)  
directedwebnode 属性 [187](#)  
discriminantnode 属性 [214](#)  
distinctnode 属性 [111](#)  
distributionnode 属性 [169](#)

## E

E-Plot 节点  
    属性 [185](#)  
ensemblenode 属性 [142](#)  
eplotnode 属性 [185](#)  
evaluationnode 属性 [170](#)  
Excel 导出节点  
    属性 [367](#), [369](#)  
Excel 源节点  
    属性 [84](#)  
excelexportnode 属性 [367](#), [369](#)  
excelimportnode 属性 [84](#)  
exportModelToFile [36](#)  
extensionexportnode 属性 [368](#)  
extensionimportnode 属性 [86](#)  
extensionmodelnode 属性 [216](#)  
extensionoutputnode 属性 [342](#)  
extensionprocessnode 属性 [113](#)

## F

factornode 属性 [218](#)  
featureselectionnode 属性 [4](#), [220](#)  
fillernode 属性 [143](#)  
filternode 属性 [144](#)  
fixedfilenode 属性 [88](#)  
flags  
    命令行自变量 [55](#)  
    组合多个标志 [60](#)  
flatfilenode 属性 [369](#)  
for 命令 [43](#)

## G

genlinnode 属性 [222](#)

gle 属性 [231](#)  
GLE 模型  
节点脚本编制属性 [231](#), [304](#)  
GLMM 模型  
节点脚本编制属性 [227](#), [303](#)  
glmmnode 属性 [227](#)  
gmm 属性 [381](#), [385](#)  
graphboardnode 属性 [172](#)  
gsdata\_import 节点属性 [92](#)

## H

HDBSCAN 节点  
属性 [381](#)  
hdbscannode 属性 [381](#)  
hdbscannugget 属性 [312](#)  
histogramnode 属性 [176](#)  
historynode 属性 [145](#)

## I

IBM Cognos TM1 源节点  
属性 [95](#), [96](#)  
IBM Cognos 源节点  
属性 [77](#)  
IBM SPSS Analytic Server 存储库  
命令行自变量 [59](#)  
IBM SPSS Collaboration and Deployment Services  
Repository  
命令行自变量 [59](#)  
脚本编制 [43](#)  
IBM SPSS Modeler  
从命令行运行 [55](#)  
IBM SPSS Statistics 变换节点  
属性 [377](#)  
IBM SPSS Statistics 模型  
节点脚本编制属性 [378](#)  
IBM SPSS Statistics 源节点  
属性 [377](#)  
IBM SPSS Statistics 输出节点  
属性 [378](#)  
Isotonic-AS 节点  
属性 [395](#)  
isotonicasnode 属性 [395](#)

## J

JSON 内容模型 [50](#)  
JSON 源节点  
属性 [92](#)  
jsonimportnode 属性 [92](#)  
Jython [13](#)

## K

K 均值 AS 模型  
节点脚本编制属性 [239](#), [395](#)  
K-Means 模型  
节点脚本编制属性 [238](#), [304](#)  
KDE 建模节点  
属性 [383](#)  
KDE 模型  
节点脚本编制属性 [312](#)

KDE 模拟节点  
属性 [343](#), [384](#)  
kdeapply 属性 [312](#)  
kdeexport 属性 [343](#), [384](#)  
kdemodel 属性 [383](#)  
kmeansasnode 属性 [239](#), [395](#)  
kmeansnode 属性 [238](#)  
KNN 模型  
节点脚本编制属性 [304](#)  
knnnode 属性 [240](#)  
kohonen 模型  
节点脚本编制属性 [241](#)  
Kohonen 模型  
节点脚本编制属性 [305](#)  
kohonennode 属性 [241](#)

## L

linear-AS 属性 [244](#)  
Logistic 回归模型  
节点脚本编制属性 [246](#), [305](#)  
logregnode 属性 [246](#)  
lowertoupper 函数 [43](#)  
LSVM 模型  
节点脚本编制属性 [251](#)  
lsvmnode 属性 [251](#)

## M

mapvisualization 属性 [177](#)  
matrixnode 属性 [344](#)  
meansnode 属性 [346](#)  
mergenode 属性 [114](#)  
Microsoft 模型  
节点脚本编制属性 [313](#), [315](#)  
MS Logistic 回归  
节点脚本编制属性 [313](#), [315](#)  
MS 决策树  
节点脚本编制属性 [313](#), [315](#)  
MS 序列聚类  
节点脚本编制属性 [315](#)  
MS 时间序列  
节点脚本编制属性 [315](#)  
MS 神经网络  
节点脚本编制属性 [313](#), [315](#)  
MS 线性回归  
节点脚本编制属性 [313](#), [315](#)  
msassocnode 属性 [313](#)  
msbayesnode 属性 [313](#)  
msclusternode 属性 [313](#)  
mslogisticnode 属性 [313](#)  
msneuralnetworknode 属性 [313](#)  
msregressionnode 属性 [313](#)  
mssequenceclusternode 属性 [313](#)  
mstimeseriesnode 属性 [313](#)  
mstreenode 属性 [313](#)  
MultiLayerPerceptron-AS 节点  
属性 [396](#)  
multilayerperceptronnode 属性 [396](#)  
multiplotnode 属性 [180](#)



## N

netezza GLM 属性 [325](#)  
Nettezza K-Means 模型  
节点脚本编制属性 [325, 338](#)  
Nettezza KNN 模型  
节点脚本编制属性 [325, 338](#)  
Nettezza PCA 模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 决策树模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 分裂式聚类模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 回归树模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 广义线性模型  
节点脚本编制属性 [325](#)  
Nettezza 时间序列模型  
节点脚本编制属性 [325](#)  
Nettezza 朴素贝叶斯模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 模型  
节点脚本编制属性 [325](#)  
Nettezza 线性回归模型  
节点脚本编制属性 [325, 338](#)  
Nettezza 贝叶斯网络模型  
节点脚本编制属性 [325, 338](#)  
netezzabayesnode 属性 [325](#)  
netezzadectreenode 属性 [325](#)  
netezzadivclusternode 属性 [325](#)  
netezzakmeansnode 属性 [325](#)  
netezzaknnnode 属性 [325](#)  
netezzalinereregressionnode 属性 [325](#)  
netezzanaivebayesnode 属性 [325](#)  
netezzapcanode 属性 [325](#)  
netezzaregtreenode 属性 [325](#)  
netezzatimeseriesnode 属性 [325](#)  
neuralnetnode 属性 [252](#)  
neuralnetworknode 属性 [255](#)  
numericpredictornode 属性 [199](#)

## O

ocsvmnode 属性 [385](#)  
oraabnnode 属性 [317](#)  
oraainode 属性 [317](#)  
oraapriorinode 属性 [317](#)  
Oracle AI 模型  
节点脚本编制属性 [317](#)  
Oracle Apriori 模型  
节点脚本编制属性 [317, 324](#)  
Oracle KMeans 模型  
节点脚本编制属性 [317, 324](#)  
Oracle MDL 模型  
节点脚本编制属性 [317, 324](#)  
Oracle NMF 模型  
节点脚本编制属性 [317, 324](#)  
Oracle O-Cluster  
节点脚本编制属性 [317, 324](#)  
Oracle 决策树模型  
节点脚本编制属性 [317, 324](#)  
Oracle 广义线性模型  
节点脚本编制属性 [317](#)  
Oracle 支持向量机模型

Oracle 支持向量机模型 (*continued*)  
节点脚本编制属性 [317, 324](#)  
Oracle 朴素贝叶斯模型  
节点脚本编制属性 [317, 324](#)  
Oracle 模型  
节点脚本编制属性 [317](#)  
Oracle 自适应贝叶斯模型  
节点脚本编制属性 [317, 324](#)  
oradecisiontreenode 属性 [317](#)  
oraglmnode 属性 [317](#)  
orakmeansnode 属性 [317](#)  
oramdlnode 属性 [317](#)  
oranbnnode 属性 [317](#)  
oranmfnode 属性 [317](#)  
oraoclusternode 属性 [317](#)  
orasvmnode 属性 [317](#)  
outputfilenode 属性 [369](#)

## P

partitionnode 属性 [146](#)  
PCA 模型  
节点脚本编制属性 [218, 302](#)  
plotnode 属性 [181](#)  
Python  
脚本编制 [13](#)  
Python 模型  
节点脚本编制属性 [307, 311, 312](#)  
高斯混合节点脚本编写属性 [304](#)

## Q

QUEST 模型  
节点脚本编制属性 [257, 307](#)  
questnode 属性 [257](#)

## R

R 输出节点  
属性 [349](#)  
randomtrees 属性 [259](#)  
reclassifynode 属性 [147](#)  
regressionnode 属性 [261](#)  
reordernode 属性 [147](#)  
reportnode 属性 [348](#)  
reprojectnode 属性 [148](#)  
restructurenode 属性 [149](#)  
retrieve 命令 [43](#)  
RFM “汇总”节点  
属性 [116](#)  
RFM 分析节点  
属性 [149](#)  
rfmaggregatenode 属性 [116](#)  
rfmanalysisnode 属性 [149](#)  
rfnode 属性 [387](#)  
routputnode 属性 [349](#)  
Rprocessnode 属性 [117](#)

## S

samplenode 属性 [118](#)  
SAS 导出节点  
属性 [370](#)

- SAS 源节点
  - 属性 [92](#)
- sasexportnode 属性 [370](#)
- sasimportnode 属性 [92](#)
- selectnode 属性 [120](#)
- sequencenode 属性 [263](#)
- setglobalsnode 属性 [350](#)
- settoflagnode 属性 [150](#)
- simevalnode 属性 [350](#)
- simfitnode 属性 [351](#)
- simgennode 属性 [93](#)
- SLRM 模型
  - 节点脚本编制属性 [264](#), [308](#)
- slrmnode 属性 [264](#)
- SMOTE 节点
  - 属性 [388](#)
- smotenode 属性 [388](#)
- sortnode 属性 [120](#)
- spacetimeboxes 属性 [121](#)
- statisticsexportnode 属性 [379](#)
- statisticsimportnode 属性 [4](#), [377](#)
- statisticsmodelnode 属性 [378](#)
- statisticsnode 属性 [352](#)
- statisticsoutputnode 属性 [378](#)
- statisticstransformnode 属性 [377](#)
- store 命令 [43](#)
- STP 节点
  - 属性 [266](#)
- STP 节点块
  - 属性 [309](#)
- stpnode 属性 [266](#)
- stream.nodes 属性 [43](#)
- streamingtimeseries 属性 [122](#)
- streamingts 属性 [128](#)
- SuperNode
  - 流 [25](#)
- SuperNodes
  - 参数 [399](#)
  - 属性 [399](#)
  - 流 [25](#)
  - 脚本 [1](#), [4](#), [5](#), [25](#)
  - 脚本编制 [399](#)
  - 设置属性 [399](#)
- SVM 模型
  - 节点脚本编制属性 [271](#)
- svmnode 属性 [271](#)

## T

- t-SNE 节点
  - 属性 [185](#), [389](#)
- tablenode 属性 [354](#)
- tcm 模型
  - 节点脚本编制属性 [309](#)
- tcmnode 属性 [273](#)
- timeintervalsnode 属性 [151](#)
- timeplotnode 属性 [183](#)
- timeseriesnode 属性 [286](#)
- tm1import 节点属性 [96](#)
- tm1odataimport 节点属性 [95](#)
- transformnode 属性 [356](#)
- transposenode 属性 [157](#)
- Tree-AS 模型
  - 节点脚本编制属性 [288](#), [310](#)

- treeas 属性 [288](#)
- ts 属性 [278](#)
- tsnenode 属性 [185](#), [389](#)
- twcimport 节点属性 [97](#)
- twostepAS 属性 [291](#)
- twostepnode 属性 [290](#)
- typenode 属性 [4](#), [159](#)

## U

- userinputnode 属性 [98](#)

## V

- variablefilenode 属性 [99](#)

## W

- Web 节点
  - 属性 [187](#)
- webnode 属性 [187](#)

## X

- XGBoost Linear 节点
  - 属性 [391](#)
- XGBoost Tree 节点
  - 属性 [392](#)
- XGBoost-AS 节点
  - 属性 [396](#)
- xgboostasnode 属性 [396](#)
- xgboostlinearnode 属性 [391](#)
- xgboosttreenode 属性 [392](#)
- XML 内容模型 [48](#)
- XML 导出节点
  - 属性 [375](#)
- XML 源节点
  - 属性 [104](#)
- xmlexportnode 属性 [375](#)
- xmlimportnode 属性 [104](#)



