

*IBM SPSS Modeler 17
Python Handbuch
für Scripterstellung
und Automatisierung*

IBM

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen“ auf Seite 327 gelesen werden.

Produktinformation

Diese Ausgabe bezieht sich auf Version 17, Release 0, Modifikation 0 von IBM(r) SPSS(r) Modeler und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs *IBM SPSS Modeler 17, Python Scripting and Automation Guide*, herausgegeben von International Business Machines Corporation, USA

© Copyright International Business Machines Corporation 2015

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
TSC Germany
Kst. 2877
Januar 2015

Inhaltsverzeichnis

Kapitel 1. Scripts und die Scriptsprache 1

Scripterstellung - Überblick	1
Scripttypen	1
Stream-Scripts	2
Beispiel für Stream-Script: Trainieren eines neuronalen Netzes	3
Standalone-Scripts	4
Beispiel für Standalone-Script: Speichern und Laden von Modellen	4
Beispiel für Standalone-Script: Generieren eines Merkmalauswahlmodells	5
Superknotenscripts	5
Beispiel für Superknotenscript	6
Verwendung von Schleifen und bedingte Ausführung in Streams	6
Verwendung von Schleifen in Streams	7
Bedingte Ausführung in Streams	11
Ausführen und Unterbrechen von Scripts	12
Suchen und Ersetzen	13

Kapitel 2. Scriptsprache 17

Scriptsprache - Überblick	17
Python und Jython	17
Python-Scripting	18
Operationen	18
Listen	18
Zeichenfolgen	19
Anmerkungen	21
Anweisungssyntax	21
IDs	21
Codeblöcke	22
Übergeben von Argumenten an ein Script	22
Beispiele	22
Mathematische Methoden	23
Verwendung von Nicht-ASCII-Zeichen	25
Objektorientierte Programmierung	26
Definieren einer Klasse	26
Erstellen einer Klasseninstanz	27
Hinzufügen von Attributen zu einer Klasseninstanz	27
Definieren von Klassenattributen und Methoden	27
Ausgeblendete Variablen	28
Vererbung	28

Kapitel 3. Scripting in IBM SPSS Modeler. 29

Scripttypen	29
Streams, Superknotenstreams und Diagramme	29
Streams	29
Superknotenstreams	29
Diagramme	29
Ausführen eines Streams	30
Scriptingkontext	30
Referenzieren vorhandener Knoten	31
Suchen von Knoten	31

Festlegen von Eigenschaften	32
Erstellen von Knoten und Ändern von Streams	33
Erstellen von Knoten	33
Aktivieren und Aufheben von Links für Knoten	34
Importieren, Ersetzen und Löschen von Knoten	35
Traversieren durch Knoten in einem Stream	36
Entfernen von Elementen	37
Abrufen von Informationen zu Knoten	37

Kapitel 4. Scripting-API. 41

Einführung in die Scripting-API	41
Beispiel: Suchen nach Knoten mit einem benutzerdefinierten Filter	41
Metadaten: Informationen zu Daten	41
Zugriff auf generierte Objekte	44
Fehlerbehandlung	46
Stream-, Sitzungs- und Superknotenparameter	46
Globale Werte	50
Arbeiten mit mehreren Streams: Standalone-Scripts	51

Kapitel 5. Tipps zur Scripterstellung . . . 53

Ändern der Streamausführung	53
Verwendung von Schleifen bei Knoten	53
Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository	54
Erstellen eines verschlüsselten Kennworts	55
Scriptprüfung	56
Scripts in der Befehlszeile	56
Kompatibilität zu früheren Versionen	56
Zugriff auf Streamausführungsergebnisse	56
Tabelleninhaltsmodell	57
XML-Inhaltsmodell	59
JSON-Inhaltsmodell	60
Inhaltsmodell für Spaltenstatistikdaten und Inhaltsmodell für paarweise Statistikdaten	62

Kapitel 6. Befehlszeilenargumente . . . 65

Aufrufen der Software	65
Verwenden von Befehlszeilenargumenten	65
Systemargumente	66
Parameterargumente	67
Argumente zum Herstellen einer Serververbindung	68
Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung	69
Argumente zum Herstellen einer IBM SPSS Analytic Server-Verbindung	70
Kombinieren mehrerer Argumente	70

Kapitel 7. Eigenschaftsreferenz 71

Überblick über die Eigenschaften	71
Syntax für Eigenschaften	71
Beispiele für Knoten- und Streameigenschaften	73
Überblick über Knoteneigenschaften	73

Allgemeine Knoteneigenschaften	74
--	----

Kapitel 8. Streameigenschaften 75

Kapitel 9. Eigenschaften von Quellenknoten 79

Allgemeine Eigenschaften von Quellenknoten	79
Eigenschaften von "asimport"	83
Eigenschaften des Knotens "cognosimport"	83
Eigenschaften von "databasenode"	85
Eigenschaften von "datacollectionimportnode"	87
Eigenschaften von "excelimportnode"	90
Eigenschaften von "evimportnode"	91
Eigenschaften von "fixedfilenode"	91
Knoteneigenschaften von "gsdata_import"	94
Eigenschaften von "sasimportnode"	94
Eigenschaften von "simgennode"	95
Eigenschaften von "statisticsimportnode"	98
Eigenschaften des Knotens "tmlimport"	98
Eigenschaften von "userinputnode"	98
Eigenschaften von "variablefilenode"	99
Eigenschaften von "xmlimportnode"	102
Eigenschaften von "dataviewimport"	103

Kapitel 10. Eigenschaften von Datensatzoperationsknoten 105

Eigenschaften von "appendnode"	105
Eigenschaften von "aggregatenode"	105
Eigenschaften von "balancenode"	106
Eigenschaften von "derive_stbnode"	107
Eigenschaften von "distinctnode"	109
Eigenschaften von "mergenode"	110
Eigenschaften von "rfmaggregatenode"	112
Eigenschaften von "Rprocessnode"	114
Eigenschaften von "samplennode"	114
Eigenschaften von "selectnode"	116
Eigenschaften von "sortnode"	116
Eigenschaften von "streamingts"	117

Kapitel 11. Eigenschaften von Feldoperationsknoten. 121

Eigenschaften von "anonymizenode"	121
Eigenschaften von "autodatapreprenode"	122
Eigenschaften von "astimeintervalsnode"	125
Eigenschaften von "binningnode"	125
Eigenschaften von "derivenode"	128
Eigenschaften von "ensemblennode"	130
Eigenschaften von "fillernode"	131
Eigenschaften von "filternode"	132
Eigenschaften von "historynode"	133
Eigenschaften von "partitionnode"	134
Eigenschaften von "reclassifynode"	135
Eigenschaften von "reordernode"	136
Eigenschaften von "reprojectnode"	136
Eigenschaften von "restructurenode"	137
Eigenschaften von "rfmanalysisnode"	137
Eigenschaften von "settoflagnode"	139
Eigenschaften von "statisticstransformnode"	139
Eigenschaften von "timeintervalsnode"	140

Eigenschaften von "transposenode"	144
Eigenschaften von "typenode"	145

Kapitel 12. Eigenschaften von Diagrammknoten 151

Allgemeine Eigenschaften von Diagrammknoten	151
Eigenschaften von "collectionnode"	152
Eigenschaften von "distributionnode"	153
Eigenschaften von "evaluationnode"	154
Eigenschaften von "graphboardnode"	156
Eigenschaften von "histogramnode"	158
Eigenschaften von "multiplotnode"	159
Eigenschaften von "plotnode"	160
Eigenschaften von "timeplotnode"	162
Eigenschaften von "webnode"	164

Kapitel 13. Eigenschaften von Modellierungsknoten. 167

Allgemeine Eigenschaften von Modellierungsknoten	167
Eigenschaften von "anomalydetectionnode"	167
Eigenschaften von "apriorinode"	169
Eigenschaften von "associationrulesnode"	170
Eigenschaften von "autoclassifiernode"	173
Festlegen der Algorithmeigenschaften	174
Eigenschaften von "autoclusternode"	175
Eigenschaften von "autonumericnode"	177
Eigenschaften von "bayesnetnode"	178
Eigenschaften von "buildr"	179
Eigenschaften von "c50node"	180
Eigenschaften von "carmanode"	181
Eigenschaften von "cartnode"	183
Eigenschaften von "chaidnode"	185
Eigenschaften von "coxregnode"	187
Eigenschaften von "decisionlistnode"	189
Eigenschaften von "discriminantnode"	190
Eigenschaften von "factornode"	192
Eigenschaften von "featureselectionnode"	193
Eigenschaften von "genlinnode"	195
Eigenschaften von "glimmnode"	199
Eigenschaften von "kmeansnode"	202
Eigenschaften von "knnnode"	203
Eigenschaften von "kohonennode"	205
Eigenschaften von "linearnode"	206
Eigenschaften von "logregnode"	207
Eigenschaften von "neuralnetnode"	212
Eigenschaften von "neuralnetworknode"	214
Eigenschaften von "questnode"	216
Eigenschaften von "regressionnode"	218
Eigenschaften von "sequencenode"	220
Eigenschaften von "slrmnode"	221
Eigenschaften von "statisticsmodellenode"	222
Eigenschaften von "stpnode"	222
Eigenschaften von "svmnode"	227
Eigenschaften von "tcmnode"	228
Eigenschaften von "timeseriesnode"	231
Eigenschaften von "twostepnode"	233
Eigenschaften von "twostepAS"	234

Kapitel 14. Eigenschaften von Modellnuggetknoten 237

Eigenschaften von "applyanomalydetectionnode"	237
Eigenschaften von "applyapriorinode"	237
Eigenschaften von "applyautoclassifiernode"	238
Eigenschaften von "applyautoclusternode"	238
Eigenschaften von "applyautonumericnode"	238
Eigenschaften von "applybayesnetnode"	239
Eigenschaften von "applyc50node"	239
Eigenschaften von "applycarmanode"	239
Eigenschaften von "applycartnode"	239
Eigenschaften von "applychaidnode"	240
Eigenschaften von "applycoxregnode"	240
Eigenschaften von "applydecisionlistnode"	241
Eigenschaften von "applydiscriminantnode"	241
Eigenschaften von "applyfactornode"	241
Eigenschaften von "applyfeatureselectionnode"	241
Eigenschaften von "applygeneralizedlinearnode"	242
Eigenschaften von "applyglmnode"	242
Eigenschaften von "applyassociationrulesnode"	242
Eigenschaften von "applykmeansnode"	243
Eigenschaften von "applyknnnode"	243
Eigenschaften von "applykohonenode"	243
Eigenschaften von "applylinearnode"	243
Eigenschaften von "applylogregnode"	244
Eigenschaften von "applyneuralnetnode"	244
Eigenschaften von "applyneuralnetworknode"	245
Eigenschaften von "applyquestnode"	245
Eigenschaften von "applyr"	245
Eigenschaften von "applyregressionnode"	246
Eigenschaften von "applyselflearningnode"	246
Eigenschaften von "applysequencenode"	246
Eigenschaften von "applysvmnode"	246
Eigenschaften von "applystpnode"	247
Eigenschaften von "applytimeseriesnode"	247
Eigenschaften von "applytwostepnode"	247
Eigenschaften von "applytwostepAS"	247

Kapitel 15. Eigenschaften von Datenbankmodellierungsknoten 249

Knoteneigenschaften für Microsoft-Modellierung	249
Eigenschaften von Microsoft-Modellierungsknoten	249
Eigenschaften von Microsoft-Modellnuggets	251
Knoteneigenschaften für Oracle-Modellierung	253
Eigenschaften von Oracle-Modellierungsknoten	253
Eigenschaften von Oracle-Modellnuggets	259
Knoteneigenschaften für IBM DB2-Modellierung	260
Eigenschaften von IBM DB2-Modellierungsknoten	260
Eigenschaften von IBM DB2-Modellnuggets	265
Knoteneigenschaften für IBM Netezza Analytics-Modellierung	266
Eigenschaften von Netezza-Modellierungsknoten	266
Eigenschaften von Netezza-Modellnuggets	276

Kapitel 16. Eigenschaften von Ausgabeknoten 279

Eigenschaften von "analysisnode"	279
----------------------------------	-----

Eigenschaften von "dataauditnode"	280
Eigenschaften von "matrixnode"	282
Eigenschaften von "meansnode"	284
Eigenschaften von "reportnode"	285
Eigenschaften von "routputnode"	286
Eigenschaften von "setglobalsnode"	287
Eigenschaften von "simevalnode"	287
Eigenschaften von "simfitnode"	288
Eigenschaften von "statisticsnode"	289
Eigenschaften von "statisticsoutputnode"	290
Eigenschaften von "tablnode"	290
Eigenschaften von "transformnode"	292

Kapitel 17. Eigenschaften von Exportknoten 295

Allgemeine Eigenschaften von Exportknoten	295
Eigenschaften von "asexport"	295
Eigenschaften von "cognosexportnode"	295
Eigenschaften von "databaseexportnode"	297
Eigenschaften von "datacollectionexportnode"	301
Eigenschaften von "excelexportnode"	301
Eigenschaften von "outputfilenode"	302
Eigenschaften von "sasexportnode"	303
Eigenschaften von "statisticsexportnode"	304
Eigenschaften des Knotens "tmlexport"	304
Eigenschaften von "xmlexportnode"	304

Kapitel 18. IBM SPSS Statistics-Knoteneigenschaften 307

Eigenschaften von "statisticsimportnode"	307
Eigenschaften von "statisticstransformnode"	307
Eigenschaften von "statisticsmodelnode"	308
Eigenschaften von "statisticsoutputnode"	308
Eigenschaften von "statisticsexportnode"	309

Kapitel 19. Superknoteneigenschaften 311

Anhang A. Knotennamenreferenz . . . 313

Modellnuggetnamen	313
Vermeidung doppelter Modellnamen	315
Namen der Ausgabetypen	315

Anhang B. Migration von traditionellem Scripting zu Python-Scripting . . 317

Übersicht über die Migration traditioneller Scripts	317
Allgemeine Unterschiede	317
Scriptingkontext	317
Befehle und Funktionen	317
Literale und Kommentare	318
Operatoren	318
Bedingte Befehle und Schleifenbefehle	319
Variablen	320
Knoten-, Ausgabe- und Modelltypen	320
Eigenschaftsnamen	320
Knotenreferenzen	320
Abrufen und Festlegen von Eigenschaften	321
Bearbeiten von Streams	321
Knotenoperationen	322
Verwendung von Schleifen	323

Streams ausführen.	323
Zugriff auf Objekte über das Dateisystem und das Repository	324
Streamoperationen.	325
Modelloperationen	325
Dokumentaushabeoperationen.	325
Weitere Unterschiede zwischen traditionellem Scripting und Python-Scripting	326

Bemerkungen	327
Marken	328
Index	329

Kapitel 1. Scripts und die Scriptsprache

Scripterstellung - Überblick

Die Scripterstellung in IBM® SPSS Modeler ist ein leistungsstarkes Tool, mit dem Prozesse in der Benutzerschnittstelle automatisiert werden. Scripts können dieselben Arten von Aktionen durchführen, die Sie mit einer Maus oder einer Tastatur durchführen. So können Sie Aufgaben automatisieren, die bei einer manuellen Durchführung sehr viele Wiederholungen verlangen oder sehr viel Zeit beanspruchen.

Scripts können zu folgenden Zwecken verwendet werden:

- Eine bestimmte Reihenfolge für die Knotenausführung in einem Stream erzwingen.
- Die Eigenschaften von Knoten festlegen und Ableitungen durchführen, indem Sie ein Subset von CLEM (Control Language for Expression Manipulation) verwenden.
- Eine automatische Abfolge von Aktionen festlegen, für die normalerweise Benutzeraktivitäten erforderlich sind. So können Sie beispielsweise ein Modell erstellen und dieses anschließend testen.
- Komplexe Prozesse einrichten, für die häufige Interventionen des Benutzers notwendig sind, wie dies beispielsweise bei Kreuzvalidierungen der Fall ist, bei denen ein Modell wiederholt generiert und getestet werden muss.
- Prozesse einrichten, mit denen Streams bearbeitet werden. Sie können zum Beispiel einen Modelltrainings-Stream ausführen und automatisch den entsprechenden Modelltest-Stream erstellen.

In diesem Kapitel finden Sie allgemeine Beschreibungen und Beispiele für Scripts auf der Streamebene, Standalone-Scripts und Scripts innerhalb von Superknoten auf der IBM SPSS Modeler-Benutzerschnittstelle. Weitere Informationen zu Scriptsprache, Syntax und Befehlen finden Sie in den nachfolgenden Kapiteln.

Hinweis: Sie können keine Scripts importieren und ausführen, die in IBM SPSS Statistics innerhalb von IBM SPSS Modeler erstellt wurden.

Scripttypen

IBM SPSS Modeler verwendet drei Scripttypen:

- **Stream-Scripts** werden als Streameigenschaft gespeichert und daher zusammen mit einem bestimmten Stream gespeichert und geladen. Beispielsweise können Sie ein Stream-Script schreiben, das das Trainieren und Anwenden eines Modellnuggets automatisiert. Außerdem können Sie angeben, dass bei jeder Ausführung eines bestimmten Streams statt des Inhalts des Streamerstellungsbereichs das Script ausgeführt werden soll.
- **Standalone-Scripts** sind mit keinem bestimmten Stream verknüpft und werden in externen Textdateien gespeichert. Mit einem Standalone-Script können beispielsweise mehrere Streams gemeinsam bearbeitet werden.
- **Superknotenscripts** werden als Streameigenschaft von Superknoten gespeichert. Superknotenscripts stehen nur in Endsuperknoten zur Verfügung. Mit Superknotenscripts kann die Ausführungssequenz der Superknoteninhalte gesteuert werden. Bei Superknoten, bei denen es sich nicht um Endknoten handelt (also Quellen- oder Prozessknoten), können Sie Eigenschaften für den Superknoten bzw. die Knoten, die er enthält, direkt im Stream-Script definieren.

Stream-Scripts

Mit Scripts können in einem bestimmten Stream enthaltene Operationen angepasst und zusammen mit dem Stream gespeichert werden. Stream-Scripts können verwendet werden, um eine bestimmte Ausführungsreihenfolge der in einem Stream enthaltenen Endknoten vorzugeben. Die Bearbeitung des mit dem aktuellen Stream gespeicherten Scripts erfolgt im Dialogfeld "Script" des Streams.

So können Sie im Dialogfeld "Streameigenschaften" auf die Registerkarte für das Stream-Script zugreifen:

1. Wählen Sie im Menü "Extras" folgende Optionsfolge aus:

Streameigenschaften > Ausführung

2. Klicken Sie auf die Registerkarte **Ausführung**, um mit den Scripts des aktuellen Streams zu arbeiten.

Mit den Symbolleistenymbolen oben im Dialogfeld "script" des Streams können Sie folgende Operationen durchführen:

- Inhalte eines bereits vorhandenen Standalone-Scripts in das Fenster importieren.
- Script als Textdatei speichern.
- Script drucken.
- Standardscript anhängen.
- Script bearbeiten (Rückgängig machen, Ausschneiden, Kopieren, Einfügen und andere gängige Editierfunktionen).
- Gesamtes aktuelles Script ausführen.
- Ausgewählte Zeilen eines Scripts ausführen.
- Script während der Ausführung stoppen. (Dieses Symbol ist nur während einer Scriptausführung aktiviert.)
- Syntax des Scripts überprüfen und etwaige Fehler zur Untersuchung im unteren Fensterbereich des Dialogfelds anzeigen.

Ab Version 16.0 verwendet SPSS Modeler die Scripting-Sprache Python. Alle Versionen davor verwendeten eine spezielle Scripting-Sprache von SPSS Modeler, die jetzt als traditionelles Scripting bezeichnet wird. Wählen Sie je nach Typ des verwendeten Scripts auf der Registerkarte **Ausführung** den Ausführungsmodus **Standard (optionales Script)** und dann entweder **Python** oder **Traditionell** aus.

Außerdem können Sie angeben, ob dieses Script bei Ausführung des Streams ausgeführt werden soll oder nicht. Mit der Option **Dieses Script ausführen** legen Sie fest, dass das Script bei jedem Ausführen des Streams gestartet und die im Script angegebene Ausführungsreihenfolge verwendet werden soll. Diese Einstellung führt zu einer Automatisierung auf Streamebene und sorgt für eine schnellere Modellbildung. In der Standardeinstellung wird das Script allerdings während der Streamausführung ignoriert. Auch wenn Sie die Option **Dieses Script ignorieren** auswählen, können Sie das Script stets direkt über dieses Dialogfeld ausführen.

Der Scripteditor umfasst die folgenden Funktionen zur Unterstützung von Script-Authoring:

- Syntaxhervorhebung; Schlüsselwörter, Literalwerte (wie Zeichenfolgen und Zahlen) und Kommentare werden hervorgehoben.
- Zeilennummerierung.
- Blockabgleich; wenn der Cursor an den Anfang eines Programmblocks gesetzt wird, wird der entsprechende Endblock ebenfalls hervorgehoben.
- Vorschlag für Auto-Vervollständigen.

Die von der Syntaxhervorhebung verwendeten Farben und Textstile können mit den Anzeigevorgaben von IBM SPSS Modeler angepasst werden. Sie können die Anzeigevorgaben aufrufen, indem Sie **Extras > Optionen > Benutzeroptionen** auswählen und auf die Registerkarte **Syntax** klicken.

Eine Liste vorgeschlagener Syntaxvervollständigungen kann aufgerufen werden, indem Sie im Kontextmenü **Automatisch vorschlagen** auswählen oder Strg+Leertaste drücken. Mit den Cursorstasten können Sie sich in der Liste nach oben und unten bewegen. Zum Einfügen des ausgewählten Texts drücken Sie dann die Eingabetaste. Drücken Sie Esc, um die automatische Vervollständigung zu verlassen, ohne den vorhandenen Text zu ändern.

Die Registerkarte **Debug** zeigt Debugnachrichten an und kann verwendet werden, um den Scriptstatus auszuwerten, sobald das Script ausgeführt wurde. Die Registerkarte **Debug** besteht aus einem schreibgeschützten Textbereich und einem einzeiligen Eingabetextfeld. Der Textbereich zeigt Text an, der von den Scripts an die Standardausgabe oder an die Standard-Fehlerausgabe gesendet wird, z. B. über Fehler- nachrichtentext. Das Eingabetextfeld übernimmt die Eingabe des Benutzers. Diese Eingabe wird dann im Kontext des Scripts ausgewertet, das zuletzt im Dialog ausgeführt wurde (so genannter *Scriptingkontext*). Der Textbereich enthält den Befehl und die resultierende Ausgabe, sodass der Benutzer einen Trace der Befehle sehen kann. Das Eingabetextfeld enthält immer die Eingabeaufforderung (--> für traditionelles Scripting).

In den folgenden Fällen wird ein neuer Scriptingkontext erstellt:

- Ein Script wird über die Schaltfläche "Dieses Script ausführen" oder die Schaltfläche "Ausgewählte Zeilen ausführen" ausgeführt.
- Die Scriptsprache wird geändert.

Wenn ein neuer Scriptingkontext erstellt wird, wird der Inhalt des Textbereichs gelöscht.

Anmerkung: Durch die Ausführung eines Streams außerhalb des Scriptbereichs wird der Scriptkontext des Scriptbereichs nicht geändert. Die Werte von Variablen, die als Teil dieser Ausführung erzeugt werden, sind im Scriptdialog nicht sichtbar.

Beispiel für Stream-Script: Trainieren eines neuronalen Netzes

Ein Stream kann dazu genutzt werden, um bei der Ausführung ein neuronales Netzmodell zu trainieren. Um das Modell zu testen, müssen Sie den Modellknoten ausführen, um das Modell in den Stream einzufügen, die entsprechenden Verbindungen herzustellen und einen Analyseknoden auszuführen.

Mit einem IBM SPSS Modeler-Script können Sie das Testen des Modellnuggets nach seiner Erstellung automatisieren. Beispielsweise kann folgendes Script für den Demostream *druglearn.str* (verfügbar im Ordner */Demos/streams/* unter Ihrer IBM SPSS Modeler-Installation) aus dem Dialogfeld "Streameigenschaften" (**Extras > Streameigenschaften > Script**) ausgeführt werden:

```
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([])
```

In der folgenden Auflistung werden die einzelnen Zeilen dieses Scriptbeispiels beschrieben.

- Die erste Zeile definiert eine Variable, die auf den aktuellen Stream verweist.
- In Zeile 2 sucht das Script den neuronalen Netzbuilderknoten.
- In Zeile 3 erstellt das Script eine Liste, in der die Ausführungsergebnisse gespeichert werden können.
- In Zeile 4 wird das neuronale Netzmodellnugget erstellt. Es wird in der Liste gespeichert, die in Zeile 3 definiert wird.
- In Zeile 5 wird ein Modellanwendungsknoten für das Modellnugget erstellt und im Streamerstellungsbereich angeordnet.

- In Zeile 6 wird der Analyseknoden Drug erstellt.
- In Zeile 7 sucht das Script den Typknoden.
- In Zeile 8 stellt das Script eine Verbindung zum in Zeile 5 erstellten Modellanwendungsknoten zwischen dem Typknoden und Analyseknoden her.
- Schließlich wird der Analyseknoden ausgeführt, um den Analysebericht zu erstellen.

Sie können mithilfe eines Scripts einen völlig neuen Stream, ausgehend von einem leeren Erstellungsreich, erstellen und ausführen. Weitere Informationen zur Scriptsprache im Allgemeinen finden Sie in Scriptsprache. Weitere Informationen zu Scriptbefehlen im Besonderen finden Sie in Scriptbefehle.

Standalone-Scripts

Im Dialogfeld "Standalone-Script" wird ein Script erstellt oder bearbeitet, das als Textdatei gespeichert wird. Es zeigt den Namen der Datei und bietet Optionen zum Laden, Speichern, Importieren und Ausführen von Scripts.

So öffnen Sie das Dialogfeld für Standalone-Scripts:

Wählen Sie im Hauptmenü Folgendes:

Extras > Standalone-Script

Für Standalone-Scripts stehen dieselbe Symbolleiste und dieselben Optionen zur Überprüfung der Script-Syntax zur Verfügung wie für Stream-Scripts. Weitere Informationen finden Sie im Thema „Stream-Scripts“ auf Seite 2.

Beispiel für Standalone-Script: Speichern und Laden von Modellen

Standalone-Scripts sind bei der Streambearbeitung hilfreich. Nehmen wir an, Sie besitzen zwei Streams - einen, der ein Modell erstellt, und einen anderen, der das aus dem ersten Stream mit vorhandenen Datenfeldern generierte Regelset anhand von Diagrammen untersucht. Ein Standalone-Script für dieses Szenario könnte wie folgt aussehen:

```
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/16/Demos/"

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

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

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

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

Anmerkung: Weitere Informationen zur Scriptsprache im Allgemeinen finden Sie in *Scriptsprache - Überblick*. Weitere Informationen zu Scriptbefehlen im Besonderen finden Sie in *Scriptbefehle*.

Beispiel für Standalone-Script: Generieren eines Merkmalauswahlmodells

In diesem Beispiel wird ausgehend von einem leeren Erstellungsbereich ein Stream erstellt, der ein Merkmalauswahlmodell generiert, das Modell anwendet und eine Tabelle erstellt, in der die 15 wichtigsten Ziele in Bezug auf das angegebene Ziel aufgeführt werden.

```
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([])
```

Das Script erstellt einen Quellenknoten zum Einlesen der Daten, verwendet einen Typknoten, um die Rolle (Verwendung) des Felds `response_01` auf `Target` zu setzen und erstellt anschließend einen Merkmalauswahlknoten und führt diesen aus. Außerdem verbindet das Script die Knoten und positioniert sie im Streamerstellungsbereich, um ein lesbares Layout zu erstellen. Anschließend wird das resultierende Modellnugget mit einem Tabellenknoten verbunden, in dem die 15 wichtigsten Felder aufgeführt sind, die durch die Eigenschaften `selection_mode` und `top_n` bestimmt wurden. Weitere Informationen finden Sie im Thema „Eigenschaften von `featureselectionnode`“ auf Seite 193.

Superknotenscripts

Mithilfe der Scriptsprache von IBM SPSS Modeler können Sie Scripts innerhalb jedes beliebigen Endsuperknotens erstellen und speichern. Diese Scripts stehen ausschließlich für Endsuperknoten zur Verfügung und werden häufig beim Erstellen von Vorlagenstreams oder zum Erzwingen einer bestimmten Ausführungsreihenfolge für die Superknoteninhalte verwendet. Mit Superknotenscripts können Sie außerdem mehrere Scripts in einem Stream ausführen.

Beispiel: Angenommen, Sie müssen die Ausführungsreihenfolge für einen komplexen Stream angeben und Ihr Superknoten enthält mehrere Knoten, darunter einen Globalwerteknoten, der ausgeführt werden muss, bevor ein neues Feld, das in einem Plotknoten verwendet wird, abgeleitet wird. In diesem Fall können Sie ein Superknotenscript erstellen, das zuerst den Globalwerteknoten ausführt. Die durch diesen Knoten berechneten Werte, wie Durchschnitt oder Standardabweichung, können anschließend bei der Ausführung des Plotknotens verwendet werden.

Innerhalb eines Superknotenscripts können Sie Knoteneigenschaften auf dieselbe Weise angeben wie bei anderen Scripts. Alternativ können Sie die Eigenschaften für einen beliebigen Superknoten oder den darin gekapselten Knoten direkt über ein Stream-Script ändern und definieren. Weitere Informationen finden Sie in Kapitel 19, „Superknoteneigenschaften“, auf Seite 311. Diese Methode funktioniert für Quellen- und Prozesssuperknoten ebenso wie für Endsuperknoten.

Hinweis: Da nur Endsuperknoten ihre eigenen Scripts ausführen können, steht die Registerkarte "Scripts" des Dialogfelds "Superknoten" nur für Endsuperknoten zur Verfügung.

So öffnen Sie das Dialogfeld "Superknotenscript" im Haupterstellungsbereich:

Wählen Sie einen Endsuperknoten im Streamerstellungsbereich aus und wählen Sie folgende Option im Menü "Superknoten":

Superknotenscript...

So öffnen Sie das Dialogfeld "Superknotenscript" im vergrößerten Superknoten-Erstellungsbereich:

Klicken Sie mit der rechten Maustaste auf den Superknoten-Erstellungsbereich und wählen Sie aus dem Kontextmenü folgende Optionen:

Superknotenscript...

Beispiel für Superknotenscript

Das folgende Superknotenscript gibt die Reihenfolge an, in der die Endknoten innerhalb des Superknotens ausgeführt werden sollen. Mit dieser Reihenfolge wird sichergestellt, dass der Globalwerteknoten zuerst ausgeführt wird und somit die von diesem Knoten berechneten Werte bei der Ausführung eines weiteren Knotens verwendet werden können.

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

Verwendung von Schleifen und bedingte Ausführung in Streams

Ab Version 16.0 ermöglicht SPSS Modeler es Ihnen, einige grundlegende Scripts in einem Stream zu erstellen, indem Sie Werte in verschiedenen Dialogfeldern auswählen, statt die Anweisungen direkt in der Scriptsprache schreiben zu müssen. Die beiden Haupttypen von Scripts, die Sie auf diese Weise erstellen können, sind einfache Schleifen sowie eine Möglichkeit zum Ausführen von Knoten, wenn eine Bedingung erfüllt ist.

Sie können Regeln zur Verwendung von Schleifen und für die bedingte Ausführung in einem Stream kombinieren. Sie haben z. B. Daten über den Kfz-Verkauf von Herstellern weltweit. Sie könnten eine Schleife definieren, um die Daten in einem Stream zu verarbeiten, Details nach dem Herstellerland ermitteln und die Daten in unterschiedlichen Diagrammen ausgeben mit Details wie Umsatzvolumen nach Modell, Emissionsstufen nach Hersteller und Motorgröße usw. Wenn Sie nur Daten europäischer Hersteller analysieren wollen, könnten Sie auch Bedingungen zur Schleife hinzufügen, die verhindern, dass Diagramme für amerikanische und asiatische Hersteller erstellt werden.

Anmerkung: Da sowohl die Verwendung von Schleifen als auch die bedingte Ausführung auf Hintergrundscripts basieren, werden sie nur auf einen gesamten Stream bei dessen Ausführung angewendet.

- **Verwendung von Schleifen.** Sie können mit Schleifen sich wiederholende Tasks automatisieren. Dabei könnte z. B. eine bestimmte Anzahl von Knoten zu einem Stream hinzugefügt und jeweils ein Knoten-

parameter einzeln geändert werden. Alternativ könnten Sie die wiederholte Ausführung eines Streams oder einer Verzweigung für eine bestimmte Anzahl von Malen steuern (siehe folgende Beispiele):

- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie die Quelle bei jeder Ausführung.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie bei jeder Ausführung den Wert einer Variablen.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und geben Sie bei jeder Ausführung ein Extrafeld ein.
- Erstellen Sie ein Modell eine bestimmte Anzahl Male und ändern Sie bei jeder Erstellung eine Modelleinstellung.
- **Bedingte Ausführung.** Damit können Sie steuern, wie Endknoten in Abhängigkeit von vordefinierten Bedingungen ausgeführt werden (siehe folgende Beispiele):
 - Steuern Sie, ob ein Knoten ausgeführt wird, in Abhängigkeit davon, ob ein bestimmter Wert "true" oder "false" ist.
 - Definieren Sie, ob Schleifen parallel oder sequenziell ausgeführt werden.

Sowohl die Verwendung von Schleifen als auch die bedingte Ausführung werden auf der Registerkarte **Ausführung** im Dialogfeld **Streameigenschaften** definiert. Alle Knoten, die in bedingten oder Schleifenanforderungen verwendet werden, werden mit einem zusätzlichen Symbol im Streamerstellungsbereich angezeigt, um darauf hinzuweisen, dass sie Teil einer Schleifen- oder einer bedingten Ausführung sind.

Sie können auf drei Arten auf die Registerkarte **Ausführung** zugreifen:

- Über die Menüs oben im Hauptdialogfeld:
 1. Wählen Sie im Menü "Extras" folgende Optionsfolge aus:
Streameigenschaften > Ausführung
 2. Klicken Sie auf die Registerkarte **Ausführung**, um mit den Scripts des aktuellen Streams zu arbeiten.
- Aus einem Stream:
 1. Klicken Sie mit der rechten Maustaste auf einen Knoten und wählen Sie **Verwendung von Schleifen/bedingte Ausführung** aus.
 2. Wählen Sie die entsprechende Option aus dem Untermenü aus.
- Klicken Sie in der Grafiksymbolliste oben im Hauptdialogfeld auf das Symbol für die Streameigenschaften.

Wenn Sie zum ersten Mal Details zur Schleifen- oder zur bedingten Ausführung eingeben, wählen Sie auf der Registerkarte **Ausführung** den Ausführungsmodus **Verwendung von Schleifen/bedingte Ausführung** aus und wählen Sie dann die Unterregisterkarte **Bedingt** oder **Verwendung von Schleifen** aus.

Verwendung von Schleifen in Streams

Mit Schleifen können Sie sich wiederholende Tasks in Streams automatisieren (siehe folgende Beispiele):

- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie die Quelle bei jeder Ausführung.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und ändern Sie bei jeder Ausführung den Wert einer Variablen.
- Führen Sie den Stream eine bestimmte Anzahl Male aus und geben Sie bei jeder Ausführung ein Extrafeld ein.
- Erstellen Sie ein Modell eine bestimmte Anzahl Male und ändern Sie bei jeder Erstellung eine Modelleinstellung.

Sie definieren die zu erfüllenden Bedingungen auf der Unterregisterkarte **Verwendung von Schleifen** der Registerkarte **Ausführung** des Streams. Um die Unterregisterkarte anzuzeigen, wählen Sie den Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aus.

Alle definierten Anforderungen für die Verwendung von Schleifen werden bei der Ausführung des Streams wirksam, wenn der Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aktiviert wurde. Optional können Sie den Script-Code für Ihre Schleifenanforderungen generieren und ihn in den Scripteditor einfügen, indem Sie unten rechts auf der Unterregisterkarte **Verwendung von Schleifen** auf **Einfügen...** klicken. Die Anzeige der Hauptregisterkarte **Ausführung** ändert sich und der Ausführungsmodus **Standard (optionales Script)** wird mit dem Script im oberen Teil der Registerkarte angezeigt. Dies bedeutet, dass Sie eine Schleifenstruktur mithilfe der verschiedenen Optionen im Dialogfeld **Verwendung von Schleifen** definieren können, bevor Sie ein Script generieren, das Sie im Scripteditor weiter anpassen können. Wenn Sie auf **Einfügen...** klicken, werden auch alle Bedingungen für die bedingte Ausführung, die Sie definiert haben, im generierten Script angezeigt.

Wichtig: Die Variablen für die Verwendung von Schleifen, die Sie in einem SPSS Modeler-Stream festlegen, können überschrieben werden, wenn Sie den Stream in einem Job von IBM SPSS Collaboration and Deployment Services ausführen. Der Grund hierfür ist, dass der Jobeditoreintrag von IBM SPSS Collaboration and Deployment Services den SPSS Modeler-Eintrag überschreibt. Wenn Sie z. B. eine Variable für die Verwendung von Schleifen im Stream so festlegen, dass ein unterschiedlicher Ausgabedateiname für jede Schleife erstellt wird, werden die Dateien in SPSS Modeler ordnungsgemäß benannt, jedoch vom festen Eintrag überschrieben, der auf der Registerkarte **Ergebnis** von IBM SPSS Collaboration and Deployment Services Deployment Manager eingegeben wird.

So richten Sie eine Schleife ein:

1. Erstellen Sie einen Iterationsschlüssel, um die Hauptschleifenstruktur zu definieren, die in einem Stream ausgeführt werden soll. Weitere Informationen finden Sie in Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams.
2. Definieren Sie bei Bedarf eine oder mehrere Iterationsvariablen. Weitere Informationen finden Sie in Erstellen einer Iterationsvariablen zur Verwendung von Schleifen in Stream.
3. Die Iterationen und die Variablen, die Sie erstellt haben, werden im Hauptteil der Unterregisterkarte angezeigt. Standardmäßig werden Iterationen in der aufgeführten Reihenfolge ausgeführt. Um eine Iteration in der Liste nach oben oder unten zu verschieben, wählen Sie sie durch Anklicken aus und ändern Sie die Reihenfolge mit dem Auf- oder Abwärtspfeil in der rechten Spalte der Unterregisterkarte.

Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams

Sie verwenden einen Iterationsschlüssel, um die Hauptschleifenstruktur zu definieren, die in einem Stream ausgeführt werden soll. Bei der Analyse von Kfz-Verkäufen könnten Sie z. B. einen Streamparameter namens *Herstellerland* erstellen und diesen als Iterationsschlüssel verwenden. Beim Ausführen des Streams wird dieser Schlüssel während jeder Iteration auf die einzelnen Länderwerte in Ihren Daten gesetzt. Verwenden Sie das Dialogfeld **Iterationsschlüssel definieren**, um den Schlüssel zu definieren.

Zum Öffnen des Dialogfelds wählen Sie entweder die Schaltfläche **Iterationsschlüssel...** unten links auf der Unterregisterkarte **Verwendung von Schleifen** aus oder klicken Sie mit der rechten Maustaste auf einen beliebigen Knoten im Stream und wählen Sie entweder **Verwendung von Schleifen/Bedingte Ausführung** > **Iterationsschlüssel definieren (Felder)** oder **Verwendung von Schleifen/Bedingte Ausführung** > **Iterationsschlüssel definieren (Werte)** aus. Wenn Sie das Dialogfeld vom Stream aus öffnen, sind einige der Felder wie der Knotenname möglicherweise bereits ausgefüllt.

Füllen Sie die folgenden Felder aus, um einen Iterationsschlüssel zu definieren:

Iteration nach. Sie können eine der folgenden Optionen auswählen:

- **Streamparameter - Felder.** Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert eines vorhandenen Streamparameters nacheinander auf jedes angegebene Feld setzt.

- **Streamparameter - Werte.** Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert eines vorhandenen Streamparameters nacheinander auf jeden angegebenen Wert setzt.
- **Knoteneigenschaft - Felder.** Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert einer Knoteneigenschaft nacheinander auf jedes angegebene Feld setzt.
- **Knoteneigenschaft - Werte.** Verwenden Sie diese Option, um eine Schleife zu erstellen, die den Wert einer Knoteneigenschaft nacheinander auf jeden angegebenen Wert setzt.

Festzulegen. Wählen Sie das Element aus, dessen Wert bei jeder Schleifenausführung festgelegt werden soll. Sie können eine der folgenden Optionen auswählen:

- **Parameter.** Nur verfügbar, wenn Sie **Streamparameter - Felder** oder **Streamparameter - Werte** auswählen. Wählen Sie den erforderlichen Parameter aus der Liste der verfügbaren Parameter aus.
- **Knoten.** Nur verfügbar, wenn Sie **Knoteneigenschaft - Felder** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie den Knoten aus, für die Sie eine Schleife definieren wollen. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs-, Ausgabe-, Export- oder Modellanwendungsknoten.
- **Eigenschaft.** Nur verfügbar, wenn Sie **Knoteneigenschaft - Felder** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie die Eigenschaft des Knotens aus der Liste aus.

Zu verwendende Felder. Nur verfügbar, wenn Sie **Streamparameter - Felder** oder **Knoteneigenschaft - Felder** auswählen. Wählen Sie die Felder in einem Knoten aus, um die Iterationswerte anzugeben. Sie können eine der folgenden Optionen auswählen:

- **Knoten.** Nur verfügbar, wenn Sie **Streamparameter - Felder** auswählen. Wählen Sie den Knoten mit den Details aus, für die Sie eine Schleife konfigurieren möchten. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs-, Ausgabe-, Export- oder Modellanwendungsknoten.
- **Feldliste.** Klicken Sie auf die Listenschaltfläche in der rechten Spalte, um das Dialogfeld **Felder auswählen** anzuzeigen, in dem Sie die Felder im Knoten zur Angabe der Iterationsdaten auswählen können. Weitere Informationen finden Sie in „Auswählen von Feldern für Iterationen“ auf Seite 10.

Zu verwendende Werte. Nur verfügbar, wenn Sie **Streamparameter - Werte** oder **Knoteneigenschaft - Werte** auswählen. Wählen Sie die Werte im ausgewählten Feld aus, die als Iterationswerte verwendet werden sollen. Sie können eine der folgenden Optionen auswählen:

- **Knoten.** Nur verfügbar, wenn Sie **Streamparameter - Werte** auswählen. Wählen Sie den Knoten mit den Details aus, für die Sie eine Schleife konfigurieren möchten. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur bestimmte Knotentypen angezeigt werden. Wählen Sie dazu eine der folgenden Kategorien aus: Quellen-, Prozess-, Diagramm-, Modellierungs-, Ausgabe-, Export- oder Modellanwendungsknoten.
- **Feldliste.** Wählen Sie das Feld im Knoten aus, das die Iterationsdaten angeben soll.
- **Werteliste.** Klicken Sie auf die Listenschaltfläche in der rechten Spalte, um das Dialogfeld **Werte auswählen** anzuzeigen, in dem Sie die Werte im Feld zur Angabe der Iterationsdaten auswählen können.

Erstellen einer Iterationsvariablen zur Verwendung von Schleifen in Streams

Sie können Iterationsvariablen verwenden, um die Werte von Streamparametern oder Eigenschaften ausgewählter Knoten in einem Stream bei jeder Schleifenausführung zu ändern. Wenn Ihre Streamschleife z. B. Kfz-Verkaufsdaten analysiert und *Herstellerland* als Iterationsschlüssel verwendet, können Sie eine Diagrammausgabe erhalten, die den Umsatz nach Modell anzeigt, und eine andere Diagrammausgabebegabe, die Abgasemissionen anzeigt. In diesen Fällen könnten Sie Iterationsvariablen erstellen, die neue Titel

für die entsprechenden Diagramme erstellen, z. B. *Schwedische Fahrzeugemissionen* und *Japanische Kfz-Verkäufe nach Modell*. Verwenden Sie das Dialogfeld **Iterationsvariable definieren**, um die erforderlichen Variablen zu definieren.

Zum Öffnen des Dialogfelds wählen Sie entweder die Schaltfläche **Variable hinzufügen...** unten links auf der Unterregisterkarte **Verwendung von Schleifen** aus oder klicken Sie mit der rechten Maustaste auf einen beliebigen Knoten im Stream und wählen Sie **Verwendung von Schleifen/Bedingte Ausführung > Iterationsvariable definieren** aus.

Füllen Sie die folgenden Felder aus, um eine Iterationsvariable zu definieren:

Ändern. Wählen Sie den Typ von Attribut aus, den Sie ändern wollen. Sie können zwischen **Streamparameter** oder **Knoteneigenschaft** wählen.

- Wenn Sie **Streamparameter** auswählen, wählen Sie den erforderlichen Parameter aus und definieren Sie dann mit einer der folgenden Optionen (sofern für Ihren Stream verfügbar), auf welchen Wert dieser Parameter bei jeder Schleifeniteration gesetzt werden soll:
 - **Globale Variable.** Wählen Sie die globale Variable aus, auf die der Streamparameter gesetzt werden soll.
 - **Tabellenausgabezeile.** Um einen Streamparameter auf den Wert in einer Tabellenausgabezeile zu setzen, wählen Sie die Tabelle aus der Liste aus und geben Sie die zu verwendende Zeile und Spalte ein.
 - **Manuell eingeben.** Wählen Sie diese Option aus, wenn Sie manuell einen Wert eingeben wollen, den dieser Parameter in den einzelnen Iterationen annehmen soll. Wenn Sie zur Unterregisterkarte **Verwendung von Schleifen** zurückkehren, wird eine neue Spalte erstellt, in die Sie den erforderlichen Text eingeben können.
- Wenn Sie **Knoteneigenschaft** auswählen, wählen Sie den erforderlichen Knoten und eine seiner Eigenschaften aus und legen Sie dann den Wert fest, der für diese Eigenschaft verwendet werden soll. Definieren Sie den neuen Eigenschaftswert mit einer der folgenden Optionen:
 - **Allein.** Der Eigenschaftswert verwendet den Iterationsschlüsselwert. Weitere Informationen finden Sie in „Erstellen eines Iterationsschlüssels zur Verwendung von Schleifen in Streams“ auf Seite 8.
 - **Als Präfix für Stamm.** Verwendet den Iterationsschlüsselwert als Präfix für den Wert, den Sie in das Feld **Stamm** eingeben.
 - **Als Suffix für Stamm.** Verwendet den Iterationsschlüsselwert als Suffix für den Wert, den Sie in das Feld **Stamm** eingeben.

Wenn Sie die Präfix- oder die Suffix-Option ausgewählt haben, werden Sie aufgefordert, den zusätzlichen Text in das Feld **Stamm** einzugeben. Wenn Ihr Iterationsschlüssel z. B. *Herstellerland* lautet und Sie die Option **Als Präfix für Stamm** auswählen, könnten Sie in dieses Feld - *Verkauf nach Modell* eingeben.

Auswählen von Feldern für Iterationen

Beim Erstellen von Iterationen können Sie über das Dialogfeld **Felder auswählen** ein oder mehrere Felder auswählen.

Sortieren nach: Sie können verfügbare Felder für die Anzeige sortieren, indem Sie eine der folgenden Optionen auswählen:

- **Natürlich:** Zeigt die Felder in der Reihenfolge an, in der im aktuellen Datenstream an den aktuellen Knoten übergeben wurden.
- **Name:** Verwendet eine alphabetische Reihenfolge zum Sortieren der Felder für die Anzeige.
- **Typ:** Zeigt Felder sortiert nach ihrem Messniveau an. Diese Option ist hilfreich bei der Auswahl von Feldern mit einem bestimmten Messniveau.

Wählen Sie Felder einzeln aus der Liste aus oder wählen Sie bei gedrückter Umschalttaste oder bei gedrückter Steuertaste mehrere Felder aus. Sie können auch die Schaltflächen unter der Liste verwenden, um Gruppen von Feldern basierend auf deren Messniveau auszuwählen oder um alle Felder in der Tabelle aus- oder abzuwählen.

Die zur Auswahl verfügbaren Felder werden gefiltert, sodass nur die Felder angezeigt werden, die für den verwendeten Streamparameter oder die verwendete Knoteneigenschaft geeignet sind. Wenn Sie z. B. einen Streamparameter mit dem Speichertyp "Zeichenfolge" verwenden, werden nur Felder mit diesem Speichertyp angezeigt.

Bedingte Ausführung in Streams

Mit bedingter Ausführung können Sie steuern, wie Endknoten ausgeführt werden, und zwar in Abhängigkeit davon, ob die Streaminhalte den definierten Bedingungen entsprechen. Beispiele:


- Steuern Sie, ob ein Knoten ausgeführt wird, in Abhängigkeit davon, ob ein bestimmter Wert "true" oder "false" ist.
- Definieren Sie, ob Schleifen parallel oder sequenziell ausgeführt werden.

Sie definieren die zu erfüllenden Bedingungen auf der Unterregisterkarte **Bedingt** der Registerkarte **Ausführung** des Streams. Um die Unterregisterkarte anzuzeigen, wählen Sie den Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aus.

Alle definierten Anforderungen für die bedingte Ausführung werden bei der Ausführung des Streams wirksam, wenn der Ausführungsmodus **Verwendung von Schleifen/Bedingte Ausführung** aktiviert wurde. Optional können Sie den Script-Code für die Anforderungen Ihrer bedingten Ausführung generieren und ihn in den Scripteditor einfügen, indem Sie unten rechts auf der Unterregisterkarte **Bedingt** auf **Einfügen...** klicken. Die Anzeige der Hauptregisterkarte **Ausführung** ändert sich und der Ausführungsmodus **Standard (optionales Script)** wird mit dem Script im oberen Teil der Registerkarte angezeigt. Dies bedeutet, dass Sie Bedingungen mithilfe der verschiedenen Optionen im Dialogfeld **Verwendung von Schleifen** definieren können, bevor Sie ein Script generieren, das Sie im Scripteditor weiter anpassen können. Wenn Sie auf **Einfügen...** klicken, werden auch alle von Ihnen definierten Schleifenanforderungen im generierten Script angezeigt.

So definieren Sie eine Bedingung:

1. Klicken Sie in der rechten Spalte der Unterregisterkarte **Bedingt** auf die Schaltfläche **Neue Bedingung**

hinzufügen , um das Dialogfeld **Bedingte Ausführungsanweisung hinzufügen** zu öffnen. In diesem Dialogfeld geben Sie die Bedingung an, die erfüllt sein muss, damit der Knoten ausgeführt wird.

2. Geben Sie im Dialogfeld **Bedingte Ausführungsanweisung hinzufügen** Folgendes an:

- a. **Knoten.** Wählen Sie den Knoten aus, für die Sie eine bedingte Ausführung konfigurieren wollen. Klicken Sie auf die Suchschaltfläche, um das Dialogfeld für die Knotenauswahl zu öffnen, und wählen Sie den gewünschten Knoten aus. Werden zu viele Knoten angezeigt, können Sie die Anzeige so filtern, dass nur Knoten einer der folgenden Kategorien angezeigt werden: Export-, Diagramm-, Modellierungs- oder Ausgabeknoten.
- b. **Bedingung basierend auf.** Geben Sie die Bedingung an, die erfüllt sein muss, damit der Knoten ausgeführt wird. Sie können unter vier Optionen wählen: **Streamparameter**, **Globale Variable**, **Tabellenausgabezelle** oder **Immer wahr**. Welche Details Sie in der unteren Hälfte des Dialogfelds angeben können, wird von der ausgewählten Bedingung gesteuert.
 - **Streamparameter.** Wählen Sie den Parameter aus der Liste der verfügbaren Parameter aus und wählen Sie dann den Operator für diesen Parameter aus. Beispielsweise kann der Operator **Größer als**, **Gleich**, **Kleiner als**, **Zwischen** usw. sein. Dann geben Sie je nach Operator den Wert oder Minimal- oder Maximalwerte ein.

- **Globale Variable.** Wählen Sie die Variable aus der Liste der verfügbaren Variablen aus. Beispielsweise kann die Variable **Mittelwert**, **Summe**, **Minimum**, **Maximum** oder **Standardabweichung** sein. Wählen Sie dann den Operator und die erforderlichen Werte aus.
 - **Tabellenausgabezeile.** Wählen Sie den Tabellenknoten aus der Liste der verfügbaren Elemente aus und wählen Sie dann die Zeile oder Spalte in der Tabelle aus. Wählen Sie dann den Operator und die erforderlichen Werte aus.
 - **Immer wahr.** Wählen Sie diese Option aus, wenn der Knoten immer ausgeführt werden muss. Wenn Sie diese Option auswählen, müssen keine weiteren Parameter mehr ausgewählt werden.
3. Wiederholen Sie Schritt 1 und 2 so oft, bis Sie alle erforderlichen Bedingungen definiert haben. Der ausgewählte Knoten und die Bedingung, die erfüllt werden muss, damit der Knoten ausgeführt wird, werden im Hauptteil der Unterregisterkarte in den Spalten **Ausführungsknoten** bzw. **Wenn diese Bedingung wahr ist** angezeigt.
 4. Standardmäßig werden Knoten und Bedingungen in der aufgeführten Reihenfolge ausgeführt. Um einen Knoten und eine Bedingung in der Liste nach oben oder unten zu verschieben, wählen Sie diese durch Anklicken aus und ändern Sie die Reihenfolge mit dem Auf- oder Abwärtspfeil in der rechten Spalte der Unterregisterkarte.

Darüber hinaus können Sie die folgenden Optionen unten in der Unterregisterkarte **Bedingt** festlegen:

- **Alle der Reihenfolge nach auswerten.** Wählen Sie diese Option aus, um jede Bedingung in der auf der Unterregisterkarte aufgeführten Reihenfolge auszuwerten. Die Knoten, für die die Bedingungen erfüllt sind, werden ausgeführt, sobald alle Bedingungen ausgewertet wurden.
- **Jeweils nur einen ausführen.** Nur verfügbar, wenn **Alle der Reihenfolge nach auswerten** ausgewählt wurde. Wenn Sie diese Option auswählen und eine Bedingung als "True" ausgewertet wird, wird der Knoten ausgeführt, dem diese Bedingung zugeordnet ist, bevor die nächste Bedingung ausgewertet wird.
- **Auswerten bis zum ersten Treffer.** Wenn Sie diese Option auswählen, wird nur der erste Knoten ausgeführt, für den die angegebenen Bedingungen als "True" ausgewertet werden.

Ausführen und Unterbrechen von Scripts

Es gibt mehrere Möglichkeiten zur Ausführung von Scripts. Beispielsweise führt die Schaltfläche "Dieses Script ausführen" im Dialogfeld für das Stream-Script oder Standalone-Script das vollständige Script aus:



Abbildung 1. Schaltfläche "Dieses Script ausführen"

Die Schaltfläche "Ausgewählte Zeilen ausführen" führt eine einzelne Zeile oder einen Block benachbarter Zeilen aus, die Sie im Script ausgewählt haben:



Abbildung 2. Schaltfläche "Ausgewählte Zeilen ausführen"

Zum Ausführen von Scripts stehen folgende Methoden zur Auswahl:

- Klicken Sie im Dialogfeld für ein Stream-Script oder ein Standalone-Script auf die Schaltfläche "Dieses Script ausführen" oder "Ausgewählte Zeilen ausführen".
- Ausführen eines Streams mit **Dieses Script ausführen** als Standardausführungsmethode.
- Verwenden Sie das Flag `-execute` beim Start im interaktiven Modus. Weitere Informationen finden Sie im Thema „Verwenden von Befehlszeilenargumenten“ auf Seite 65.

Hinweis: Ein Superknotenscript wird bei der Ausführung des Superknotens ausgeführt, sofern Sie **Dieses Script ausführen** im Superknotenscript-Dialogfeld ausgewählt haben.

Unterbrechen der Scriptausführung

Während der Scriptausführung ist im Dialogfeld "Stream-Script" die rote Stoppschaltfläche aktiviert. Mit dieser Schaltfläche können Sie die Ausführung des Scripts und aller aktuellen Streams abbrechen.

Suchen und Ersetzen

Das Dialogfeld "Suchen/Ersetzen" ist in Situationen verfügbar, in denen Sie Script- oder Ausdruckstext bearbeiten, u. a. im Script-Editor, im CLEM Expression Builder und bei der Definition von Vorlagen im Berichtsknoten. Während der Bearbeitung von Text in einem dieser Bereiche können Sie durch Drücken von Strg+F das Dialogfeld aufrufen. Achten Sie dabei darauf, dass sich der Cursor über einem Textbereich befindet. So können Sie beispielsweise bei der Arbeit in einem Füllerknoten das Dialogfeld aus einem der Textbereiche auf der Registerkarte "Einstellungen" aufrufen oder über das Textfeld im Expression Builder.

1. Achten Sie darauf, dass sich der Cursor in einem Textfeld befindet, und drücken Sie Strg+F, um das Dialogfeld "Suchen/Ersetzen" aufzurufen.
2. Geben Sie den Text ein, nach dem Sie suchen möchten, oder treffen Sie eine Auswahl aus der Drop-down-Liste der kürzlich durchsuchten Elemente.
3. Geben Sie, falls erforderlich, den Ersatztext ein.
4. Klicken Sie auf **Weitersuchen**, um die Suche zu starten.
5. Klicken Sie auf **Ersetzen**, um die aktuelle Auswahl zu ersetzen, oder **Alle ersetzen**, um alle bzw. die ausgewählten Instanzen zu ersetzen.
6. Das Dialogfeld wird nach jedem Vorgang geschlossen. Drücken Sie in einem Textbereich die Taste F3, um den letzten Suchvorgang zu wiederholen, bzw. drücken Sie Strg+F, um erneut auf das Dialogfeld zuzugreifen.

Suchoptionen

Groß-/Kleinschreibung beachten. Gibt an, ob beim Suchvorgang die Groß- und Kleinschreibung berücksichtigt wird; beispielsweise, ob *myvar* als identisch mit *myVar* betrachtet wird. Ersetzungstext wird unabhängig von dieser Einstellung stets genau so eingefügt, wie er eingegeben wurde.

Nur ganze Wörter. Gibt an, ob beim Suchvorgang auch Wortteile gefunden werden. Wenn diese Option ausgewählt ist, werden bei einer Suche nach *Tag* Terme wie *Tagesordnung* oder *Tag-und-Nacht-Gleiche* nicht als Treffer ausgegeben.

Reguläre Ausdrücke. Gibt an, ob die Syntax für reguläre Ausdrücke verwendet werden soll (siehe nächsten Abschnitt). Bei Auswahl dieser Option wird die Option **Nur ganze Wörter** inaktiviert und ihr Wert ignoriert.

Nur ausgewählter Text. Legt den Suchumfang bei Verwendung der Option **Alle ersetzen** fest.

Syntax für reguläre Ausdrücke

Mithilfe von regulären Ausdrücken können Sie nach Sonderzeichen (z. B. Tabulatoren oder Zeilenumbrüche), Zeichenklassen bzw. -bereichen, wie *a* bis *d*, beliebige Ziffer oder Nichtziffer, und nach Begrenzungen, wie beispielsweise Zeilenanfang bzw. Zeilenende, suchen. Folgende Arten von Ausdrücken werden unterstützt:

Tabelle 1. Zeichenübereinstimmungen.

Zeichen	Übereinstimmung/Bedeutung
x	Das Zeichen x
\\	Umgekehrter Schrägstrich
\0n	Das Zeichen mit dem Oktalwert 0n (0 <= n <= 7)
\0nn	Das Zeichen mit dem Oktalwert 0nn (0 <= n <= 7)
\0mnn	Das Zeichen mit dem Oktalwert 0mnn (0 <= m <= 3; 0 <= n <= 7)
\xhh	Das Zeichen mit dem Hexadezimalwert 0xhh
\uhhhh	Das Zeichen mit dem Hexadezimalwert 0xhhhh
\t	Das Tabulatorzeichen ('\u0009')
\n	Das Zeilenvorschubzeichen ('\u000A')
\r	Das Rücklaufzeichen ('\u000D')
\f	Das Formularvorschubzeichen ('\u000C')
\a	Das Alarmzeichen ('\u0007')
\e	Das Escapezeichen ('\u001B')
\cx	Das Steuerzeichen, das x entspricht

Tabelle 2. Übereinstimmende Zeichenklassen.

Zeichenklassen	Übereinstimmung/Bedeutung
[abc]	a, b oder c (einfache Klasse)
[^abc]	Beliebiges Zeichen mit Ausnahme von a, b oder c (Differenzmenge)
[a-zA-Z]	a bis einschließlich z bzw. A bis einschließlich Z (Bereich)
[a-d[m-p]]	a bis d bzw. m bis P (Vereinigungsmenge) Alternative Angabemöglichkeit: [a-dm-p]
[a-z&&[def]]	a bis z und d, e bzw. f (Schnittmenge)
[a-z&&[^bc]]	a bis z mit Ausnahme von b und c (Differenzmenge) Alternative Angabemöglichkeit: [a-d-z]
[a-z&&[^m-p]]	a bis z mit Ausnahme von m bis p (Differenzmenge) Alternative Angabemöglichkeit: [a-lq-z]

Tabelle 3. Vordefinierte Zeichenklassen.

Vordefinierte Zeichenklassen	Übereinstimmung/Bedeutung
.	Beliebiges Zeichen (evtl. Übereinstimmung mit Zeilenabschlusszeichen)
\d	Beliebige Ziffer: [0-9]
\D	Nichtziffer: [^0-9]
\s	Ein Leerzeichen: [\t\n\x0B\f\r]
\S	Ein Nichtleerzeichen: [^\s]
\w	Ein Zeichen: [a-zA-Z_0-9]
\W	Ein Nichtzeichen: [^\w]

Tabelle 4. Übereinstimmungen mit Begrenzungszeichen.

Zeichen(folgen) für Begrenzungszeichen	Übereinstimmung/Bedeutung
<code>^</code>	Zeilenanfang
<code>\$</code>	Zeilenende
<code>\b</code>	Wortgrenze
<code>\B</code>	Nichtwortgrenze
<code>\A</code>	Beginn der Eingabe
<code>\Z</code>	Ende der Eingabe mit Ausnahme des letzten Abschlusszeichens, sofern vorhanden
<code>\z</code>	Ende der Eingabe

Kapitel 2. Scriptsprache

Scriptsprache - Überblick

Mit den Scriptfunktionen für IBM SPSS Modeler können Sie Scripts erstellen, die mit der SPSS Modeler-Benutzerschnittstelle arbeiten, Ausgabeobjekte manipulieren und Befehlssyntax ausführen. Sie können Scripts direkt aus SPSS Modeler ausführen.

Scripts in IBM SPSS Modeler werden in der Scriptsprache Python geschrieben. Die von IBM SPSS Modeler verwendete Java-basierte Implementierung von Python wird als Jython bezeichnet. Die Scriptsprache besteht aus folgenden Elementen:

- Format für die Referenzierung von Knoten, Streams, Projekten, Ausgaben und anderen IBM SPSS Modeler-Objekten
- Set mit Scriptanweisungen bzw. Befehlen, die zur Bearbeitung dieser Objekte verwendet werden können.
- Script-Ausdrucksprache für die Festlegung der Werte von Variablen, Parametern und anderen Objekten.
- Unterstützung für Kommentare, Fortsetzungen und Blöcke mit Literaltext.

In den folgenden Abschnitten werden die Python-Scriptsprache, die Jython-Implementierung von Python und die grundlegende Syntax für das erste Arbeiten mit Scripting in IBM SPSS Modeler beschrieben. Informationen zu speziellen Eigenschaften und Befehlen finden Sie in den nachfolgenden Abschnitten.

Python und Jython

Jython ist eine Implementierung der Python-Scriptsprache, die in Java geschrieben ist und in die Java-Plattform integriert ist. Python ist eine leistungsfähige objektorientierte Scriptsprache. Jython ist nützlich, da es die Produktivitätsfunktionen einer ausgereiften Scriptsprache bereitstellt und anders als Python in einer beliebigen Umgebung ausgeführt werden kann, die Java Virtual Machine (JVM) unterstützt. Dies bedeutet, dass die Java-Bibliotheken in JVM beim Schreiben von Programmen zur Verfügung stehen. Mit Jython können Sie sich diesen Unterschied zunutze machen und die Syntax und die meisten Funktionen der Python-Sprache verwenden.

Als Scriptsprache ist Python (und deren Jython-Implementierung) einfach zu erlernen und effizient zu codieren und sie hat die minimal erforderliche Struktur zum Erstellen eines lauffähigen Programms. Code kann zeilenweise interaktiv eingegeben werden. Python ist eine interpretierte Scriptsprache; es gibt keinen Vorkompilierungsschritt wie in Java. Python-Programme sind einfache Textdateien, die bei ihrer Eingabe (nach der Analyse auf Syntaxfehler) interpretiert werden. Einfache Ausdrücke wie z. B. definierte Werte und auch komplexere Aktionen wie Funktionsdefinitionen werden unverzüglich ausgeführt und sind sofort einsatzbereit. Änderungen am Code können schnell getestet werden. Die Scriptinterpretation hat jedoch einige Nachteile. So ist beispielsweise die Verwendung einer nicht definierten Variablen kein Compilerfehler und wird deshalb erst bei der Ausführung der Anweisung erkannt, in der die Variable verwendet wird. In diesem Fall kann das Programm bearbeitet und ausgeführt werden, um den Fehler zu beheben.

Python betrachtet alles, d. h. alle Daten und den gesamten Code, als Objekt. Sie können diese Objekte daher mit Codezeilen manipulieren. Einige Typen wie Zahlen und Zeichenfolgen werden praktischerweise als Werte und nicht als Objekte betrachtet; dies wird von Python unterstützt. Es gibt einen unterstützten Nullwert. Dieser Nullwert hat den reservierten Namen None.

Eine umfassendere Einführung in Python- und Jython-Scripting sowie einige Beispielscripts finden Sie in <http://www.ibm.com/developerworks/java/tutorials/j-jython1> und <http://www.ibm.com/developerworks/java/tutorials/j-jython2>.

Python-Scripting

Dieses Handbuch zur Python-Scriptsprache bietet eine Einführung in die Komponenten, die am häufigsten beim Scripting in IBM SPSS Modeler verwendet werden, einschließlich der Konzepte und Programmiergrundlagen. Es liefert Ihnen ausreichend Kenntnisse, um mit der Entwicklung Ihrer eigenen Python-Scripts zu beginnen, die in IBM SPSS Modeler verwendet werden sollen.

Operationen

Die Zuordnung erfolgt mittels eines Gleichheitszeichens (=). Wenn Sie z. B. den Wert "3" einer Variablen namens "x" zuweisen wollen, würden Sie die folgende Anweisung verwenden:

```
x = 3
```

Das Gleichheitszeichen wird auch für die Zuordnung von Zeichenfolgedaten zu einer Variablen verwendet. Wenn Sie z. B. den Wert "ein Zeichenfolgewert" einer Variablen namens "y" zuweisen wollen, würden Sie die folgende Anweisung verwenden:

```
y = "ein Zeichenfolgewert"
```

In der folgenden Tabelle werden einige der gängigen Vergleichs- und numerischen Operationen sowie deren Beschreibungen aufgelistet.

Tabelle 5. Allgemeine Vergleichsoperationen und numerische Operationen

Operation	Beschreibung
$x < y$	Ist x kleiner als y?
$x > y$	Ist x größer als y?
$x \leq y$	Ist x kleiner-gleich y?
$x \geq y$	Ist x größer-gleich y?
$x == y$	Ist x gleich y?
$x != y$	Ist x ungleich y?
$x \lt;> y$	Ist x ungleich y?
$x + y$	y zu x addieren
$x - y$	y von x substrahieren
$x * y$	x mit y multiplizieren
x / y	x durch y dividieren
$x ** y$	x hoch y

Listen

Listen sind Folgen von Elementen. Eine Liste kann eine beliebige Anzahl von Elementen enthalten und die Elemente der Liste können einen beliebigen Objekttyp haben. Listen kann man sich auch als Arrays vorstellen. Die Anzahl der Elemente in einer Liste kann sich durch Hinzufügen, Entfernen oder Ersetzen von Elementen verringern oder erhöhen.

Beispiele

```
[]
```

Eine beliebige Liste ohne Inhalt.

[1]	Eine Liste mit einem einzigen Element: einer ganzen Zahl.
["Mike", 10, "Don", 20]	Eine Liste mit vier Elementen: zwei Zeichenfolgeelemente und zwei ganzzahlige Elemente.
[[], [7], [8, 9]]	Eine aus Listen bestehende Liste. Jede Unterliste ist entweder eine Liste ohne Inhalt oder eine Liste mit ganzzahligen Elementen.
x = 7; y = 2; z = 3; [1, x, y, x + y]	Eine Liste mit ganzen Zahlen. Dieses Beispiel veranschaulicht die Verwendung von Variablen und Ausdrücken.

Sie können eine Liste einer Variablen zuordnen. Beispiel:

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

Sie können dann auf bestimmte Elemente der Liste zugreifen. Beispiel:

```
mylist[0]
```

Dies resultiert in der folgenden Ausgabe:

```
one
```

Die Zahl in eckigen Klammern ([]) wird als *Index* bezeichnet und verweist auf ein bestimmtes Element der Liste. Die Elemente einer Liste werden mit 0 beginnend indexiert.

Sie können auch einen Bereich von Elementen einer Liste auswählen. Dies wird als *Slicing* bezeichnet. `x[1:3]` beispielsweise wählt das zweite und das dritte Element von `x` aus. Der Index für das Bereichsende muss um 1 größer sein als der Index des letzten auszuwählenden Elements.

Zeichenfolgen

Eine *Zeichenfolge* ist eine unveränderliche Folge von Zeichen, die als Wert behandelt wird. Zeichenfolgen unterstützen alle Funktionen und Operationen mit unveränderlichen Folgen, die in einer neuen Zeichenfolge resultieren. Beispiel: `"abcdef"[1:4]` wird als `"bcd"` ausgegeben.

In Python werden Zeichen als Zeichenfolgen der Länge 1 dargestellt.

Zeichenfolgeliterale werden durch die Verwendung von ein- oder dreifachen Anführungszeichen definiert. Zeichenfolgen, die mit einfachen Anführungszeichen definiert sind, können nicht mehrere Zeilen umfassen, Zeichenfolgen in dreifachen Anführungszeichen dagegen schon. Eine Zeichenfolge kann in einfache Anführungszeichen (') oder doppelte Anführungszeichen (") eingeschlossen werden. Ein Hervorhebungszeichen kann das andere Hervorhebungszeichen ohne Escape-Zeichen enthalten oder das Hervorhebungszeichen wird durch ein Escapezeichen entwertet, d. h., ihm wird der umgekehrte Schrägstrich (\) vorangestellt.

Beispiele

```
"Dies ist eine Zeichenfolge"
'Dies ist auch eine Zeichenfolge'
"Es ist eine Zeichenfolge"
'Dieses Handbuch heißt "Handbuch für Python-Scripting und -Automatisierung".'
"Dies ist ein durch Escapezeichen entwertetes Anführungszeichen (\") in einer Zeichenfolge in Anführungszeichen"
```

Mehrere durch Leerzeichen voneinander getrennte Zeichenfolgen werden vom Python-Parser automatisch verkettet. Dies vereinfacht die Eingabe langer Zeichenfolgen und das Mischen unterschiedlicher Anführungszeichen in einer einzigen Zeichenfolge. Beispiel:

```
"Diese Zeichenfolge verwendet ' und " 'diese Zeichenfolge verwendet ".'
```

Dies resultiert in der folgenden Ausgabe:

Diese Zeichenfolge verwendet ' und diese Zeichenfolge verwendet ".

Zeichenfolgen unterstützen mehrere nützliche Methoden. Einige dieser Methoden werden in der folgenden Tabelle genannt.

Tabelle 6. Zeichenfolgemethoden

Methoden	Verwendung
<code>s.capitalize()</code>	Anfangsbuchstabe von <code>s</code> wird großgeschrieben.
<code>s.count(ss {,start {,end}})</code>	Zählt die Vorkommen von <code>ss</code> in <code>s[start:end]</code>
<code>s.startswith(str {, start {, end}})</code> <code>s.endswith(str {, start {, end}})</code>	Testet, ob <code>s</code> mit <code>str</code> beginnt. Testet, ob <code>s</code> mit <code>str</code> endet.
<code>s.expandtabs({size})</code>	Ersetzt Tabstopps durch Leerzeichen. Standardgröße (<code>size</code>) ist 8.
<code>s.find(str {, start {, end}})</code> <code>s.rfind(str {, start {, end}})</code>	Sucht den ersten Index von <code>str</code> in <code>s</code> ; wird er nicht gefunden, lautet das Ergebnis -1. <code>rfind</code> sucht von rechts nach links.
<code>s.index(str {, start {, end}})</code> <code>s.rindex(str {, start {, end}})</code>	Sucht den ersten Index von <code>str</code> in <code>s</code> ; wenn er nicht gefunden wird, wird <code>ValueError</code> ausgelöst. <code>rindex</code> sucht von rechts nach links.
<code>s.isalnum</code>	Testet, ob die Zeichenfolge alphanumerisch ist.
<code>s.isalpha</code>	Testet, ob die Zeichenfolge alphabetisch ist.
<code>s.isnum</code>	Testet, ob die Zeichenfolge numerisch ist.
<code>s.isupper</code>	Testet, ob die Zeichenfolge ganz in Großbuchstaben geschrieben ist.
<code>s.islower</code>	Testet, ob die Zeichenfolge ganz in Kleinbuchstaben geschrieben ist.
<code>s.isspace</code>	Testet, ob die Zeichenfolge nur aus Leerzeichen besteht.
<code>s.istitle</code>	Testet, ob die Zeichenfolge eine Folge von alphanumerischen Zeichen mit Großschreibung des ersten Buchstaben ist.
<code>s.lower()</code> <code>s.upper()</code> <code>s.swapcase()</code> <code>s.title()</code>	Konvertiert alles in Kleinbuchstaben. Konvertiert alles in Großbuchstaben. Invertiert die Groß-/Kleinschreibung. Konvertiert alles in Schreibung mit großem Anfangsbuchstaben.
<code>s.join(seq)</code>	Verknüpft die Zeichenfolgen in <code>seq</code> mit <code>s</code> als Trennzeichen.
<code>s.splitlines({keep})</code>	Teilt <code>s</code> in Zeilen auf. Wenn 'keep' true ist, werden die neuen Zeilen beibehalten.
<code>s.split({sep {, max}})</code>	Teilt <code>s</code> mithilfe von <code>sep</code> (Standardeinstellung von <code>sep</code> ist ein Leerzeichen) bis zu <code>max</code> Male in "Wörter" auf.
<code>s.ljust(width)</code> <code>s.rjust(width)</code> <code>s.center(width)</code> <code>s.zfill(width)</code>	Richtet die Zeichenfolge in einem Feld der Breite <code>width</code> links aus. Richtet die Zeichenfolge in einem Feld der Breite <code>width</code> rechts aus. Zentriert die Zeichenfolge in einem Feld der Breite <code>width</code> . Füllt mit 0 auf.

Table 6. Zeichenfolgemethoden (Forts.)

Methoden	Verwendung
s.lstrip() s.rstrip() s.strip()	Entfernt führende Leerzeichen. Entfernt folgende Leerzeichen. Entfernt führende und folgende Leerzeichen.
s.translate(str {,delc})	Setzt s mithilfe einer Tabelle um, nachdem Zeichen in delc entfernt wurden. str sollte eine Zeichenfolge der Länge == 256 sein.
s.replace(old, new {, max})	Ersetzt alle oder maximal max Vorkommen der Zeichenfolge old durch die Zeichenfolge new.

Anmerkungen

Anmerkungen sind Kommentare, die durch das Rautenzeichen (Hashzeichen) (#) eingeleitet werden. Der Text, der in derselben Zeile auf die Raute folgt, wird als Teil der Anmerkung betrachtet und ignoriert. Eine Anmerkung kann in einer beliebigen Spalte beginnen. Das folgende Beispiel veranschaulicht die Verwendung von Anmerkungen:

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

Anweisungssyntax

Die Anweisungssyntax für Python ist sehr einfach. Im Allgemeinen ist jede Quellcodezeile eine einzelne Anweisung. Außer bei expression- und assignment-Anweisungen wird jede Anweisung durch ein Schlüsselwort wie if oder for eingeleitet. Leerzeilen oder Anmerkungszeilen können an beliebiger Stelle zwischen Anweisungen im Code eingefügt werden. Wenn mehrere Anweisungen in einer Zeile stehen, müssen sie durch ein Semikolon (;) voneinander getrennt werden.

Sehr lange Anweisungen können in weiteren Zeilen fortgesetzt werden. In diesem Fall muss die Anweisung, die in der nächsten Zeile fortgesetzt werden soll, mit einem umgekehrten Schrägstrich (\) enden. Beispiel:

```
x = "Eine laaaaaaaaaaaaaaaaaaange Zeichenfolge" + \
    "noch eine laaaaaaaaaaaaaaaaaaange Zeichenfolge"
```

Wenn eine Struktur in runde Klammern (()), eckige Klammern ([]) oder geschweifte Klammern ({}), eingeschlossen ist, kann die Anweisung ohne umgekehrten Schrägstrich nach einem Komma in der nächsten Zeile fortgesetzt werden. Beispiel:

```
x = (1, 2, 3, "hallo",
    "auf Wiedersehen", 4, 5, 6)
```

IDs

IDs werden verwendet, um Variablen, Funktionen, Klassen und Schlüsselwörter zu bezeichnen. IDs können eine beliebige Länge haben. Sie müssen jedoch entweder mit einem Groß- oder Kleinbuchstaben oder mit einem Unterstrich (_) beginnen. Namen, die mit einem Unterstrich beginnen, sind im Allgemeinen für interne oder nicht öffentliche Namen reserviert. Nach dem ersten Zeichen kann die ID eine beliebige Anzahl von Buchstaben, Ziffern und Unterstrichen in beliebiger Kombination enthalten.

Es gibt in Python einige reservierte Wörter, die nicht zur Benennung von Variablen, Funktionen oder Klassen verwendet werden können. Sie sind in die folgenden Kategorien aufgeteilt:

- **Einführungszeichen für Anweisungen:** assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, pass, print, raise, return, try und while
- **Einführungszeichen für Parameter:** as, import und in
- **Operatoren:** and, in, is, lambda, not und or

Bei inkorrektter Schlüsselwortverwendung tritt in der Regel ein Syntaxfehler auf.

Codeblöcke

Codeblöcke sind Gruppen von Anweisungen, die an Stellen verwendet werden, an denen einzelne Anweisungen erwartet werden. Codeblöcke können auf jede der folgenden Anweisungen folgen: `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` und `class`. Diese Anweisungen führen den Codeblock mit dem Doppelpunkt (`:`) ein. Beispiel:

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

Zur Begrenzung von Codeblöcken wird Einrückung verwendet (anstelle der geschweiften Klammern, die in Java verwendet werden). Alle Zeilen in einem Block müssen an dieselbe Position eingerückt werden. Eine Änderung der Einrückung bedeutet das Ende eines Codeblocks. Gewöhnlich wird pro Ebene um vier Leerschritte eingerückt. Zur Einrückung der Zeilen empfiehlt es sich, Leerzeichen anstelle von Tabstopps zu verwenden. Leerzeichen und Tabstopps dürfen nicht gemischt werden. Die Zeilen im äußersten Block eines Moduls, müssen in Spalte 1 beginnen, da sonst ein Syntaxfehler auftritt.

Die Anweisungen, aus denen ein Codeblock besteht (und die auf einen Doppelpunkt folgen), können auch in einer einzigen Zeile durch Semikolons getrennt angeordnet werden. Beispiel:

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

Übergeben von Argumenten an ein Script

Die Übergabe von Argumenten an ein Script ist nützlich, da dadurch ein Script wiederholt ohne Änderung verwendet werden kann. Die Argumente, die in der Befehlszeile übergeben werden, werden als Werte in der Liste `sys.argv` übergeben. Die Anzahl der übergebenen Werte kann über den Befehl `len(sys.argv)` abgerufen werden. Beispiel:

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

In diesem Beispiel importiert der Befehl `import` die gesamte Klasse `sys`, sodass die für diese Klasse vorhandenen Methoden wie `argv` verwendet werden können.

Das Script in diesem Beispiel kann über die folgende Befehlszeile aufgerufen werden:

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

Daraus resultiert die folgende Ausgabe:

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

Beispiele

Das Schlüsselwort `print` gibt die ihm direkt nachfolgenden Argumente aus. Wenn auf die Anweisung ein Komma folgt, enthält die Ausgabe keinen Zeilenumbruch. Beispiel:

```
print "Dies veranschaulicht die Verwendung eines",
print " Kommas am Ende einer Druckanweisung."
```

Dies resultiert in der folgenden Ausgabe:

Dies veranschaulicht die Verwendung eines Kommas am Ende einer Druckanweisung.

Die Anweisung `for` wird zum Durchlaufen eines Codeblocks verwendet. Beispiel:

```
mylist1 = ["eins", "zwei", "drei"]
for lv in mylist1:
    print lv
    continue
```

In diesem Beispiel werden drei Zeichenfolgen der Liste `mylist1` zugeordnet. Die Elemente der Liste werden dann ausgegeben, wobei jeweils ein Element in jeder Zeile steht. Dies resultiert in der folgenden Ausgabe:

```
eins
zwei
drei
```

In diesem Beispiel nimmt der Iterator `lv` nacheinander den Wert jedes Elements in der Liste `mylist1` an, während die For-Schleife den Codeblock für jedes Element implementiert. Ein Iterator kann eine gültige Kennung beliebiger Länge sein.

Die Anweisung `if` ist eine bedingte Anweisung. Sie wertet die Bedingung aus und gibt je nach Ergebnis der Bewertung entweder `"true"` oder `"false"` zurück. Beispiel:

```
mylist1 = ["eins", "zwei", "drei"]
for lv in mylist1:
    if lv == "zwei":
        print "Der Wert von lv ist zwei ", lv
    else:
        print "Der Wert von lv ist nicht zwei, sondern ", lv
    continue
```

In diesem Beispiel wird der Wert des Iterators `lv` ausgewertet. Wenn `lv` den Wert `zwei` hat, wird eine andere Zeichenfolge zurückgegeben als in den Fällen, in denen `lv` nicht den Wert `zwei` hat. Dies resultiert in der folgenden Ausgabe:

```
Der Wert von lv ist nicht zwei, sondern eins
Der Wert von lv ist zwei
Der Wert von lv ist nicht zwei, sondern drei
```

Mathematische Methoden

Aus dem Modul `math` können Sie auf nützliche mathematische Methoden zugreifen. Einige dieser Methoden werden in der folgenden Tabelle genannt. Sofern nichts anderes angegeben ist, werden alle Werte als Gleitkommazahlen zurückgegeben.

Tabelle 7. Mathematische Methoden

Methode	Verwendung
<code>math.ceil(x)</code>	Gibt die obere Grenze (ceiling) von <code>x</code> als Gleitkommazahl zurück. Dies ist die kleinste Ganzzahl, die größer-gleich <code>x</code> ist.
<code>math.copysign(x, y)</code>	Gibt <code>x</code> mit dem Vorzeichen von <code>y</code> zurück. <code>copysign(1, -0.0)</code> gibt <code>-1</code> zurück.
<code>math.fabs(x)</code>	Gibt den absoluten Wert von <code>x</code> zurück.
<code>math.factorial(x)</code>	Gibt <code>x</code> -Fakultät zurück. Wenn <code>x</code> eine negative Zahl oder keine Ganzzahl ist, tritt ein <code>ValueError</code> auf.
<code>math.floor(x)</code>	Gibt die Untergrenze (floor) von <code>x</code> als Gleitkommazahl zurück. Dies ist die größte Ganzzahl, die kleiner-gleich <code>x</code> ist.

Tabelle 7. Mathematische Methoden (Forts.)

Methoden	Verwendung
<code>math.frexp(x)</code>	Gibt die Mantisse (m) und den Exponenten (e) von x als Paar (m, e) zurück. m ist eine Gleitkommazahl und e ist eine Ganzzahl, sodass sich genau $x = m * 2^{**e}$ ergibt. Wenn x Null ist, wird (0.0, 0) zurückgegeben, sonst wird $0.5 \leq \text{abs}(m) < 1$ zurückgegeben.
<code>math.fsum(iterable)</code>	Gibt eine genaue Gleitkommasumme von Werten in <code>iterable</code> zurück.
<code>math.isinf(x)</code>	Prüft, ob die Gleitkommazahl x positiv oder negativ unendlich ist.
<code>math.isnan(x)</code>	Prüft, ob die Gleitkommazahl x eine Nichtzahl (NaN -not a number) ist.
<code>math.ldexp(x, i)</code>	Gibt $x * (2^{**i})$ zurück. Dies ist im Wesentlichen die Umkehrfunktion von <code>frexp</code> .
<code>math.modf(x)</code>	Gibt die ganzzahligen und die Bruchteile von x zurück. Beide Ergebnisse übernehmen das Vorzeichen von x und sind Gleitkommazahlen.
<code>math.trunc(x)</code>	Gibt den reellen Wert x zurück, der auf eine Ganzzahl abgeschnitten wurde.
<code>math.exp(x)</code>	Gibt e^{**x} zurück.
<code>math.log(x[, base])</code>	Gibt den Logarithmus von x zur Basis base zurück. Ohne Angabe von base wird der natürliche Logarithmus von x zurückgegeben.
<code>math.log1p(x)</code>	Gibt den natürlichen Logarithmus von 1+x (base e) zurück.
<code>math.log10(x)</code>	Gibt den Logarithmus von x zur Basis 10 zurück.
<code>math.pow(x, y)</code>	Gibt x hoch y zurück. <code>pow(1.0, x)</code> und <code>pow(x, 0.0)</code> geben immer 1 zurück, selbst wenn x Null oder eine Nichtzahl ist.
<code>math.sqrt(x)</code>	Gibt die Quadratwurzel von x zurück.

Zusätzlich zu den mathematischen Funktionen gibt es einige nützliche trigonometrische Methoden. Diese Methoden werden in der folgenden Tabelle dargestellt.

Tabelle 8. Trigonometrische Methoden

Methoden	Verwendung
<code>math.acos(x)</code>	Gibt den Arkuskosinus von x in Radianten zurück.
<code>math.asin(x)</code>	Gibt den Arkussinus von x in Radianten zurück.
<code>math.atan(x)</code>	Gibt den Arkustangens von x in Radianten zurück.
<code>math.atan2(y, x)</code>	Gibt <code>atan(y / x)</code> in Radianten zurück.
<code>math.cos(x)</code>	Gibt den Kosinus von x in Radianten zurück.
<code>math.hypot(x, y)</code>	Gibt die euklidische Norm $\text{sqrt}(x*x + y*y)$ zurück. Dies ist die Länge des Vektors vom Ursprung zum Punkt (x, y).
<code>math.sin(x)</code>	Gibt den Sinus von x in Radianten zurück.
<code>math.tan(x)</code>	Gibt den Tangens von x in Radianten zurück.
<code>math.degrees(x)</code>	Konvertiert den Winkel x von Radianten in Grad.

Table 8. Trigonometrische Methoden (Forts.)

Method	Verwendung
<code>math.radians(x)</code>	Konvertiert den Winkel x von Grad in Radianten.
<code>math.acosh(x)</code>	Umkehrfunktion des Hyperbelkosinus von x zurückgeben
<code>math.asinh(x)</code>	Umkehrfunktion des Hyperbelsinus von x zurückgeben
<code>math.atanh(x)</code>	Umkehrfunktion des Hyperbeltangens von x zurückgeben
<code>math.cosh(x)</code>	Hyperbelkosinus von x zurückgeben
<code>math.sinh(x)</code>	Hyperbelsinus von x zurückgeben
<code>math.tanh(x)</code>	Hyperbeltangens von x zurückgeben

Es gibt auch zwei mathematische Konstanten. Der Wert von `math.pi` ist die mathematische Konstante Pi. Der Wert von `math.e` ist die mathematische Konstante e.

Verwendung von Nicht-ASCII-Zeichen

Damit Nicht-ASCII-Zeichen verwendet werden können, ist bei Python eine explizite Codierung und Decodierung von Zeichenfolgen in Unicode erforderlich. In IBM SPSS Modeler wird davon ausgegangen, dass Python-Skripts in UTF-8 codiert sind, einer Standard-Unicode-Codierung, die Nicht-ASCII-Zeichen unterstützt. Das folgende Skript wird kompiliert, da der Python-Compiler von SPSS Modeler auf UTF-8 gesetzt wurde.

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

Der resultierende Knoten hat jedoch eine falsche Beschriftung.



ãfã, 'ãf^ãf ãf%ãf%

Abbildung 3. Falsch dargestellte Knotenbeschriftung mit Nicht-ASCII

Die Beschriftung ist falsch, da das Zeichenfolgeliteral von Python in eine ASCII-Zeichenfolge konvertiert wurde.

Python erlaubt die Angabe von Unicode-Zeichenfolgeliteralen, indem das Zeichenpräfix `u` vor dem Zeichenfolgeliteral hinzugefügt wird:

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

Hiermit wird eine Unicode-Zeichenfolge erstellt und die Beschriftung wird korrekt dargestellt.



テストノード

Abbildung 4. Korrekt dargestellte Knotenbeschriftung mit Nicht-ASCII

Die Verwendung von Python und Unicode ist ein umfassendes Thema, das über den Rahmen dieses Dokuments hinausgeht. Es gibt viele Handbücher und Onlinere Ressourcen, die sich ausführlich mit diesem Thema befassen.

Objektorientierte Programmierung

Objektorientierte Programmierung basiert auf der Erstellung eines Modells des Zielproblems in Ihren Programmen. Objektorientierte Programmierung verringert Programmierfehler und fördert die Wiederverwendung von Code. Python ist eine objektorientierte Sprache. In Python definierte Objekte haben die folgenden Funktionen:

- **Identität.** Jedes Objekt muss eindeutig sein und diese Bedingung muss testbar sein. Zu diesem Zweck gibt es die Tests `is` und `is not`.
- **Status.** Jedes Objekt muss den Status speichern können. Zu diesem Zweck gibt es Attribute wie Felder und Instanzvariablen.
- **Verhalten.** Jedes Objekt muss seinen Status manipulieren können. Zu diesem Zweck gibt es Methoden.

Python umfasst die folgenden Funktionen zur Unterstützung objektorientierter Programmierung:

- **Klassenbasierte Objekterstellung.** Klassen sind Vorlagen für die Erstellung von Objekten. Objekte sind Datenstrukturen mit zugehörigem Verhalten.
- **Vererbung mit Polymorphie.** Python unterstützt sowohl Einfach- als auch Mehrfachvererbung. Alle Python-Instanzdefinitionsmethoden sind polymorph und können von Unterklassen außer Kraft gesetzt werden.
- **Kapselung mit Ausblenden von Daten.** Python erlaubt das Ausblenden von Attributen. Wenn Attribute ausgeblendet sind, kann von außerhalb der Klasse nur über Methoden der Klasse auf sie zugegriffen werden. Klassen implementieren Methoden, um die Daten zu ändern.

Definieren einer Klasse

In einer Python-Klasse können sowohl Variablen als auch Methoden definiert werden. Anders als in Java können Sie in Python eine beliebige Anzahl öffentlicher Klassen pro Quellendatei (oder *Modul*) definieren. Ein Modul in Python kann daher als Entsprechung eines Pakets in Java betrachtet werden.

In Python werden Klassen mit der Anweisung `class` definiert. Die Anweisung `class` hat das folgende Format:

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

ODER

```
class name (superclasses):  
    Zuweisung  
    .  
    .  
    Funktion  
    .  
    .
```

Beim Definieren einer Klasse haben Sie die Möglichkeit, null oder mehr *Zuordnungsanweisungen* anzugeben. Diese erstellen Klassenattribute, die von allen Instanzen der Klasse gemeinsam genutzt werden. Sie können auch null oder mehr *Funktionsdefinitionen* angeben. Diese Funktionsdefinitionen erstellen Methoden. Die Superklassenliste ist optional.

Der Klassenname sollte in seinem Bereich, d. h. in einem Modul, einer Funktion oder einer Klasse, eindeutig sein. Sie können mehrere Variablen zum Verweis auf dieselbe Klasse definieren.

Erstellen einer Klasseninstanz

Klassen werden zur Aufnahme von (gemeinsam genutzten) Klassenattributen oder zum Erstellen von Klasseninstanzen verwendet. Wenn Sie eine Instanz einer Klasse erstellen wollen, rufen Sie die Klasse so auf, als wäre sie eine Funktion. Beispiel einer Klasse:

```
class MyClass:
    pass
```

Hier wird die Anweisung `pass` verwendet, da eine Anweisung zum Abschließen der Klasse erforderlich ist. Vom Programm aus ist jedoch keine Aktion erforderlich.

Die folgende Anweisung erstellt eine Instanz der Klasse `MyClass`:

```
x = MyClass()
```

Hinzufügen von Attributen zu einer Klasseninstanz

Anders als in Java können Clients in Python Attribute einer Instanz einer Klasse hinzufügen. Nur diese eine Instanz wird geändert. Wenn Sie z. B. Attribute einer Instanz `x` hinzufügen wollen, legen Sie neue Werte für diese Instanz fest:

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

Definieren von Klassenattributen und Methoden

Jede Variable, die in einer Klasse gebunden ist, ist ein *Klassenattribut*. Jede in einer Klasse definierte Funktion ist eine *Methode*. Methoden erhalten als erstes Argument eine Instanz der Klasse, die normalerweise `self` genannt wird. Sie könnten z. B. den folgenden Code eingeben, um einige Klassenattribute und Methoden zu definieren:

```
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
```

In einer Klasse sollten Sie alle Verweise auf Klassenattribute mit dem Klassennamen qualifizieren, z. B. `MyClass.attr1`. Alle Verweise auf Instanzattribute sollten mit der Variablen `self` qualifiziert werden, z. B. `self.text`. Außerhalb der Klasse sollte Sie alle Verweise auf Klassenattribute mit dem Klassennamen

(z. B. `MyClass.attr1`) oder mit einer Instanz der Klasse qualifizieren (z. B. `x.attr1`, wobei `x` eine Instanz der Klasse ist). Außerhalb der Klasse sollten alle Verweise auf Instanzvariablen mit einer Instanz der Klasse qualifiziert werden, z. B. `x.text`.

Ausgeblendete Variablen

Daten können durch das Erstellen *nicht öffentlicher* Variablen ausgeblendet werden. Auf nicht öffentliche Variablen kann nur von der Klasse selbst zugegriffen werden. Wenn Sie Namen der Form `__xxx` oder `__xxx_yyy` deklarieren, d. h. mit zwei vorausgehenden Unterstrichen, fügt der Python-Parser dem deklarierten Namen automatisch den Klassennamen hinzu und erstellt so ausgeblendete Variablen. Beispiel:

```
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
```

Anders als in Java müssen in Python alle Verweise auf Instanzvariablen mit `self` qualifiziert werden; es gibt keine implizierte Verwendung von `this`.

Vererbung

Die Fähigkeit zur Vererbung von Klassen ist elementar für die objektorientierte Programmierung. Python unterstützt sowohl Einfach- als auch Mehrfachvererbung. *Einfachvererbung* bedeutet, dass es nur eine Superklasse geben kann. *Mehrfachvererbung* bedeutet, dass es mehrere Superklasse geben kann.

Die Vererbung wird durch Unterklassenbildung anderer Klassen implementiert. Eine beliebige Anzahl von Python-Klassen können Superklassen sein. In der Jython-Implementierung von Python kann die Vererbung direkt oder indirekt nur von einer Java-Klasse erfolgen. Es muss keine Superklasse angegeben werden.

Jedes Attribut oder jede Methode in einer Superklasse ist auch in jeder Unterklasse enthalten und kann von der Klasse selbst oder von jedem beliebigen Client verwendet werden, sofern das Attribut oder die Methode nicht ausgeblendet ist. Jede Instanz einer Unterklasse kann überall verwendet werden, wo auch eine Instanz einer Superklasse verwendet werden kann; dies ist ein Beispiel für *Polymorphie*. Durch diese Funktionen wird die Wiederverwendung ermöglicht und die Erweiterung erleichtert.

Beispiel

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass    #single inheritance

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

Kapitel 3. Scripting in IBM SPSS Modeler

Scripttypen

In IBM SPSS Modeler gibt es drei Typen von Scripts:

- *Stream-Scripts* werden verwendet, um die Ausführung eines einzelnen Streams zu steuern. Sie werden im Stream gespeichert.
- *Superknotenscripts* werden verwendet, um das Verhalten von Superknoten zu steuern.
- *Standalone- oder Sitzungsscripts* können verwendet werden, um die Ausführung über eine Reihe unterschiedlicher Streams zu koordinieren.

Es stehen verschiedene Methoden zur Verfügung, die in Scripts in IBM SPSS Modeler verwendet werden können. Mit diesen können Sie auf eine Vielzahl von SPSS Modeler-Funktionen zugreifen. Diese Methoden werden auch in Kapitel 4, „Scripting-API“, auf Seite 41 zur Erstellung erweiterter Funktionen verwendet.

Streams, Superknotenstreams und Diagramme

Meistens bedeutet der Begriff *Stream* dasselbe, unabhängig davon, ob es sich um einen Stream handelt, der aus einer Datei geladen oder der in einem Superknoten verwendet wird. Im Allgemeinen ist damit eine Sammlung von Knoten gemeint, die miteinander verbunden sind und ausgeführt werden können. Beim Scripting werden jedoch nicht alle Operationen an allen Stellen unterstützt. Ein Scriptautor sollte sich deshalb bewusst sein, welche Streamvariante er verwendet.

Streams

Ein Stream ist der Hauptdokumenttyp von IBM SPSS Modeler. Er kann gespeichert, geladen, bearbeitet und ausgeführt werden. Streams können auch Parameter, globale Werte, ein Script und weitere zugehörige Informationen haben.

Superknotenstreams

Ein *Superknotenstream* ist der Typ von Stream, der in einem Superknoten verwendet wird. Wie ein normaler Stream enthält er Knoten, die miteinander verbunden sind. Superknotenstreams unterscheiden sich durch eine Reihe von Punkten von einem normalen Stream:

- Parameter und Scripts sind dem Superknoten zugeordnet, dem der Superknotenstream gehört, und nicht dem Superknotenstream selbst.
- Superknotenstreams haben je nach Typ des Superknotens zusätzliche Ein- und Ausgabe-Verbindungsknoten. Diese Verbindungsknoten werden verwendet, um Informationen in den und aus dem Superknotenstream zu leiten. Sie werden automatisch bei der Erstellung des Superknotens erstellt.

Diagramme

Der Begriff *Diagramm* deckt die Funktionen ab, die sowohl von normalen Streams als auch von Superknotenstreams unterstützt werden, z. B. das Hinzufügen und Entfernen von Knoten und das Ändern von Verbindungen zwischen den Knoten.

Ausführen eines Streams

Das folgende Beispiel führt alle ausführbaren Knoten im Stream aus. Es ist der einfachste Typ von Stream-Script:

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

Das folgende Beispiel führt ebenfalls alle ausführbaren Knoten im Stream aus:

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

In diesem Beispiel wird der Stream in einer Variablen namens `stream` gespeichert. Das Speichern des Streams in einer Variablen ist hilfreich, da typischerweise ein Script verwendet wird, um entweder den Stream oder die Knoten in einem Stream zu ändern. Durch die Erstellung einer Variablen, die die Stream-ergebnisse speichert, ergibt sich ein knapperes Script.

Scriptingkontext

Das Modul `modeler.script` stellt den Kontext bereit, in dem ein Script ausgeführt wird. Das Modul wird zur Laufzeit automatisch in ein SPSS Modeler-Script importiert. Das Modul definiert vier Funktionen, die ein Script mit Zugriff auf seine Ausführungsumgebung bereitstellen:

- Die Funktion `session()` gibt die Sitzung für das Script zurück. Die Sitzung definiert Informationen wie die Ländereinstellung und das SPSS Modeler-Back-End (entweder ein lokaler Prozess oder eine vernetzte SPSS Modeler Server-Instanz) für die Ausführung von Streams.
- Die Funktion `stream()` kann in Verbindung mit Stream- und Superknotenscripts verwendet werden. Diese Funktion gibt den Stream zurück, dem das ausgeführte Streamscript oder Superknotenscript gehört.
- Die Funktion `diagram()` kann in Verbindung mit Superknotenscripts verwendet werden. Diese Funktion gibt das Diagramm im Superknoten zurück. Bei anderen Scripttypen hat diese Funktion dieselbe Rückgabe wie die Funktion `stream()`.
- Die Funktion `supernode()` kann in Verbindung mit Superknotenscripts verwendet werden. Diese Funktion gibt den Superknoten zurück, dem das ausgeführte Script gehört.

Die vier Funktionen und ihre Ausgaben sind in der folgenden Tabelle zusammengefasst.

Tabelle 9. Zusammenfassung der `modeler.script`-Funktionen

Scripttyp	<code>session()</code>	<code>stream()</code>	<code>diagram()</code>	<code>supernode()</code>
Standalone	Gibt eine Sitzung zurück	Gibt den aktuellen verwalteten Stream zum Zeitpunkt des Scriptaufrufs zurück (z. B. den Stream, der über die Stapelmodusoption <code>-stream</code> übergeben wird), bzw. <code>None</code> .	Wie bei <code>stream()</code>	Nicht zutreffend
Stream	Gibt eine Sitzung zurück	Gibt einen Stream zurück	Wie bei <code>stream()</code>	Nicht zutreffend
Superknoten	Gibt eine Sitzung zurück	Gibt einen Stream zurück	Gibt einen Superknotenstream zurück	Gibt einen Superknoten zurück

Das Modul `modeler.script` definiert auch eine Möglichkeit zum Beenden des Scripts mit einem Beendigungscode. Die Funktion `exit(exit-code)` stoppt die Ausführung des Scripts und gibt den angegebenen ganzzahligen Beendigungscode zurück.

Eine der für einen Stream definierten Methoden ist `runAll(List)`. Diese Methode führt alle ausführbaren Knoten aus. Alle Modelle oder Ausgaben, die durch die Ausführung der Knoten generiert werden, werden der angegebenen Liste hinzugefügt.

Üblicherweise generiert eine Streamausführung Ausgaben wie Modelle, Grafiken oder andere Ausgaben. Zur Erfassung dieser Ausgabe kann ein Script eine Variable angeben, die in eine Liste initialisiert wird. Beispiel:

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

Wenn die Ausführung abgeschlossen ist, kann von der Ergebnisliste auf die von der Ausführung generierten Objekte zugegriffen werden.

Referenzieren vorhandener Knoten

Ein Stream ist oft mit einigen Parametern vordefiniert, die geändert werden müssen, bevor der Stream ausgeführt werden kann. Die Änderung dieser Parameter umfasst die folgenden Tasks:

1. Suchen der Knoten im entsprechenden Stream.
2. Ändern der Knoten- und/oder Streameinstellungen.

Suchen von Knoten

Streams bieten eine Reihe von Möglichkeiten zum Suchen eines vorhandenen Knotens. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 10. Methoden zum Lokalisieren eines vorhandenen Knotens

Methoden	Rückgabebetyp	Beschreibung
<code>s.findAll(type, label)</code>	Sammlung	Gibt eine Liste aller Knoten mit dem angegebenen Typ und der angegebenen Beschriftung zurück. Entweder der Typ oder die Beschriftung kann None sein. In diesem Fall wird der jeweils andere Parameter verwendet.
<code>s.findAll(filter, recursive)</code>	Sammlung	Gibt eine Sammlung aller Knoten zurück, die vom angegebenen Filter akzeptiert werden. Wenn das rekursive Flag True lautet, werden auch Superknoten im angegebenen Stream gesucht.
<code>s.findById(id)</code>	Knoten	Gibt den Knoten mit der angegebenen ID zurück bzw. None, wenn kein derartiger Knoten vorhanden ist. Die Suche ist auf den aktuellen Stream eingeschränkt.

Tabelle 10. Methoden zum Lokalisieren eines vorhandenen Knotens (Forts.)

Methoden	Rückgabebetyp	Beschreibung
<code>s.findByType(type, label)</code>	Knoten	Gibt den Knoten mit dem angegebenen Typ und/oder der angegebenen Beschriftung zurück. Entweder der Typ oder der Name kann None sein. In diesem Fall wird der jeweils andere Parameter verwendet. Wenn sich für mehreren Knoten eine Übereinstimmung ergibt, wird ein beliebiger ausgewählt und zurückgegeben. Wenn sich für keinen Knoten eine Übereinstimmung ergibt, lautet der Rückgabewert None.
<code>s.findDownstream(fromNodes)</code>	Sammlung	Sucht in der angegebenen Liste von Knoten und gibt das Set von Knoten an, die den angegebenen Knoten nachfolgen. Die zurückgegebene Liste enthält die ursprünglich bereitgestellten Knoten.
<code>s.findUpstream(fromNodes)</code>	Sammlung	Sucht in der angegebenen Liste von Knoten und gibt das Set von Knoten an, die den angegebenen Knoten vorausgehen. Die zurückgegebene Liste enthält die ursprünglich bereitgestellten Knoten.

Wenn z. B. ein Stream einen einzigen Filterknoten enthält, auf den das Script zugreifen muss, kann der Filterknoten mithilfe des folgenden Scripts gefunden werden:

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

Wenn die ID des Knotens (siehe Registerkarte "Anmerkungen" im Knotendialogfeld) bekannt ist, kann anhand der ID nach dem Knoten gesucht werden. Beispiel:

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

Festlegen von Eigenschaften

Knoten, Streams, Modelle und Ausgaben haben Eigenschaften, die abgerufen und in den meisten Fällen auch festgelegt werden können. Eigenschaften werden üblicherweise verwendet, um das Verhalten oder Aussehen des Objekts zu ändern. Die verfügbaren Methoden für den Zugriff und das Festlegen von Objekteigenschaften sind in der folgenden Tabelle zusammengefasst.

Tabelle 11. Methoden zum Zugreifen auf Objekteigenschaften und zum Festlegen dieser Eigenschaften

Methoden	Rückgabebetyp	Beschreibung
<code>p.getPropertyValue(propertyName)</code>	Objekt	Gibt den Wert der angegebenen Eigenschaft zurück, bzw. None, wenn keine solche Eigenschaft vorhanden ist.
<code>p.setPropertyValue(propertyName, value)</code>	Nicht zutreffend	Definiert den Wert der angegebenen Eigenschaft.

Tabelle 11. Methoden zum Zugreifen auf Objekteigenschaften und zum Festlegen dieser Eigenschaften (Forts.)

Methoden	Rückgabotyp	Beschreibung
<code>p.setPropertyValues(properties)</code>	Nicht zutreffend	Definiert die Werte der angegebenen Eigenschaften. Jeder Eintrag in der Eigenschaftszuordnung besteht aus einem Schlüssel, der den Eigenschaftsnamen und den Wert darstellt, der der entsprechenden Eigenschaft zugewiesen werden sollte.
<code>p.getKeyedPropertyValue(propertyName, keyName)</code>	Objekt	Gibt den Wert der angegebenen Eigenschaft und des zugehörigen Schlüssels zurück, bzw. None, wenn keine solche Eigenschaft oder kein solcher Schlüssel vorhanden ist.
<code>p.setKeyedPropertyValue(propertyName, keyName, value)</code>	Nicht zutreffend	Definiert den Wert der angegebenen Eigenschaft und des Schlüssels.

Wenn Sie z. B. den Wert eines Knotens **Variable Datei** am Anfang eines Streams festlegen wollen, können Sie das folgende Script verwenden:

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

Alternativ könnten Sie ein Feld aus einem Filterknoten filtern. In diesem Fall wird der Wert auch in den Feldnamen eingegeben. Beispiel:

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

Erstellen von Knoten und Ändern von Streams

In einigen Situationen wollen Sie möglicherweise vorhandenen Streams neue Knoten hinzufügen. Die Hinzufügung von Knoten zu vorhandenen Streams umfasst die folgenden Tasks:

1. Erstellen der Knoten.
2. Aktivieren von Links für die Knoten in der vorhandenen Streamfolge.

Erstellen von Knoten

Streams bieten eine Reihe von Möglichkeiten zum Erstellen von Knoten. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 12. Methoden zum Erstellen von Knoten

Methoden	Rückgabotyp	Beschreibung
<code>s.create(nodeType, name)</code>	Knoten	Erstellt einen Knoten des angegebenen Typs und fügt ihn dem angegebenen Stream hinzu.
<code>s.createAt(nodeType, name, x, y)</code>	Knoten	Erstellt einen Knoten des angegebenen Typs und fügt ihn dem angegebenen Stream an der angegebenen Position hinzu. Bei $x < 0$ oder $y < 0$ ist die Position nicht festgelegt.

Table 12. Methoden zum Erstellen von Knoten (Forts.)

Method	Return type	Description
<code>s.createModelApplier(modelOutput, name)</code>	Knoten	Erstellt einen Modellanwendungsknoten, der vom angegebenen Modellausgabeobjekt abgeleitet wird.

You can use the following script to create a new type node in a stream, for example:

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

Activating and Removing Links for Nodes

When a new node is created in a stream, it must be connected to other nodes in the stream before it can be used. Streams offer a number of options for activating and removing links for nodes. These methods are summarized in the following table.

Table 13. Methoden zum Aktivieren und Aufheben von Links für Knoten

Method	Return type	Description
<code>s.link(source, target)</code>	Nicht zutreffend	Erstellt einen neuen Link zwischen dem Quellen- und dem Zielknoten.
<code>s.link(source, targets)</code>	Nicht zutreffend	Erstellt neue Links zwischen dem Quellenknoten und jedem Zielknoten in der angegebenen Liste.
<code>s.linkBetween(inserted, source, target)</code>	Nicht zutreffend	Verbindet einen Knoten zwischen zwei anderen Knoteninstanzen (dem Quellen- und dem Zielknoten) und legt die Position des eingefügten Knotens zwischen diesen beiden fest. Jeder direkte Link zwischen dem Quellen- und dem Zielknoten wird zuerst entfernt.
<code>s.linkPath(path)</code>	Nicht zutreffend	Erstellt einen neuen Pfad zwischen Knoteninstanzen. Der erste Knoten wird mit dem zweiten verbunden, der zweite wird mit dem dritten verbunden usw.
<code>s.unlink(source, target)</code>	Nicht zutreffend	Entfernt jeden direkten Link zwischen dem Quellen- und dem Zielknoten.
<code>s.unlink(source, targets)</code>	Nicht zutreffend	Entfernt alle direkten Links zwischen dem Quellenknoten und jedem Objekt in der Zielliste.
<code>s.unlinkPath(path)</code>	Nicht zutreffend	Entfernt jeden Pfad zwischen Knoteninstanzen.
<code>s.disconnect(node)</code>	Nicht zutreffend	Entfernt alle Links zwischen dem angegebenen Knoten und allen anderen Knoten im angegebenen Stream.

Tabelle 13. Methoden zum Aktivieren und Aufheben von Links für Knoten (Forts.)

Methoden	Rückgabebetyp	Beschreibung
<code>s.isValidLink(source, target)</code>	<i>boolesch</i>	Gibt True zurück, wenn die Erstellung eines Links zwischen dem angegebenen Quellen- und Zielknoten zulässig wäre. Diese Methode prüft, dass beide Objekte zum angegebenen Stream gehören, dass der Quellenknoten einen Link bereitstellen kann, dass der Zielknoten einen Link empfangen kann und dass die Erstellung eines solchen Links keinen Zirkelbezug im Stream verursacht.

Das folgende Beispielscript führt die folgenden fünf Tasks durch:

1. Es erstellt einen Eingabeknoten "Variable Datei", einen Filterknoten und einen Tabellenausgabeknoten.
2. Es verbindet die Knoten miteinander.
3. Es legt den Dateinamen im Eingabeknoten "Variable Datei" fest.
4. Es filtert das Feld "Drug" aus der Ergebnisausgabe.
5. Es führt den Tabellenknoten aus.

```
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)
```

Importieren, Ersetzen und Löschen von Knoten

Neben dem Erstellen und Verbinden von Knoten ist es oft auch erforderlich, Knoten zu ersetzen oder aus dem Stream zu löschen. Die verfügbaren Methoden zum Importieren, Ersetzen und Löschen von Knoten sind in der folgenden Tabelle zusammengefasst.

Tabelle 14. Methoden zum Importieren, Ersetzen oder Löschen von Knoten

Methoden	Rückgabebetyp	Beschreibung
<code>s.replace(originalNode, replacementNode, discardOriginal)</code>	Nicht zutreffend	Ersetzt den angegebenen Knoten im angegebenen Stream. Sowohl der Originalknoten als auch der Ersetzungsknoten müssen dem angegebenen Stream gehören.

Tabelle 14. Methoden zum Importieren, Ersetzen oder Löschen von Knoten (Forts.)

Methoden	Rückgabebetyp	Beschreibung
<code>s.insert(source, nodes, newIDs)</code>	Liste	Fügt Kopien der Knoten in der angegebenen Liste ein. Es wird angenommen, dass alle Knoten in der angegebenen Liste im angegebenen Stream enthalten sind. Das Flag <code>newIDs</code> gibt an, ob für jeden Knoten eine neue ID generiert werden soll oder ob die vorhandene ID kopiert und verwendet werden soll. Es wird angenommen, dass alle Knoten in einem Stream eine eindeutige ID haben. Dieses Flag muss daher auf <code>True</code> gesetzt werden, wenn der Quellenstream dem angegebenen Stream entspricht. Die Methode gibt die Liste neu eingefügter Knoten zurück, wobei die Reihenfolge der Knoten nicht definiert ist (d. h. die Reihenfolge entspricht nicht unbedingt der Reihenfolge der Knoten in der Eingabeliste).
<code>s.delete(node)</code>	Nicht zutreffend	Löscht den angegebenen Knoten aus dem angegebenen Stream. Der Knoten muss dem angegebenen Stream gehören.
<code>s.deleteAll(nodes)</code>	Nicht zutreffend	Löscht alle angegebenen Knoten aus dem angegebenen Stream. Alle Knoten in der Sammlung müssen dem angegebenen Stream gehören.
<code>s.clear()</code>	Nicht zutreffend	Löscht alle Knoten aus dem angegebenen Stream.

Traversieren durch Knoten in einem Stream

Eine allgemeine Voraussetzung ist die Ermittlung von Knoten, die einem bestimmten Knoten vorangehen oder nachfolgen. Der Stream bietet eine Reihe von Methoden, die verwendet werden können, um diese Knoten zu ermitteln. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 15. Methoden zum Angeben vorausgehender und nachfolgender Streams

Methoden	Rückgabebetyp	Beschreibung
<code>s.iterator()</code>	Iterator	Gibt einen Iterator über die Knotenobjekte zurück, die im angegebenen Stream enthalten sind. Wenn der Stream zwischen Aufrufen der Funktion <code>next()</code> geändert wird, kann das Verhalten des Iterators undefiniert sein.
<code>s.predecessorAt(node, index)</code>	Knoten	Gibt den angegebenen unmittelbaren Vorgänger des angegebenen Knotens zurück, bzw. <code>None</code> , wenn der Index außerhalb des gültigen Bereichs liegt.
<code>s.predecessorCount(node)</code>	Ganzz	Gibt die Anzahl der unmittelbaren Vorgänger des angegebenen Knotens zurück.

Tabelle 15. Methoden zum Angeben vorausgehender und nachfolgender Streams (Forts.)

Methoden	Rückgabotyp	Beschreibung
s.predecessors(node)	Liste	Gibt die unmittelbaren Vorgänger des angegebenen Knotens zurück.
s.successorAt(node, index)	Knoten	Gibt den angegebenen unmittelbaren Nachfolger des angegebenen Knotens zurück, bzw. None, wenn der Index außerhalb des gültigen Bereichs liegt.
s.successorCount(node)	Ganzz	Gibt die Anzahl der unmittelbaren Nachfolger des angegebenen Knotens zurück.
s.successors(node)	Liste	Gibt die unmittelbaren Nachfolger des angegebenen Knotens zurück.

Entfernen von Elementen

Traditionelles Scripting unterstützt verschiedene Verwendungen des Befehls `clear`, z. B.:

- `clear outputs` löscht alle Ausgabeelemente aus der Managerpalette.
- `clear generated palette` löscht alle Modellnuggets aus der Modellpalette.
- `clear stream` entfernt den Inhalt eines Streams.

Python-Scripting unterstützt ein ähnliches Set von Funktionen. Mit dem Befehl `removeAll()` werden die Stream-, Ausgabe- und Modellmanager gelöscht. Beispiele:

- So löschen Sie den Stream-Manager:


```
session = modeler.script.session()
session.getStreamManager.removeAll()
```
- So löschen Sie den Ausgabemanager:


```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```
- So löschen Sie den Modellmanager:


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

Abrufen von Informationen zu Knoten

Knoten fallen in eine Reihe unterschiedlicher Kategorien wie Datenimport- und -exportknoten, Modellerstellungsknoten und andere Typen von Knoten. Jeder Knoten stellt eine Reihe von Methoden bereit, mit denen Informationen zum Knoten abgerufen werden können.

Die Methoden, mit denen die ID, der Name und die Beschriftung eines Knotens abgerufen werden können, sind in der folgenden Tabelle zusammengefasst.

Tabelle 16. Methoden zum Abrufen von ID, Name und Beschriftung eines Knotens

Methoden	Rückgabebetyp	Beschreibung
n.getLabel()	Zeichenfolge	Gibt die Anzeigebeschriftung des angegebenen Knotens zurück. Die Beschriftung entspricht nur dann dem Wert der Eigenschaft custom_name, wenn diese Eigenschaft eine nicht leere Zeichenfolge ist und die Eigenschaft use_custom_name nicht definiert ist; andernfalls entspricht die Beschriftung dem Wert von getName().
n.setLabel(label)	Nicht zutreffend	Legt die Anzeigebeschriftung des angegebenen Knotens fest. Wenn die neue Beschriftung eine nicht leere Zeichenfolge ist, wird sie der Eigenschaft custom_name zugewiesen und die Eigenschaft use_custom_name wird auf False gesetzt, damit die angegebene Beschriftung Vorrang hat; andernfalls wird der Eigenschaft custom_name eine leere Zeichenfolge zugewiesen und die Eigenschaft use_custom_name wird auf True gesetzt.
n.getName()	Zeichenfolge	Gibt den Namen des angegebenen Knotens zurück.
n.getID()	Zeichenfolge	Gibt die ID des angegebenen Knotens zurück. Bei jeder Erstellung eines neuen Knotens wird eine neue ID erzeugt. Die ID wird im Knoten als persistent definiert, wenn dieser als Teil eines Streams gespeichert wird, damit beim Öffnen des Streams die Knoten-IDs beibehalten werden. Wenn jedoch ein gespeicherter Knoten in einen Stream eingefügt wird, gilt der eingefügte Knoten als neues Objekt und ihm wird eine neue ID zugewiesen.

Die Methoden, mit denen weitere Informationen zu einem Knoten abgerufen werden können, sind in der folgenden Tabelle zusammengefasst.

Tabelle 17. Methoden zum Abrufen von Informationen zu einem Knoten

Methoden	Rückgabebetyp	Beschreibung
n.getTypeName()	Zeichenfolge	Gibt den Scriptnamen dieses Knotens zurück. Dies ist derselbe Name, der zum Erstellen einer neuen Instanz dieses Knoten verwendet werden könnte.
n.isInitial()	boolesch	Gibt True zurück, wenn dies ein Ursprungsknoten ist, d. h. ein Knoten, der am Anfang eines Streams auftritt.

Tabelle 17. Methoden zum Abrufen von Informationen zu einem Knoten (Forts.)

Methoden	Rückgabebetyp	Beschreibung
<code>n.isInline()</code>	<i>boolesch</i>	Gibt True zurück, wenn dies ein <i>Inline-Knoten</i> ist, d. h. ein Knoten, der in der Mitte eines Streams auftritt.
<code>n.isTerminal()</code>	<i>boolesch</i>	Gibt True zurück, wenn dies ein <i>Endknoten</i> ist, d. h. ein Knoten, der am Ende eines Streams auftritt.
<code>n.getXPosition()</code>	<i>Ganzz</i>	Gibt den Offset der X-Position des Knotens im Stream an.
<code>n.getYPosition()</code>	<i>Ganzz</i>	Gibt den Offset der Y-Position des Knotens im Stream an.
<code>n.setXYPosition(x, y)</code>	Nicht zutreffend	Legt die Position des Knotens im Stream fest.
<code>n.setPositionBetween(source, target)</code>	Nicht zutreffend	Legt die Position des Knotens im Stream so fest, dass er zwischen den angegebenen Knoten angeordnet ist.
<code>n.isCacheEnabled()</code>	<i>boolesch</i>	Gibt True zurück, wenn der Cache aktiviert ist; andernfalls wird False zurückgegeben.
<code>n.setCacheEnabled(val)</code>	Nicht zutreffend	Aktiviert oder inaktiviert den Cache für dieses Objekt. Wenn der Cache voll ist und das Caching inaktiviert wird, wird der Cache geleert.
<code>n.isCacheFull()</code>	<i>boolesch</i>	Gibt True zurück, wenn der Cache voll ist; andernfalls wird False zurückgegeben.
<code>n.flushCache()</code>	Nicht zutreffend	Leert den Cache dieses Knotens. Hat keine Auswirkung, wenn der Cache nicht aktiviert oder nicht voll ist.

Kapitel 4. Scripting-API

Einführung in die Scripting-API

Die Scripting-API bietet Zugriff auf eine Vielzahl von SPSS Modeler-Funktionen. Alle bisher beschriebenen Methoden sind Teil der API und können ohne weitere Importe implizit im Script aufgerufen werden. Wenn Sie jedoch auf die API-Klassen verweisen wollen, müssen Sie die API mit der folgenden Anweisung explizit importieren:

```
import modeler.api
```

Diese Importanweisung ist für viele Beispiele der Scripting-API erforderlich.

Eine vollständige Beschreibung der Klassen, Methoden und Parameter, die über die Scripting-API verfügbar sind, finden Sie im Handbuch *IBM SPSS Modeler 17 Python Scripting API Reference Guide*.

Beispiel: Suchen nach Knoten mit einem benutzerdefinierten Filter

Im Abschnitt „Suchen von Knoten“ auf Seite 31 ist ein Beispiel für die Suche nach einem Knoten in einem Stream enthalten, wobei der Typname des Knotens als Suchkriterium verwendet wird. In einigen Situationen ist eine allgemeinere Suche erforderlich. Diese kann mit der Klasse `NodeFilter` und der Streammethode `findAll()` implementiert werden. Diese Art von Suche umfasst die folgenden beiden Schritte:

1. Erstellen einer neuen Klasse, die `NodeFilter` erweitert und eine benutzerdefinierte Version der Methode `accept()` implementiert.
2. Aufrufen der Streammethode `findAll()` mit einer Instanz dieser neuen Klasse. Dadurch werden alle Knoten zurückgegeben, die die in der Methode `accept()` definierten Kriterien erfüllen.

Im folgenden Beispiel wird gezeigt, wie in einem Stream nach Knoten mit aktiviertem Knotencache gesucht werden kann. Die Liste der zurückgegebenen Knoten könnte verwendet werden, um die Caches dieser Knoten zu leeren oder zu inaktivieren.

```
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)
```

Metadaten: Informationen zu Daten

Da Knoten in einem Stream miteinander verbunden sind, sind Informationen zu den in jedem Knoten verfügbaren Spalten oder Feldern verfügbar. In der Modeler-Benutzerschnittstelle können Sie so z. B. auswählen, nach welchen Felder sortiert oder kumuliert werden soll. Diese Informationen werden als Datenmodell bezeichnet.

Scripts können auch auf das Datenmodell zugreifen, indem sie die Felder betrachten, die in einen Knoten eintreten oder aus einem Knoten austreten. Bei einigen Knoten sind das Eingabe- und das Ausgabedatenmodell gleich. Beispielsweise ordnet ein Knoten "sort" die Datensätze einfach um, ändert jedoch nicht das Datenmodell. Einige, wie der Knoten "derive", können neue Felder hinzufügen. Andere, wie der Knoten "filter", können Felder umbenennen oder entfernen.

Im folgenden Beispiel nimmt das Script den IBM SPSS Modeler-Standardstream `druglearn.str` und erstellt für jedes Feld ein Modell, wobei eines der Eingabefelder gelöscht wird. Die Vorgehensweise sieht dabei folgendermaßen aus:

1. Zugreifen auf das Ausgabedatenmodell aus dem Typknoten
2. Durchlaufen jedes Felds im Ausgabedatenmodell in einer Schleife
3. Ändern des Knotens "filter" für jedes Eingabefeld
4. Ändern des Namens des zu erstellenden Modells
5. Ausführen des Modellerstellungsknotens

Anmerkung: Bevor Sie das Script im Stream `druglearn.str` ausführen, müssen Sie die Scriptsprache auf Python setzen. (Der Stream wurde in einer Vorgängerversion von IBM SPSS Modeler erstellt; die Scriptsprache des Streams ist also auf "Legacy" eingestellt.)

```
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([])
```

Das Datenmodellobjekt bietet eine Reihe von Methoden für den Zugriff auf Informationen zu den Feldern oder Spalten im Datenmodell. Diese Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 18. Methoden des Datenmodellobjekts für den Zugriff auf Informationen zu Feldern oder Spalten

Methoden	Rückgabtyp	Beschreibung
<code>d.getColumnCount()</code>	<i>Ganzz</i>	Gibt die Anzahl der Spalten im Datenmodell zurück.
<code>d.columnIterator()</code>	Iterator	Gibt einen Iterator zurück, der seinerseits jede Spalte in der "natürlichen" Einfügereihenfolge zurückgibt. Der Iterator gibt Spalteninstanzen zurück.
<code>d.nameIterator()</code>	Iterator	Gibt einen Iterator zurück, der seinerseits den Namen jeder Spalte in der "natürlichen" Einfügereihenfolge zurückgibt.

Tabelle 18. Methoden des Datenmodellobjekts für den Zugriff auf Informationen zu Feldern oder Spalten (Forts.)

Methoden	Rückgabotyp	Beschreibung
d.contains(name)	boolesch	Gibt True zurück, wenn eine Spalte mit dem angegebenen Namen in diesem Datenmodell vorhanden ist; andernfalls wird False zurückgegeben.
d.getColumn(name)	Spalte	Gibt die Spalte mit dem angegebenen Namen zurück.
d.getColumnGroup(name)	Spaltengruppe	Gibt die angegebene Spaltengruppe zurück, bzw. None, wenn keine solche Spaltengruppe vorhanden ist.
d.getColumnGroupCount()	Ganzz	Gibt die Anzahl der Spaltengruppen in diesem Datenmodell zurück.
d.columnGroupIterator()	Iterator	Gibt einen Iterator zurück, der nacheinander jede Spaltengruppe zurückgibt.
d.toArray()	Spalte[]	Gibt das Datenmodell als Array von Spalten zurück. Die Spalten sind in ihrer "natürlichen" Einfügereihenfolge geordnet.

Jedes Feld (Spaltenobjekt) umfasst eine Reihe von Methoden für den Zugriff auf Informationen zu der Spalte. Die folgende Tabelle enthält eine Auswahl davon.

Tabelle 19. Spaltenobjektmethoden für den Zugriff auf Informationen zu der Spalte

Methoden	Rückgabotyp	Beschreibung
c.getColumnName()	Zeichenfolge	Gibt den Namen der Spalte zurück.
c.getColumnLabel()	Zeichenfolge	Gibt die Beschriftung der Spalte zurück bzw. eine leere Zeichenfolge, wenn der Spalte keine Beschriftung zugeordnet ist.
c.getMeasureType()	Maßtyp	Gibt den Maßtyp für die Spalte zurück.
c.getStorageType()	Speichertyp	Gibt den Speichertyp für die Spalte zurück.
c.isMeasureDiscrete()	boolesch	Gibt True zurück, wenn die Spalte diskret ist. Spalten, die entweder ein Set oder ein Flag sind, werden als diskret betrachtet.
c.isModelOutputColumn()	boolesch	Gibt True zurück, wenn die Spalte eine Modellausgabespalte ist.
c.isStorageDatetime()	boolesch	Gibt True zurück, wenn der Speicher der Spalte ein Uhrzeit-, ein Datum- oder ein Zeitmarkenwert ist.
c.isStorageNumeric()	boolesch	Gibt True zurück, wenn der Speicher der Spalte eine ganze oder eine reelle Zahl ist.
c.isValidValue(value)	boolesch	Gibt True zurück, wenn der angegebene Wert für diesen Speicher gültig ist, und gibt valid zurück, wenn die gültigen Spaltenwerte bekannt sind.

Tabelle 19. Spaltenobjektmethoden für den Zugriff auf Informationen zu der Spalte (Forts.)

Methoden	Rückgabebetyp	Beschreibung
<code>c.getModelingRole()</code>	Modellierungsrolle	Gibt die Modellierungsrolle für die Spalte zurück.
<code>c.getSetValues()</code>	Objekt[]	Gibt ein Array gültiger Werte für die Spalte zurück, bzw. None, wenn die Werte nicht bekannt sind oder wenn die Spalte kein Set ist.
<code>c.getValueLabel(value)</code>	Zeichenfolge	Gibt die Beschriftung für den Wert in der Spalte zurück bzw. eine leere Zeichenfolge, wenn dem Wert keine Beschriftung zugeordnet ist.
<code>c.getFalseFlag()</code>	Objekt	Gibt den Indikatorwert "false" für die Spalte zurück, bzw. None, wenn der Wert nicht bekannt ist oder wenn die Spalte kein Flag ist.
<code>c.getTrueFlag()</code>	Objekt	Gibt den Indikatorwert "true" für die Spalte zurück, bzw. None, wenn der Wert nicht bekannt ist oder wenn die Spalte kein Flag ist.
<code>c.getLowerBound()</code>	Objekt	Gibt den unteren Grenzwert für die Werte in der Spalte zurück, bzw. None, wenn der Wert nicht bekannt ist oder wenn die Spalte nicht fortlaufend ist.
<code>c.getUpperBound()</code>	Objekt	Gibt den oberen Grenzwert für die Werte in der Spalte zurück, bzw. None, wenn der Wert nicht bekannt ist oder wenn die Spalte nicht fortlaufend ist.

Für die meisten der Methoden, die auf Informationen zu einer Spalte zugreifen, sind entsprechende Methoden im Datenmodellobjekt selbst definiert. Die folgenden beiden Anweisungen sind beispielsweise funktional entsprechend:

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

Zugriff auf generierte Objekte

Bei der Ausführung eines Streams werden in der Regel zusätzliche Ausgabeobjekte erzeugt. Diese zusätzlichen Objekte könnten ein neues Modell oder eine Ausgabe sein, die Informationen bereitstellt, die in nachfolgenden Ausführungen verwendet werden.

Im Beispiel unten wird der Stream `druglearn.str` erneut als Ausgangspunkt für den Stream verwendet. In diesem Beispiel werden alle Knoten im Stream ausgeführt und die Ergebnisse werden in einer Liste gespeichert. Das Script durchläuft die Ergebnisse dann in einer Schleife und alle Modellausgaben aus der Ausführung werden als IBM SPSS Modeler-Modelldatei (`.gm`) gespeichert und das Modell wird im PMML-Format exportiert.

```
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)

```

Die Klasse der ausführbaren Komponente bietet eine praktische Möglichkeit zum Ausführen verschiedener allgemeiner Tasks. Die in dieser Klasse verfügbaren Methoden werden in der folgenden Tabelle zusammengefasst.

Tabelle 20. Methode der Klasse der ausführbaren Komponente zum Ausführen allgemeiner Tasks

Methoden	Rückgabertyp	Beschreibung
t.createStream(Name, autoVerbindung, autoVerwaltung)	Stream	Erstellt einen neuen Stream und gibt ihn zurück. Für Code, der Streams nicht öffentlich erstellen muss, ohne sie dem Benutzer sichtbar zu machen, sollte das Flag autoManage auf False gesetzt werden.
t.exportDocumentToFile(Dokumentaushabe, Dateiname, Dateiformat)	Nicht zutreffend	Exportiert die Streambeschreibung unter Verwendung des angegebenen Dateiformats in eine Datei.
t.exportModelToFile(modelOutput, filename, fileFormat)	Nicht zutreffend	Exportiert das Modell unter Verwendung des angegebenen Dateiformats in eine Datei.
t.exportStreamToFile(stream, filename, fileFormat)	Nicht zutreffend	Exportiert den Stream unter Verwendung des angegebenen Dateiformats in eine Datei.
t.insertNodeFromFile(filename, diagram)	Knoten	Liest einen Knoten aus der angegebenen Datei, gibt ihn zurück und fügt ihn in das angegebene Diagramm ein. Damit können sowohl Knoten- als auch Superknotenobjekte gelesen werden.
t.openDocumentFromFile(filename, autoManage)	Dokumentaushabe	Liest ein Dokument aus der angegebenen Datei und gibt es zurück.
t.openModelFromFile(filename, autoManage)	Modellaushabe	Liest ein Modell aus der angegebenen Datei und gibt es zurück.
t.openStreamFromFile(filename, autoManage)	Stream	Liest einen Stream aus der angegebenen Datei und gibt ihn zurück.
t.saveDocumentToFile(documentOutput, filename)	Nicht zutreffend	Speichert das Dokument an der angegebenen Dateiposition.

Tabelle 20. Methode der Klasse der ausführbaren Komponente zum Ausführen allgemeiner Tasks (Forts.)

Methodenname	Rückgabebetyp	Beschreibung
t.saveModelToFile(modelOutput, filename)	Nicht zutreffend	Speichert das Modell an der angegebenen Dateiposition.
t.saveStreamToFile(stream, filename)	Nicht zutreffend	Speichert den Stream an der angegebenen Dateiposition.

Fehlerbehandlung

Die Python-Sprache bietet Fehlerbehandlung über den Codeblock `try...except`. Dieser kann in Scripts verwendet werden, um Ausnahmebedingungen abzufangen und um Probleme zu behandeln, die sonst zur Beendigung des Scripts führen würden.

Im Beispielscript unten wird versucht, ein Modell aus IBM SPSS Collaboration and Deployment Services Repository abzurufen. Diese Operation kann dazu führen, dass eine Ausnahmebedingung ausgelöst wird. Beispielsweise könnten die Berechtigungsnachweise für die Repository-Anmeldung nicht korrekt eingerichtet sein oder der Repository-Pfad ist falsch. Im Script kann dies dazu führen, dass eine `ModelerException`-Ausnahmebedingung ausgelöst wird (alle von IBM SPSS Modeler generierten Ausnahmebedingungen sind von `modeler.api.ModelerException` abgeleitet).

```
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()
```

Anmerkung: Einige Scriptoperation können dazu führen, dass Java-Standardausnahmebedingungen ausgelöst werden. Diese sind nicht von `ModelerException` abgeleitet. Zum Abfangen aller Java-Ausnahmebedingungen kann ein zusätzlicher `except`-Block verwendet werden. Beispiel:

```
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()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

Stream-, Sitzungs- und Superknotenparameter

Parameter sind eine praktische Möglichkeit, um Werte zur Laufzeit zu übergeben, statt sie direkt im Script fest zu codieren. Parameter und ihre Werte werden auf dieselbe Weise definiert wie für Streams, d. h. als Einträge in der Parametertabelle eines Streams oder Superknotens oder als Parameter in der Befehlszeile. Die Stream- und Superknotenklassen implementieren eine Gruppe von Funktionen, die vom `ParameterProvider`-Objekt definiert werden (siehe folgende Tabelle). Die Sitzung stellt einen Aufruf `getParameters()` bereit, der ein Objekt zurückgibt, das diese Funktionen definiert.

Table 21. Vom ParameterProvider-Objekt definierte Funktionen

Methoden	Rückgabotyp	Beschreibung
p.parameterIterator()	Iterator	Gibt einen Iterator von Parameternamen für dieses Objekt zurück.
p.getParameterDefinition(parameterName)	ParameterDefinition	Gibt die Parameterdefinition für den Parameter mit dem angegebenen Namen zurück bzw. None, wenn kein solcher Parameter in diesem Provider existiert. Das Ergebnis kann eine Momentaufnahme der Definition zum Zeitpunkt des Aufrufs der Methode sein und spiegelt nicht unbedingt nachfolgende Änderungen am Parameter durch den Provider wider.
p.getParameterLabel(parameterName)	Zeichenfolge	Gibt die Beschriftung des angegebenen Parameters zurück bzw. None, wenn kein solcher Parameter vorhanden ist.
p.setParameterLabel(parameterName, label)	Nicht zutreffend	Definiert die Beschriftung des angegebenen Parameters.
p.getParameterStorage(parameterName)	ParameterStorage	Gibt den Speicher des angegebenen Parameters zurück bzw. None, wenn kein solcher Parameter vorhanden ist.
p.setParameterStorage(parameterName, storage)	Nicht zutreffend	Definiert den Speicher des angegebenen Parameters.
p.getParameterType(parameterName)	ParameterType	Gibt den Typ des angegebenen Parameters zurück bzw. None, wenn kein solcher Parameter vorhanden ist.
p.setParameterType(parameterName, type)	Nicht zutreffend	Definiert den Typ des angegebenen Parameters.
p.getParameterValue(parameterName)	Objekt	Gibt den Wert des angegebenen Parameters zurück bzw. None, wenn kein solcher Parameter vorhanden ist.
p.setParameterValue(parameterName, value)	Nicht zutreffend	Definiert den Wert des angegebenen Parameters.

Im folgenden Beispiel aggregiert das Script einige Telekommunikationsdaten, um die Region mit den niedrigsten durchschnittlichen Einnahmen zu ermitteln. Dann wird ein Streamparameter mit dieser Region definiert. Dieser Streamparameter wird dann in einem Auswahlknoten verwendet, um diese Region aus den Daten auszuschließen, bevor mit den restlichen Daten ein Abwanderungsmodell erstellt wird.

Dieses Beispiel ist konstruiert, da das Script den Auswahlknoten selbst generiert und daher den korrekten Wert direkt in den Ausdruck für den Auswahlknoten generiert haben könnte. Streams sind jedoch in der Regel vordefiniert, weshalb die Festlegung von Parametern auf diese Weise ein nützliches Beispiel darstellt.

Der erste Teil des Beispielscripts erstellt den Streamparameter, der die Region mit den niedrigsten durchschnittlichen Einnahmen enthält. Das Script erstellt auch die Knoten in der Aggregationsverzweigung und in der Modellerstellungsverzweigung und verbindet diese miteinander.

```
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)

```

Das Beispielscript erzeugt den folgenden Stream.

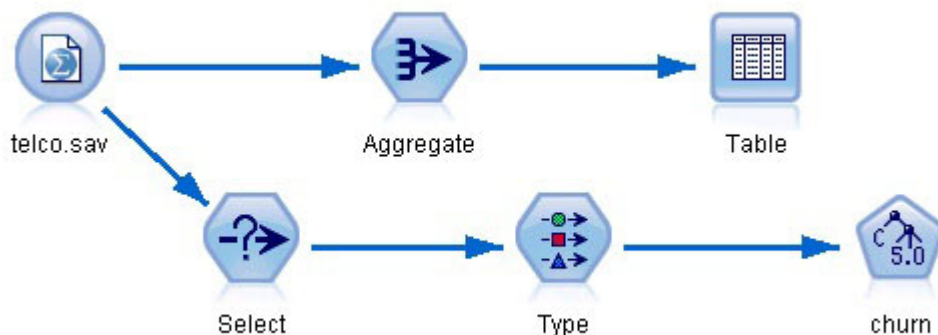


Abbildung 5. Stream, der aus dem Beispielscript resultiert

Der folgende Teil des Beispielscripts führt den Tabellenknoten am Ende der Aggregationsverzweigung aus.

```

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

```

Der folgende Teil des Beispielscripts greift auf die Tabellenausgabe zu, die durch die Ausführung des Tabellenknotens generiert wurde. Das Script durchläuft dann die Zeilen in der Tabelle und sucht nach den niedrigsten Durchschnittseinnahmen.

```

# 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

```

Der folgende Teil des Scripts verwendet die Region mit den niedrigsten Durchschnittseinnahmen zur Definition des zuvor erstellten Streamparameters "LowestRegion". Das Script führt dann das Modellerstellungsprogramm aus, wobei die angegebene Region aus den Trainingsdaten ausgeschlossen wird.

```

# 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([])

```

Das vollständige Beispielscript ist im Folgenden aufgeführt.

```

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([])

```

Globale Werte

Globale Werte werden zum Berechnen verschiedener Auswertungsstatistiken für angegebene Felder verwendet. Diese Auswertungswerte sind überall im Stream zugänglich. Globale Werte sind den Streamparametern darin ähnlich, dass über den Stream anhand des Namens auf sie zugegriffen wird. Sie unterscheiden sich von Streamparametern darin, dass die zugehörigen Werte automatisch bei der Ausführung eines Globalwerteknotens aktualisiert werden und nicht vom Scripting oder von der Befehlszeile zugewiesen werden. Der Zugriff auf die globalen Werte für einen Stream erfolgt über den Aufruf der Methode `getGlobalValues()` des Streams.

Das `GlobalValues`-Objekt definiert die Funktionen, die in der folgenden Tabelle aufgelistet werden.

Tabelle 22. Vom `GlobalValues`-Objekt definierte Funktionen

Methodenname	Rückgabotyp	Beschreibung
<code>g.fieldNameIterator()</code>	Iterator	Gibt einen Iterator für jeden Feldnamen mit mindestens einem globalen Wert zurück.
<code>g.getValue(type, fieldName)</code>	Objekt	Gibt den globalen Wert für den angegebenen Typ und Feldnamen zurück bzw. <code>None</code> , wenn kein Wert gefunden werden kann. Als zurückgegebener Wert wird im Allgemeinen eine Zahl erwartet. Einige künftige Funktionen könnten jedoch auch andere Wertetypen zurückgeben.
<code>g.getValues(fieldName)</code>	Zuordnung	Gibt eine Zuordnung zurück, die die bekannten Einträge für den angegebenen Feldnamen enthält, bzw. <code>None</code> , wenn es keine vorhandenen Einträge für das Feld gibt.

GlobalValues.Type definiert den Typ der verfügbaren Auswertungsstatistiken. Die folgenden Auswertungsstatistikdaten sind verfügbar:

- MAX: Maximalwert des Felds.
- MEAN: Mittelwert des Felds.
- MIN: Minimalwert des Felds.
- STDDEV: Standardabweichung des Felds.
- SUM: Summe der Werte im Feld.

Das folgende Script z. B. greift auf den Mittelwert des Felds "income" zu, das von einem Globalwerteknoten berechnet wird:

```
import modeler.api

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

Arbeiten mit mehreren Streams: Standalone-Scripts

Zum Arbeiten mit mehreren Streams muss ein Standalone-Script verwendet werden. Das Standalone-Script kann in der Benutzerschnittstelle von IBM SPSS Modeler bearbeitet und ausgeführt oder als Befehlszeilenparameter im Stapelmodus übergeben werden.

Das folgende Standalone-Script öffnet zwei Streams. Einer dieser Streams erstellt ein Modell, der zweite plottet die Verteilung der vorhergesagten Werte.

```
# Change to the appropriate location for your system
demosDir = "C:/Programme/IBM/SPSS/Modeler/16/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()
```

Kapitel 5. Tipps zur Scripterstellung

In diesem Abschnitt erhalten Sie einen Überblick über Tipps und Verfahren für die Verwendung von Scripts, wie beispielsweise die Änderung der Streamausführung, die Verwendung von verschlüsselten Kennwörtern in Scripts und den Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository.

Ändern der Streamausführung

Wenn ein Stream ausgeführt wird, werden die Terminal-Knoten in einer für die Standardsituation optimierten Reihenfolge ausgeführt. In bestimmten Fällen kann eine andere Ausführungsreihenfolge wünschenswert sein. Um die Ausführungsreihenfolge eines Streams zu ändern, führen Sie im Dialogfeld "Streameigenschaften" auf der Registerkarte "Ausführung" folgende Schritte aus:

1. Starten Sie mit einem leeren Script.
2. Klicken Sie in der Symbolleiste auf die Schaltfläche **Standardscript anhängen**, um ein Standard-Stream-Script hinzuzufügen.
3. Bringen Sie die im Standard-Stream-Script enthaltenen Anweisungen in die für die Ausführung gewünschte Reihenfolge.

Verwendung von Schleifen bei Knoten

Sie können eine for-Schleife verwenden, um alle Knoten in einem Stream in einer Schleife zu durchlaufen. Die beiden folgenden Scriptbeispiele durchlaufen alle Knoten in einer Schleife und ändern dabei die Feldnamen in allen Filterknoten in Großbuchstaben.

Diese Scripts können in jedem Stream verwendet werden, der einen Filterknoten enthält, selbst wenn tatsächlich gar keine Felder gefiltert werden. Fügen Sie einfach einen Filterknoten hinzu, der alle Felder weigibt, um die Feldnamen durchgängig in Großbuchstaben zu ändern.

```
# 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)
```

Das Script durchläuft alle Knoten im aktuellen Stream und prüft jeweils, ob es sich bei den einzelnen Knoten um einen Filter handelt. Wenn ja, durchläuft das Script die einzelnen Felder im Knoten und verwendet die Funktion `field.upper()` oder `field.getColumnName().upper()`, um den Namen in Großbuchstaben zu ändern.

Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository

Wenn Sie IBM SPSS Collaboration and Deployment Services Repository lizenziert haben, können Sie mithilfe von Scriptbefehlen Objekte im Repository speichern, abrufen, sperren und entsperren. Mit dem Repository können Sie die Lebensdauer von Data-Mining-Modellen und verwandten Vorhersageobjekten im Zusammenhang mit Unternehmensanwendungen, Tools und Lösungen verwalten.

Verbinden mit IBM SPSS Collaboration and Deployment Services Repository

Um auf das Repository zugreifen zu können, müssen Sie zunächst eine gültige Verbindung einrichten, entweder über das Menü "Extras" der IBM SPSS Modeler-Benutzerschnittstelle oder über die Befehlszeile. (Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung“ auf Seite 69.)

Speichern und Abrufen von Objekten

Innerhalb eines Scripts können Sie mit den Befehlen `retrieve` und `store` auf verschiedene Objekte zugreifen, beispielsweise auf Streams, Modelle, Ausgaben, Knoten und Projekte. Die Syntax lautet wie folgt:

```
store object as REPOSITORY-PFAD {label BESCHRIFTUNG}
store object as URI [#1.label]
retrieve object REPOSITORY-PFAD {label BESCHRIFTUNG | version VERSION}
retrieve object URI [(#m.marker | #1.label)]
```

REPOSITORY-PFAD gibt die Position des Objekts im Repository an. Der Pfad muss in Anführungszeichen eingeschlossen sein und es müssen normale Schrägstriche als Trennzeichen verwendet werden. Die Groß- und Kleinschreibung wird nicht berücksichtigt.

```
store stream as "/folder_1/folder_2/mystream.str"
store model Drug as "/myfolder/drugmodel"
store model Drug as "/myfolder/drugmodel.gm" label "final"
store node DRUG1n as "/samples/drug1ntypenode"
store project as "/CRISPDM/DrugExample.cpj"
store output "Data Audit von [6 Felder]" as "/eigener Ordner/Mein Audit"
```

Optional können Erweiterungen wie `.str` oder `.gm` in den Objektnamen aufgenommen werden. Dies ist jedoch nicht erforderlich, solange der Name konsistent ist. Wenn beispielsweise ein Modell ohne Erweiterung gespeichert wird, muss es mit demselben Namen wieder abgerufen werden:

```
store model "/myfolder/drugmodel"
retrieve model "/myfolder/drugmodel"
```

Im Unterschied zu:

```
store model "/myfolder/drugmodel.gm"
retrieve model "/myfolder/drugmodel.gm" version "0:2005-10-12 14:15:41.281"
```

Beachten Sie: Beim Abrufen von Objekten wird immer die aktuellste Version des Objekts ausgegeben, es sei denn, Sie geben eine Version oder eine Beschriftung an. Beim Abrufen eines Knotenobjekts wird der Knoten automatisch in den aktuellen Stream eingefügt. Beim Abrufen eines Streamobjekts müssen Sie ein Standalone-Script verwenden. Streamobjekte können nicht aus Stream-Scripts abgerufen werden.

Sperren und Entsperrern von Objekten

Mit einem Script können Sie ein Objekt sperren, um zu verhindern, dass andere Benutzer seine bestehenden Versionen aktualisieren oder neue Versionen erstellen. Außerdem können Sie ein Objekt entsperren, das Sie gesperrt haben.

Die Syntax zum Sperren und Entsperren eines Objekts:

```
lock REPOSITORY-PFAD  
lock URI
```

```
unlock REPOSITORY-PFAD  
unlock URI
```

Wie beim Speichern und Abrufen von Objekten gibt REPOSITORY-PFAD die Position des Objekts im Repository an. Der Pfad muss in Anführungszeichen eingeschlossen sein und es müssen normale Schrägstriche als Trennzeichen verwendet werden. Die Groß- und Kleinschreibung wird nicht berücksichtigt.

```
lock "/myfolder/Stream1.str"
```

```
unlock "/myfolder/Stream1.str"
```

Alternativ können Sie statt eines Repository-Pfads einen URI (Uniform Resource Identifier) verwenden, um die Position des Objekts anzugeben. Der URI muss das Präfix `spsscr:` enthalten und muss vollständig in Anführungszeichen eingeschlossen sein. Nur normale Schrägstriche sind als Pfadtrennzeichen zulässig und Leerzeichen müssen codiert werden. Statt eines Leerzeichens muss im Pfad also `%20` verwendet werden. Die Groß- und Kleinschreibung wird beim URI nicht berücksichtigt. Beispiele:

```
lock "spsscr:///myfolder/Stream1.str"
```

```
unlock "spsscr:///myfolder/Stream1.str"
```

Beachten Sie, dass das Sperren von Objekten für alle Versionen eines Objekts gilt - Sie können keine einzelnen Versionen sperren oder entsperren.

Erstellen eines verschlüsselten Kennworts

In bestimmten Fällen müssen Sie möglicherweise ein Kennwort in ein Script aufnehmen, beispielsweise um auf eine kennwortgeschützte Datenquelle zuzugreifen. Verschlüsselte Kennwörter können in folgenden Elementen verwendet werden:

- Knoteneigenschaften für Datenbankquellenknoten und Ausgabeknoten
- Befehlszeilenargumente für die Anmeldung beim Server
- Die Datenbankverbindungseigenschaften, die in einer *.par*-Datei (die über die Registerkarte "Veröffentlichen" eines Exportknotens generierte Parameterdatei) gespeichert sind.

Über die Benutzerschnittstelle steht ein Tool zur Verfügung, mit dem Sie verschlüsselte Kennwörter auf der Grundlage des Blowfish-Algorithmus erstellen können. (Weitere Informationen finden Sie unter <http://www.schneier.com/blowfish.html>.) Nach der Verschlüsselung können Sie das Kennwort in Scriptdateien und Befehlszeilenargumente kopieren und dort speichern. Die Knoteneigenschaft `epassword`, die für `database`node und `databaseexport`node verwendet wird, speichert das verschlüsselte Kennwort.

1. Um ein verschlüsseltes Kennwort zu erstellen, wählen Sie im Menü "Extras" folgende Befehlsfolge aus:
Kennwort verschlüsseln...
2. Geben Sie ein Kennwort im Textfeld "Kennwort" ein.
3. Klicken Sie auf **Verschlüsseln**, um eine Zufallsverschlüsselung des Kennworts zu generieren.
4. Klicken Sie auf die Schaltfläche "Kopieren", um das verschlüsselte Kennwort in die Zwischenablage zu kopieren.
5. Fügen Sie das Kennwort in das gewünschte Script bzw. den gewünschten Parameter ein.

Scriptprüfung

Die Syntax aller Scripttypen können Sie sehr schnell prüfen, indem Sie in der Symbolleiste des Dialogfelds "Standalone-Script" auf die rote Prüfschaltfläche klicken.



Abbildung 6. Symbolleistenschaltflächen für Stream-Scripts

Die Scriptprüfung informiert Sie über alle in Ihrem Code enthaltenen Fehler und macht Verbesserungsvorschläge. Um die den Fehler enthaltende Zeile anzuzeigen, klicken Sie auf das in der unteren Hälfte des Dialogfelds angezeigte Feedback. Der Fehler wird dann rot hervorgehoben.

Scripts in der Befehlszeile

Mit Scripts können Sie Vorgänge ausführen, die normalerweise über die Benutzerschnittstelle durchgeführt werden. Geben Sie in der Befehlszeile beim Start von IBM SPSS Modeler einfach einen Standalone-Stream an und führen Sie ihn aus. Beispiel:

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

Das Flag `-script` lädt das angegebene Script, während das Flag `-execute` alle im Script enthaltenen Befehle ausführt.

Kompatibilität zu früheren Versionen

In früheren IBM SPSS Modeler-Versionen erstellte Scripts laufen in der aktuellen Version normalerweise unverändert. Allerdings können nun automatisch Modellnuggets in den Stream aufgenommen werden (das ist die Standardeinstellung) und ein vorhandenes Nugget dieses Typs im Stream ersetzen oder ergänzen. Ob dies tatsächlich geschieht, hängt von den Einstellungen der Optionen **Modell zu Stream hinzufügen** und **Bisheriges Modell ersetzen** ab (**Extras > Optionen > Benutzeroptionen > Benachrichtigungen**). Es kann beispielsweise erforderlich sein, ein Script aus einer früheren Version zu modifizieren, bei der die Nugget-Ersetzung durch Löschen des vorhandenen Nuggets und Einsetzen des neuen erfolgt.

In der aktuellen Version erstellte Scripts funktionieren eventuell nicht in früheren Versionen.

Wenn ein in einer älteren Version erstelltes Script einen Befehl verwendet, der mittlerweile ersetzt wurde (oder nicht mehr verwendet wird), wird die alte Form weiterhin unterstützt, es wird jedoch eine Warnnachricht angezeigt. Beispielsweise wurde das alte Schlüsselwort `generated` durch `model` und `clear generated` durch `clear generated palette` ersetzt. Scripts, die die alten Formen verwenden, werden weiterhin ausgeführt, es wird jedoch eine Warnnachricht angezeigt.

Zugriff auf Streamausführungsergebnisse

Viele IBM SPSS Modeler-Knoten erzeugen Ausgabeobjekte, wie z. B. Modelle, Diagramme und Tabellendaten. Viele dieser Ausgabedaten enthalten nützliche Werte, mit denen Scripts die nachfolgende Ausführung steuern können. Diese Werte werden in Containern mit Inhalt (einfach als "Container" bezeichnet) gruppiert, auf die über Tags oder IDs zugegriffen werden kann, die die einzelnen Container bezeichnen. Wie auf diese Werte zugegriffen wird, hängt vom Format oder "Inhaltsmodell" ab, das der jeweilige Container verwendet.

Viele Ausgaben von Vorhersagemodellen verwenden z. B. eine Variante von XML mit der Bezeichnung PMML zum Darstellen von Informationen zum Modell, wie z. B., welche Felder ein Entscheidungsbaum an jeder Aufteilung verwendet oder wie und mit welcher Stärke die Neuronen in einem neuronalen Netz verbunden sind. Modellausgaben, die PMML verwenden, stellen ein XML-Inhaltsmodell bereit, über das auf diese Informationen zugegriffen werden kann. Beispiel:

```

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]

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")

# 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")

```

IBM SPSS Modeler unterstützt die folgenden Inhaltsmodelle beim Scripting:

- Das **Tabelleninhaltsmodell** ermöglicht Zugriff auf die einfachen Tabellendaten, die als Zeilen und Spalten dargestellt werden
- Das **XML-Inhaltsmodell** ermöglicht Zugriff auf Inhalt im XML-Format
- Das **JSON-Inhaltsmodell** ermöglicht Zugriff auf Inhalt im JSON-Format
- Das **Inhaltsmodell für Spaltenstatistikdaten** ermöglicht Zugriff auf Auswertungsstatistikdaten zu einem bestimmten Feld
- Das **Inhaltsmodell für paarweise Spaltenstatistikdaten** ermöglicht Zugriff auf Auswertungsstatistikdaten zwischen zwei Feldern oder Werte zwischen zwei separaten Feldern

Tabelleninhaltsmodell

Das Tabelleninhaltsmodell stellt ein einfaches Modell für den Zugriff auf einfache Zeilen- und Spaltendaten bereit. Die Werte in einer bestimmten Spalte müssen alle denselben Speichertyp haben (z. B. Zeichenfolgen oder Ganzzahlen).

API

Tabelle 23. API

Ergebnis	Methode	Beschreibung
Ganzzahl	getRowCount()	Gibt die Anzahl der Zeilen in dieser Tabelle zurück.
Ganzzahl	getColumnCount()	Gibt die Anzahl der Spalten in dieser Tabelle zurück.
Zeichenfolge	getColumnName(Ganzzahl Spaltenindex)	Gibt den Namen der Spalte am angegebenen Spaltenindex zurück. Der Spaltenindex beginnt bei 0.
Speichertyp	getStorageType(Ganzzahl Spaltenindex)	Gibt den Speichertyp der Spalte am angegebenen Index zurück. Der Spaltenindex beginnt bei 0.
Objekt	getValueAt(Ganzzahl Zeilenindex, Ganzzahl Spaltenindex)	Gibt den Wert am angegebenen Zeilen- und Spaltenindex zurück. Der Zeilenindex und der Spaltenindex beginnen bei 0.
void	reset()	Führt eine Flushoperation für den internen Speicher aus, der diesem Inhaltsmodell zugeordnet ist.

Knoten und Ausgaben

In dieser Tabelle sind Knoten aufgelistet, die die Ausgaben erstellen, die diesen Typ von Inhaltsmodell enthalten.

Tabelle 24. Knoten und Ausgaben

Knotenname	Ausgabename	Container-ID
table	Tabelle	"table"

Beispielscript

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

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

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

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

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

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

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

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

Die Ausgabe auf der Registerkarte für die Scripting-Fehlerbehebung sieht ungefähr wie folgt aus:

```
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-Inhaltsmodell

Das XML-Inhaltsmodell bietet Zugriff auf XML-basierten Inhalt.

Das XML-Inhaltsmodell unterstützt den Zugriff auf Komponenten auf der Basis von XPath-Ausdrücken. XPath-Ausdrücke sind Zeichenfolgen, die definieren, welche Elemente oder Attribute vom Aufrufenden benötigt werden. Das XML-Inhaltsmodell blendet die Details der Erstellung verschiedener Objekte und der Kompilierung von Ausdrücken, die normalerweise für die XPath-Unterstützung benötigt werden, aus. Dies erleichtert das Aufrufen aus Python-Scriptumgebungen.

Das XML-Inhaltsmodell enthält eine Funktion, die das XML-Dokument als Zeichenfolge zurückgibt. Dadurch können Python-Scriptbenutzer ihre bevorzugte Python-Bibliothek zum Analysieren des XML-Codes verwenden.

API

Table 25. API

Ergebnis	Methode	Beschreibung
Zeichenfolge	<code>getXMLAsString()</code>	Gibt die XML-Daten als Zeichenfolge zurück.
Zahl	<code>getNumericValue(String xpath)</code>	Gibt das Ergebnis der Auswertung des Pfads mit dem numerischen Datentyp zurück (z. B. Zählen der Anzahl Elemente, die mit dem Pfadausdruck übereinstimmen).
boolesch	<code>getBooleanValue(String xpath)</code>	Gibt das boolesche Ergebnis der Auswertung des angegebenen Pfadausdrucks zurück.
Zeichenfolge	<code>getStringValue(String xpath, String attribute)</code>	Gibt entweder den Attributwert oder den XML-Knotenwert zurück, der mit dem angegebenen Pfad übereinstimmt.
Liste von Zeichenfolgen	<code>getStringValues(String xpath, String attribute)</code>	Gibt eine Liste aller Attributwerte oder XML-Knotenwerte zurück, die mit dem angegebenen Pfad übereinstimmen.
Liste von Zeichenfolgen	<code>getValuesList(String xpath, <List of strings> attributes, boolean includeValue)</code>	Gibt eine Liste aller Attributwerte, die mit dem angegebenen Pfad übereinstimmen, zusammen mit dem XML-Knotenwert (falls erforderlich) zurück.
Hashtabelle (Schlüssel:Zeichenfolge, Wert:Liste von Zeichenfolgen)	<code>getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)</code>	Gibt eine Hashtabelle zurück, die das Schlüsselattribut oder den XML-Knotenwert als Schlüssel und die Liste angegebener Attributwerte als Tabellenwerte verwendet.
boolesch	<code>isNamespaceAware()</code>	Gibt zurück, ob die XML-Parser Namespaces erkennen sollen. Die Standardeinstellung ist <code>False</code> .
void	<code>setNamespaceAware(boolean value)</code>	Legt fest, ob die XML-Parser Namespaces erkennen sollen. Hierdurch wird auch <code>reset()</code> aufgerufen, um sicherzustellen, dass Änderungen von nachfolgenden Aufrufen berücksichtigt werden.

Tabelle 25. API (Forts.)

Ergebnis	Methode	Beschreibung
void	reset()	Führt eine Flushoperation für den internen Speicher aus, der diesem Inhaltsmodell zugeordnet ist (beispielsweise im Cache gespeichertes DOM-Objekt).

Knoten und Ausgaben

In dieser Tabelle sind Knoten aufgelistet, die die Ausgaben erstellen, die diesen Typ von Inhaltsmodell enthalten.

Tabelle 26. Knoten und Ausgaben

Knotenname	Ausgabename	Container-ID
Die meisten Modellerstellungsprogramme	Die meisten generierten Modelle	"PMML"
"autodataprep"	entfällt	"PMML"

Beispielscript

Der Python-Scriptcode für den Zugriff auf den Inhalt könnte wie folgt aussehen:

```
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-Inhaltsmodell

Das JSON-Inhaltsmodell bietet Unterstützung für Inhalt im JSON-Format. Es stellt eine Basis-API zur Verfügung, die Aufrufenden die Extraktion von Werten ermöglicht. Dabei wird angenommen, dass sie wissen, auf welche Werte zugegriffen werden soll.

API

Tabelle 27. API

Ergebnis	Methode	Beschreibung
Zeichenfolge	getJSONAsString()	Gibt den JSON-Inhalt als Zeichenfolge zurück.
Objekt	getObjectAt(<List of objects> path, JSONArtifact artifact) löst eine Ausnahmebedingung aus	Gibt das Objekt im angegebenen Pfad zurück. Das angegebene Stammartefakt kann null sein. In diesem Fall wird der Stamm des Inhalts verwendet. Der zurückgegebene Wert kann eine Literalzeichenfolge, eine Ganzzahl, eine reelle Zahl oder ein boolescher Wert oder ein JSON-Artefakt (ein JSON-Objekt oder ein JSON-Array) sein.

Tabelle 27. API (Forts.)

Ergebnis	Methode	Beschreibung
Hashtabelle (Schlüssel:Objekt, Wert:Objekt)	<code>getChildValuesAt(<List of objects> path, JSONArtifact artifact)</code> löst eine Ausnahmebedingung aus	Gibt die untergeordneten Werte des angegebenen Pfads zurück, wenn der Pfad zu einem JSON-Objekt führt, oder andernfalls null. Die Schlüssel in der Tabelle sind Zeichenfolgen, während der zugeordnete Wert eine Literalzeichenfolge, eine Ganzzahl, eine reelle Zahl oder ein boolescher Wert oder ein JSON-Artefakt (ein JSON-Objekt oder ein JSON-Array) sein kann.
Liste von Objekten	<code>getChildrenAt(<List of objects> path, JSONArtifact artifact)</code> löst eine Ausnahmebedingung aus	Gibt die Liste von Objekten im angegebenen Pfad zurück, wenn der Pfad zu einem JSON-Array führt, oder andernfalls null. Die zurückgegebenen Werte können eine Literalzeichenfolge, eine Ganzzahl, eine reelle Zahl oder ein boolescher Wert oder ein JSON-Artefakt (ein JSON-Objekt oder ein JSON-Array) sein.
void	<code>reset()</code>	Führt eine Flushoperation für den internen Speicher aus, der diesem Inhaltsmodell zugeordnet ist (beispielsweise im Cache gespeichertes DOM-Objekt).

Beispielscript

Wenn ein Ausgabeerstellungsprogrammknoden vorhanden ist, der Ausgabe auf der Basis des JSON-Formats erstellt, könnte der folgende Code zum Zugriff auf Informationen zu einer Gruppe von Büchern verwendet werden:

```

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

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

# Alternatively, get the book object and use it as the root
# for subsequent entries
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Get all child values for aspecific book
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Get the third book entry. Assumes the top-level "books" value
# contains a JSON array which can be indexed
bookInfo = cm.getObjectAt(["books", 2], None)

# Get a list of all child entries
allBooks = cm.getChildrenAt(["books"], None)

```

Inhaltsmodell für Spaltenstatistikdaten und Inhaltsmodell für paarweise Statistikdaten

Das Inhaltsmodell für Spaltenstatistikdaten ermöglicht Zugriff auf Statistikdaten, die für jedes Feld berechnet werden können (univariate Statistik). Das Inhaltsmodell für paarweise Statistikdaten ermöglicht Zugriff auf Statistikdaten, die zwischen Paaren von Feldern oder Werten in einem Feld berechnet werden können.

Folgende Statistikdatenmaße sind möglich:

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

Einige Werte sind nur für Einzelspaltenstatistikdaten geeignet, andere nur für paarweise Statistikdaten.

Knoten, durch die sie erzeugt werden:

- Der **Statistiknoten** erzeugt Spaltenstatistikdaten und kann paarweise Statistikdaten erzeugen, wenn Korrelationsfelder angegeben werden.
- Der **Data Audit-Knoten** erzeugt Spaltenstatistikdaten und kann paarweise Statistikdaten erzeugen, wenn ein Überlagerungsfeld angegeben wird.
- Der **Mittelwertknoten** erzeugt paarweise Statistikdaten, wenn Feldpaare verglichen werden oder wenn die Werte eines Felds mit anderen Feldzusammenfassungen verglichen werden.

Welche Inhaltsmodelle und Statistikdaten verfügbar sind, hängt von den Fähigkeiten des jeweiligen Knotens und den Einstellungen im Knoten ab.

ColumnStatsContentModel-API

Tabelle 28. ColumnStatsContentModel-API.

Ergebnis	Methode	Beschreibung
List<StatisticType>	getAvailableStatistics()	Gibt die verfügbaren Statistikdaten in diesem Modell zurück. Nicht alle Felder enthalten notwendigerweise Werte für alle Statistikdaten.
List<String>	getAvailableColumns()	Gibt die Namen der Spalten zurück, für die Statistikdaten berechnet wurden.
Zahl	getStatistic(String column, StatisticType statistic)	Gibt die statistischen Werte zurück, die der Spalte zugeordnet sind.
void	reset()	Führt eine Flushoperation für den internen Speicher aus, der diesem Inhaltsmodell zugeordnet ist.

PairwiseStatsContentModel-API

Tabelle 29. PairwiseStatsContentModel-API.

Ergebnis	Methode	Beschreibung
List<StatisticType>	getAvailableStatistics()	Gibt die verfügbaren Statistikdaten in diesem Modell zurück. Nicht alle Felder enthalten notwendigerweise Werte für alle Statistikdaten.
List<String>	getAvailablePrimaryColumns()	Gibt die Namen der Primärspalten zurück, für die Statistikdaten berechnet wurden.
List<Object>	getAvailablePrimaryValues()	Gibt die Werte der Primärspalte zurück, für die Statistikdaten berechnet wurden.
List<String>	getAvailableSecondaryColumns()	Gibt die Namen der Sekundärspalten zurück, für die Statistikdaten berechnet wurden.
Zahl	getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)	Gibt die statistischen Werte zurück, die den Spalten zugeordnet sind.
Zahl	getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)	Gibt die statistischen Werte zurück, die dem Primärspaltenwert und der Sekundärspalte zugeordnet sind.
void	reset()	Führt eine Flushoperation für den internen Speicher aus, der diesem Inhaltsmodell zugeordnet ist.

Knoten und Ausgaben

In dieser Tabelle sind Knoten aufgelistet, die die Ausgaben erstellen, die diesen Typ von Inhaltsmodell enthalten.

Tabelle 30. Knoten und Ausgaben.

Knotenname	Ausgabename	Container-ID	Hinweise
"means" (Mittelwertknoten)	"means"	"columnStatistics"	
"means" (Mittelwertknoten)	"means"	"pairwiseStatistics"	
"dataaudit" (Data Audit-Knoten)	"means"	"columnStatistics"	
"statistics" (Statistiknoten)	"statistics"	"columnStatistics"	Wird nur generiert, wenn bestimmte Felder untersucht werden.
"statistics" (Statistiknoten)	"statistics"	"pairwiseStatistics"	Wird nur generiert, wenn Felder korreliert werden.

Beispielscript

```

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

```

Kapitel 6. Befehlszeilenargumente

Aufrufen der Software

Sie können die Befehlszeile Ihres Betriebssystems wie folgt verwenden, um IBM SPSS Modeler zu starten:

1. Öffnen Sie auf einem Computer, auf dem IBM SPSS Modeler installiert ist, ein DOS- oder Befehlszeilenfenster.
2. Um die IBM SPSS Modeler-Schnittstelle im interaktiven Modus zu starten, geben Sie den Befehl `modelerclient` gefolgt von den erforderlichen Argumenten ein. Beispiel:

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

Mithilfe der verfügbaren Argumente (Flags) können Sie eine Verbindung zu einem Server herstellen, Streams laden, Scripts ausführen oder je nach Bedarf weitere Parameter angeben.

Verwenden von Befehlszeilenargumenten

Sie können Befehlszeilenargumente (auch als *Flags* bezeichnet) an den ursprünglichen Befehl `modelerclient` anhängen, um die Vorgehensweise beim Aufrufen von IBM SPSS Modeler zu ändern.

Es sind mehrere Typen von Befehlszeilenargumenten verfügbar, die später in diesem Abschnitt beschrieben werden.

Tabelle 31. Typen von Befehlszeilenargumenten.

Argumenttyp	Wo beschrieben
Systemargumente	Weitere Informationen finden Sie im Thema „Systemargumente“ auf Seite 66.
Parameterargumente	Weitere Informationen finden Sie im Thema „Parameterargumente“ auf Seite 67.
Argumente zum Herstellen einer Serververbindung	Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer Serververbindung“ auf Seite 68.
Argumente zum Herstellen einer Verbindung mit IBM SPSS Collaboration and Deployment Services Repository	Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung“ auf Seite 69.
Argumente zum Herstellen einer Verbindung mit IBM SPSS Analytic Server	Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Analytic Server-Verbindung“ auf Seite 70.

Beispielsweise können Sie mit den Flags `-server`, `-stream` und `-execute` wie folgt eine Verbindung zu einem Server herstellen und dann einen Stream laden und ausführen:

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

Beachten Sie: Bei der Ausführung unter einer lokalen Clientinstallation sind die Argumente für die Serververbindung nicht erforderlich.

Parameterwerte, die Leerzeichen enthalten können, können in doppelte Anführungszeichen eingeschlossen werden, z. B.:

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

Sie können auch IBM SPSS Modeler-Statusmodi und -Scripts auf diese Weise ausführen, nämlich mit den Flags `-state` bzw. `-script`.

Anmerkung: Wenn Sie einen strukturierten Parameter in einem Befehl verwenden, müssen Sie vor Anführungszeichen einen umgekehrten Schrägstrich (`\`) angeben. Dadurch wird verhindert, dass die Anführungszeichen während der Interpretation der Zeichenfolge entfernt werden.

Fehlersuche bei Befehlszeilenargumenten

Um die Fehlersuche in einer Befehlszeile durchzuführen, starten Sie IBM SPSS Modeler mithilfe des Befehls `modelerclient` mit den gewünschten Argumenten. Dadurch können Sie prüfen, ob die Befehle erwartungsgemäß ausgeführt werden. Außerdem können Sie die Werte jedes Parameters bestätigen, der von der Befehlszeile in das Dialogfeld "Sitzungsparameter" (Menü "Extras", "Sitzungsparameter festlegen") übergeben wird.

Systemargumente

In der nachstehenden Tabelle werden die Systemargumente beschrieben, die für das Aufrufen der Benutzerschnittstelle über die Befehlszeile zur Verfügung stehen.

Tabelle 32. Systemargumente

Argument	Verhalten/Beschreibung
@ <Befehlsdatei>	Das Symbol @, gefolgt von einem Dateinamen, bezeichnet eine Liste von Befehlen. Wenn der Befehl <code>modelerclient</code> auf ein Argument mit dem Symbol @ trifft, werden die Befehle in dieser Datei so abgearbeitet, als hätten Sie diese Befehle direkt in der Befehlszeile eingegeben. Weitere Informationen finden Sie im Thema „Kombinieren mehrerer Argumente“ auf Seite 70.
-directory <Verzeichnis>	Bestimmt das Standardarbeitsverzeichnis. Im lokalen Modus wird dieses Verzeichnis sowohl für Daten als auch für die Ausgabe herangezogen. Beispiel: <code>-directory c:/</code> oder <code>-directory c:\</code>
-server_directory <Verzeichnis>	Bestimmt das Serverstandardverzeichnis für Daten. Das Arbeitsverzeichnis, das mithilfe des Flag <code>-directory</code> angegeben wird, wird für die Ausgabe genutzt.
-execute	Nach dem Starten: Alle Streams, Statusangaben oder Scripts ausführen, die beim Starten geladen waren. Wird ein Script zusätzlich zu einem Stream oder einem Status geladen, wird nur das Script ausgeführt.
-stream <Stream>	Beim Starten: Angegebenen Stream laden. Sie können mehrere Streams angeben; der zuletzt genannte Stream wird dabei als aktueller Stream festgelegt.
-script <Script>	Beim Starten: Angegebenes Standalone-Script laden. Sie können dieses Script zusätzlich zu einem Stream oder einem Status angeben (siehe unten); beim Starten kann jedoch nur ein einziges Script geladen werden.
-model <Modell>	Beim Starten: Angegebenes generiertes Modell (Datei im Format <code>.gm</code>) laden.
-state <Status>	Beim Starten: Angegebenen gespeicherten Status laden.
-project <Projekt>	Angegebenes Projekt laden. Beim Starten kann nur ein einziges Projekt geladen werden.
-output <Ausgabe>	Beim Starten: Gespeichertes Ausgabeobjekt (Formatdatei <code>.cou</code>) laden.
-help	Liste der Befehlszeilenargumente abrufen. Wenn diese Option angegeben ist, werden alle anderen Argumente ignoriert und der Hilfebildschirm wird geöffnet.
-P <Name>=<Wert>	Bestimmt einen Startparameter. Kann auch zum Festlegen von Knoteneigenschaften (Slotparameter) herangezogen werden.

Hinweis: Standardverzeichnisse können auch über die Benutzerschnittstelle festgelegt werden. Wählen Sie hierzu im Menü "Datei" die Option **Arbeitsverzeichnis festlegen** bzw. **Serververzeichnis festlegen** aus.

Laden mehrerer Dateien

Über die Befehlszeile können Sie beim Start mehrere Streams, Status und Ausgaben laden, indem Sie für jedes geladene Objekt das relevante Argument wiederholen. Sollen beispielsweise zwei Streams mit den Bezeichnungen *report.str* und *train.str* geladen werden, geben Sie den folgenden Befehl ein:

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

Laden von Objekten aus dem IBM SPSS Collaboration and Deployment Services Repository

Da Sie bestimmte Objekte aus einer Datei oder aus dem IBM SPSS Collaboration and Deployment Services Repository (sofern lizenziert) laden können, gibt das Dateinamenspräfix `spsscr:` und optional `file:` (für Objekte auf Datenträgern) IBM SPSS Modeler an, wo nach dem Objekt gesucht werden soll. Das Präfix funktioniert mit folgenden Flags:

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

Das Präfix wird zur Erstellung eines URI verwendet, der die Position des Objekts angibt. Beispiel: `-stream "spsscr:///folder_1/scoring_stream.str"`. Bei Verwendung des Präfix `spsscr:` ist es erforderlich, dass in demselben Befehl eine gültige Verbindung zu IBM SPSS Collaboration and Deployment Services Repository angegeben wurde. Der vollständige Befehl sieht also etwa wie folgt aus:

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

Beachten Sie: In der Befehlszeile *müssen* Sie einen URI verwenden. Das einfachere `REPOSITORY_PATH` wird nicht unterstützt. (Es funktioniert nur innerhalb von Scripts.) Weitere Details zu URIs für Objekte im IBM SPSS Collaboration and Deployment Services Repository finden Sie im Thema „Zugriff auf Objekte in IBM SPSS Collaboration and Deployment Services Repository“ auf Seite 54.

Parameterargumente

Bei der Ausführung von IBM SPSS Modeler über die Befehlszeile können Parameter als Flags herangezogen werden. In Befehlszeilenargumenten wird das Flag `-P` verwendet, um einen Parameter im Format `-P <Name>=<Wert>` zu kennzeichnen.

Die folgenden Parameter stehen zur Auswahl:

- **Einfache Parameter** (oder Parameter, die direkt in CLEM-Ausdrücken verwendet werden).
- **Slotparameter** (auch als **Knoteneigenschaften** bezeichnet). Mit diesen Parametern werden die Einstellungen für die Knoten im Stream bearbeitet. Weitere Informationen finden Sie im Thema „Überblick über Knoteneigenschaften“ auf Seite 73.
- **Befehlszeilenparameter** dienen zum Ändern der Vorgehensweise beim Aufrufen von IBM SPSS Modeler.

Geben Sie beispielsweise die Benutzernamen und Kennwörter für Datenquellen in Form von Befehlszeilen-Flags an:

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

Das Format stimmt mit dem Parameter `datasource` der Knoteneigenschaft `databasenode` überein. Weitere Informationen finden Sie in „Eigenschaften von "databasenode"“ auf Seite 85.

Anmerkung: Wenn der Knoten benannt ist, müssen Sie seinen Namen in Anführungszeichen einschließen und den Anführungszeichen einen umgekehrten Schrägstrich (\) als Escapezeichen voranstellen. Wenn z. B. der Datenquellenknoten im vorherigen Beispiel den Namen *Quelle_ABC* hat, würde der Eintrag wie folgt lauten:

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

Ein umgekehrter Schrägstrich ist auch vor den Anführungszeichen erforderlich, die einen strukturierten Parameter angeben, wie im folgenden TM1-Datenquellenbeispiel gezeigt wird:

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

Argumente zum Herstellen einer Serververbindung

Das Flag `-server` besagt, dass IBM SPSS Modeler eine Verbindung zu einem öffentlichen Server aufbauen soll. Mit den Flags `-hostname`, `-use_ssl`, `-port`, `-username`, `-password` und `-domain` legen Sie fest, auf welche Weise IBM SPSS Modeler diese Verbindung zum öffentlichen Server herstellen soll. Wenn kein Argument vom Typ `-server` angegeben wurde, wird der Standardserver bzw. der lokale Server verwendet.

Beispiele

So stellen Sie eine Verbindung mit einem öffentlichen Server her:

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

So stellen Sie eine Verbindung mit einem Server-Cluster her:

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

Beachten Sie, dass zum Herstellen einer Verbindung mit einem Server-Cluster der Coordinator of Processes über IBM SPSS Collaboration and Deployment Services erforderlich ist. Das Argument `-cluster` muss also in Verbindung mit den Optionen für eine Repository-Verbindung (`spsscr_*`) verwendet werden. Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung“ auf Seite 69.

Tabelle 33. Argumente zum Herstellen einer Serververbindung.

Argument	Verhalten/Beschreibung
<code>-server</code>	Startet IBM SPSS Modeler im Servermodus. Hierzu wird eine Verbindung zu einem öffentlichen Server mit den Flags <code>-hostname</code> , <code>-port</code> , <code>-username</code> , <code>-password</code> und <code>-domain</code> hergestellt.
<code>-hostname <Name></code>	Hostname des Server-Computers. Nur im Servermodus verfügbar.
<code>-use_ssl</code>	Gibt an, dass die Verbindung SSL (Secure Socket Layer) verwenden sollte. Dieses Flag ist optional, bei der Standardeinstellung wird SSL <i>nicht</i> verwendet.
<code>-port <Nummer></code>	Portnummer des angegebenen Servers. Nur im Servermodus verfügbar.
<code>-cluster <Name></code>	Gibt eine Verbindung zu einem Server-Cluster und nicht zu einem benannten Server an; dieses Argument ist eine Alternative zu den Argumenten <code>hostname</code> , <code>port</code> und <code>use_ssl</code> . Bei dem Namen handelt es sich um den Clusternamen oder um einen eindeutigen URI, der den Cluster im IBM SPSS Collaboration and Deployment Services Repository identifiziert. Der Server-Cluster wird von Coordinator of Processes über IBM SPSS Collaboration and Deployment Services verwaltet. Weitere Informationen finden Sie im Thema „Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung“ auf Seite 69.

Tabelle 33. Argumente zum Herstellen einer Serververbindung (Forts.).

Argument	Verhalten/Beschreibung
-username <Name>	Benutzername, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar.
-password <Kennwort>	Kennwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. <i>Hinweis:</i> Falls das Argument -password nicht verwendet wird, werden Sie aufgefordert, ein Kennwort einzugeben.
-epassword <codierte Kennwortzeichenfolge>	Codiertes Kennwort, mit dem die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar. <i>Hinweis:</i> Ein codiertes Kennwort kann in IBM SPSS Modeler mit den Befehlen im Menü "Extras" erzeugt werden.
-domain <Name>	Domäne, mit der die Anmeldung beim Server erfolgt. Nur im Servermodus verfügbar.
-P <Name>=<Wert>	Bestimmt einen Startparameter. Kann auch zum Festlegen von Knoteneigenschaften (Slotparameter) herangezogen werden.

Argumente zum Herstellen einer IBM SPSS Collaboration and Deployment Services Repository-Verbindung

Wenn Sie Objekte aus IBM SPSS Collaboration and Deployment Services mithilfe der Befehlszeile speichern oder abrufen möchten, müssen Sie eine gültige Verbindung zum IBM SPSS Collaboration and Deployment Services Repository angeben. Beispiel:

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

In der folgenden Tabelle sind die Argumente aufgeführt, die zum Einrichten der Verbindung verwendet werden können.

Tabelle 34. Argumente zum Herstellen einer Verbindung mit IBM SPSS Collaboration and Deployment Services Repository

Argument	Verhalten/Beschreibung
-spsscr_hostname <Hostname oder IP-Adresse>	Der Hostname bzw. die IP-Adresse des Servers, auf dem IBM SPSS Collaboration and Deployment Services Repository installiert ist.
-spsscr_port <Nummer>	Die Nummer des Ports, an dem IBM SPSS Collaboration and Deployment Services Repository Verbindungen akzeptiert (üblicherweise standardmäßig 8080).
-spsscr_use_ssl	Gibt an, dass die Verbindung SSL (Secure Socket Layer) verwenden sollte. Dieses Flag ist optional, bei der Standardeinstellung wird SSL <i>nicht</i> verwendet.
-spsscr_username <Name>	Benutzername, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
-spsscr_password <Kennwort>	Kennwort, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
-spsscr_epassword <codiertes Kennwort>	Codiertes Kennwort, mit dem die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt.
-spsscr_domain <Name>	Domäne, mit der die Anmeldung beim IBM SPSS Collaboration and Deployment Services Repository erfolgt. Dieses Flag ist optional. Verwenden Sie es nur, wenn Sie sich nicht mithilfe von LDAP oder Active Directory anmelden.

Argumente zum Herstellen einer IBM SPSS Analytic Server-Verbindung

Wenn Sie Objekte aus IBM SPSS Analytic Server mithilfe der Befehlszeile speichern oder abrufen möchten, müssen Sie eine gültige Verbindung zu IBM SPSS Analytic Server angeben.

Anmerkung: Der Speicherort von Analytic Server wird von SPSS Modeler Server abgerufen und kann auf dem Client nicht geändert werden.

In der folgenden Tabelle sind die Argumente aufgeführt, die zum Einrichten der Verbindung verwendet werden können.

Tabelle 35. Argumente zum Herstellen einer Verbindung mit IBM SPSS Analytic Server

Argument	Verhalten/Beschreibung
-analytic_server_username	Benutzername, mit dem die Anmeldung bei IBM SPSS Analytic Server erfolgt.
-analytic_server_password	Kennwort, mit dem die Anmeldung bei IBM SPSS Analytic Server erfolgt.
-analytic_server_epassword	Codiertes Kennwort, mit dem die Anmeldung bei IBM SPSS Analytic Server erfolgt.
-analytic_server_credential	Berechtigungsnachweise, mit denen die Anmeldung bei IBM SPSS Analytic Server erfolgt.

Kombinieren mehrerer Argumente

Sie können mehrere Argumente in einer einzigen Befehlsdatei kombinieren, die mit dem Symbol @, gefolgt vom Dateinamen, beim Aufrufen angegeben wird. Auf diese Weise können Sie das Aufrufen über die Befehlszeile verkürzen und die im Betriebssystem geltenden Einschränkungen für die Befehlslänge umgehen. Beim nachstehenden Startbefehl werden beispielsweise die Argumente verwendet, die in der durch <Befehlsdateiname> referenzierten Datei angegeben sind.

```
modelerclient @<Befehlsdateiname>
```

Schließen Sie den Dateinamen und den Pfad in Anführungszeichen ein, falls Leerzeichen erforderlich sind, beispielsweise:

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

Die Befehlsdatei kann alle Argumente umfassen, die zuvor beim Starten einzeln angegeben wurden, und zwar mit jeweils einem Argument pro Zeile. Beispiel:

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

Beim Schreiben und Referenzieren von Befehlsdateien sind die folgenden Einschränkungen zu beachten:

- Geben Sie nur je einen Befehl pro Zeile ein.
- Betten Sie kein @Befehlsdatei-Argument in eine Befehlsdatei ein.

Kapitel 7. Eigenschaftsreferenz

Überblick über die Eigenschaften

Für Knoten, Streams, Superknoten und Projekte können Sie eine Reihe von verschiedenen Eigenschaften festlegen. Einige Eigenschaften wie Name, Anmerkung und QuickInfo gelten für alle Knoten, während andere sich nur auf bestimmte Knotentypen beziehen. Wieder andere Eigenschaften beziehen sich auf Streamoperationen auf hoher Ebene wie Zwischenspeichern oder das Verhalten von Superknoten. Der Zugriff auf Eigenschaften erfolgt über die Standardbenutzerschnittstelle (z. B. beim Öffnen eines Dialogfelds zum Bearbeiten von Optionen für einen Knoten). Eigenschaften können auf vielfältige Weise verwendet werden.

- Eigenschaften lassen sich mit Scripts ändern, wie in diesem Abschnitt beschrieben. Weitere Informationen finden Sie in „Syntax für Eigenschaften“.
- Knoteneigenschaften können in Superknotenparametern verwendet werden.
- Knoteneigenschaften können auch als Teil einer Befehlszeilenoption (mit dem Flag -P) beim Starten von IBM SPSS Modeler verwendet werden.

Im Zusammenhang mit Scripts in IBM SPSS Modeler werden Knoten- und Streameigenschaften häufig als **Slotparameter** bezeichnet. In diesem Handbuch werden sie als Knoten- oder Streameigenschaften beschrieben.

Weitere Informationen zur Scriptsprache finden Sie in Scriptsprache.

Syntax für Eigenschaften

Eigenschaften können mit der folgenden Syntax festgelegt werden.

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

oder:

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

Der Wert von Eigenschaften kann mit der folgenden Syntax abgerufen werden:

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

oder:

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

Dabei ist OBJECT ein Knoten oder eine Ausgabe, PROPERTY ist der Name der Knoteneigenschaft, auf die Ihr Ausdruck verweist, und KEY ist der Schlüsselwert für verschlüsselte Eigenschaften. Beispielsweise wird die folgende Syntax verwendet, um den Filterknoten zu suchen und dann den Standard so festzulegen, dass alle Felder eingeschlossen werden und das Feld Age in nachfolgenden Daten gefiltert wird:

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

Alle in IBM SPSS Modeler verwendeten Knoten können mit der Streamfunktion `findByType(TYPE, LABEL)` gesucht werden. Mindestens eine Instanz von TYPE oder LABEL muss angegeben werden.

Strukturierte Eigenschaften

Es gibt zwei Möglichkeiten, wie Scripts strukturierte Eigenschaften verwenden können, um eine größere Klarheit bei der Analyse zu erreichen:

- Den Namen der Eigenschaften für komplexe Knoten, wie Typ-, Filter- und Balancierungsknoten, strukturieren.
- Ein Format zum Festlegen mehrerer Eigenschaften gleichzeitig angeben.

Strukturieren komplexer Benutzerschnittstellen

Die Scripts für Knoten mit Tabellen und anderen komplexen Benutzerschnittstellen (z. B. Typ-, Filter- und Balancierungsknoten) müssen eine bestimmte Struktur besitzen, damit die Analyse ordnungsgemäß erfolgt. Für diese Eigenschaften ist ein komplexerer Name als für einen einzelnen IDs erforderlich. Dieser Name wird als Schlüssel bezeichnet. Innerhalb eines Filterknotens wird beispielsweise jedes verfügbare Feld (auf der vorausgehenden Seite) ein- oder ausgeschaltet. Um auf diese Information zurückzugreifen, speichert der Filterknoten ein Informationselement pro Feld (ob das Feld jeweils wahr oder falsch ist). Diese Eigenschaft kann den Wert True oder False besitzen (bzw. zugewiesen bekommen). Angenommen, ein Filterknoten namens mynode weist (auf der vorausgehenden Seite) ein Feld mit dem Namen Age auf. Um dieses auszuschalten, legen Sie für die Eigenschaft include mit dem Schlüssel Age wie folgt den Wert False fest:

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

Strukturieren zum Festlegen mehrerer Eigenschaften

Für mehrere Knoten können Sie mehr als einen Knoten oder eine Stromeigenschaft gleichzeitig zuweisen. Dies wird als **Multiset-Befehl** oder **Blockset** bezeichnet. Weitere Informationen finden Sie im Thema Befehl "set".

In manchen Fällen kann eine strukturierte Eigenschaft äußerst komplex sein. Ein Beispiel:

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

Ein weiterer Vorteil strukturierter Eigenschaften besteht darin, dass mehrere Eigenschaften an einem Knoten festgelegt werden können, bevor der Knoten stabil ist. Standardmäßig legt ein Multiset alle Eigenschaften in einem Block fest, bevor je nach individueller Eigenschafteneinstellung Vorgänge ausgeführt werden. Beispiel: Wenn die Feldeigenschaften beim Definieren eines Knotes "Datei (fest)" in zwei Schritten festgelegt werden, treten Fehler auf, da der Knoten erst konsistent ist, wenn beide Einstellungen gültig sind. Durch Definieren der Eigenschaften als Multiset wird dieses Problem umgangen, indem beide Eigenschaften festgelegt werden, bevor das Datenmodell aktualisiert wird.

Abkürzungen

Für die Syntax der Knoteneigenschaften werden Standardabkürzungen verwendet. Sich mit den Abkürzungen vertraut zu machen kann beim Erstellen von Scripts sehr hilfreich sein.

Tabelle 36. In der Syntax verwendete Standardabkürzungen

Abkürzung	Bedeutung
abs	Absoluter Wert
len	Länge
min	Minimum
max	Maximum
correl	Korrelation
covar	Kovarianz
num	Zahl oder numerisch
pct	Prozent oder Prozentsatz
transp	Transparenz
xval	Kreuzvalidierung

Tabelle 36. In der Syntax verwendete Standardabkürzungen (Forts.)

Abkürzung	Bedeutung
var	Varianz oder Variable (in Quellenknoten)

Beispiele für Knoten- und Streameigenschaften

Mit IBM SPSS Modeler können Knoten- und Streameigenschaften auf vielfältige Weise verwendet werden. Meistens werden sie in einem Script, entweder einem **Standalone-Script** zur Automatisierung mehrerer Streams oder Operationen oder einem **Stream-Script** zur Automatisierung von Prozessen innerhalb eines einzelnen Streams, verwendet. Superknotenparameter können ebenfalls innerhalb des Superknotens anhand der Knoteneigenschaften angegeben werden. Auf niedrigster Ebene können Eigenschaften auch als Befehlszeilenoption zum Starten von IBM SPSS Modeler verwendet werden. Mit dem Argument `-p` als Teil des Befehlszeilenaufrufs können Sie eine Streameigenschaft verwenden, um eine Einstellung im Stream zu ändern.

Tabelle 37. Beispiele für Knoten- und Streameigenschaften

Eigenschaft	Bedeutung
<code>s.max_size</code>	Bezieht sich auf die Eigenschaft <code>max_size</code> des Knotens <code>s</code> .
<code>s:samplenode.max_size</code>	Bezieht sich auf die Eigenschaft <code>max_size</code> des Knotens <code>s</code> , bei dem es sich um einen Stichprobenknoten handeln muss.
<code>:samplenode.max_size</code>	Bezieht sich auf die Eigenschaft <code>max_size</code> des Stichprobenknotens im aktuellen Stream (es darf nur ein Stichprobenknoten vorhanden sein).
<code>s:sample.max_size</code>	Bezieht sich auf die Eigenschaft <code>max_size</code> des Knotens <code>s</code> , bei dem es sich um einen Stichprobenknoten handeln muss.
<code>t.direction.Alter</code>	Bezieht sich auf die Rolle des Felds <code>Alter</code> im Typknoten <code>t</code> .
<code>:.max_size</code>	*** NICHT ZULÄSSIG *** Sie müssen entweder den Knotennamen oder den Knotentyp angeben.

Das Beispiel `s:sample.max_size` veranschaulicht, dass Knotentypen nicht vollständig ausgeschrieben werden müssen.

Im Beispiel `t.direction.Age` wird deutlich, dass einige Slotnamen selbst strukturiert werden können, und zwar wenn die Attribute eines Knotens komplexer sind als nur individuelle Slots mit individuellen Werten. Solche Slots werden als **strukturierte** oder **komplexe** Eigenschaften bezeichnet.

Überblick über Knoteneigenschaften

Jeder Knotentyp besitzt eine eigene Gruppe zulässiger Eigenschaften und jede Eigenschaft besitzt einen Typ. Dabei kann es sich um einen allgemeinen Typ einer Zahl, eines Flags, oder einer Zeichenfolgen handeln. In diesem Fall wird für die Einstellungen der Eigenschaft der richtige Typ erzwungen. Wenn sie nicht erzwungen werden können, wird ein Fehler ausgegeben. Alternativ kann die Eigenschaftsreferenz den Bereich zulässiger Werte, wie `Discard`, `PairAndDiscard` und `IncludeAsText`, angeben. In diesem Fall tritt bei Verwendung eines anderen Werts ein Fehler auf. Flag-Eigenschaften sollten anhand der Werte `true` und `false` gelesen bzw. festgelegt werden. (Abweichungen wie beispielsweise `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` oder `0` werden beim Festlegen der Werte ebenfalls erkannt, können jedoch beim Lesen der Eigenschaftswerte in einigen Fällen zu Fehlern führen. Alle anderen Werte werden als wahr betrachtet. Durch die durchgängige Verwendung von `true` und `false` können Verwechslungen vermieden werden.) Die Referenztabelle in diesem Handbuch weisen strukturierte Eigenschaften als solche in der Spalte *Eigenschaftsbeschreibung* aus und geben ihr Verwendungsformat an.

Allgemeine Knoteneigenschaften

Zahlreiche Eigenschaften beziehen sich in IBM SPSS Modeler auf alle Knoten (einschließlich Superknoten).

Table 38. Allgemeine Knoteneigenschaften.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
use_custom_name	Flag	
name	Zeichenfolge	Schreibgeschützte Eigenschaft zum Lesen des Namens (entweder automatisch oder benutzerdefiniert) für einen Knoten im Erstellungsbereich.
custom_name	Zeichenfolge	Legt einen benutzerdefinierten Namen für den Knoten fest.
tooltip	Zeichenfolge	
annotation	Zeichenfolge	
keywords	Zeichenfolge	Strukturierter Slot, der eine Liste der mit dem Objekt verknüpften Schlüsselwörter angibt (Beispiel: ["Keyword1" "Keyword2"]).
cache_enabled	Flag	
node_type	source_supernode process_supernode terminal_supernode alle Knotennamen, wie für Scripts angegeben	Schreibgeschützte Eigenschaft, die den Bezug zu einem Knoten nach Typ herstellt. Statt auf den Knoten nur mit dem Namen zu verweisen, wie real_income, können Sie beispielsweise auch den Typ, wie userinputnode oder filternode, angeben.

Superknotenspezifische Eigenschaften werden wie alle anderen Knoten separat erörtert. Weitere Informationen finden Sie in Kapitel 19, „Superknoteneigenschaften“, auf Seite 311.

Kapitel 8. Streameigenschaften

Verschiedene Streameigenschaften können durch Scripts gesteuert werden. Um Streameigenschaften zu referenzieren, müssen Sie die Ausführungsmethode für die Verwendung von Scripts festlegen:

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

Beispiel

Die Knoteneigenschaft dient zur Referenzierung der Knoten im aktuellen Stream. Das folgende Stream-Script ist ein Beispiel:

```
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)
```

Im oben genannten Beispiel wird anhand der Knoteneigenschaft eine Liste aller Knoten im Stream erstellt und diese Liste in die Streamanmerkungen geschrieben. Die erzeugte Anmerkung sieht wie folgt aus:

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"
```

In der folgenden Tabelle werden die Streameigenschaften beschrieben.

Tabelle 39. Streameigenschaften.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
execute_method	Normal Script	

Tabelle 39. Streameigenschaften (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	
date_baseline	Zahl	
date_2digit_baseline	Zahl	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	Flag	
import_datetime_as_string	Flag	
decimal_places	Zahl	
decimal_symbol	Default Period Comma	
angles_in_radians	Flag	
use_max_set_size	Flag	
max_set_size	Zahl	
ruleset_evaluation	Voting FirstHit	

Tabella 39. Streameigenschaften (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
refresh_source_nodes	Flag	Dient zur automatischen Aktualisierung von Quellenknoten nach Ausführung des Streams.
script	Zeichenfolge	
annotation	Zeichenfolge	
name	Zeichenfolge	Anmerkung: Diese Eigenschaft ist schreibgeschützt. Wenn Sie den Namen eines Streams ändern möchten, speichern Sie ihn unter einem anderen Namen.
parameters		Diese Eigenschaft dient zur Aktualisierung von Streamparametern innerhalb eines Standalone-Scripts.
nodes		Details finden Sie weiter unten.
encoding	SystemDefault "UTF-8"	
stream_rewriting	boolesch	
stream_rewriting_maximise_sql	boolesch	
stream_rewriting_optimise_clem_execution	boolesch	
stream_rewriting_optimise_syntax_execution	boolesch	
enable_parallelism	boolesch	
sql_generation	boolesch	
database_caching	boolesch	
sql_logging	boolesch	
sql_generation_logging	boolesch	
sql_log_native	boolesch	
sql_log_prettyprint	boolesch	
record_count_suppress_input	boolesch	
record_count_feedback_interval	Ganzzahl	
use_stream_auto_create_node_settings	boolesch	Bei "true" werden die für den Stream spezifischen Einstellungen verwendet. Andernfalls werden die Vorgaben verwendet.
create_model_applier_for_new_models	boolesch	Bei "true" wird ein neuer Modellanwender hinzugefügt, wenn ein Modellerstellungsprogramm ein neues Modell erstellt und es keine aktiven Update-Links hat. Anmerkung: Wenn Sie IBM SPSS Modeler Batch Version 15 verwenden, müssen Sie den Modellanwender explizit in Ihrem Script hinzufügen.
create_model_applier_update_links	createEnabled createDisabled doNotCreate	Definiert den Typ von Link, wenn ein Modellanwendungsknoten automatisch hinzugefügt wird.

Table 39. Stream properties (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
create_source_node_from_builders	<i>boolesch</i>	Bei "true" wird ein neuer Quellenknoten hinzugefügt, wenn ein Quellenerstellungsprogramm eine neue Quellenausgabe erstellt und sie keine aktiven Update-Links hat.
create_source_node_update_links	createEnabled createDisabled doNotCreate	Definiert den Typ von Link, wenn ein Quellenknoten automatisch hinzugefügt wird.
has_coordinate_system	<i>boolesch</i>	Bei "true" wird ein Koordinatensystem auf den gesamten Datenstrom angewendet.
coordinate_system	<i>Zeichenfolge</i>	Name des ausgewählten projizierten Koordinatensystems.

Kapitel 9. Eigenschaften von Quellenknoten

Allgemeine Eigenschaften von Quellenknoten

Eigenschaften, die alle Quellenknoten besitzen, sind unten aufgelistet. Die darauf folgenden Themen enthalten Informationen zu bestimmten Knoten.

Beispiel 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")
```

Beispiel 2

Dieses Script geht davon aus, dass die angegebene Datendatei ein Feld mit der Bezeichnung Region enthält, das eine mehrzeilige Zeichenfolge darstellt.

```
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")
```

Tabelle 40. Allgemeine Eigenschaften von Quellenknoten.

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
direction	Input Target Both None Partition Split Frequency RecordID	Verschlüsselte Eigenschaft für Feldrollen. Format: NODE.direction.FIELDNAME Anmerkung: Die Werte In und Out werden nicht mehr verwendet. In zukünftigen Versionen fällt möglicherweise die Unterstützung dafür weg.

Tabelle 40. Allgemeine Eigenschaften von Quellenknoten (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
type	Range Flag Set Typeless Discrete Ordered Set Default	Feldtyp. Wenn diese Eigenschaft auf <i>Default</i> (Standard) gesetzt wird, werden alle Eigenschafteneinstellungen vom Typ <i>values</i> gelöscht, und wenn <i>value_mode</i> den Wert <i>Default</i> (Angaben) besitzt, wird er auf <i>Read</i> (Lesen) zurückgesetzt. Wenn <i>value_mode</i> bereits auf <i>Pass</i> (Übergeben) oder <i>Read</i> (Lesen) gesetzt ist, hat das Einstellen von <i>type</i> keinerlei Auswirkung. Format: NODE.type.FIELDNAME
storage	Unknown String Integer Real Zeit Datum Timestamp	Schreibgeschützte verschlüsselte Eigenschaft für Feldspeichertyp. Format: NODE.storage.FIELDNAME
check	None Nullify Coerce Discard Warn Abort	Verschlüsselte Eigenschaft für das Überprüfen von Feldtyp und Bereich. Format: NODE.check.FIELDNAME
values	[Wert Wert]	Bei einem stetigen Feld (Bereich) ist der erste Wert das Minimum und der letzte das Maximum. Geben Sie für nominale Felder (Setfelder) alle Werte an. Bei Flagfeldern steht der erste Wert für <i>falsch</i> und der letzte für <i>wahr</i> . Bei automatischer Festlegung dieser Eigenschaft wird die Eigenschaft <i>value_mode</i> auf <i>Specify</i> (Angaben) festgelegt. Die Speicherung wird basierend auf dem ersten Wert in der Liste festgelegt. Wenn z. B. der erste Wert eine <i>Zeichenfolge</i> ist, wird die Speicherung auf "String" gesetzt. Format: NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	Bestimmt, wie Werte beim nächsten Datendurchlauf für ein Feld festgelegt werden. Format: NODE.value_mode.FIELDNAME Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf <i>Angaben</i> festlegen können. Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft <i>Werte</i> fest.
default_value_mode	Read Pass	Gibt die Standardmethode für das Festlegen von Werten für alle Felder an. Format: NODE.default_value_mode Diese Einstellung kann mithilfe der Eigenschaft <i>value_mode</i> für bestimmte Felder überschrieben werden.

Tabelle 40. Allgemeine Eigenschaften von Quellenknoten (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
extend_values	Flag	Gilt, wenn value_mode auf <i>Read</i> (Lesen) gesetzt ist. Setzen Sie den Wert auf <i>T</i> , um neu gelesene Werte zu bereits für das Feld vorhandenen Werten hinzuzufügen. Setzen Sie den Wert auf <i>F</i> , um vorhandene Werte zu verwerfen und sie durch neu gelesene Werte zu ersetzen. Format: NODE.extend_values.FIELDNAME
value_labels	Zeichenfolge	Wird zur Angabe einer Wertbeschriftung verwendet. Beachten Sie, dass Werte zuerst angegeben werden müssen.
enable_missing	Flag	Bei Festlegung auf <i>T</i> wird die Verfolgung von fehlenden Werten für das Feld aktiviert. Format: NODE.enable_missing.FIELDNAME
missing_values	[Wert Wert ...]	Gibt Datenwerte an, die fehlende Daten kennzeichnen. Format: NODE.missing_values.FIELDNAME
range_missing	Flag	Wenn diese Eigenschaft auf <i>T</i> gesetzt ist, wird angegeben, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist. Format: NODE.range_missing.FIELDNAME
missing_lower	Zeichenfolge	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an. Format: NODE.missing_lower.FIELDNAME
missing_upper	Zeichenfolge	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an. Format: NODE.missing_upper.FIELDNAME
null_missing	Flag	Wenn diese Eigenschaft auf <i>T</i> gesetzt ist, werden Nullen (undefinierte Werte, die in der Software als \$null\$ angezeigt werden) als fehlende Werte betrachtet. Format: NODE.null_missing.FIELDNAME
whitespace_missing	Flag	Wenn diese Eigenschaft auf <i>T</i> festgelegt ist, werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet. Format: NODE.whitespace_missing.FIELDNAME
Beschreibung	Zeichenfolge	Dient zur Angabe einer Feldbeschriftung oder Beschreibung.

Tabelle 40. Allgemeine Eigenschaften von Quellenknoten (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
default_include	Flag	Verschlüsselte Eigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden: NODE.default_include Beispiel: set mynode:filternode.default_include = false
include	Flag	Verschlüsselte Eigenschaft, mit der bestimmt wird, ob einzelne Felder aufgenommen oder gefiltert werden: NODE.include.FIELDNAME.
new_name	Zeichenfolge	
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	Diese verschlüsselte Eigenschaft ähnelt type dahingehend, dass sie zum Definieren der dem Feld zugeordneten Messung verwendet werden kann. Das Python-Scripting unterscheidet sich dadurch, dass der Setter-Funktion auch einer der MeasureType-Werte übergeben werden kann, während die Getter-Funktion immer für MeasureType-Werte zurückgibt.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Bei Sammlungsfeldern (Listen mit einer Tiefe von 0) definiert diese verschlüsselte Eigenschaft den Messtyp, der den zugrunde liegenden Werten zugeordnet ist.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Bei georäumlichen Feldern definiert diese verschlüsselte Eigenschaft den Typ des durch dieses Feld dargestellten georäumlichen Objekts. Dies sollte konsistent mit der Listentiefe der Werte sein.
has_coordinate_system	boolesch	Bei georäumlichen Feldern definiert diese Eigenschaft, ob dieses Feld ein Koordinatensystem hat
coordinate_system	Zeichenfolge	Bei georäumlichen Feldern definiert diese verschlüsselte Eigenschaft das Koordinatensystem für dieses Feld.

Tabelle 40. Allgemeine Eigenschaften von Quellenknoten (Forts.).

Eigenschaftsname	Datentyp	Eigenschaftsbeschreibung
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	Diese verschlüsselte Eigenschaft ähnelt custom_storage dahingehend, dass sie zum Definieren der Speicherüberschreibung für das Feld verwendet werden kann. Das Python-Scripting unterscheidet sich dadurch, dass der Setter-Funktion auch einer der StorageType-Werte übergeben werden kann, während die Getter-Funktion immer für StorageType-Werte zurückgibt.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Bei Listenfeldern gibt diese verschlüsselte Eigenschaft den Speichertyp der zugrunde liegenden Werte an.
custom_list_depth	Ganzzahl	Bei Listenfeldern gibt diese verschlüsselte Eigenschaft die Tiefe des Felds an.

Eigenschaften von "asimport"

Die Analytic Server-Quelle ermöglicht Ihnen die Ausführung eines Streams unter HDFS (Hadoop Distributed File System).

Beispiel

```
node = stream.create("asimport", "My node")
node.setPropertyValue("data_source", "Drug1n")
```

Tabelle 41. Eigenschaften von "asimport".

Eigenschaften von asimport	Datentyp	Eigenschaftsbeschreibung
data_source	Zeichenfolge	Der Name der Datenquelle.

Eigenschaften des Knotens "cognosimport"



Der IBM Cognos BI-Quellenknoten importiert Daten aus Cognos BI-Datenbanken.

Beispiel

```
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]"])
```


Tabelle 42. Eigenschaften des Knotens "cognosimport".

Eigenschaften des Knotens cognosimport	Datentyp	Eigenschaftsbeschreibung
mode	Data Bericht	Gibt an, ob Cognos BI-Daten (Standard) oder Berichte importiert werden sollen.
cognos_connection	{ <i>"Zeichenfolge"</i> , <i>flag</i> , <i>"Zeichenfolge"</i> , <i>"Zeichenfolge"</i> , <i>"Zeichenfolge"</i> }	<p>Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: {"Cognos-Server-URL", login_mode, "Namespace", "Benutzername", "Kennwort"}</p> <p>Dabei gilt Folgendes: Cognos-Server-URL ist die URL des Cognos-Servers, der die Quelle enthält. login_mode gibt an, ob eine anonyme Anmeldung verwendet wird, und ist entweder true oder false; bei Angabe von true sollten die folgenden Felder auf "" gesetzt werden. Namespace gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. Benutzername und Kennwort sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.</p> <p>Anstelle von login_mode sind die folgenden Modi verfügbar:</p> <ul style="list-style-type: none"> anonymousMode. Beispiel: { 'Cognos-Server-URL', 'anonymousMode', "Namespace", "Benutzername", "Kennwort" } credentialMode. Beispiel: { 'Cognos-Server-URL', 'credentialMode', "Namespace", "Benutzername", "Kennwort" } storedCredentialMode. Beispiel: { 'Cognos-Server-URL', 'storedCredentialMode', "gespeicherter_Berechtigungsnachweisname" } <p>Dabei ist gespeicherter_Berechtigungsnachweisname der Name eines Cognos-Berechtigungsnachweises im Repository.</p>
cognos_package_name	<i>Zeichenfolge</i>	<p>Pfad und Name des Cognos-Pakets, von dem Sie Datenobjekte importieren, z. B.: /Public Folders/GOSALES</p> <p>Hinweis: Nur normale Schrägstriche sind gültig.</p>
cognos_items	{ <i>"Feld"</i> , <i>"Feld"</i> , ... , <i>"Feld"</i> }	<p>Der Name eines oder mehrerer Datenobjekte, die importiert werden sollen. Das Format von <i>Feld</i> ist [<i>Namespace</i>].[<i>Abfragesubjekt</i>].[<i>Abfrageelement</i>]</p>

Tabelle 42. Eigenschaften des Knotens "cognosimport" (Forts.).

Eigenschaften des Knotens cognosimport	Datentyp	Eigenschaftsbeschreibung
cognos_filters	Feld	Der Name eines oder mehrerer Filter, die vor dem Datenimport angewendet werden sollen.
cognos_data_parameters	Liste	Werte für Eingabeaufforderungsparameter für Daten. Paare aus Name und Wert sind in geschweifte Klammern eingeschlossen und mehrere Paare werden durch Kommas getrennt. Die gesamte Zeichenfolge steht in eckigen Klammern. Format: [{"Param1", "Wert"},...,{"ParamN", "Wert"}]
cognos_report_directory	Feld	Der Cognos-Pfad eines Ordners bzw. Pakets, aus dem Berichte importiert werden sollen. Beispiel: /Public Folders/GOSALES Hinweis: Nur normale Schrägstriche sind gültig.
cognos_report_name	Feld	Der Pfad und Name innerhalb des Speicherorts eines zu importierenden Berichts.
cognos_report_parameters	Liste	Werte für Berichtsparameter. Paare aus Name und Wert sind in geschweifte Klammern eingeschlossen und mehrere Paare werden durch Kommas getrennt. Die gesamte Zeichenfolge steht in eckigen Klammern. Format: [{"Param1", "Wert"},...,{"ParamN", "Wert"}]

Eigenschaften von "datenbanknode"



Mit dem Datenbankknoten lassen sich Daten aus einer Reihe von anderen Paketen importieren, die ODBC (Open Database Connectivity) verwenden, darunter u. a. Microsoft SQL Server, DB2 und Oracle.

Beispiel

```
import modeler.api
stream = modeler.script.stream()
nnode = 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")
```

Tabelle 43. Eigenschaften von "databasenode".

Eigenschaften von databasenode	Datentyp	Eigenschaftsbeschreibung
mode	Table Query	Legen Sie <i>Table</i> (Tabelle) fest, um die Verbindung zu einer Datenbanktabelle über Dialogfeldsteuerelemente herzustellen, oder <i>Query</i> (Abfrage), um die ausgewählte Datenbank mit SQL abzufragen.
datasource	Zeichenfolge	Datenbankname (siehe auch Hinweis unten).
username	Zeichenfolge	Datenbankverbindungsdetails (siehe auch Hinweis unten).
password	Zeichenfolge	
credential	Zeichenfolge	Name des in IBM SPSS Collaboration and Deployment Services gespeicherten Berechtigungsnachweises. Er kann anstelle der Eigenschaften username und password verwendet werden. Der Benutzername und das Kennwort des Berechtigungsnachweises müssen mit dem Benutzernamen und Kennwort übereinstimmen, die zum Zugreifen auf die Datenbank erforderlich sind.
use_credential		Setzen Sie diese Eigenschaft auf True oder False.
epassword	Zeichenfolge	Gibt ein codiertes Kennwort an, als Alternative zum festen Codieren eines Kennworts in einem Skript. Weitere Informationen finden Sie im Thema „Erstellen eines verschlüsselten Kennworts“ auf Seite 55. Diese Eigenschaft ist während der Ausführung schreibgeschützt.
tablename	Zeichenfolge	Name der Tabelle, auf die Sie zugreifen wollen.
strip_spaces	None Left Right Both	Optionen zum Verwerfen von führenden und nachfolgenden Leerzeichen in Zeichenfolgen.
use_quotes	AsNeeded Always Never	Geben Sie an, ob die Tabellen- und Spaltennamen in Anführungszeichen eingeschlossen sind, wenn Abfragen an die Datenbank gesendet werden (wenn sie z. B. Leerzeichen oder Satzzeichen enthalten).
query	Zeichenfolge	Gibt den SQL-Code für die Abfrage an, die Sie senden wollen.

Anmerkung: Wenn der Datenbankname (in der Eigenschaft datasource mindestens ein Leerzeichen, mindestens einen Punkt oder mindestens einen Unterstrich enthält, können Sie ihn in Zeichenfolgen aus Backslash und doppeltem Anführungszeichen einschließen, damit er als Zeichenfolge behandelt wird. Beispiele: "\"db2v9.7.6_linux\" oder "\"TDATA 131\"".

Anmerkung: Wenn der Datenbankname (in der Eigenschaft datasource) Leerzeichen enthält, können Sie anstatt jeweils einzelner Eigenschaften für Datenquelle, Benutzername und Kennwort eine einzige Datenquelleneigenschaft im folgenden Format verwenden:

Tabelle 44. Datenquellenspezifische Eigenschaften von "datenbasenode".

Eigenschaften von datenbasenode	Datentyp	Eigenschaftsbeschreibung
datasource	Zeichenfolge	Format: {Datenbankname,Benutzername,Kennwort[,true false]} Der letzte Parameter ist für die Verwendung bei verschlüsselten Kennwörtern gedacht. Wenn er auf true gesetzt ist, wird das Kennwort vor der Nutzung verschlüsselt.

Verwenden Sie dieses Format auch für Änderungen der Datenquelle; wenn Sie allerdings nur den Benutzernamen oder das Kennwort ändern möchten, können Sie die Eigenschaften Benutzername oder Kennwort verwenden.

Eigenschaften von "datacollectionimportnode"



Der IBM SPSS Data Collection-Datenimportknoten importiert Umfragedaten auf der Grundlage des von den IBM SPSS Data Collection-Marktforschungsprodukten verwendeten IBM Data Model. Um diesen Knoten verwenden zu können, muss die IBM SPSS Data Collection Data Library installiert sein.

Abbildung 7.
Dimensionsdaten-
importknoten

Beispiel

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Programme/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:/Programme/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

Tabelle 45. Eigenschaften von "datacollectionimportnode".

Eigenschaften von datacollectionimportnode	Datentyp	Eigenschaftsbeschreibung
metadata_name	Zeichenfolge	Der Name des MDSC. Der spezielle Wert DimensionsMDD gibt an, dass das standardmäßige IBM SPSS Data Collection-Metadatendokument verwendet werden soll. Weitere mögliche Werte: mrAD0Dsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC Der spezielle Wert none zeigt an, dass es keinen MDSC gibt.
metadata_file	Zeichenfolge	Name der Datei, in der die Metadaten gespeichert werden.
casedata_name	Zeichenfolge	Der Name des CDSC. Mögliche Werte: mrAD0Dsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXm1Dsc Der spezielle Wert none zeigt an, dass es keinen CDSC gibt.
casedata_source_type	Unknown File Folder UDL DSN	Gibt den Quellentyp des CDSC an.
casedata_file	Zeichenfolge	Wenn casedata_source_type den Wert <i>File</i> aufweist, wird hier die Datei angegeben, die die Falldaten enthält.
casedata_folder	Zeichenfolge	Wenn casedata_source_type den Wert <i>Folder</i> aufweist, wird hier der Ordner angegeben, der die Falldaten enthält.
casedata_udl_string	Zeichenfolge	Wenn casedata_source_type den Wert <i>UDL</i> aufweist, wird hier die OLD-DB-Verbindungszeichenfolge für die Datenquelle angegeben, die die Falldaten enthält.
casedata_dsn_string	Zeichenfolge	Wenn casedata_source_type den Wert <i>DSN</i> , aufweist, wird hier die ODBC-Verbindungszeichenfolge für die Datenquelle angegeben.

Tabelle 45. Eigenschaften von "datacollectionimportnode" (Forts.).

Eigenschaften von datacollectionimportnode	Datentyp	Eigenschaftsbeschreibung
casedata_project	Zeichenfolge	Beim Lesen von Falldaten aus einer IBM SPSS Data Collection-Datenbank, können Sie den Namen des Projekts eingeben. Bei allen anderen Falldatentypen sollte diese Einstellung leer bleiben.
version_import_mode	All Latest Specify	Gibt an, wie mit Versionen umgegangen werden soll.
specific_version	Zeichenfolge	Wenn version_import_mode den Wert <i>Specify</i> aufweist, wird hier die Version der zu importierenden Falldaten festgelegt.
use_language	Zeichenfolge	Gibt an, ob Beschriftungen einer bestimmten Sprache verwendet werden sollen.
language	Zeichenfolge	Wenn use_language den Wert "true" aufweist, wird hier der beim Import zu verwendende Sprachcode festgelegt. Bei diesem Sprachcode sollte es sich um einen der in den Falldaten verfügbaren Sprachcodes handeln.
use_context	Zeichenfolge	Gibt an, ob ein bestimmter Kontext importiert werden sollte. Kontexte dienen dazu, die den Antworten zugeordnete Beschreibung zu variieren.
context	Zeichenfolge	Wenn use_context den Wert "true" aufweist, wird hier der zu importierende Kontext festgelegt. Bei diesem Kontext sollte es sich um einen der in den Falldaten verfügbaren Kontexte handeln.
use_label_type	Zeichenfolge	Gibt an, ob ein bestimmter Beschriftungstyp importiert werden sollte.
label_type	Zeichenfolge	Wenn use_label_type den Wert "true" aufweist, wird hier der zu importierende Beschriftungstyp festgelegt. Bei diesem Beschriftungstyp sollte es sich um einen der in den Falldaten verfügbaren Beschriftungstypen handeln.
user_id	Zeichenfolge	Bei Datenbanken, bei denen eine explizite Anmeldung erforderlich ist, können Sie eine Benutzer-ID und ein Kennwort für den Zugriff auf die Datenquelle angeben.
password	Zeichenfolge	
import_system_variables	Common None All	Gibt an, welche Systemvariablen importiert werden sollen.
import_codes_variables	Flag	
import_sourcefile_variables	Flag	
import_multi_response	MultipleFlags Single	

Eigenschaften von "excelimportnode"



Der Excel-Importknoten importiert Daten aus einer beliebigen Version von Microsoft Excel. Es ist keine ODBC-Datenquelle erforderlich.

Beispiele

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

Tabelle 46. Eigenschaften von "excelimportnode".

Eigenschaften von excelimportnode	Datentyp	Eigenschaftsbeschreibung
excel_file_type	Excel2003 Excel2007	
full_filename	Zeichenfolge	Der vollständige Dateiname mit Pfad.
use_named_range	boolesch	Gibt an, ob ein benannter Bereich verwendet werden soll. Bei "true" wird die Eigenschaft named_range zur Angabe des zu lesenden Bereichs verwendet. Andere Einstellungen für Arbeitsblatt und Datenbereich werden ignoriert.
named_range	Zeichenfolge	
worksheet_mode	Index Name	Gibt an, ob das Arbeitsblatt durch Index oder durch Namen definiert wird.
worksheet_index	Ganzzahl	Index des zu lesenden Arbeitsblattes, beginnend mit 0 für das erste Arbeitsblatt, 1 für das zweite usw.
worksheet_name	Zeichenfolge	Name des zu lesenden Arbeitsblattes.
data_range_mode	FirstNonBlank ExplicitRange	Gibt an, wie der Bereich festgelegt werden sollte.
blank_rows	StopReading ReturnBlankRows	Wenn data_range_mode den Wert FirstNonBlank aufweist, wird hier angegeben, wie mit leeren Zeilen umgegangen werden soll.
explicit_range_start	Zeichenfolge	Wenn data_range_mode den Wert ExplicitRange aufweist, wird hier der Startpunkt des zu lesenden Bereichs angegeben.
explicit_range_end	Zeichenfolge	

Tabelle 46. Eigenschaften von "excelimportnode" (Forts.).

Eigenschaften von excelimportnode	Datentyp	Eigenschaftsbeschreibung
read_field_names	boolesch	Gibt an, ob die erste Zeile im angegebenen Bereich als Feldnamen (Spaltennamen) verwendet werden soll.

Eigenschaften von "evimportnode"



Der Enterprise-Ansichtsknoten erstellt eine Verbindung mit einem IBM SPSS Collaboration and Deployment Services Repository, was es Ihnen ermöglicht, Enterprise-Ansichtsdaten in einen Stream einzulesen und ein Modell in ein Szenario zu packen, auf das andere Benutzer über das Repository zugreifen können.

Anmerkung: Der Enterprise-Ansichtsknoten wurde in SPSS Modeler 16.0 durch den Datenansichtsknoten ersetzt. Der Enterprise-Ansichtsknoten wird für in vorherigen Releases gespeicherte Streams weiterhin unterstützt. Ihnen wird jedoch empfohlen, den Datenansichtsknoten zu verwenden, wenn Streams aktualisiert oder neue Streams erstellt werden.

Beispiel

```
node = stream.create("evimport", "My node")
node.setPropertyValue("connection", ["Training data", "/Application views/Marketing", "LATEST", "Analytic", "/Data Providers/Marketing"])
node.setPropertyValue("tablename", "cust1")
```

Tabelle 47. Eigenschaften von "evimportnode".

Eigenschaften von evimportnode	Datentyp	Eigenschaftsbeschreibung
connection	Liste	Strukturierte Eigenschaft - Liste der Parameter, die eine Enterprise-Ansichtsverbindung ergeben. Format: evimportnode.connection = [description, app_view_path, app_view_version_label, environment, DPD_path]
tablename	Zeichenfolge	Der Name einer Tabelle in der Anwendungsansicht.

Eigenschaften von "fixedfilenode"



Der Knoten des Typs "Datei (fest)" importiert Daten aus Textdateien mit festen Feldern, also aus Dateien, deren Felder nicht begrenzt sind, sondern an derselben Position beginnen und eine feste Länge haben. Maschinell erzeugte Daten oder Legacydaten werden häufig im Format mit festen Feldern gespeichert.

Beispiel

```
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)

```

Tabelle 48. Eigenschaften von "fixedfilenode".

Eigenschaften von fixedfilenode	Datentyp	Eigenschaftsbeschreibung
record_len	Zahl	Legt die Zahl der Zeichen in jedem Datensatz fest.
line_oriented	Flag	Überspringt am Ende jedes Datensatzes das Zeilenwechselzeichen.
decimal_symbol	Default Comma Period	Der Typ des in Ihrer Datenquelle verwendeten Dezimaltrennzeichens.
skip_header	Zahl	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeilen an. Dies ist nützlich, um Spaltenkopfzeilen zu ignorieren.
auto_recognize_datetime	Flag	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quelldaten identifiziert werden.
lines_to_scan	Zahl	
fields	Liste	Strukturierte Eigenschaft.
full_filename	Zeichenfolge	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.
strip_spaces	None Left Right Both	Verwirft beim Importieren führende und nachfolgende Leerzeichen in Zeichenfolgen.
invalid_char_mode	Discard Replace	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Codierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
invalid_char_replacement	Zeichenfolge	
use_custom_values	Flag	
custom_storage	Unknown String Integer Real Zeit Datum Timestamp	

Tabelle 48. Eigenschaften von "fixedfilenode" (Forts.).

Eigenschaften von fixedfilenode	Datentyp	Eigenschaftsbeschreibung
custom_date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Diese Eigenschaft ist nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
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"	Diese Eigenschaft ist nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
custom_decimal_symbol	<i>Feld</i>	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
encoding	StreamDefault SystemDefault "UTF-8"	Legt die Textcodierungsmethode fest.

Knoteneigenschaften von "gsdata_import"



Mit dem georäumlichen Quellenknoten können Sie Kartendaten oder räumliche Daten in Ihre Datenmining-Sitzung einführen.

Tabelle 49. Knoteneigenschaften von "gsdata_import"

Knoteneigenschaften von gsdata_import	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Geben Sie den Dateipfad zu der .shp-Datei ein, die Sie laden wollen.
map_service_URL	Zeichenfolge	Geben Sie die Kartenservice-URL ein, zu der Sie die Verbindung herstellen wollen.
map_name	Zeichenfolge	Nur, wenn map_service_URL verwendet wird; enthält die Ordnerstruktur auf der höchsten Ebene des Kartenservice.

Eigenschaften von "sasimportnode"



Der SAS-Importknoten importiert SAS-Daten in IBM SPSS Modeler.

Beispiel

```
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)
```

Tabelle 50. Eigenschaften von "sasimportnode".

Eigenschaften von sasimportnode	Datentyp	Eigenschaftsbeschreibung
format	Windows UNIX Transport SAS7 SAS8 SAS9	Format der zu importierenden Datei.
full_filename	Zeichenfolge	Der vollständige, von Ihnen eingegebene Dateiname, einschließlich der Pfadangabe.
member_name	Zeichenfolge	Geben Sie ein Mitglied an, das aus der oben angegebenen SAS-Transportdatei importiert werden soll.
read_formats	Flag	Liest Datenformate (wie z. B. Variablenbeschriftungen) aus der angegebenen Formatdatei.
full_format_filename	Zeichenfolge	

Tabelle 50. Eigenschaften von "sasimportnode" (Forts.).

Eigenschaften von sasimportnode	Datentyp	Eigenschaftsbeschreibung
import_names	NamesAndLabels LabelsasNames	Gibt die Methode für die Zuordnung von Variablennamen und -beschriftungen beim Import an.

Eigenschaften von "simgennode"



Der Simulationsgenerierungsknoten bietet eine einfache Möglichkeit zum Generieren simulierter Daten - entweder völlig neu mit benutzerspezifischen statistischen Verteilungen oder automatisch mit den Verteilungen, die durch die Ausführung eines Simulationsanpassungsknoten an vorhandenen historischen Daten gewonnen wurden. Dies ist hilfreich, wenn Sie das Ergebnis eines Vorhersagemodells bei Unsicherheiten in den Modelleingaben auswerten wollen.

Tabelle 51. Eigenschaften von "simgennode".

Eigenschaften von simgennode	Datentyp	Eigenschaftsbeschreibung
fields	Strukturierte Eigenschaft	Siehe Beispiel
correlations	Strukturierte Eigenschaft	Siehe Beispiel
max_cases	Ganzzahl	Minimalwert ist 1.000, Maximalwert ist 2.147.483.647
create_iteration_field	boolesch	
iteration_field_name	Zeichenfolge	
replicate_results	boolesch	
random_seed	Ganzzahl	
overwrite_when_refitting	boolesch	
parameter_xml	Zeichenfolge	Gibt den Parameter Xml als Zeichenfolge zurück.
distribution	Bernoulli Beta Binomial Categorical Exponential Fixed Gamma Lognormal NegativeBinomialFailures NegativeBinomialTrials Normal Poisson Range Triangular Uniform Weibull	distribution ist eine Deklaration des Verteilungsnamens gefolgt von einer Liste, die Paare von Attributnamen und Werten enthält. Beispiel simgennode.setKeyedPropertyValue("distribution", "Age", "Gamma") Anmerkung: Sie können distribution nicht direkt festlegen; Sie verwenden diese Eigenschaft zusammen mit der Eigenschaft fields. Siehe das Beispiel für fields nach dieser Tabelle.
bernoulli_prob	Zahl	$0 \leq \text{bernoulli_prob} \leq 1$
beta_shape1	Zahl	Muss ≥ 0 sein.
beta_shape2	Zahl	Muss ≥ 0 sein.
beta_min	Zahl	Optional. Muss kleiner als beta_max sein.
beta_max	Zahl	Optional. Muss größer als beta_min sein.
binomial_n	Ganzzahl	Muss > 0 sein.

Tabelle 51. Eigenschaften von "simgennode" (Forts.).

Eigenschaften von simgennode	Datentyp	Eigenschaftsbeschreibung
binomial_prob	Zahl	$0 \leq \text{binomial_prob} \leq 1$
binomial_min	Zahl	Optional. Muss kleiner als binomial_max sein.
binomial_max	Zahl	Optional. Muss größer als binomial_min sein.
exponential_scale	Zahl	Muss > 0 sein.
exponential_min	Zahl	Optional. Muss kleiner als exponential_max sein.
exponential_max	Zahl	Optional. Muss größer als exponential_min sein.
fixed_value	Zeichenfolge	
gamma_shape	Zahl	Muss ≥ 0 sein.
gamma_scale	Zahl	Muss ≥ 0 sein.
gamma_min	Zahl	Optional. Muss kleiner als gamma_max sein.
gamma_max	Zahl	Optional. Muss größer als gamma_min sein.
lognormal_shape1	Zahl	Muss ≥ 0 sein.
lognormal_shape2	Zahl	Muss ≥ 0 sein.
lognormal_min	Zahl	Optional. Muss kleiner als lognormal_max sein.
lognormal_max	Zahl	Optional. Muss größer als lognormal_min sein.
negative_bin_failures_threshold	Zahl	Muss ≥ 0 sein.
negative_bin_failures_prob	Zahl	$0 \leq \text{negative_bin_failures_prob} \leq 1$
negative_bin_failures_min	Zahl	Optional. Muss kleiner als negative_bin_failures_max sein.
negative_bin_failures_max	Zahl	Optional. Muss größer als negative_bin_failures_min sein.
negative_bin_trials_threshold	Zahl	Muss ≥ 0 sein.
negative_bin_trials_prob	Zahl	$0 \leq \text{negative_bin_trials_prob} \leq 1$
negative_bin_trials_min	Zahl	Optional. Muss kleiner als negative_bin_trials_max sein.
negative_bin_trials_max	Zahl	Optional. Muss kleiner als negative_bin_trials_min sein.
normal_mean	Zahl	
normal_sd	Zahl	Muss > 0 sein.
normal_min	Zahl	Optional. Muss kleiner als normal_max sein.
normal_max	Zahl	Optional. Muss größer als normal_min sein.
poisson_mean	Zahl	Muss ≥ 0 sein.
poisson_min	Zahl	Optional. Muss kleiner als poisson_max sein.

Tabelle 51. Eigenschaften von "simgennode" (Forts.).

Eigenschaften von simgennode	Datentyp	Eigenschaftsbeschreibung
poisson_max	Zahl	Optional. Muss größer als poisson_min sein.
triangular_mode	Zahl	$\text{triangular_min} \leq \text{triangular_mode} \leq \text{triangular_max}$
triangular_min	Zahl	Muss kleiner als triangular_mode sein.
triangular_max	Zahl	Muss größer als triangular_mode sein.
uniform_min	Zahl	Muss kleiner als uniform_max sein.
uniform_max	Zahl	Muss größer als uniform_min sein.
weibull_rate	Zahl	Muss ≥ 0 sein.
weibull_scale	Zahl	Muss ≥ 0 sein.
weibull_location	Zahl	Muss ≥ 0 sein.
weibull_min	Zahl	Optional. Muss kleiner als weibull_max sein.
weibull_max	Zahl	Optional. Muss größer als weibull_min sein.

Beispiel für 'fields'

Hierbei handelt es sich um einen strukturierten Slotparameter mit der folgenden Syntax:

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

Jede Verteilung ist folgendermaßen definiert:

```
[Verteilungsname, [{Par1}, {Par2}, {Par3}]
```

```
simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [{"Age", "nteger", False, ["Uniform"], 15, 74}])
```

Sie können z. B. das folgende Script verwenden, um einen Knoten zu erstellen, der ein einzelnes Feld mit einer Binomialverteilung generiert:

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

Die Binomialverteilung verarbeitet zwei Parameter: n und prob. Da die Binomialverteilung Mindest- und Maximalwerte nicht unterstützt, werden sie als leere Zeichenfolge bereitgestellt.

Beispiel für 'correlations'

Hierbei handelt es sich um einen strukturierten Slotparameter mit der folgenden Syntax:

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

Die Korrelation kann eine beliebige Zahl zwischen +1 und -1 sein. Sie können beliebig viele Korrelationen angeben. Jede nicht angegebene Korrelation wird auf Null gesetzt. Wenn Felder unbekannt sind, sollte

der Korrelationswert in der Korrelationsmatrix (oder -tabelle) festgelegt werden. Er wird als roter Text angezeigt. Wenn Felder unbekannt sind, kann der Knoten nicht ausgeführt werden.

Eigenschaften von "statisticsimportnode"



Der IBM SPSS Statistics-Dateiknoten liest Daten aus dem Dateiformat *.sav* ein, das von IBM SPSS Statistics verwendet wird, sowie in IBM SPSS Modeler gespeicherte Cache-Dateien, die ebenfalls dasselbe Format verwenden.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften von "statisticsimportnode"“ auf Seite 307.

Eigenschaften des Knotens "tm1import"



Der IBM Cognos TM1-Quellenknoten importiert Daten aus Cognos TM1-Datenbanken.

Tabelle 52. Eigenschaften des Knotens "tm1import".

Eigenschaften des Knotens tm1import	Datentyp	Eigenschaftsbeschreibung
pm_host	Zeichenfolge	Der Hostname. Beispiel: TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	{"Feld", "Feld", ... ,"Feld"}	Eine Listeneigenschaft mit den Verbindungsdetails für den TM1-Server. Format: ["TM1-Servername", "TM1-Benutzername", "TM1-Kennwort"] Beispiel: TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])
selected_view	{"Feld" "Feld"}	Eine Listeneigenschaft, die die Details der Eigenschaften des ausgewählten TM1-Cubes und den Namen der Cube-Ansicht enthält, aus der Daten nach SPSS importiert werden. Beispiel: TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])

Eigenschaften von "userinputnode"



Der Benutzereingabeknoten bietet eine einfache Möglichkeit, künstliche Daten zu erstellen. Dazu können entweder neue Daten ohne Vorlage erstellt oder vorhandene Daten geändert werden. Diese Funktion ist nützlich, wenn Sie z. B. ein Testdataset für die Modellierung erstellen möchten.

Beispiel

```

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")

```

Table 53. Eigenschaften von "userinputnode".

Eigenschaften von userinputnode	Datentyp	Eigenschaftsbeschreibung
data		
names		Strukturierter Slot, der eine vom Knoten erstellte Liste der Feldnamen festlegt oder zurückgibt.
custom_storage	Unknown String Integer Real Zeit Datum Timestamp	Schlüsselslot, der den Speichertyp für ein Feld festlegt oder zurückgibt.
data_mode	Combined Ordered	Wenn Combined angegeben wird, werden Datensätze für jede Kombination aus Set-Werten und Min./Max.-Werten erstellt. Die Anzahl der erstellten Datensätze entspricht dem Produkt der Anzahl der Werte in jedem Feld. Wenn Ordered angegeben wird, wird zur Erstellung einer Datenzeile aus jeder Spalte für jeden Datensatz genau ein Wert abgerufen. Die Anzahl der erstellten Datensätze entspricht der höchsten Anzahl von Werten, die einem Feld zugeordnet sind. Alle Felder mit weniger Datenwerten werden mit Nullwerten aufgefüllt.
values		Anmerkung: Diese Eigenschaft wurde zugunsten von userinputnode.data verworfen und sollte nicht mehr verwendet werden.

Eigenschaften von "variablefilenode"



Der Variablendateiknoten liest Daten aus Textdateien mit freien Feldern, also aus Dateien, deren Datensätze eine konstante Anzahl von Feldern, aber eine variable Anzahl von Zeichen enthalten. Dieser Knoten ist außerdem nützlich für Dateien mit fester Länge, Überschriftentext und bestimmten Anmerkungen.

Beispiel

```

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])

```

Tabelle 54. Eigenschaften von "variablefilenode".

Eigenschaften von variablefilenode	Datentyp	Eigenschaftsbeschreibung
skip_header	Zahl	Gibt die Anzahl der ab dem Beginn des ersten Datensatzes zu ignorierenden Zeichen an.
num_fields_auto	Flag	Stellt die Anzahl der Felder in jedem Datensatz automatisch fest. Datensätze müssen mit einem Zeilenwechselzeichen abgeschlossen werden.
num_fields	Zahl	Legt die Anzahl der Felder in jedem Datensatz manuell fest.
delimit_space	Flag	Gibt an, welches Zeichen in der Datei für die Feldbegrenzungen verwendet wird.
delimit_tab	Flag	
delimit_new_line	Flag	
delimit_non_printing	Flag	
delimit_comma	Flag	Wenn das Komma sowohl als Feldtrennzeichen als auch als Dezimaltrennzeichen für Streams festgelegt ist, setzen Sie delimit_other auf true und legen Sie ein Komma als Trennzeichen fest, indem Sie die Eigenschaft other verwenden.
delimit_other	Flag	Hier können Sie ein benutzerdefiniertes Trennzeichen festlegen, indem Sie die Eigenschaft other verwenden.
other	Zeichenfolge	Gibt an, welches Trennzeichen verwendet wird, wenn delimit_other auf true gesetzt ist.
decimal_symbol	Default Comma Period	Legt das in der Datenquelle verwendete Dezimaltrennzeichen fest.
multi_blank	Flag	Behandelt mehrere angrenzende leere Trennzeichen als ein einziges Trennzeichen.
read_field_names	Flag	Behandelt die erste Zeile der Datendatei als Beschriftungen für die Spalten.
strip_spaces	None Left Right Both	Verwirft beim Importieren führende und nachfolgende Leerzeichen in Zeichenfolgen.
invalid_char_mode	Discard Replace	Entfernt ungültige Zeichen (null, 0 oder jedes nicht in der aktuellen Codierung enthaltene Zeichen) aus der Dateneingabe oder ersetzt ungültige Zeichen durch das festgelegte, aus einem Zeichen bestehende Symbol.
invalid_char_replacement	Zeichenfolge	
break_case_by_newline	Flag	Gibt an, dass das Zeilenvorschubzeichen als Zeilenbegrenzer verwendet wird.
lines_to_scan	Zahl	Gibt an, wie viele Zeilen nach angegebenen Datentypen durchsucht werden sollen.

Tabelle 54. Eigenschaften von "variablefilenode" (Forts.).

Eigenschaften von variablefilenode	Datentyp	Eigenschaftsbeschreibung
auto_recognize_datetime	Flag	Gibt an, ob Datums- oder Zeitangaben automatisch in den Quelldaten identifiziert werden.
quotes_1	Discard PairAndDiscard IncludeAsText	Gibt an, wie einfache Anführungszeichen beim Import behandelt werden.
quotes_2	Discard PairAndDiscard IncludeAsText	Gibt an, wie doppelte Anführungszeichen beim Import behandelt werden.
full_filename	Zeichenfolge	Vollständiger Name der zu lesenden Datei, einschließlich Pfadangabe.
use_custom_values	Flag	
custom_storage	Unknown String Integer Real Zeit Datum Timestamp	
custom_date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.

Tabelle 54. Eigenschaften von "variablefilenode" (Forts.).

Eigenschaften von variablefilenode	Datentyp	Eigenschaftsbeschreibung
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"	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
custom_decimal_symbol	Feld	Nur anwendbar, wenn ein benutzerdefinierter Speichertyp angegeben wurde.
encoding	StreamDefault SystemDefault "UTF-8"	Legt die Textcodierungsmethode fest.

Eigenschaften von "xmlimportnode"



Der XML-Quellenknoten importiert Daten im XML-Format in den Stream. Sie können eine einzelne Datei oder alle Dateien in einem Verzeichnis importieren. Optional können Sie eine Schemadatei angeben, aus der die XML-Struktur gelesen werden soll.

Beispiel

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

Tabelle 55. Eigenschaften von "xmlimportnode".

Eigenschaften von xmlimportnode	Datentyp	Eigenschaftsbeschreibung
read	single directory	Liest eine einzige Datendatei (Standard) oder alle XML-Dateien in einem Verzeichnis.
recurse	Flag	Legt fest, ob zusätzlich XML-Dateien aus allen Unterverzeichnissen des angegebenen Verzeichnisses gelesen werden sollen.
full_filename	Zeichenfolge	(erforderlich) Vollständiger Pfad und Dateiname der zu importierenden XML-Datei (falls read = single).
directory_name	Zeichenfolge	(erforderlich) Vollständiger Pfad und Dateiname des Verzeichnisses, in dem sich die zu importierenden XML-Dateien befinden (falls read = directory).

Tabelle 55. Eigenschaften von "xmlimportnode" (Forts.).

Eigenschaften von xmlimportnode	Datentyp	Eigenschaftsbeschreibung
full_schema_filename	Zeichenfolge	Vollständiger Pfad und Dateiname der XSD- oder DTD-Datei, aus der die XML-Struktur gelesen werden soll. Wenn Sie diesen Parameter auslassen, wird die Struktur aus der XML-Quellendatei gelesen.
records	Zeichenfolge	XPath-Ausdruck (z. B. /author/name), der die Datensatzgrenze definiert. Jedes Mal, wenn dieses Element in der Quellendatei gefunden wird, wird ein neuer Datensatz erstellt.
mode	read specify	Alle Daten lesen (Standard) oder festlegen, welche Objekte gelesen werden sollen.
fields		Liste der zu importierenden Objekte (Elemente und Attribute). Jedes Objekt in der Liste ist ein XPath-Ausdruck.

Eigenschaften von "dataviewimport"



Der Datenansichtsknoten importiert Datenansichtsdaten in IBM SPSS Modeler.

Beispiel

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

dvnnode = stream.createAt("dataviewimport", "Data View", 96, 96)
dvnnode.setPropertyValue("analytic_data_source",
["", "/folder/adv", "LATEST"])
dvnnode.setPropertyValue("table_name", ["", "com.ibm.spss.Table"])
dvnnode.setPropertyValue("data_access_plan",
["", "DataAccessPlan"])
dvnnode.setPropertyValue("optional_attributes",
[["", "NewDerivedAttribute"]])
dvnnode.setPropertyValue("include_xml", True)
dvnnode.setPropertyValue("include_xml_field", "xml_data")
```

Tabelle 56. Eigenschaften von "dataviewimport"

Eigenschaften von dataviewimport	Datentyp	Eigenschaftsbeschreibung
analytic_data_source	Zeichenfolge	Das in IBM SPSS Collaboration and Deployment Services gespeicherte Analysedatenansichtsobjekt. Der Pfadname und die Versionsbeschriftung für die zu verwendende Version. ["Objekt-ID", "Vollständiger Pfad", "Version"]

Tabelle 56. Eigenschaften von "dataviewimport" (Forts.)

Eigenschaften von dataviewimport	Datentyp	Eigenschaftsbeschreibung
table_name	Zeichenfolge	Die in der Analysedatenansicht verwendete Datenansichtstabelle. Der Tabellename muss über ein Paket qualifiziert sein. Sie können das Paket durch Exportieren der BOM-Datei aus dem IBM SPSS Collaboration and Deployment Services Deployment Manager-Client und Suchen in der Datei default.bom im exportierten ZIP-Archiv abrufen. Der Paketname sollte immer derselbe sein, sofern die BOM-Datei nicht aus IBM Operational Decision Management (iLOG) importiert wurde. ["Objekt-ID", "Name"]
data_access_plan	Zeichenfolge	Der Datenzugriffsplan zum Bereitstellen von Daten für die Analysedatenansicht. ["Objekt-ID", "Name"]
optional_attributes	Zeichenfolge	Eine Liste einzubeziehender abgeleiteter Attribute. [["ID1", "Name1"], ["ID2", "Name2"]]
include_xml	boolesch	true, wenn ein Feld mit XOM-Instanzdaten eingefügt werden soll. Sofern keine IBM Analytical Decision Management iLOG-Knoten verwendet werden, ist die empfohlene Einstellung false. Durch Aktivieren dieser Option wird möglicherweise eine große Menge zusätzlicher Verarbeitungsvorgänge erforderlich.
include_xml_field	Zeichenfolge	Der Name des Felds, das hinzugefügt werden soll, wenn include_xml auf true gesetzt ist.

Kapitel 10. Eigenschaften von Datensatzoperationsknoten

Eigenschaften von "appendnode"



Der Anhangknoten verkettet Gruppen von Datensätzen miteinander. Er ist insbesondere nützlich für die Kombination von Datensätzen mit ähnlicher Struktur, aber unterschiedlichen Daten.

Beispiel

```
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")
```

Tabelle 57. Eigenschaften von "appendnode".

Eigenschaften von appendnode	Datentyp	Eigenschaftsbeschreibung
match_by	Position Name	Datensätze können Sie auf der Grundlage der Position der Felder in der Hauptdatenquelle oder auf der Grundlage der Namen von Feldern der Eingabedatensätze anhängen.
match_case	Flag	Aktiviert für die Übereinstimmung von Feldnamen die Unterscheidung zwischen Groß- und Kleinschreibung.
include_fields_from	Main All	
create_tag_field	Flag	
tag_field_name	Zeichenfolge	

Eigenschaften von "aggregatenode"



Der Aggregatknoten ersetzt eine Sequenz von Eingabedatensätzen durch zusammengefasste, aggregierte Ausgabedatensätze.

Beispiel

```
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")
```

Tabelle 58. Eigenschaften von "aggregatenode".

Eigenschaften von aggregatenode	Datentyp	Eigenschaftsbeschreibung
keys	<i>list</i>	Listet Felder auf, die als Schlüssel für die Aggregation verwendet werden können. Bei den Schlüsselfeldern Geschlecht und Region beispielsweise erhält jede eindeutige Kombination von M und W mit den Regionen N und S (vier eindeutige Kombinationen) einen aggregierten Datensatz.
contiguous	<i>Flag</i>	Wählen Sie diese Option aus, wenn Sie wissen, dass alle Datensätze mit denselben Schlüsselwerten in der Eingabe als zusammenhängende Gruppe vorliegen (z. B., wenn die Eingabe nach Schlüsselfeldern sortiert ist). Dadurch lässt sich eventuell die Leistungsfähigkeit verbessern.
aggregates		Strukturierte Eigenschaft, die die numerischen Felder auflistet, deren Werte aggregiert werden, sowie die ausgewählten Aggregationsmodi.
aggregate_exprs		Verschlüsselte Eigenschaft, die den abgeleiteten Feldnamen mit dem Aggregatausdruck verschlüsselt, der zur Berechnung verwendet wird. Beispiel: aggregatenode.setKeyedPropertyValue("aggregate_exprs", "Na_MAX", "MAX('Na')")
extension	<i>Zeichenfolge</i>	Geben Sie ein Präfix oder Suffix für doppelt aggregierte Felder an (Beispiel unten).
add_as	Suffix Prefix	
inc_record_count	<i>Flag</i>	Erstellt ein zusätzliches Feld, das angibt, wie viele Eingabedatensätze aus den einzelnen Aggregatdatensätzen aggregiert wurden.
count_field	<i>Zeichenfolge</i>	Gibt den Namen des Felds für die Datensatzanzahl an.
allow_approximation	<i>boolesch</i>	Ermöglicht eine Approximation von Reihenfolgestatistiken, wenn eine Aggregation in Analytic Server ausgeführt wird
bin_count	<i>Ganzzahl</i>	Gibt die Anzahl bei der Approximation zu verwendender Klassen an

Eigenschaften von "balancenode"



Der Balancierungsknoten korrigiert Unausgewogenheiten in einem Dataset, sodass dieses eine bestimmte Bedingung erfüllt. Die Balancierungsanweisung passt den Anteil der Datensätze, bei denen eine Bedingung wahr ist, um den angegebenen Faktor an.

Beispiel

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

Tabelle 59. Eigenschaften von "balancenode".

Eigenschaften von balancenode	Datentyp	Eigenschaftsbeschreibung
directives		Strukturierte Eigenschaft, die für eine auf der angegebenen Zahl basierenden Gewichtung des Anteils von Feldwerten verwendet wird (siehe Beispiel unten).
training_data_only	Flag	Gibt an, dass nur Trainingsdaten balanciert werden sollen. Wenn im Stream kein Partitionsfeld vorhanden ist, wird diese Option ignoriert.

Diese Knoteneigenschaft besitzt folgendes Format:

[[Zahl, Zeichenfolge] \ [Zahl, Zeichenfolge] \ ... [Zahl, Zeichenfolge]].

Anmerkung: Wenn Zeichenfolgen (mithilfe von doppelten Anführungszeichen) in den Ausdruck eingebettet werden, muss ihnen das Escapezeichen " \ " vorangestellt werden. Das Zeichen " \ " dient außerdem als Fortsetzungszeichen, mit dem Sie die Argumente übersichtlich aufführen können.

Eigenschaften von "derive_stbnode"



Der Knoten "Space-Time-Boxes" leitet Space-Time-Boxes aus den Feldern für den Breitengrad, den Längengrad und die Zeitmarke ab. Sie können auch mehrere Space-Time-Boxes als Aufenthaltsorte angeben.

Beispiel

```
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_1HOURL", "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)
```

Tabelle 60. Eigenschaften des Knotens "Space-Time-Boxes"

Eigenschaften von derive_stbnode	Datentyp	Eigenschaftsbeschreibung
mode	IndividualRecords Hangouts	
latitude_field	Feld	
longitude_field	Feld	
timestamp_field	Feld	

Tabelle 60. Eigenschaften des Knotens "Space-Time-Boxes" (Forts.)

Eigenschaften von derive_stbnode	Datentyp	Eigenschaftsbeschreibung
hangout_density	Dichte	Eine einfache Dichte. Gültige Dichtewerte siehe densities.
densities	[Dichte,Dichte,..., Dichte]	Jede Dichte ist eine Zeichenfolge, z. B. STB_GH8_1DAY. Anmerkung: Es gibt Einschränkungen dazu, welche Dichten gültig sind. Für den Geohash-Teil können Werte von GH1 bis GH15 verwendet werden. Für den Zeitteil können die folgenden Werte verwendet werden: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	Feld	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Muss eine Zeichenfolge sein.
min_events	Ganzzahl	Der gültige ganzzahlige Minimalwert ist 2.
qualifying_pct	Ganzzahl	Muss im Bereich von 1 bis 100 liegen.
add_extension_as	Präfix Suffix	

Tabelle 60. Eigenschaften des Knotens "Space-Time-Boxes" (Forts.)

Eigenschaften von <code>derive_stbnode</code>	Datentyp	Eigenschaftsbeschreibung
<code>name_extension</code>	<i>Zeichenfolge</i>	

Eigenschaften von "distinctnode"



Der Duplikatknoten entfernt doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Datenstream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden.

Beispiel

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

Tabelle 61. Eigenschaften von "distinctnode".

Eigenschaften von <code>distinctnode</code>	Datentyp	Eigenschaftsbeschreibung
<code>mode</code>	Include Discard	Duplikatknoten entfernen doppelte Datensätze, entweder indem jeweils der erste Datensatz an den Datenstream übergeben wird oder aber indem der erste Datensatz verworfen wird und stattdessen etwaige Duplikate an den Stream übergeben werden.
<code>grouping_fields</code>	<i>list</i>	Listet die Felder auf, die verwendet werden, um zu bestimmen, ob die Datensätze identisch sind. Anmerkung: Diese Eigenschaft wird ab IBM SPSS Modeler 16 nicht mehr unterstützt.
<code>composite_value</code>	Strukturierter Slot	Siehe unten stehendes Beispiel.
<code>composite_values</code>	Strukturierter Slot	Siehe unten stehendes Beispiel.
<code>inc_record_count</code>	<i>Flag</i>	Erstellt ein zusätzliches Feld, das angibt, wie viele Eingabedatensätze aus den einzelnen Aggregatdatensätzen aggregiert wurden.
<code>count_field</code>	<i>Zeichenfolge</i>	Gibt den Namen des Felds für die Datensatzanzahl an.
<code>sort_keys</code>	Strukturierte Eigenschaft.	Anmerkung: Diese Eigenschaft wird ab IBM SPSS Modeler 16 nicht mehr unterstützt.
<code>default_ascending</code>	<i>Flag</i>	
<code>low_distinct_key_count</code>	<i>Flag</i>	Gibt an, dass Sie nur über eine kleine Anzahl an Datensätzen und/oder eine kleine Anzahl an eindeutigen Werten der Schlüsselfelder verfügen.
<code>keys_pre_sorted</code>	<i>Flag</i>	Gibt an, dass alle Datensätze mit denselben Schlüsselwerten in der Eingabe zusammengefasst werden.
<code>disable_sql_generation</code>	<i>Flag</i>	

Beispiel für die Eigenschaft `composite_value`

Die Eigenschaft `composite_value` hat das folgende allgemeine Format:

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

FÜLLOPTION hat das Format [Fülltyp, Option1, Option2, ...].

Beispiele:

```
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"])
```

Für die benutzerdefinierten Optionen sind mehrere Argumente erforderlich, die als Liste hinzugefügt werden. Beispiel:

```
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"])
```

Beispiel für die Eigenschaft `composite_values`

Die Eigenschaft `composite_values` hat das folgende allgemeine Format:

```
node.setPropertyValue("composite_values", [
    [FELD1, [FÜLLOPTION1]],
    [FELD2, [FÜLLOPTION2]],
    .
    .
    .
])
```

Beispiel:

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

Eigenschaften von "mergenode"



Der Zusammenführungsknoten erstellt aus mehreren Eingabedatensätzen einen einzelnen Ausgabedatensatz mit einigen oder allen der Eingabefelder. Er wird zum Zusammenführen von Daten aus verschiedenen Quellen verwendet, beispielsweise Daten über Auslandskunden und erworbene demografische Daten.

Beispiel

```
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"}])

```

Tabelle 62. Eigenschaften von "mergenode".

Eigenschaften von mergenode	Datentyp	Eigenschaftsbeschreibung
method	Order Keys Condition Rankedcondition	Geben Sie an, ob die Datensätze in der Reihenfolge zusammengeführt werden sollen, in der sie in den Datendateien aufgeführt sind, ob eines oder mehrere Felder verwendet werden sollen, um Datensätze mit demselben Wert in den Schlüsselfeldern zusammenzuführen, ob die Datensätze zusammengeführt werden, wenn eine bestimmte Bedingung erfüllt ist, oder ob jede Zeilenpaarung im primären und allen sekundären Datasets zusammengeführt werden soll. Der Rangfolgeausdruck wird verwendet, um mehrere Übereinstimmungen von niedrig nach hoch zu sortieren.
condition	<i>Zeichenfolge</i>	Wenn method auf Condition gesetzt ist, wird hier die Bedingung für das Einschließen oder Verwerfen von Datensätzen angegeben.
key_fields	<i>list</i>	
common_keys	<i>Flag</i>	
join	Inner FullOuter PartialOuter Anti	
outer_join_tag.n	<i>Flag</i>	Bei dieser Eigenschaft ist <i>n</i> der im Dialogfeld für die Datasetauswahl angezeigte Tagname. Beachten Sie, dass mehrere Tagnamen angegeben werden können, da jede beliebige Zahl von Datasets unvollständige Datensätze beitragen könnte.
single_large_input	<i>Flag</i>	Gibt an, ob die Optimierung verwendet werden soll, wenn eine Eingabe vorhanden ist, die im Vergleich mit den anderen Eingaben relativ groß ist.
single_large_input_tag	<i>Zeichenfolge</i>	Geben Sie den Tagnamen an, der im Dialogfeld "Großes Dataset auswählen" angezeigt wird. Beachten Sie, dass die Verwendung dieser Eigenschaft leicht von der Eigenschaft outer_join_tag abweicht (Flag gegenüber Zeichenfolge), da nur ein einziges Eingabedataset angegeben werden kann.
use_existing_sort_keys	<i>Flag</i>	Gibt an, ob die Eingaben bereits nach einem oder mehreren Schlüsselfeldern sortiert sind.
existing_sort_keys	[{'Zeichenfolge', 'Ascending'} \ {'Zeichenfolge', 'Descending'}]	Gibt die bereits sortierten Felder und ihre Sortierrichtung an.

Tabelle 62. Eigenschaften von "mergenode" (Forts.).

Eigenschaften von mergenode	Datentyp	Eigenschaftsbeschreibung
primary_dataset	Zeichenfolge	Wenn method auf Rankedcondition gesetzt ist, wählen Sie die Primärdatei in der Zusammenführung aus. Dies kann als die linke Seite einer Outer Join-Zusammenführung angesehen werden.
add_tag_duplicate	boolesch	Wenn method auf Rankedcondition gesetzt und dafür Y festgelegt ist und wenn das resultierende zusammengeführte Dataset mehrere gleichnamige Felder aus unterschiedlichen Datenquellen enthält, werden die entsprechenden Tags aus den Datenquellen am Anfang der Feldspaltenheader hinzugefügt.
merge_condition	Zeichenfolge	
ranking_expression	Zeichenfolge	
Num_matches	Ganzzahl	Die Anzahl zurückzugebender Übereinstimmungen basierend auf merge_condition und ranking_expression. Minimum: 1, Maximum: 100.

Eigenschaften von "rfmaggregatenode"



Mit dem Knoten "RFM-Aggregat" (Recency-, Frequency-, Monetary-Aggregat) können Sie Daten über die früheren Transaktionen von Kunden verwenden, alle nicht benötigten Daten entfernen und alle verbliebenen Transaktionsdaten zu einer einzigen Zeile zusammenfassen, die angibt, wann der betreffende Kunde zuletzt mit Ihnen in Geschäftskontakt stand, wie viele Transaktionen er vorgenommen hat und wie hoch der Gesamtwert dieser Transaktionen ist.

Beispiel

```
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")
```

Tabelle 63. Eigenschaften von "rfmaggregatenode".

Eigenschaften von rfmaggregatenode	Datentyp	Eigenschaftsbeschreibung
relative_to	Fixed Today	Dient zur Angabe des Datums, ausgehend von dem die Aktualität der Transaktionen berechnet werden soll.
reference_date	Datum	Nur verfügbar, wenn unter relative_to die Option Fixed festgelegt wurde.
contiguous	Flag	Wenn die Daten vorsortiert sind, sodass alle Datensätze mit derselben ID zusammen im Datenstream erscheinen, können Sie mit dieser Option die Verarbeitung beschleunigen.

Tabelle 63. Eigenschaften von "rfmaggregatenode" (Forts.).

Eigenschaften von rfmaggregatenode	Datentyp	Eigenschaftsbeschreibung
id_field	Feld	Dient zur Angabe des für die Identifizierung des Kunden und seiner Transaktionen zu verwendenden Felds.
date_field	Feld	Dient zur Angabe des Datumsfelds, das für die Berechnung der Aktualität verwendet werden soll.
value_field	Feld	Dient zur Angabe des Felds, das für die Berechnung der Geldwerts verwendet werden soll.
extension	Zeichenfolge	Geben Sie ein Präfix oder Suffix für doppelt aggregierte Felder an.
add_as	Suffix Prefix	Gibt an, ob die Erweiterung (extension) als Suffix oder als Präfix hinzugefügt werden soll.
discard_low_value_records	Flag	Ermöglicht die Verwendung der Einstellung discard_records_below.
discard_records_below	Zahl	Dient zur Angabe eines Mindestwerts für die bei der Berechnung der RFM-Gesamtwerte verwendeten Transaktionsdetails. Die für den Wert geltenden Einheiten beziehen sich auf das ausgewählte Feld value.
only_recent_transactions	Flag	Dient zur Aktivierung der Einstellung specify_transaction_date bzw. transaction_within_last.
specify_transaction_date	Flag	
transaction_date_after	Datum	Nur verfügbar, wenn specify_transaction_date ausgewählt wurde. Dient zur Angabe des Transaktionsdatums, nach dem die Datensätze in die Analyse aufgenommen werden sollen.
transaction_within_last	Zahl	Nur verfügbar, wenn transaction_within_last ausgewählt wurde. Hier können Sie anhand von Anzahl und Typ der Zeiträume (Tage, Wochen, Monate oder Jahre) angeben, wie weit die in die Analyse aufzunehmenden Datensätze ausgehend von "Aktualität (Recency) berechnen relativ zu" zurückliegen dürfen.
transaction_scale	Days Weeks Months Years	Nur verfügbar, wenn transaction_within_last ausgewählt wurde. Hier können Sie anhand von Anzahl und Typ der Zeiträume (Tage, Wochen, Monate oder Jahre) angeben, wie weit die in die Analyse aufzunehmenden Datensätze ausgehend von "Aktualität (Recency) berechnen relativ zu" zurückliegen dürfen.
save_r2	Flag	Zeigt für jeden Kunden das Datum der zweitaktuellsten Transaktion an.
save_r3	Flag	Nur verfügbar, wenn save_r2 ausgewählt wurde. Zeigt für jeden Kunden das Datum der drittaktuellsten Transaktion an.

Eigenschaften von "Rprocessnode"



Mit dem Knoten "R-Prozess" können Sie Daten aus einem IBM(r) SPSS(r) Modeler-Stream beziehen und diese mit ihrem eigenen benutzerdefinierten R-Script ändern. Nachdem die Daten geändert wurden, werden sie an den Stream zurückgegeben.

Beispiel

```
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")
```

Tabelle 64. Eigenschaften von "Rprocessnode".

Eigenschaften von Rprocessnode	Datentyp	Eigenschaftsbeschreibung
Syntax	<i>Zeichenfolge</i>	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	<i>Flag</i>	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	<i>Flag</i>	

Eigenschaften von "samplenode"



Der Stichprobenknoten wählt ein Subset der Datensätze aus. Es wird eine Vielzahl von Stichprobentypen unterstützt, darunter geschichtete, gruppierte (Clusterstichproben) und nichtzufällige (strukturierte) Stichproben. Eine Stichprobenziehung kann nützlich zur Verbesserung der Leistungsfähigkeit und zur Auswahl von verwandten Datensätzen bzw. Transaktionen für die Analyse sein.

Beispiel

```
/* 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]])

```

Table 65. Eigenschaften von "sampleneode".

Eigenschaften von <code>sampleneode</code>	Datentyp	Eigenschaftsbeschreibung
<code>method</code>	Simple Complex	
<code>mode</code>	Include Discard	Einschließen oder Verwerfen von Datensätzen, die die angegebene Bedingung erfüllen.
<code>sample_type</code>	First OneInN RandomPct	Gibt die Methode der Stichprobenziehung an.
<code>first_n</code>	<i>Ganzzahl</i>	Datensätze bis zum angegebenen Abbruchpunkt werden eingeschlossen oder verworfen.
<code>one_in_n</code>	<i>Zahl</i>	Jeden <i>n</i> -ten Datensatz einschließen oder verwerfen.
<code>rand_pct</code>	<i>Zahl</i>	Geben Sie den Prozentsatz der einzuschließenden oder zu verwerfenden Datensätze an.
<code>use_max_size</code>	<i>Flag</i>	Aktivieren Sie die Verwendung der Einstellung <code>maximum_size</code> .
<code>maximum_size</code>	<i>Ganzzahl</i>	Geben Sie die größte Stichprobe an, die in den Datenstream eingeschlossen oder verworfen werden soll. Diese Option ist redundant und wird daher inaktiviert, wenn <code>First</code> und <code>Include</code> angegeben werden.
<code>set_random_seed</code>	<i>Flag</i>	Aktiviert die Verwendung der Einstellung für den Zufallsstartwert.
<code>random_seed</code>	<i>Ganzzahl</i>	Geben Sie den Wert an, der als Startwert für den Zufallsgenerator verwendet wird.
<code>complex_sample_type</code>	Random Systematic	
<code>sample_units</code>	Proportions Counts	
<code>sample_size_proportions</code>	Fixed Custom Variable	
<code>sample_size_counts</code>	Fixed Custom Variable	
<code>fixed_proportions</code>	<i>Zahl</i>	
<code>fixed_counts</code>	<i>Ganzzahl</i>	
<code>variable_proportions</code>	<i>Feld</i>	
<code>variable_counts</code>	<i>Feld</i>	
<code>use_min_stratum_size</code>	<i>Flag</i>	
<code>minimum_stratum_size</code>	<i>Ganzzahl</i>	Diese Option gilt nur, wenn eine komplexe Stichprobe mit <code>Sample units=Proportions</code> gezogen wird.
<code>use_max_stratum_size</code>	<i>Flag</i>	

Tabelle 65. Eigenschaften von "samplenode" (Forts.).

Eigenschaften von samplenode	Datentyp	Eigenschaftsbeschreibung
maximum_stratum_size	Ganzzahl	Diese Option gilt nur, wenn eine komplexe Stichprobe mit Sample units=Proportions gezogen wird.
clusters	Feld	
stratify_by	[Feld1 ... FeldN]	
specify_input_weight	Flag	
input_weight	Feld	
new_output_weight	Zeichenfolge	
sizes_proportions	[[string Zeichenfolgewart]{string Zeichenfolgewart}...]	Wenn sample_units=proportions und sample_size_proportions=Custom festgelegt wurden, wird hiermit ein Wert für jede mögliche Kombination der Werte von Schichtungsfeldern angegeben.
default_proportion	Zahl	
sizes_counts	[[string Zeichenfolgewart]{string Zeichenfolgewart}...]	Dient zur Angabe eines Werts für jede mögliche Kombination der Werte von Schichtungsfeldern. Die Verwendung ist ähnlich wie bei sizes_proportions, es wird jedoch statt eines Anteils eine Ganzzahl angegeben.
default_count	Zahl	

Eigenschaften von "selectnode"



Der Auswahlknoten wählt auf der Grundlage einer bestimmten Bedingung ein Subset von Datensätzen aus einem Datenstream aus oder verwirft sie. Sie können beispielsweise die Datensätze auswählen, die zu einer bestimmten Verkaufsregion gehören.

Beispiel

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

Tabelle 66. Eigenschaften von "selectnode".

Eigenschaften von selectnode	Datentyp	Eigenschaftsbeschreibung
mode	Include Discard	Definiert, ob die ausgewählten Datensätze eingeschlossen oder verworfen werden sollen.
condition	Zeichenfolge	Bedingung für das Einschließen oder Verwerfen von Datensätzen.

Eigenschaften von "sortnode"



Der Sortierknoten sortiert Datensätze anhand der Werte eines oder mehrerer Felder in aufsteigender oder absteigender Reihenfolge.

Beispiel

```
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"}])
```

Tabelle 67. Eigenschaften von "sortnode".

Eigenschaften von sortnode	Datentyp	Eigenschaftsbeschreibung
keys	<i>list</i>	Gibt die Felder an, nach denen sortiert werden soll. Wenn keine Richtung angegeben ist, wird der Standard verwendet.
default_ascending	<i>Flag</i>	Gibt die Standardsortierreihenfolge an.
use_existing_keys	<i>Flag</i>	Gibt an, ob die Sortierung durch Verwendung der vorherigen Sortierreihenfolge für bereits sortierte Felder optimiert werden soll.
existing_keys		Gibt die bereits sortieren Felder und ihre Sortierrichtung an. Verwendet dasselbe Format wie die Eigenschaft keys.

Eigenschaften von "streamingts"



Der Streaming-ZR-Knoten erstellt und bewertet Zeitreihenmodelle in einem Schritt, ohne dass ein Zeitintervallknoten erforderlich ist.

Beispiel

```
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")
```

Tabelle 68. Eigenschaften von "streamingts".

Eigenschaften von streamingts	Datentyp	Eigenschaftsbeschreibung
custom_fields	<i>Flag</i>	Bei custom_fields=false werden die Einstellungen aus einem vorausgehenden Typknoten verwendet. Bei custom_fields=true müssen targets und inputs angegeben werden.
targets	[Feld1...FeldN]	
inputs	[Feld1...FeldN]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	<i>Flag</i>	
conf_limit_pct	<i>reell</i>	

Tabelle 68. Eigenschaften von "streamingts" (Forts.).

Eigenschaften von streamingts	Datentyp	Eigenschaftsbeschreibung
use_time_intervals_node	Flag	Bei use_time_intervals_node=true werden die Einstellungen aus einem vorausgehenden Zeitintervallknoten verwendet. Bei use_time_intervals_node=false müssen interval_offset_position, interval_offset, und interval_type angegeben werden.
interval_offset_position	LastObservation LastRecord	LastObservation bezieht sich auf Letzte gültige Beobachtung . LastRecord bezieht sich auf Vom letzten Datensatz rückwärts zählen .
interval_offset	Zahl	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
Ereignisse	Felder	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	Flag	
detect_outliers	Flag	
expert_outlier_additive	Flag	
expert_outlier_level_shift	Flag	
expert_outlier_innovational	Flag	
expert_outlier_transient	Flag	
expert_outlier_seasonal_additive	Flag	
expert_outlier_local_trend	Flag	
expert_outlier_additive_patch	Flag	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_d	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_q	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arima_sp	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten

Tabelle 68. Eigenschaften von "streamingts" (Forts.).

Eigenschaften von streamingts	Datentyp	Eigenschaftsbeschreibung
arma_sd	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arma_sq	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arma_transformation_type	None SquareRoot NaturalLog	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
arma_include_constant	Flag	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten
tf_arma_p.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_d.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_q.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_sp.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_sd.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_sq.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_delay.Feldname	Ganzzahl	Dieselbe Eigenschaft wie für den Zeitreihenmodellierungsknoten. Für Transferfunktionen.
tf_arma_transformation_type.feldname	None SquareRoot NaturalLog	
arma_detect_outlier_mode	None Automatic	
arma_outlier_additive	Flag	
arma_outlier_level_shift	Flag	
arma_outlier_innovational	Flag	
arma_outlier_transient	Flag	
arma_outlier_seasonal_additive	Flag	
arma_outlier_local_trend	Flag	
arma_outlier_additive_patch	Flag	
deployment_force_rebuild	Flag	
deployment_rebuild_mode	Count Percent	
deployment_rebuild_count	Zahl	
deployment_rebuild_pct	Zahl	

Table 68. Properties of "streamings" (Continued).

Eigenschaften von streamings	Datentyp	Eigenschaftsbeschreibung
deployment_rebuild_field	<Feld>	

Kapitel 11. Eigenschaften von Feldoperationsknoten

Eigenschaften von "anonymizenode"



Der Anonymisierungsknoten ändert die Art und Weise, wie Feldnamen und -werte weiter unten im Stream dargestellt werden, und verschleiert damit die ursprünglichen Daten. Dies kann sinnvoll sein, wenn andere Benutzer in die Lage versetzt werden sollen, Modelle unter Verwendung vertraulicher Daten wie beispielsweise Kundennamen zu erstellen.

Beispiel

```
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")
```

Table 69. Eigenschaften von "anonymizenode"

Eigenschaften von anonymizenode	Datentyp	Eigenschaftsbeschreibung
enable_anonymize	Flag	Bei Festlegung auf True wird die Anonymisierung von Feldwerten aktiviert (entspricht der Auswahl von Ja für das betreffende Feld in der Spalte "Werte anonymisieren").
use_prefix	Flag	Bei Festlegung auf True wird ein benutzerdefiniertes Präfix verwendet, sofern eines angegeben wurde. Gilt für Felder, die mit der Hash-Methode anonymisiert werden, und entspricht der Auswahl des Optionsfelds Benutzerdefiniert im Dialogfeld "Werte ersetzen" für das betreffende Feld.
prefix	Zeichenfolge	Entspricht der Eingabe eines Präfixes in das Textfeld im Dialogfeld "Werte ersetzen". Das Standardpräfix ist der Standardwert, wenn keine anderen Angaben gemacht wurden.
transformation	Random Fixed	Bestimmt, ob die Transformationsparameter für ein durch die Transformationsmethode anonymisiertes Feld zufällig oder fest sein sollen.
set_random_seed	Flag	Bei Festlegung auf True wird der angegebene Startwert für den Zufallsgenerator verwendet (sofern außerdem transformation auf Random gesetzt ist).
random_seed	Ganzzahl	Wenn set_random_seed auf True gesetzt ist, wird dieser Wert als Startwert für den Zufallsgenerator verwendet.
Skala	Zahl	Wenn transformation auf Fixed gesetzt ist, wird dieser Wert als Wert für "Skalieren um" verwendet. Der Höchstwert für die Skalierung ist normalerweise 10; er kann jedoch gesenkt werden, um einen Überlauf zu vermeiden.

Tabelle 69. Eigenschaften von "anonymizenode" (Forts.)

Eigenschaften von anonymizenode	Datentyp	Eigenschaftsbeschreibung
translate	Zahl	Wenn transformation auf Fixed gesetzt ist, wird dieser Wert als Wert für die Verschiebung ("translate") verwendet. Der Höchstwert für die Verschiebung ist normalerweise 10; er kann jedoch gesenkt werden, um einen Überlauf zu vermeiden.

Eigenschaften von "autodatapreinode"



Der Knoten "Automated Data Preparation" (ADP) kann Ihre Daten analysieren und Korrekturen identifizieren, problematische oder vermutlich überflüssige Felder ausschließen, wie erforderlich neue Attribute ableiten und die Leistung durch intelligente Prüf- und Stichprobenverfahren verbessern. Sie können den Knoten vollständig automatisiert nutzen, damit er Korrekturen wählen und anwenden kann. Sie können die Änderungen aber auch prüfen, bevor sie durchgeführt werden, und wie gewünscht akzeptieren, ablehnen oder ändern.

Beispiel

```
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")
```

Tabelle 70. Eigenschaften von "autodatapreinode"

Eigenschaften von autodatapreinode	Datentyp	Eigenschaftsbeschreibung
objective	Balanced Speed Accuracy Custom	
custom_fields	Flag	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
target	Feld	Gibt ein einzelnes Zielfeld an.
inputs	[Feld1 ... FeldN]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
use_frequency	Flag	
frequency_field	Feld	
use_weight	Flag	
weight_field	Feld	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	Flag	Zugriff auf alle Datums- und Zeitfelder kontrollieren

Tabelle 70. Eigenschaften von "autodatapreprobe" (Forts.)

Eigenschaften von autodatapreprobe	Datentyp	Eigenschaftsbeschreibung
compute_time_until_date	Flag	
reference_date	Today Fixed	
fixed_date	Datum	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	Flag	
reference_time	CurrentTime Fixed	
fixed_time	Zeit	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	Flag	
extract_month_from_date	Flag	
extract_day_from_date	Flag	
extract_hour_from_time	Flag	
extract_minute_from_time	Flag	
extract_second_from_time	Flag	
exclude_low_quality_inputs	Flag	
exclude_too_many_missing	Flag	
maximum_percentage_missing	Zahl	
exclude_too_many_categories	Flag	
maximum_number_categories	Zahl	
exclude_if_large_category	Flag	
maximum_percentage_category	Zahl	
prepare_inputs_and_target	Flag	
adjust_type_inputs	Flag	
adjust_type_target	Flag	
reorder_nominal_inputs	Flag	
reorder_nominal_target	Flag	
replace_outliers_inputs	Flag	
replace_outliers_target	Flag	
replace_missing_continuous_inputs	Flag	
replace_missing_continuous_target	Flag	
replace_missing_nominal_inputs	Flag	
replace_missing_nominal_target	Flag	

Tabelle 70. Eigenschaften von "autodataprepnode" (Forts.)

Eigenschaften von autodataprepnode	Datentyp	Eigenschaftsbeschreibung
replace_missing_ordinal_inputs	Flag	
replace_missing_ordinal_target	Flag	
maximum_values_for_ordinal	Zahl	
minimum_values_for_continuous	Zahl	
outlier_cutoff_value	Zahl	
outlier_method	Replace Delete	
rescale_continuous_inputs	Flag	
rescaling_method	MinMax ZScore	
min_max_minimum	Zahl	
min_max_maximum	Zahl	
z_score_final_mean	Zahl	
z_score_final_sd	Zahl	
rescale_continuous_target	Flag	
target_final_mean	Zahl	
target_final_sd	Zahl	
transform_select_input_fields	Flag	
maximize_association_with_target	Flag	
p_value_for_merging	Zahl	
merge_ordinal_features	Flag	
merge_nominal_features	Flag	
minimum_cases_in_category	Zahl	
bin_continuous_fields	Flag	
p_value_for_binning	Zahl	
perform_feature_selection	Flag	
p_value_for_selection	Zahl	
perform_feature_construction	Flag	
transformed_target_name_extension	Zeichenfolge	
transformed_inputs_name_extension	Zeichenfolge	
constructed_features_root_name	Zeichenfolge	
years_duration_name_extension	Zeichenfolge	
months_duration_name_extension	Zeichenfolge	
days_duration_name_extension	Zeichenfolge	
hours_duration_name_extension	Zeichenfolge	
minutes_duration_name_extension	Zeichenfolge	
seconds_duration_name_extension	Zeichenfolge	
year_cyclical_name_extension	Zeichenfolge	
month_cyclical_name_extension	Zeichenfolge	
day_cyclical_name_extension	Zeichenfolge	

Tabelle 70. Eigenschaften von "autodatapreinode" (Forts.)

Eigenschaften von autodatapreinode	Datentyp	Eigenschaftsbeschreibung
hour_cyclical_name_extension	Zeichenfolge	
minute_cyclical_name_extension	Zeichenfolge	
second_cyclical_name_extension	Zeichenfolge	

Eigenschaften von "astimeintervalsnode"



Der ursprüngliche Zeitintervallknoten ist nicht mit Analytic Server (AS) kompatibel. Der AS-Zeitintervallknoten (neu in SPSS Modeler Release 17.0) enthält ein Subset der Funktionen des vorhandenen Zeitintervallknotens, die mit Analytic Server verwendet werden können.

Verwenden Sie den AS-Zeitintervallknoten, um Intervalle anzugeben und ein neues Zeitfeld für Schätzung oder Vorhersage abzuleiten. Ein vollständiger Bereich von Zeitintervallen (von Sekunden bis zu Jahren) wird unterstützt.

Tabelle 71. Eigenschaften von "astimeintervalsnode"

Eigenschaften von astimeintervalsnode	Datentyp	Eigenschaftsbeschreibung
time_field	Feld	Kann nur ein einzelnes stetiges Feld akzeptieren. Dieses Feld wird vom Knoten als Aggregationsschlüssel für das Umwandeln des Intervalls verwendet. Wird hier ein Feld für ganze Zahlen verwendet, wird es als Zeitindex interpretiert.
dimensions	[Feld1 Feld2 ... Feldn]	Diese Felder werden zum Erstellen einzelner Zeitreihen basierend auf den Feldwerten verwendet.
fields_to_aggregate	[Feld1 Feld2 ... Feldn]	Diese Felder werden als Teil der Änderung des Zeitraums für das Zeitfeld aggregiert. Alle nicht in diese Auswahlfunktion eingeschlossenen Felder werden aus den Daten herausgefiltert, die den Knoten verlassen.

Eigenschaften von "binningnode"



Der Klassierknoten erstellt automatisch neue nominale Felder (Setfelder) auf der Grundlage der Werte eines oder mehrerer bestehender stetiger Felder (numerischer Bereich). Sie können beispielsweise ein stetiges Einkommensfeld in ein neues kategoriales Feld transformieren, das Einkommensgruppen als Abweichungen vom Mittelwert enthält. Nach der Erstellung von Klassen für das neue Feld können Sie einen Ableitungsknoten anhand der Trennwerte generieren.

Beispiel

```
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)
```

Table 72. Eigenschaften von "binningnode"

Eigenschaften von binningnode	Datentyp	Eigenschaftsbeschreibung
fields	[Feld1 Feld2 ... Feldn]	Stetige Felder (numerischer Bereich) mit ausstehender Transformation. Sie können mehrere Felder gleichzeitig klassieren.
method	FixedWidth EqualCount Rank SDev Optimal	Methode, die zur Ermittlung der Trennwerte für neue Feld-Bins (Kategorien) verwendet wird.
rcalculate_bins	Always IfNecessary	Gibt an, ob bei jeder Ausführung des Knotens die Klassen neu berechnet und die Daten in die relevante Klasse eingeordnet werden sollen oder ob Daten nur zu bestehenden Klassen und etwaig hinzugefügten neuen Klassen hinzugefügt werden sollen.
fixed_width_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_BIN</i> .
fixed_width_add_as	Suffix Prefix	Gibt an, ob die Erweiterung am Ende (Suffix) oder am Anfang (Präfix) des Feldnamens eingefügt werden soll. Die Standarderweiterung lautet <i>income_BIN</i> .
fixed_bin_method	Width Anzahl	
fixed_bin_count	Ganzzahl	Gibt eine Ganzzahl an, die zur Bestimmung der Anzahl der Klassen (Kategorien) mit fester Breite für die neuen Felder verwendet wird.
fixed_bin_width	reell	Wert (ganzzahlig oder reell), der zu Berechnung der Breite der Klasse verwendet wird.
equal_count_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_TILE</i> .
equal_count_add_as	Suffix Prefix	Gibt eine Erweiterung (Suffix oder Präfix) an, die für die mithilfe von Standard-N-Perzentilen generierten Felder verwendet wird. Die Standarderweiterung ist <i>_TILE</i> plus <i>N</i> ; dabei steht <i>N</i> für die Nummer des Perzentils.
tile4	Flag	Generiert vier Quantilklassen, die jeweils 25 % der Fälle enthalten.
tile5	Flag	Generiert fünf Quintilklassen.
tile10	Flag	Generiert 10 Dezilklassen.
tile20	Flag	Generiert 20 Vingtilklassen.
tile100	Flag	Generiert 100 Perzentilklassen.
use_custom_tile	Flag	
custom_tile_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_TILEN</i> .
custom_tile_add_as	Suffix Prefix	
custom_tile	Ganzzahl	

Tabelle 72. Eigenschaften von "binningnode" (Forts.)

Eigenschaften von binningnode	Datentyp	Eigenschaftsbeschreibung
equal_count_method	RecordCount ValueSum	Die Methode RecordCount versucht, jeder Klasse eine gleich große Anzahl von Datensätzen zuzuweisen, während ValueSum Datensätze so zuweist, dass die Summe der Werte in jeder Klasse gleich groß ist.
tied_values_method	Next Current Zufallsfunktionen	Gibt an, in welche Klasse Daten mit gebundenen Werten (Werten mit Gleichstand) eingeordnet werden sollen.
rank_order	Ascending Descending	Diese Eigenschaft beinhaltet Ascending (der niedrigste Wert wird mit "1" gekennzeichnet) oder Descending (der höchste Wert wird mit "1" gekennzeichnet).
rank_add_as	Suffix Prefix	Mit dieser Option werden Rang, relativer Rang und Prozentsatzrang angewendet.
Rang	Flag	
rank_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_RANK</i> .
rank_fractional	Flag	Weist Fällen Ränge zu, wobei der Wert des neuen Felds gleich dem Rang dividiert durch die Summe der Gewichtungen der nicht fehlenden Fälle ist. Relative Ränge fallen in den Bereich 0-1.
rank_fractional_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_F_RANK</i> .
rank_pct	Flag	Die einzelnen Ränge werden durch die Anzahl der Datensätze mit gültigen Werten dividiert und mit 100 multipliziert. Als Prozentsatz angegebene Bruchzahlränge fallen in den Bereich 1-100.
rank_pct_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_P_RANK</i> .
sdev_name_extension	Zeichenfolge	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	Zeichenfolge	Die Standarderweiterung lautet <i>_OPTIMAL</i> .
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	Feld	Als Supervisorfeld ausgewähltes Feld, mit dem die für die Klassierung ausgewählten Felder in Bezug stehen.
optimal_merge_bins	Flag	Gibt an, dass alle Klassen mit kleinen Fallzahlen zu einer größeren, benachbarten Klasse hinzugefügt werden.
optimal_small_bin_threshold	Ganzzahl	
optimal_pre_bin	Flag	Gibt an, dass eine Vorklassierung des Datasets durchgeführt werden soll.

Tabelle 72. Eigenschaften von "binningnode" (Forts.)

Eigenschaften von binningnode	Datentyp	Eigenschaftsbeschreibung
optimal_max_bins	Ganzzahl	Gibt eine Obergrenze an, um das Erstellen einer übermäßig großen Anzahl von Feldern zu verhindern.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Eigenschaften von "derivenode"



Der Ableitungsknoten ändert Datenwerte oder erstellt neue Felder aus einem oder mehreren bestehenden Feldern. Er erstellt Felder vom Typ "Formel", "Flag", "Nominal", "Status", "Anzahl" und "Bedingt".

Beispiel 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\"")
```

Beispiel 2

Dieses Script nimmt an, dass zwei numerische Spalten mit den Namen XPos und YPos vorhanden sind, die die X- und Y-Koordinaten eines Punkts (z. B. dem Ort eines Ereignisses) darstellen. Das Script erstellt einen Ableitungsknoten, der eine georäumliche Spalte aus den X- und Y-Koordinaten berechnet, die diesen Punkt in einem bestimmten Koordinatensystem darstellen:

```
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")
```


Tabelle 73. Eigenschaften von "derivenode"

Eigenschaften von <i>derivenode</i>	Datentyp	Eigenschaftsbeschreibung
<i>new_name</i>	<i>Zeichenfolge</i>	Name des neuen Felds.
<i>mode</i>	Einzeln Multiple	Gibt eines oder mehrere Felder an.
<i>fields</i>	<i>list</i>	Wird nur im Modus "Multiple" (Mehrere) zur Auswahl mehrerer Felder verwendet.
<i>name_extension</i>	<i>Zeichenfolge</i>	Gibt die Erweiterung für die neuen Feldnamen an.
<i>add_as</i>	Suffix Prefix	Fügt die Erweiterung als Präfix (am Anfang) oder als Suffix (am Ende) des Feldnamens ein.
<i>result_type</i>	Formula Flag Set State Count Bedingung	Die sechs Typen neuer Felder, die Sie erstellen können.
<i>formula_expr</i>	<i>Zeichenfolge</i>	Ausdruck zum Berechnen eines neuen Feldwerts in einem Ableitungsknoten.
<i>flag_expr</i>	<i>Zeichenfolge</i>	
<i>flag_true</i>	<i>Zeichenfolge</i>	
<i>flag_false</i>	<i>Zeichenfolge</i>	
<i>set_default</i>	<i>Zeichenfolge</i>	
<i>set_value_cond</i>	<i>Zeichenfolge</i>	Wird zur Bereitstellung der Bedingung, die einem bestimmten Wert zugeordnet ist, strukturiert.
<i>state_on_val</i>	<i>Zeichenfolge</i>	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "On" (Ein) erfüllt ist.
<i>state_off_val</i>	<i>Zeichenfolge</i>	Dient zur Angabe des Werts für das neue Feld, wenn die Bedingung für "Off" (Aus) erfüllt ist.
<i>state_on_expression</i>	<i>Zeichenfolge</i>	
<i>state_off_expression</i>	<i>Zeichenfolge</i>	
<i>state_initial</i>	On Off	Weist jedem Datensatz des neuen Felds einen Anfangswert On (Ein) oder Off (Aus) zu. Dieser Wert kann sich ändern, wenn die einzelnen Bedingungen erfüllt werden.
<i>count_initial_val</i>	<i>Zeichenfolge</i>	
<i>count_inc_condition</i>	<i>Zeichenfolge</i>	
<i>count_inc_expression</i>	<i>Zeichenfolge</i>	
<i>count_reset_condition</i>	<i>Zeichenfolge</i>	
<i>cond_if_cond</i>	<i>Zeichenfolge</i>	
<i>cond_then_expr</i>	<i>Zeichenfolge</i>	
<i>cond_else_expr</i>	<i>Zeichenfolge</i>	

Tabelle 73. Eigenschaften von "derivenode" (Forts.)

Eigenschaften von <i>derivenode</i>	Datentyp	Eigenschaftsbeschreibung
<code>formula_measure_type</code>	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	Diese Eigenschaft kann zum Definieren der dem abgeleiteten Feld zugeordneten Messung verwendet werden kann. An die Setter-Funktion kann entweder eine Zeichenfolge oder einer der MeasureType-Werte übergeben werden. Die Getter-Funktion gibt immer für MeasureType-Werte zurück.
<code>collection_measure</code>	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Bei Sammlungsfeldern (Listen mit einer Tiefe von 0) definiert diese Eigenschaft den Messtyp, der den zugrunde liegenden Werten zugeordnet ist.
<code>geo_type</code>	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Bei georäumlichen Feldern definiert diese Eigenschaft den Typ des durch dieses Feld dargestellten georäumlichen Objekts. Dies sollte konsistent mit der Listentiefe der Werte sein.
<code>has_coordinate_system</code>	<i>boolesch</i>	Bei georäumlichen Feldern definiert diese Eigenschaft, ob dieses Feld ein Koordinatensystem hat
<code>coordinate_system</code>	<i>Zeichenfolge</i>	Bei georäumlichen Feldern definiert diese Eigenschaft das Koordinatensystem für dieses Feld

Eigenschaften von "ensemblenode"



Der Ensemble-Knoten kombiniert zwei oder mehr Modellnuggets, um genauere Vorhersagen zu erzielen, als aus einem dieser Modelle allein gewonnen werden können.

Beispiel

```
# 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")
```

Tabelle 74. Eigenschaften von "ensemblenode".

Eigenschaften von <i>ensemblenode</i>	Datentyp	Eigenschaftsbeschreibung
<code>ensemble_target_field</code>	<i>Feld</i>	Gibt das Zielfeld für alle im Ensemble verwendeten Modelle an.
<code>filter_individual_model_output</code>	<i>Flag</i>	Gibt an, ob Scoring-Ergebnisse aus einzelnen Modellen unterdrückt werden sollen.

Tabelle 74. Eigenschaften von "ensemblenode" (Forts.).

Eigenschaften von ensemblenode	Datentyp	Eigenschaftsbeschreibung
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_voting_tie_selection	Random HighestConfidence	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.
calculate_standard_error	Flag	Wenn das Zielfeld stetig ist, wird standardmäßig eine Standardfehlerberechnung durchgeführt, um den Unterschied zwischen den gemessenen oder geschätzten Werten und den wahren Werten zu berechnen sowie um zu zeigen, wie hoch die Übereinstimmung dieser Schätzungen war.

Eigenschaften von "fillernode"



Der Füllerknoten ersetzt Feldwerte und ändert den Speichertyp. Sie können auswählen, dass die Werte auf der Grundlage einer CLEM-Bedingung wie beispielsweise @BLANK(@FIELD) ersetzt werden sollen. Alternativ können Sie auswählen, dass alle Leerstellen oder Nullwerte mit einem bestimmten Wert ersetzt werden sollen. Füllerknoten werden häufig zusammen mit einem Typknoten verwendet, um fehlende Werte zu ersetzen.

Beispiel

```
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\"")
```

Tabelle 75. Eigenschaften von "fillernode"

Eigenschaften von fillernode	Datentyp	Eigenschaftsbeschreibung
fields	list	Felder aus dem Dataset, deren Werte untersucht und ersetzt werden.
replace_mode	Always Conditional Blank Null BlankAndNull	Sie können alle Werte, leere Werte, Nullwerte oder Werte ersetzen, die einer bestimmten Bedingung entsprechen.
condition	Zeichenfolge	
replace_with	Zeichenfolge	

Eigenschaften von "filternode"



Der Filterknoten filtert (verwirft) Felder, benennt Felder um und ordnet Felder von einem Quellenknoten einem anderen zu.

Beispiel

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

Verwenden der Eigenschaft default_include. Beachten Sie, dass die Festlegung des Werts der Eigenschaft default_include nicht automatisch zum Ein- oder Ausschluss aller Felder führt; es wird lediglich die Standardvorgehensweise für die ausgewählten Felder festgelegt. Diese Eigenschaft entspricht in ihrer Funktion dem Klicken auf die Schaltfläche **Felder standardmäßig einschließen** im Dialogfeld des Filterknotens. Hier ein Beispiel: Angenommen, Sie führen folgendes Script aus:

```
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)
```

Dies führt dazu, dass der Knoten die Felder *Age* und *Sex* weitergibt und alle anderen verwirft. Angenommen, Sie führen dasselbe Script erneut aus, benennen jedoch zwei andere Felder:

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

Dadurch werden zwei weitere Felder zum Filter hinzugefügt, sodass insgesamt vier Felder weitergegeben werden (*Age*, *Sex*, *BP*, *Na*). Anders ausgedrückt, wenn der Wert von default_include auf False (Falsch) gesetzt wird, bedeutet dies nicht, dass automatisch alle Felder zurückgesetzt werden.

Wenn sie stattdessen nun default_include auf True (Wahr) ändern (entweder mithilfe eines Scripts oder im Dialogfeld des Filterknotens, wird das Verhalten umgekehrt, sodass die vier oben aufgeführten Felder nicht aufgenommen, sondern stattdessen verworfen werden. Wenn Sie sich unsicher sind, sollten Sie ein wenig mit den Steuerelementen im Dialogfeld des Filterknotens herumexperimentieren. Dies kann Ihnen beim Verständnis dieser Interaktion helfen.

Tabelle 76. Eigenschaften von "filternode"

Eigenschaften von filternode	Datentyp	Eigenschaftsbeschreibung
default_include	Flag	Verschlüsselte Eigenschaft zur Angabe, ob das Standardverhalten übergeben wird oder Felder gefiltert werden: Beachten Sie, dass die Festlegung dieser Eigenschaft nicht automatisch zum Ein- oder Ausschluss aller Felder führt; es wird lediglich festgelegt, ob die ausgewählten Felder standardmäßig ein- oder ausgeschlossen werden sollen. Weitere Kommentare finden Sie im unten stehenden Beispiel:
include	Flag	Verschlüsselte Eigenschaft zum Einbeziehen und Entfernen von Feldern.
new_name	Zeichenfolge	

Eigenschaften von "historynode"



Der Verlaufsknoten erstellt neue Felder mit Daten aus Feldern in vorangegangenen Datensätzen. Verlaufsknoten werden am häufigsten für sequenzielle Daten, beispielsweise Zeitreihendaten, verwendet. Vor der Verwendung eines Verlaufsknotens sollten die Daten mithilfe eines Sortierknotens sortiert werden.

Beispiel

```
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")
```

Tabelle 77. Eigenschaften von "historynode"

Eigenschaften von historynode	Datentyp	Eigenschaftsbeschreibung
fields	list	Felder, für die Sie einen Verlauf wollen.
Offset	Zahl	Dient zur Angabe des jüngsten Datensatzes (vor dem aktuellen Datensatz), aus dem Verlaufsfeldwerte extrahiert werden sollen.
span	Zahl	Gibt an, aus wie vielen früheren Datensätzen Werte extrahiert werden sollen.
unavailable	Discard Leave Fill	Für die Behandlung von Datensätzen, die keine Verlaufswerte besitzen, bezieht sich dies normalerweise auf die ersten Datensätze oben im Dataset, für die es keine vorangegangenen Datensätze gibt, die als Verlauf dienen könnten.
fill_with	Zeichenfolge Zahl	Gibt einen Wert oder eine Zeichenfolge an, die für Datensätze verwendet werden soll, wenn kein Verlaufswert verfügbar ist.

Eigenschaften von "partitionnode"



Der Partitionsknoten erstellt ein Partitionsfeld, das Daten in getrennte Subsets für die Trainings-, Test- und Validierungsphase der Modellerstellung aufteilt.

Beispiel

```
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")
```

Tabelle 78. Eigenschaften von "partitionnode"

Eigenschaften von partitionnode	Datentyp	Eigenschaftsbeschreibung
new_name	Zeichenfolge	Der vom Knoten erstellte Name des Partitionsfelds.
create_validation	Flag	Gibt an, ob eine Validierungspartition erstellt werden soll.
training_size	Ganzzahl	Prozentsatz der Datensätze (0-100), die der Trainingspartition zugeordnet werden sollen.
testing_size	Ganzzahl	Prozentsatz der Datensätze (0-100), die der Testpartition zugeordnet werden sollen.
validation_size	Ganzzahl	Prozentsatz der Datensätze (0-100), die der Validierungspartition zugeordnet werden sollen. Wird ignoriert, wenn keine Validierungspartition erstellt wird.
training_label	Zeichenfolge	Beschriftung der Trainingspartition.
testing_label	Zeichenfolge	Beschriftung der Testpartition.
validation_label	Zeichenfolge	Beschriftung der Validierungspartition. Wird ignoriert, wenn keine Validierungspartition erstellt wird.
value_mode	System SystemAndLabel Beschriftung	Gibt die Werte an, die für die einzelnen Partitionen in den Daten verwendet werden. Beispiel: Die Trainingsstichprobe kann durch die Systemganzzahl 1, die Beschriftung Training bzw. eine Kombination aus beiden durch 1_Training repräsentiert werden.
set_random_seed	boolesch	Gibt an, ob ein benutzerdefinierter Startwert für den Zufallsgenerator verwendet werden soll.
random_seed	Ganzzahl	Ein benutzerdefinierter Startwert für den Zufallsgenerator festlegen. Damit dieser Wert verwendet wird, muss set_random_seed auf True (wahr) gesetzt sein.
enable_sql_generation	boolesch	Gibt an, ob SQL-Pushback für die Zuweisung von Datensätzen zu Partitionen verwendet werden soll.

Tabelle 78. Eigenschaften von "partitionnode" (Forts.)

Eigenschaften von partitionnode	Datentyp	Eigenschaftsbeschreibung
unique_field		Gibt das Eingabefeld an, mit dessen Hilfe sichergestellt werden soll, dass Datensätze auf zufällige, aber wiederholbare Weise zu Partitionen zugeordnet werden. Damit dieser Wert verwendet wird, muss enable_sql_generation auf True (wahr) gesetzt sein.

Eigenschaften von "reclassifynode"



Der Umcodierungsknoten transformiert ein Set kategorialer Werte in ein anderes. Die Umcodierung dient zur Reduzierung von Kategorien bzw. Neugruppierung von Daten für die Analyse.

Beispiel

```
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"])
```

Tabelle 79. Eigenschaften von "reclassifynode"

Eigenschaften von reclassifynode	Datentyp	Eigenschaftsbeschreibung
mode	Single Multiple	Single codiert die Kategorien eines einzelnen Felds um. Multiple aktiviert Optionen, die die Transformation von mehreren Feldern gleichzeitig erlauben.
replace_field	Flag	
Feld	Zeichenfolge	Wird nur im Modus "Single" verwendet.
new_name	Zeichenfolge	Wird nur im Modus "Single" verwendet.
fields	[Feld1 Feld2 ... Feldn]	Wird nur im Modus "Multiple" verwendet.
name_extension	Zeichenfolge	Wird nur im Modus "Multiple" verwendet.
add_as	Suffix Prefix	Wird nur im Modus "Multiple" verwendet.
umcodieren	Zeichenfolge	Strukturierte Eigenschaft für Feldwerte.
use_default	Flag	Standardwert verwenden.
default	Zeichenfolge	Standardwert angeben.
pick_list	[Zeichenfolge Zeichenfolge ... Zeichenfolge]	Ermöglicht einem Benutzer den Import einer Liste bekannter neuer Werte, um die Dropdown-Liste in der Tabelle zu füllen.

Eigenschaften von "reordernode"



Der Knoten "Felder ordnen" definiert die natürliche Reihenfolge, die bei der Anzeige der nachfolgenden Felder verwendet wird. Diese Reihenfolge betrifft die Anzeige von Feldern an unterschiedlichen Stellen, beispielsweise in Tabellen, Listen und in der Feldauswahl. Dieser Vorgang dient beispielsweise dazu, um bei der Arbeit mit umfangreichen Datensets die relevanten Felder deutlicher hervorzuheben.

Beispiel

```
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"])
```

Tabelle 80. Eigenschaften von "reordernode"

Eigenschaften von reordernode	Datentyp	Eigenschaftsbeschreibung
mode	Custom Auto	Sie können Werte automatisch sortieren oder eine benutzerdefinierte Reihenfolge angeben.
sort_by	Name Type Speicher	
ascending	Flag	
start_fields	[Feld1 Feld2 ... Feldn]	Nach diesen Feldern werden neue Felder eingefügt.
end_fields	[Feld1 Feld2 ... Feldn]	Vor diesen Feldern werden neue Felder eingefügt.

Eigenschaften von "reprojectnode"



In SPSS Modeler verwenden Elemente wie die räumlichen Funktionen von Expression Builder, der STP-Knoten (Spatio-Temporal Prediction - räumliche temporale Vorhersage) und der Kartenvisualisierungsknoten das projizierte Koordinatensystem. Verwenden Sie den Reprojizierungsknoten, um das Koordinatensystem von importierten Daten zu ändern, die ein geografisches Koordinatensystem verwenden.

Tabelle 81. Eigenschaften von "reprojectnode"

Eigenschaften von reprojectnode	Datentyp	Eigenschaftsbeschreibung
reproject_fields	[Feld1 Feld2 ... Feldn]	Listet alle Felder auf, die reprojiziert werden sollen.
reproject_type	Streamdefault Specify	Wählen Sie aus, wie die Felder reprojiziert werden sollen.
coordinate_system	Zeichenfolge	Name des Koordinatensystems, das auf die Felder angewendet werden soll. Beispiel: set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"

Eigenschaften von "restructurenode"



Der Knoten "Umstrukturieren" konvertiert ein nominales Feld oder ein Flagfeld in eine Gruppe von Feldern, die mit den Werten aus einem weiteren Feld ausgefüllt werden können. Beispiel: Aus einem Feld mit dem Namen *Zahlungsart*, mit den Werten *Kreditkarte*, *Bar* und *EC-Karte* werden drei neue Felder erstellt (*Kreditkarte*, *Bar*, *EC-Karte*), die jeweils den Wert der jeweiligen Zahlung enthalten.

Beispiel

```
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"])
```

Tabelle 82. Eigenschaften von "restructurenode"

Eigenschaften von restructurenode	Datentyp	Eigenschaftsbeschreibung
fields_from	[Kategorie Kategorie Kategorie] all	
include_field_name	Flag	Gibt an, ob der Feldname im umstrukturierten Feldnamen verwendet werden soll.
value_mode	OtherFields Flags	Legt den Modus für die Angabe der Werte für die umstrukturierten Felder fest. Bei OtherFields müssen Sie angeben, welche Felder verwendet werden sollen (siehe unten). Bei Flags sind die Werte numerische Flags.
value_fields	list	Erforderlich, wenn value_mode auf OtherFields gesetzt ist. Gibt an, welche Felder als Wertfelder verwendet werden sollen.

Eigenschaften von "rfmanalysisnode"



Mit dem Knoten "RFM-Analyse" (Recency-, Frequency-, Monetary-Analyse) können Sie quantitativ ermitteln, welche Kunden wahrscheinlich die besten sind, indem Sie untersuchen, wann sie zuletzt etwas von Ihnen erworben haben (Recency (Aktualität)), wie häufig sie eingekauft haben (Frequency (Häufigkeit)) und wie viel sie für alle Transaktionen zusammengekommen ausgegeben haben (Monetary (Geldwert)).

Beispiel

```
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])
```

Tabelle 83. Eigenschaften von "rfanalysisnode"

Eigenschaften von rfanalysisnode	Datentyp	Eigenschaftsbeschreibung
recency	Feld	Gibt das Feld für "Recency" (Aktualität) an. Dabei kann es sich um ein Datum, eine Zeitmarke oder eine einfache Zahl handeln.
frequency	Feld	Gibt das Feld für "Frequency" (Häufigkeit) an.
monetary	Feld	Gibt das Feld für "Monetary" (Geldwert) an.
recency_bins	Ganzzahl	Dient zur Angabe der Anzahl der zu generierenden Aktualitätsklassen.
recency_weight	Zahl	Dient zur Angabe der Gewichtung für die Aktualitätsdaten. Der Standardwert ist 100.
frequency_bins	Ganzzahl	Dient zur Angabe der Anzahl der zu generierenden Häufigkeitsklassen.
frequency_weight	Zahl	Dient zur Angabe der Gewichtung für die Häufigkeitsdaten. Der Standardwert ist 10.
monetary_bins	Ganzzahl	Dient zur Angabe der Anzahl der zu generierenden Klassen für den Geldwert.
monetary_weight	Zahl	Dient zur Angabe der Gewichtung für die Geldwertdaten. Der Standardwert lautet 1.
tied_values_method	Next Current	Gibt an, in welche Klasse Daten mit gebundenen Werten (Werten mit Gleichstand) eingeordnet werden sollen.
recalculate_bins	Always IfNecessary	
add_outliers	Flag	Nur verfügbar, wenn recalculate_bins auf IfNecessary gesetzt ist. Wenn diese Einstellung festgelegt wurde, werden Datensätze, die unterhalb der untersten Klasse liegen, zur untersten Klasse hinzugefügt und Datensätze oberhalb der höchsten Klasse werden in die höchste Klasse aufgenommen.
binned_field	Recency Frequency Monetary	
recency_thresholds	Wert Wert	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist. Dient zur Angabe der oberen und unteren Schwellenwerte für die Aktualitätsklassen. Der obere Schwellenwert einer Klasse wird als unterer Schwellenwert der nächsten Klasse verwendet. So werden beispielsweise mit [10 30 60] zwei Klassen definiert, wobei für die erste Klasse die Schwellenwerte 10 und 30 gelten und für die zweite Klasse die Schwellenwerte 30 und 60.
frequency_thresholds	Wert Wert	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.
monetary_thresholds	Wert Wert	Nur verfügbar, wenn recalculate_bins auf Always gesetzt ist.

Eigenschaften von "settoflagnode"



Der Dichotomknoten leitet mehrere Flagfelder auf der Grundlage der kategorialen Werte ab, die für ein oder mehrere nominale Felder definiert sind.

Beispiel

```
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"])
```

Tabelle 84. Eigenschaften von "settoflagnode"

Eigenschaften von settoflagnode	Datentyp	Eigenschaftsbeschreibung
fields_from	[Kategorie Kategorie Kategorie] all	
true_value	Zeichenfolge	Gibt den Wert "Wahr" an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert lautet T.
false_value	Zeichenfolge	Gibt den Falsch-Wert an, den der Knoten zum Festlegen eines Flags verwendet. Der Standardwert lautet F.
use_extension	Flag	Verwenden Sie eine Erweiterung als Suffix oder Präfix für das neue Flagfeld.
extension	Zeichenfolge	
add_as	Suffix Prefix	Gibt an, ob die Erweiterung als Suffix oder als Präfix hinzugefügt wird.
aggregate	Flag	Fasst Datensätze anhand von Schlüsselfeldern zu Gruppen zusammen. Alle in einer Gruppe vorhandenen Flagfelder werden aktiviert, wenn einer der Datensätze auf "true" gesetzt wird.
Schlüssel	list	Schlüsselfelder.

Eigenschaften von "statistictransformnode"



Der Statistics-Transformationsknoten führt eine Auswahl von IBM SPSS Statistics-Syntaxbefehlen für Datenquellen in IBM SPSS Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften von "statistictransformnode"“ auf Seite 307.

Eigenschaften von "timeintervalsnode"



Der Zeitintervallknoten gibt Intervalle an und erstellt (bei Bedarf) Beschriftungen für die Modellierung von Zeitreihendaten. Wenn die Werte nicht gleichmäßig verteilt sind, kann der Knoten nach Bedarf Werte auffüllen oder aggregieren, um ein gleichmäßiges Intervall zwischen den Datensätzen zu erzeugen.

Beispiel

```
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")
```

Tabelle 85. Eigenschaften von "timeintervalsnode".

Eigenschaften von timeintervalsnode	Datentyp	Eigenschaftsbeschreibung
interval_type	None Periods CyclicPeriods Years Quarters Months DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Label Create	Gibt an, ob Sie die Datensätze nacheinander beschriftet werden sollen oder ob die Zeitreihe auf der Grundlage eines angegebenen Datums-, Zeitmarken- oder Zeitfelds erstellt werden soll.
field	<i>Feld</i>	Gibt beim Erstellen der Serie aus den Daten das Feld an, das das Datum bzw. die Uhrzeit für jeden Datensatz anzeigt.
period_start	<i>Ganzzahl</i>	Gibt das Startintervall für Perioden bzw. zyklische Perioden an.
cycle_start	<i>Ganzzahl</i>	Startzyklus für zyklische Perioden.
year_start	<i>Ganzzahl</i>	Bei entsprechenden Intervalltypen das Jahr, in das das erste Intervall fällt.
quarter_start	<i>Ganzzahl</i>	Bei entsprechenden Intervalltypen das Quartal, in das das erste Intervall fällt.

Tabelle 85. Eigenschaften von "timeintervalnode" (Forts.).

Eigenschaften von timeintervalnode	Datentyp	Eigenschaftsbeschreibung
month_start	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	
day_start	Ganzzahl	
hour_start	Ganzzahl	
minute_start	Ganzzahl	
second_start	Ganzzahl	
periods_per_cycle	Ganzzahl	Bei zyklischen Perioden, die Anzahl innerhalb jedes Zyklus.
fiscal_year_begins	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	Gibt bei vierteljährlichen Intervallen den Monat an, in dem das Geschäftsjahr beginnt.
week_begins_on	Sunday Monday Tuesday Wednesday Donnerstag Friday Saturday Sunday	Gibt bei periodischen Intervallen (Tage pro Woche, Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) den Tag an, an dem die Woche beginnt.
day_begins_hour	Ganzzahl	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Stunde an, zu der der Tag beginnt. Kann in Verbindung mit day_begins_minute und day_begins_second verwendet werden, um einen genauen Zeitpunkt anzugeben wie beispielsweise 8:05:01. Siehe unten stehendes Anwendungsbeispiel.
day_begins_minute	Ganzzahl	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Minute an, in der der Tag beginnt (z. B. die 5 in 8:05).
day_begins_second	Ganzzahl	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Sekunde an, in der der Tag beginnt (z. B. die 17 in 8:05:17).

Table 85. Properties of "timeintervalnode" (Continued).

Eigenschaften von timeintervalnode	Datentyp	Eigenschaftsbeschreibung
days_per_week	Ganzzahl	Gibt bei periodischen Intervallen (Tage pro Woche, Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Anzahl der Tage pro Woche an.
hours_per_day	Ganzzahl	Gibt bei periodischen Intervallen (Stunden pro Tag, Minuten pro Tag und Sekunden pro Tag) die Anzahl der Stunden pro Tag an.
interval_increment	1 2 3 4 5 6 10 15 20 30	Gibt bei Minuten pro Tag und Sekunden pro Tag die Anzahl der Minuten bzw. Sekunden an, um die der Wert für jeden Datensatz erhöht werden soll.
field_name_extension	Zeichenfolge	
field_name_extension_as_prefix	Flag	
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	

Tabella 85. Eigenschaften von "timeintervalnode" (Forts.).

Eigenschaften von timeintervalnode	Datentyp	Eigenschaftsbeschreibung
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	Gibt die Aggregationsmethode für ein Feld an.
pad	Blank MeanOfRecentPoints True False	Gibt die Auffüllmethode für ein Feld an.
agg_mode	All Specify	Gibt an, ob alle Felder mit Standardfunktionen nach Bedarf aggregiert bzw. aufgefüllt werden sollen oder ob die zu verwendenden Felder und Funktionen angegeben werden sollen.
agg_range_default	Mean Sum Mode Min Max	Gibt die beim Aggregieren von stetigen Feldern zu verwendende Standardfunktion an.
agg_set_default	Mode First Last	Gibt die beim Aggregieren von nominalen Feldern zu verwendende Standardfunktion an.
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	Blank MeanOfRecentPoints	Gibt die beim Auffüllen von stetigen Feldern zu verwendende Standardfunktion an.
pad_set_default	Blank MostRecentValue	
pad_flag_default	Blank True False	
max_records_to_create	Ganzzahl	Gibt die maximale Anzahl der beim Auffüllen der Reihe zu erstellenden Datensätze an.
estimation_from_beginning	Flag	

Tabelle 85. Eigenschaften von "timeintervalsnode" (Forts.).

Eigenschaften von timeintervalsnode	Datentyp	Eigenschaftsbeschreibung
estimation_to_end	Flag	
estimation_start_offset	Ganzzahl	
estimation_num_holdouts	Ganzzahl	
create_future_records	Flag	
num_future_records	Ganzzahl	
create_future_field	Flag	
future_field_name	Zeichenfolge	

Eigenschaften von "transposenode"



Der Transponierknoten vertauscht die Daten in Zeilen und Spalten, sodass aus Datensätzen Felder und aus Feldern Datensätze werden.

Beispiel

```
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")
```

Tabelle 86. Eigenschaften von "transposenode"

Eigenschaften von transposenode	Datentyp	Eigenschaftsbeschreibung
transposed_names	Prefix Read	Neue Felder können automatisch auf der Grundlage eines angegebenen Präfixes generiert oder aus einem bestehenden Feld in den Daten eingelesen werden.
prefix	Zeichenfolge	
num_new_fields	Ganzzahl	Bei Verwendung eines Präfixes wird die maximale Anzahl der zu erstellenden Felder angegeben.
read_from_field	Feld	Felder, aus denen Namen gelesen werden. Es muss sich um ein instanziiertes Feld handeln. Andernfalls tritt bei der Ausführung des Knotens ein Fehler auf.
max_num_fields	Ganzzahl	Beim Einlesen von Namen aus einem Feld wird eine Obergrenze angegeben, um das Erstellen einer übermäßig großen Anzahl von Feldern zu verhindern.
transpose_type	Numeric Zeichenfolge Custom	Standardmäßig werden nur stetige Felder (numerischer Bereich) transponiert, Sie können jedoch stattdessen auch ein benutzerdefiniertes Subset numerischer Felder auswählen oder alle Zeichenfolgenfelder transponieren.
transpose_fields	list	Gibt die bei Verwendung der Option Custom (Angepasst) zu transponierenden Felder an.
id_field_name	Feld	

Eigenschaften von "typenode"



Der Typknoten gibt Feldmetadaten und Eigenschaften an. Sie können beispielsweise ein Messniveau (stetig, nominal, ordinal oder Flag) für die einzelnen Felder angeben, Optionen für den Umgang mit fehlenden Werten und systemdefinierten Nullwerten festlegen, die Rolle eines Felds zu Modellierungszwecken festlegen, Feld- und Wertbeschriftungen angeben oder die Werte für ein Feld angeben.

Beispiel

```
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"]])
```

Beachten Sie: In einigen Fällen müssen Sie den Knoten "type" möglicherweise vollständig instanziiieren, damit die anderen Knoten ordnungsgemäß arbeiten, beispielsweise die Eigenschaft `fields from` (Felder aus) des Dichotomknotens. Sie können einfach einen Tabellenknoten anschließen und ausführen, um die Felder zu instanziiieren:

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

Tabelle 87. Eigenschaften von "typenode".

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
direction	Input Target Both None Partition Split Frequency RecordID	Verschlüsselte Eigenschaft für Feldrollen. Anmerkung: Die Werte In und Out werden nicht mehr verwendet. In zukünftigen Versionen fällt möglicherweise die Unterstützung dafür weg.

Tabelle 87. Eigenschaften von "typenode" (Forts.).

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
type	Range Flag Set Typeless Discrete OrderedSet Default	Messniveau des Felds (bisher als "type" bezeichnet). Wenn type auf Default gesetzt wird, werden alle Parametereinstellungen vom Typ values gelöscht, und wenn value_mode den Wert Specify besitzt, wird er auf Read gesetzt. Wird value_mode auf Pass oder Read gesetzt, hat das Einstellen von type keinerlei Auswirkung auf value_mode. Anmerkung: Die intern verwendeten Datentypen unterscheiden sich von den im Typknoten sichtbaren Datentypen. Die Korrespondenz lautet wie folgt: Range -> Continuous Set -> Nominal OrderedSet -> Ordinal Discrete- > Categorical
storage	Unknown String Integer Real Zeit Datum Zeitmarke	Schreibgeschützte verschlüsselte Eigenschaft für Feldspeichertyp.
check	None Nullify Coerce Discard Warn Abort	Verschlüsselte Eigenschaft für das Überprüfen von Feldtyp und Bereich.
Werte	[Wert Wert]	Bei einem stetigen Feld ist der erste Wert das Minimum und der letzte das Maximum. Geben Sie für nominale Felder alle Werte an. Bei Flagfeldern steht der erste Wert für <i>falsch</i> und der letzte für <i>wahr</i> . Bei automatischer Festlegung dieser Eigenschaft wird die Eigenschaft value_mode auf Specify festgelegt.
value_mode	Read Pass Read+ Current Angeben	Bestimmt, wie Werte festgelegt werden. Beachten Sie, dass Sie diese Eigenschaft nicht direkt auf Angeben festlegen können. Um bestimmte Werte zu verwenden, legen Sie die Eigenschaft values fest.
extend_values	Flag	Gilt, wenn value_mode auf Read gesetzt ist. Setzen Sie den Wert auf T, um neu gelesene Werte zu bereits für das Feld vorhandenen Werten hinzuzufügen. Setzen Sie den Wert auf F, um vorhandene Werte zu verwerfen und sie durch neu gelesene Werte zu ersetzen.
enable_missing	Flag	Bei Festlegung auf T wird die Verfolgung von fehlenden Werten für das Feld aktiviert.
missing_values	[Wert Wert ...]	Gibt Datenwerte an, die fehlende Daten kennzeichnen.
range_missing	Flag	Gibt an, ob ein Bereich fehlender Werte (leer) für ein Feld definiert ist.

Tabelle 87. Eigenschaften von "typenode" (Forts.).

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
missing_lower	Zeichenfolge	Wenn range_missing wahr ist, gibt diese Eigenschaft die Untergrenze des Bereichs fehlender Werte an.
missing_upper	Zeichenfolge	Wenn range_missing wahr ist, gibt diese Eigenschaft die Obergrenze des Bereichs fehlender Werte an.
null_missing	Flag	Bei Festlegung auf T werden <i>Nullen</i> (undefinierte Werte, die in der Software als \$null\$ angezeigt werden) als fehlende Werte betrachtet.
whitespace_missing	Flag	Bei Festlegung auf T werden Werte, die nur leere Bereiche enthalten (Leerzeichen, Tabulatoren und Zeilenumbrüche), als fehlende Werte betrachtet.
description	Zeichenfolge	Gibt die Beschreibung für ein Feld an.
value_labels	[[Wert Beschriftungszeichenfolge] {Wert Beschriftungszeichenfolge} ...]	Gibt Beschriftungen für Wertpaare an.
display_places	Ganzzahl	Legt die Dezimalstellen für das Feld bei der Anzeige fest (gilt nur für Felder mit dem Speichertyp REAL). Der Wert -1 verwendet den Streamstandard.
export_places	Ganzzahl	Legt die Dezimalstellen für das Feld beim Exportieren fest (gilt nur für Felder mit dem Speichertyp REAL). Der Wert -1 verwendet den Streamstandard.
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REAL).

Tabelle 87. Eigenschaften von "typenode" (Forts.).

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Legt das Datumsformat für das Feld fest (gilt nur für Felder mit dem Speichertyp DATE oder TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Legt das Zeitformat für das Feld fest (gilt nur für Felder mit dem Speichertyp TIME oder TIMESTAMP).
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Legt das Zahlenanzeigeformat für das Feld fest.
standard_places	<i>Ganzzahl</i>	Legt die Dezimalstellen für das Feld für die Anzeige im Standardformat fest. Der Wert -1 verwendet den Streamstandard. Der vorhandene Slot display_places ändert dies zwar auch, wird aber nicht mehr verwendet.
scientific_places	<i>Ganzzahl</i>	Legt die Dezimalstellen für das Feld für die Anzeige im wissenschaftlichen Format fest. Der Wert -1 verwendet den Streamstandard.
currency_places	<i>Ganzzahl</i>	Legt die Dezimalstellen für das Feld für die Anzeige im Währungsformat fest. Der Wert -1 verwendet den Streamstandard.

Tabelle 87. Eigenschaften von "typenode" (Forts.).

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Legt das Symbol für die Zifferngruppierung für das Feld fest.
column_width	<i>Ganzzahl</i>	Legt die Spaltenbreite für das Feld fest. Mit dem Wert -1 wird die Spaltenbreite auf Auto (Automatisch) gesetzt.
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest.
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	Diese verschlüsselte Eigenschaft ähnelt type dahingehend, dass sie zum Definieren der dem Feld zugeordneten Messung verwendet werden kann. Das Python-Scripting unterscheidet sich dadurch, dass der Setter-Funktion auch einer der MeasureType-Werte übergeben werden kann, während die Getter-Funktion immer für MeasureType-Werte zurückgibt.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Bei Sammlungsfeldern (Listen mit einer Tiefe von 0) definiert diese verschlüsselte Eigenschaft den Messtyp, der den zugrunde liegenden Werten zugeordnet ist.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Bei georäumlichen Feldern definiert diese verschlüsselte Eigenschaft den Typ des durch dieses Feld dargestellten georäumlichen Objekts. Dies sollte konsistent mit der Listentiefe der Werte sein.
has_coordinate_system	<i>boolesch</i>	Bei georäumlichen Feldern definiert diese Eigenschaft, ob dieses Feld ein Koordinatensystem hat
coordinate_system	<i>Zeichenfolge</i>	Bei georäumlichen Feldern definiert diese verschlüsselte Eigenschaft das Koordinatensystem für dieses Feld.
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	Diese verschlüsselte Eigenschaft ähnelt custom_storage dahingehend, dass sie zum Definieren der Speicherüberschreibung für das Feld verwendet werden kann. Das Python-Scripting unterscheidet sich dadurch, dass der Setter-Funktion auch einer der StorageType-Werte übergeben werden kann, während die Getter-Funktion immer für StorageType-Werte zurückgibt.

Tabelle 87. Eigenschaften von "typenode" (Forts.).

Eigenschaften von typenode	Datentyp	Eigenschaftsbeschreibung
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Bei Listenfeldern gibt diese verschlüsselte Eigenschaft den Speichertyp der zugrunde liegenden Werte an.
custom_list_depth	<i>Ganzzahl</i>	Bei Listenfeldern gibt diese verschlüsselte Eigenschaft die Tiefe des Felds an.

Kapitel 12. Eigenschaften von Diagrammknoten

Allgemeine Eigenschaften von Diagrammknoten

In diesem Abschnitt werden die für Diagrammknoten verfügbaren Eigenschaften, einschließlich allgemeiner Eigenschaften sowie knotenspezifischer Eigenschaften, beschrieben.

Table 88. Allgemeine Eigenschaften von Diagrammknoten

Allgemeine Eigenschaften von Diagrammknoten	Datentyp	Eigenschaftsbeschreibung
title	Zeichenfolge	Gibt den Titel an. Beispiel: "Dies ist ein Titel."
caption	Zeichenfolge	Gibt die Titelzeile an. Beispiel: "Dies ist eine Titelzeile."
output_mode	Screen File	Gibt an, ob die Ausgabe des Diagrammknotens angezeigt oder in eine Datei geschrieben werden soll.
output_format	BMP JPEG PNG HTML output (.cou)	Gibt den Ausgabotyp an. Der zulässige Ausgabotyp ist für jeden Knoten unterschiedlich.
full_filename	Zeichenfolge	Gibt den Zielpfad und den Dateinamen für die vom Diagrammknoten generierten Ausgabe an.
use_graph_size	Flag	Gibt an, ob die Größe des Diagramms mithilfe der unten angegebenen Eigenschaften für Breite und Höhe explizit festgelegt wird. Dies betrifft nur Diagramme, die auf dem Bildschirm ausgegeben werden. Nicht für den Verteilungsknoten verfügbar.
graph_width	Zahl	Wenn use_graph_size den Wert True aufweist, wird hier die Diagrammhöhe in Pixeln festgelegt.
graph_height	Zahl	Wenn use_graph_size den Wert True aufweist, wird hier die Diagrammhöhe in Pixeln festgelegt.

Inaktivieren optionaler Felder

Optionale Felder, wie Überlagerungsfelder für Plots, können inaktiviert werden, indem der Eigenschaftswert auf " " (leere Zeichenfolge) gesetzt wird, wie im folgenden Beispiel gezeigt:

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

Angeben von Farben

Die Farben für Titel, Titelzeilen, Hintergründe und Beschriftungen können mit hexadezimalen Zeichenfolgen, die mit einem Hashzeichen (#) beginnen, festgelegt werden. Beispiel: Um den Diagrammhintergrund auf Blau festzulegen, verwenden Sie folgende Anweisung:

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

Die ersten beiden Stellen, 87, geben den roten Inhalt an, die mittleren Stellen, CE, legen den grünen Inhalt fest und die beiden letzten Stellen, EB, definieren den blauen Inhalt. Jede Stelle kann einen Wert im Bereich von 0-9 oder A-F annehmen. Zusammen können diese Werte eine Farbe des RGB-Farbraums (Rot, Grün, Blau) definieren.

Anmerkung: Beim Angeben von Farben in Rot, Grün und Blau können Sie den richtigen Farbcode mit der Felddauswahl in der Benutzerschnittstelle festlegen. Bewegen Sie die Maus über die Farbe, um eine QuickInfo mit den gewünschten Informationen anzuzeigen.

Eigenschaften von "collectionnode"



Der Sammlungsknoten zeigt die Verteilung der Werte für ein numerisches Feld im Verhältnis zu den Werten eines anderen an. (Er erstellt histogrammähnliche Diagramme.) Er eignet sich besonders für die Darstellung einer Variablen oder eines Felds, dessen Werte sich mit der Zeit verändern. Mithilfe eines 3-D-Diagramms können Sie außerdem eine symbolische Achse anlegen, auf der die Verteilungen nach Kategorie aufgetragen sind.

Beispiel

```
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)
```

Tabelle 89. Eigenschaften von "collectionnode"

Eigenschaften von collectionnode	Datentyp	Eigenschaftsbeschreibung
over_field	Feld	
over_label_auto	Flag	
over_label	Zeichenfolge	
collect_field	Feld	
collect_label_auto	Flag	
collect_label	Zeichenfolge	
three_D	Flag	
by_field	Feld	
by_label_auto	Flag	
by_label	Zeichenfolge	
operation	Sum Mean Min Max Std. Abw.	
color_field	Zeichenfolge	
panel_field	Zeichenfolge	

Tabelle 89. Eigenschaften von "collectionnode" (Forts.)

Eigenschaften von collectionnode	Datentyp	Eigenschaftsbeschreibung
animation_field	Zeichenfolge	
range_mode	Automatic UserDefined	
range_min	Zahl	
range_max	Zahl	
bins	ByNumber ByWidth	
num_bins	Zahl	
bin_width	Zahl	
use_grid	Flag	
graph_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.

Eigenschaften von "distributionnode"



Der Verteilungsknoten zeigt das Auftreten symbolischer (kategorialer) Werte wie beispielsweise Hypothekenart oder Geschlecht. Verteilungsknoten eignen sich insbesondere zum Aufzeigen von Unausgewogenheiten in den Daten, die mithilfe eines Balancierungsknotens vor dem Erstellen eines Modells ausgeglichen werden können.

Beispiel

```
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)
```

Tabelle 90. Eigenschaften von "distributionnode"

Eigenschaften von distributionnode	Datentyp	Eigenschaftsbeschreibung
plot	SelectedFields Flags	
x_field	Feld	
color_field	Feld	Überlagerungsfeld
normalize	Flag	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	Flag	

Eigenschaften von "evaluationnode"



Der Evaluierungsknoten erleichtert die Evaluation und den Vergleich von Vorhersagemodellen. Das Evaluierungsdiagramm zeigt, wie gut Modelle bestimmte Ergebnisse vorhersagen. Die Datensätze werden auf der Grundlage des vorhergesagten Werts und des Konfidenzwerts für die Vorhersage sortiert. Die Datensätze werden in gleich große Gruppen (**Quantile**) aufgeteilt. Anschließend wird der Wert des Geschäftskriteriums für jedes Quantil geplottet, vom höchsten Wert bis zum niedrigsten Wert. Mehrere Modelle werden als separate Linien im Plot dargestellt.

Beispiel

```
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")
```

Tabelle 91. Eigenschaften von "evaluationnode".

Eigenschaften von evaluationnode	Datentyp	Eigenschaftsbeschreibung
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	Flag	
field_detection_method	Metadata Name	
use_fixed_cost	Flag	
cost_value	Zahl	
cost_field	Zeichenfolge	
use_fixed_revenue	Flag	
revenue_value	Zahl	
revenue_field	Zeichenfolge	
use_fixed_weight	Flag	
weight_value	Zahl	
weight_field	Feld	

Tabelle 91. Eigenschaften von "evaluationnode" (Forts.).

Eigenschaften von evaluationnode	Datentyp	Eigenschaftsbeschreibung
n_tile	Quartiles Quintiles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	Flag	
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	
export_data	Flag	
data_filename	Zeichenfolge	
delimiter	Zeichenfolge	
new_line	Flag	
inc_field_names	Flag	
inc_best_line	Flag	
inc_business_rule	Flag	
business_rule_condition	Zeichenfolge	
plot_score_fields	Flag	
score_fields	[Feld1 ... FeldN]	
target_field	Feld	
use_hit_condition	Flag	
hit_condition	Zeichenfolge	
use_score_expression	Flag	
score_expression	Zeichenfolge	
caption_auto	Flag	

Eigenschaften von "graphboardnode"



Der Diagrammtafelknoten bietet viele verschiedene Diagrammtypen in einem einzigen Knoten. Bei Verwendung dieses Knotens können Sie die Datenfelder auswählen, die Sie untersuchen möchten, und anschließend eines der für die ausgewählten Daten verfügbaren Diagramme auswählen. Der Knoten filtert automatisch alle Diagrammtypen heraus, die nicht für die Feldauswahl geeignet sind.

Anmerkung: Wenn Sie eine Eigenschaft festlegen, die für den Diagrammtyp ungültig ist (z. B. ein `y_field` für ein Histogramm), wird diese Eigenschaft ignoriert.

Anmerkung: In der Benutzerschnittstelle enthält die Registerkarte **Detailliert** vieler verschiedener Diagrammtypen ein Feld **Übersicht**, das nicht mit Scripting festgelegt werden kann. Die folgenden Diagrammtypen unterstützen nicht das Feld **Übersicht**: 3DArea, 3DBar, 3DHistogram, 3DPie, Area, Bar, BarCounts, BarMap, Histogram, Line, LineChartMap, Pie, PieCounts, Ribbon und Scatterplot.

Beispiel

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

Tabelle 92. *graphboardnode properties*

Eigenschaften von graphboard	Datentyp	Eigenschaftsbeschreibung
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPL0M Surface	Identifiziert den Diagrammtyp.
x_field	<i>Feld</i>	Legt eine benutzerdefinierte Beschriftung für die <i>x</i> -Achse fest. Nur für Beschriftungen verfügbar.
y_field	<i>Feld</i>	Legt eine benutzerdefinierte Beschriftung für die <i>y</i> -Achse fest. Nur für Beschriftungen verfügbar.
z_field	<i>Feld</i>	In einigen 3-D-Diagrammen verwendet.
color_field	<i>Feld</i>	In Heat-Maps verwendet.

Tabelle 92. graphboardnode properties (Forts.)

Eigenschaften von graphboard	Datentyp	Eigenschaftsbeschreibung
size_field	Feld	In Blasendiagrammen verwendet.
categories_field	Feld	
values_field	Feld	
rows_field	Feld	
columns_field	Feld	
fields	Feld	
start_longitude_field	Feld	Bei Pfeilen auf einer Referenzkarte verwendet.
end_longitude_field	Feld	
start_latitude_field	Feld	
end_latitude_field	Feld	
data_key_field	Feld	In verschiedenen Karten verwendet.
panelrow_field	Zeichenfolge	
panelcol_field	Zeichenfolge	
animation_field	Zeichenfolge	
longitude_field	Feld	Bei Koordinaten in Karten verwendet.
latitude_field	Feld	
map_color_field	Feld	

Eigenschaften von "histogramnode"



Der Histogrammknoten zeigt das Auftreten bestimmter Werte in numerischen Feldern. Damit werden häufig die Daten vor der weiteren Bearbeitung und der Modellerstellung untersucht. Ähnlich wie der Verteilungsknoten kann der Histogrammknoten oft Unausgewogenheiten in den Daten aufdecken.

Beispiel

```
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)
```

Tabelle 93. Eigenschaften von "histogramnode"

Eigenschaften von histogramnode	Datentyp	Eigenschaftsbeschreibung
Feld	Feld	
color_field	Feld	

Tabelle 93. Eigenschaften von "histogramnode" (Forts.)

Eigenschaften von histogramnode	Datentyp	Eigenschaftsbeschreibung
panel_field	Feld	
animation_field	Feld	
range_mode	Automatic UserDefined	
range_min	Zahl	
range_max	Zahl	
bins	ByNumber ByWidth	
num_bins	Zahl	
bin_width	Zahl	
normalize	Flag	
separate_bands	Flag	
x_label_auto	Flag	
x_label	Zeichenfolge	
y_label_auto	Flag	
y_label	Zeichenfolge	
use_grid	Flag	
graph_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
normal_curve	Flag	Gibt an, ob die Normalverteilungskurve in der Ausgabe angezeigt werden soll.

Eigenschaften von "multiplotnode"



Ein Multiplot erstellt ein Plot, bei dem mehrere Y-Felder über einem einzelnen X-Feld dargestellt werden. Die Y-Felder werden als farbige Linien geplottet, die jeweils einem Plotknoten mit dem Stil **Linie** und dem X-Modus **Sortieren** entsprechen. Multiplots sind hilfreich, wenn die Fluktuation mehrerer Variablen im Laufe der Zeit untersucht werden soll.

Beispiel

```
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")
```

Tabelle 94. Eigenschaften von "multiplotnode"

Eigenschaften von multiplotnode	Datentyp	Eigenschaftsbeschreibung
x_field	Feld	
y_fields	list	
panel_field	Feld	
animation_field	Feld	
normalize	Flag	
use_overlay_expr	Flag	
overlay_expression	Zeichenfolge	
records_limit	Zahl	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	Flag	
x_label	Zeichenfolge	
y_label_auto	Flag	
y_label	Zeichenfolge	
use_grid	Flag	
graph_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.

Eigenschaften von "plotnode"



Der Plotknoten zeigt die Beziehung zwischen numerischen Feldern an. Sie können einen Plot mithilfe von Punkten (Streudiagramm) oder mit Linien erstellen.

Beispiel

```
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")
```


Tabelle 95. Eigenschaften von "plotnode".

Eigenschaften von plotnode	Datentyp	Eigenschaftsbeschreibung
x_field	Feld	Legt eine benutzerdefinierte Beschriftung für die x-Achse fest. Nur für Beschriftungen verfügbar.
y_field	Feld	Legt eine benutzerdefinierte Beschriftung für die y-Achse fest. Nur für Beschriftungen verfügbar.
three_D	Flag	Legt eine benutzerdefinierte Beschriftung für die y-Achse fest. Nur für Beschriftungen in 3-D-Diagrammen verfügbar.
z_field	Feld	
color_field	Feld	Überlagerungsfeld
size_field	Feld	
shape_field	Feld	
panel_field	Feld	Gibt ein nominales oder Flagfeld an, mit dem je ein separates Diagramm für jede Kategorie angelegt werden soll. Die Diagramme werden gemeinsam in einem Ausgabefenster dargestellt.
animation_field	Feld	Gibt ein nominales oder Flagfeld an, mit dem die Kategorien für die Datenwerte in Form einer Reihe von Diagrammen dargestellt werden, die nacheinander als Animation angezeigt werden.
transp_field	Feld	Gibt ein Feld an, mit dem die Kategorien für die Datenwerte in Form von verschiedenen Transparenzstufen für die einzelnen Kategorien dargestellt werden. Für Liniendiagramme nicht verfügbar.
overlay_type	None Smoother Function	Gibt an, ob eine Überlagerungsfunktion oder ein LOESS-Smoother angezeigt werden soll.
overlay_expression	Zeichenfolge	Gibt den Ausdruck an, der verwendet werden soll, wenn overlay_type auf Function gesetzt ist.
style	Point Line	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	

Tabelle 95. Eigenschaften von "plotnode" (Forts.).

Eigenschaften von plotnode	Datentyp	Eigenschaftsbeschreibung
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	Zahl	
x_range_max	Zahl	
y_range_mode	Automatic UserDefined	
y_range_min	Zahl	
y_range_max	Zahl	
z_range_mode	Automatic UserDefined	
z_range_min	Zahl	
z_range_max	Zahl	
Streuen	Flag	
records_limit	Zahl	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	Flag	
x_label	Zeichenfolge	
y_label_auto	Flag	
y_label	Zeichenfolge	
z_label_auto	Flag	
z_label	Zeichenfolge	
use_grid	Flag	
graph_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
page_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
use_overlay_expr	Flag	Wird zugunsten von overlay_type nicht mehr verwendet.

Eigenschaften von "timeplotnode"



Der Zeitdiagrammknoten zeigt ein oder mehrere Sets mit Zeitreihendaten an. Normalerweise wird zuerst mithilfe eines Zeitintervallknotens ein *TimeLabel*-Feld erstellt, das dann zur Beschriftung der *x*-Achse verwendet wird.

Beispiel

```
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)

```

Tabelle 96. Eigenschaften von "timeplotnode".

Eigenschaften von timeplotnode	Datentyp	Eigenschaftsbeschreibung
plot_series	Series Models	
use_custom_x_field	Flag	
x_field	Feld	
y_fields	list	
panel	Flag	
normalize	Flag	
line	Flag	
points	Flag	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	Flag	Sie können nur dann Glättungselemente zum Plot hinzufügen, wenn Sie panel auf True setzen.
use_records_limit	Flag	
records_limit	Ganzzahl	
symbol_size	Zahl	Gibt eine Symbolgröße an.
panel_layout	Horizontal Vertical	

Eigenschaften von "webnode"



Der Netzdiagrammknoten zeigt die Stärke der Beziehung zwischen den Werten aus mindestens zwei symbolischen (kategorialen) Feldern. Im Diagramm wird die Verbindungsstärke durch unterschiedlich breite Linien angezeigt. Mit Netzdiagrammknoten können Sie beispielsweise die Beziehung zwischen dem Kauf einer Gruppe von Artikeln auf einer e-Commerce-Website untersuchen.

Beispiel

```
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")
```

Tabelle 97. Eigenschaften von "webnode"

Eigenschaften von webnode	Datentyp	Eigenschaftsbeschreibung
use_directed_web	Flag	
fields	list	
to_field	Feld	
from_fields	list	
true_flags_only	Flag	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	Flag	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	Zahl	
links_above	Zahl	
discard_links_min	Flag	
links_min_records	Zahl	
discard_links_max	Flag	
links_max_records	Zahl	

Tabelle 97. Eigenschaften von "webnode" (Forts.)

Eigenschaften von webnode	Datentyp	Eigenschaftsbeschreibung
weak_below	Zahl	
strong_above	Zahl	
link_size_continuous	Flag	
web_display	Circular Network Directed Grid	
graph_background	Farbe	Die Standardfarben für Diagramme werden am Anfang dieses Abschnitts beschrieben.
symbol_size	Zahl	Gibt eine Symbolgröße an.

Kapitel 13. Eigenschaften von Modellierungsknoten

Allgemeine Eigenschaften von Modellierungsknoten

Folgende Eigenschaften haben einige oder alle Modellierungsknoten gemeinsam. Etwaige Ausnahmen sind in der Dokumentation für die einzelnen Modellierungsknoten angegeben.

Tabelle 98. Allgemeine Eigenschaften von Modellierungsknoten

Eigenschaft	Werte	Eigenschaftsbeschreibung
custom_fields	Flag	Bei "true" können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
target ODER targets	Feld ODER [Feld1 ... FeldN]	Gibt je nach Modelltyp ein einzelnes Zielfeld oder mehrere Zielfelder an.
inputs	[Feld1 ... FeldN]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
partition	Feld	
use_partitioned_data	Flag	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
use_split_data	Flag	
splits	[Feld1 ... FeldN]	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an. Nur wirksam, wenn use_split_data auf True gesetzt ist.
use_frequency	Flag	Gewichtungs- und Häufigkeitsfelder werden von bestimmten Modellen je nach Angabe für die einzelnen Modelltypen verwendet.
frequency_field	Feld	
use_weight	Flag	
weight_field	Feld	
use_model_name	Flag	
model_name	Zeichenfolge	Benutzerdefinierter Name für neues Modell.
mode	Simple Expert	

Eigenschaften von "anomalydetectionnode"



Der Anomalieerkennungsknoten ermittelt ungewöhnliche Fälle bzw. "Ausreißer", die nicht den Mustern der "normalen" Daten entsprechen. Mit diesem Knoten können Ausreißer ermittelt werden, selbst wenn sie keinem bereits bekannten Muster entsprechen und selbst wenn Sie nicht genau wissen, wonach Sie suchen.

Beispiel

```
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)
```

Table 99. Eigenschaften von "anomalydetectionnode"

Eigenschaften von anomalydetectionnode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	Anomalieerkennungsmodelle führen ein Screening von Datensätzen auf der Grundlage der angegebenen Eingabefelder durch. Sie verwenden kein Zielfeld. Gewichtungsfelder werden ebenfalls nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Gibt die Methode für die Bestimmung des Trennwerts zur Kennzeichnung von Datensätzen als anomal an.
index_level	Zahl	Gibt den minimalen Trennwert für die Kennzeichnung von Anomalien an.
percent_records	Zahl	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage des Prozentsatzes der Datensätze in den Trainingsdaten fest.
num_records	Zahl	Legt den Schwellenwert für die Kennzeichnung von Datensätzen auf der Grundlage der Anzahl der Datensätze in den Trainingsdaten fest.
num_fields	Ganzzahl	Die Anzahl der für die einzelnen anomalen Datensätze zu meldenden Felder.
impute_missing_values	Flag	
adjustment_coeff	Zahl	Wert, der zum Balancieren der relativen Gewichtung verwendet wird, das den stetigen und den kategorialen Feldern bei der Berechnung der Distanz zugewiesen wird.
peer_group_num_auto	Flag	Berechnet automatisch die Anzahl der Peergruppen.
min_num_peer_groups	Ganzzahl	Gibt an, wie viele Peergruppen mindestens verwendet werden, wenn peer_group_num_auto auf True gesetzt ist.
max_num_per_groups	Ganzzahl	Gibt die maximale Anzahl an Peergruppen an.
num_peer_groups	Ganzzahl	Gibt an, wie viele Peergruppen verwendet werden, wenn peer_group_num_auto auf False gesetzt ist.

Tabelle 99. Eigenschaften von "anomalydetectionnode" (Forts.)

Eigenschaften von anomalydetectionnode	Werte	Eigenschaftsbeschreibung
noise_level	Zahl	Bestimmt, wie Ausreißer bei der Clusterbildung behandelt werden. Geben Sie einen Wert zwischen 0 und 0,5 an.
noise_ratio	Zahl	Gibt an, welcher Anteil des der Komponente zugeordneten Arbeitsspeichers für die Rausch-Pufferung verwendet werden soll. Geben Sie einen Wert zwischen 0 und 0,5 an.

Eigenschaften von "apriorinode"



Der Apriori-Knoten extrahiert ein Regelset aus den Daten und daraus die Regeln mit dem höchsten Informationsgehalt. Apriori bietet fünf verschiedene Methoden zur Auswahl von Regeln und verwendet ein ausgereiftes Indizierungsschema zur effizienten Verarbeitung großer Datensets. Bei großen Problemen ist Apriori in der Regel schneller zu trainieren, es gibt keine willkürliche Begrenzung für die Anzahl der Regeln, die beibehalten werden können, und es können Regeln mit bis zu 32 Vorbedingungen verarbeitet werden. Bei Apriori müssen alle Ein- und Ausgabefelder kategorial sein; dafür bietet es jedoch eine bessere Leistung, da es für diesen Datentyp optimiert ist.

Beispiel

```
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)
```

Tabelle 100. Eigenschaften von "apriorinode"

Eigenschaften von apriorinode	Werte	Eigenschaftsbeschreibung
consequents	Feld	Apriori-Modelle verwenden anstelle von standardmäßigen Ziel- und Eingabefeldern Sukzedenzen bzw. Antezedenzen. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
antecedents	[Feld1 ... FeldN]	
min_supp	Zahl	
min_conf	Zahl	
max_antecedents	Zahl	
true_flags	Flag	
optimize	Speed Memory	
use_transactional_data	Flag	
contiguous	Flag	
id_field	Zeichenfolge	
content_field	Zeichenfolge	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	Zahl	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften von "associationrulesnode"



Der Assoziationsregelknoten ähnelt dem Apriori-Knoten, jedoch kann der Assoziationsregelknoten im Gegensatz zu Apriori Listendaten verarbeiten. Außerdem kann der Assoziationsregelknoten mit IBM SPSS Analytic Server verwendet werden, um große Datenmengen und -vielfalt zu verarbeiten und schnellere parallele Verarbeitung zu nutzen.

Tabelle 101. Eigenschaften von "associationrulesnode"

Eigenschaften von associationrulesnode	Datentyp	Eigenschaftsbeschreibung
predictions	Feld	Felder in dieser Liste können nur als Prädiktor einer Regel angezeigt werden.
conditions	[Feld1...FeldN]	Felder in dieser Liste können nur als Bedingung einer Regel angezeigt werden.
max_rule_conditions	Ganzzahl	Die maximale Anzahl Bedingungen, die in eine einzelne Regel eingeschlossen werden können. Minimum: 1, Maximum: 9.

Tabelle 101. Eigenschaften von "associationrulesnode" (Forts.)

Eigenschaften von associationrulesnode	Datentyp	Eigenschaftsbeschreibung
max_rule_predictions	Ganzzahl	Die maximale Anzahl Vorhersagen, die in eine einzelne Regel eingeschlossen werden können. Minimum: 1, Maximum: 5.
max_num_rules	Ganzzahl	Die maximale Anzahl Regeln, die als Teil der Regelerstellung berücksichtigt werden können. Minimum: 1, Maximum: 10.000.
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Deployability	Das Regelkriterium, das den Wert bestimmt, nach dem die ersten "N" Regeln im Modell ausgewählt werden.
true_flags	boolesch	Die Einstellung Y legt fest, dass nur die wahren Werte für Flagfelder während der Regelerstellung berücksichtigt werden.
rule_criterion	boolesch	Die Einstellung Y legt fest, dass die Regelkriteriumswerte für das Ausschließen von Regeln während der Modellerstellung verwendet werden.
min_confidence	Zahl	0.1 bis 100; der Prozentwert für das erforderliche Mindestkonfidenzniveau für eine Regel, die vom Modell erzeugt wird. Wenn das Modell eine Regel mit einem Konfidenzniveau erzeugt, das den hier angegebenen Wert unterschreitet, wird die Regel verworfen.
min_rule_support	Zahl	0.1 bis 100; der Prozentwert für die erforderliche Mindestregelunterstützung für eine Regel, die vom Modell erzeugt wird. Wenn das Modell eine Regel mit einem Regelunterstützungsniveau erzeugt, das den hier angegebenen Wert unterschreitet, wird die Regel verworfen.
min_condition_support	Zahl	0.1 bis 100; der Prozentwert für die erforderliche Mindestbedingungsunterstützung für eine Regel, die vom Modell erzeugt wird. Wenn das Modell eine Regel mit einem Bedingungsunterstützungsniveau erzeugt, das den hier angegebenen Wert unterschreitet, wird die Regel verworfen.
min_lift	Ganzzahl	1 bis 10; stellt den erforderlichen Mindestlift für eine Regel dar, die vom Modell erzeugt wird. Wenn das Modell eine Regel mit einem Liftniveau erzeugt, das den hier angegebenen Wert unterschreitet, wird die Regel verworfen.
exclude_rules	boolesch	Wird verwendet, um eine Liste zusammengehöriger Felder auszuwählen, aus denen das Modell keine Regeln erstellen soll. Beispiel: set :gsarsnode.exclude_rules = [{{field1,field2, field3}},{{field4, field5}}], wobei jede durch {} getrennte Liste von Feldern eine Zeile in der Tabelle ist.

Tabelle 101. Eigenschaften von "associationrulesnode" (Forts.)

Eigenschaften von associationrulesnode	Datentyp	Eigenschaftsbeschreibung
num_bins	Ganzzahl	Legen Sie die Anzahl automatischer Klassen fest, in die stetige Felder eingeteilt werden. Minimum: 2, Maximum: 10.
max_list_length	Ganzzahl	Wird auf alle Listenfelder angewendet, für die die maximale Länge nicht bekannt ist. Elemente in der Liste bis zur hier angegebenen Zahl werden in die Modellerstellung eingeschlossen; weitere Elemente werden verworfen. Minimum: 1, Maximum: 100.
output_confidence	boolesch	
output_rule_support	boolesch	
output_lift	boolesch	
output_condition_support	boolesch	
output_deployability	boolesch	
rules_to_display	upto all	Die maximale Anzahl Regeln, die in den Ausgabebetabellen angezeigt werden sollen.
display_upto	Ganzzahl	Wenn upto in rules_to_display festgelegt wird, legen Sie die Anzahl Regeln fest, die in den Ausgabebetabellen angezeigt werden sollen. Minimum: 1.
field_transformations	boolesch	
records_summary	boolesch	
rule_statistics	boolesch	
most_frequent_values	boolesch	
most_frequent_fields	boolesch	
word_cloud	boolesch	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	Ganzzahl	Minimum: 1, Maximum: 20.
max_predictions	Ganzzahl	Die maximale Anzahl Regeln, die auf jede Eingabe mit dem Score angewendet werden können.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Wählen Sie das Maß aus, das zum Festlegen der Stärke von Regeln verwendet wird.
allow_repeats	boolesch	Legt fest, ob Regeln mit derselben Vorhersage in den Score eingeschlossen werden.
check_input	NoPredictions Predictions NoCheck	

Eigenschaften von "autoclassifiernode"



Mit dem Knoten "Autom. Klassifikationsmerkmal" können Sie eine Reihe verschiedener Modelle für binäre Ergebnisse ("Ja" oder "Nein", "Abwanderung" oder "Keine Abwanderung" usw.) erstellen und vergleichen, um den besten Ansatz für die jeweilige Analyse auszuwählen. Es wird eine Reihe von Modellierungsalgorithmen unterstützt, sodass Sie die gewünschten Methoden, die spezifischen Optionen für die jeweilige Methode und die Kriterien zum Vergleich der Ergebnisse auswählen können. Der Knoten generiert eine Gruppe von Modellen, die auf den angegebenen Optionen beruhen, und erstellt anhand der von Ihnen angegebenen Kriterien eine Rangordnung der besten Kandidaten.

Beispiel

```
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)
```

Tabelle 102. Eigenschaften von "autoclassifiernode".

Eigenschaften von autoclassifiernode	Werte	Eigenschaftsbeschreibung
target	<i>Feld</i>	Für Flagziele verlangt der Knoten "Automatisches Klassifikationsmerkmal" ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem können Gewichtung- und Häufigkeitsfelder angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	<i>Ganzzahl</i>	Anzahl der Modelle, die in das Modellnugget aufgenommen werden sollen. Geben Sie eine Ganzzahl zwischen 1 und 100 an.
calculate_variable_importance	<i>Flag</i>	
enable_accuracy_limit	<i>Flag</i>	
accuracy_limit	<i>Ganzzahl</i>	Ganzzahl zwischen 0 und 100.
enable_area_under_curve_limit	<i>Flag</i>	
area_under_curve_limit	<i>Zahl</i>	Reelle Zahl zwischen 0,0 und 1,0.
enable_profit_limit	<i>Flag</i>	
profit_limit	<i>Zahl</i>	Ganzzahl größer als 0.
enable_lift_limit	<i>Flag</i>	

Tabelle 102. Eigenschaften von "autoclassifiernode" (Forts.).

Eigenschaften von autoclassifiernode	Werte	Eigenschaftsbeschreibung
lift_limit	Zahl	Reelle Zahl größer als 1,0.
enable_number_of_variables_limit	Flag	
number_of_variables_limit	Zahl	Ganzzahl größer als 0.
use_fixed_cost	Flag	
fixed_cost	Zahl	Reelle Zahl größer 0.0.
variable_cost	Feld	
use_fixed_revenue	Flag	
fixed_revenue	Zahl	Reelle Zahl größer 0.0.
variable_revenue	Feld	
use_fixed_weight	Flag	
fixed_weight	Zahl	Reelle Zahl größer als 0,0
variable_weight	Feld	
lift_percentile	Zahl	Ganzzahl zwischen 0 und 100.
enable_model_build_time_limit	Flag	
model_build_time_limit	Zahl	Ganzzahl für die Anzahl der Minuten, die maximal für die Erstellung jedes einzelnen Modells aufgewendet werden.
enable_stop_after_time_limit	Flag	
stop_after_time_limit	Zahl	Reelle Zahl für die Anzahl der Stunden, die als Obergrenze für die insgesamt verstrichene Zeit für den Durchlauf eines automatischen Klassifikationsmerkmals verwendet wird.
enable_stop_after_valid_model_produced	Flag	
use_costs	Flag	
<Algorithmus>	Flag	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
<Algorithmus>.<Eigenschaft>	Zeichenfolge	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmeigenschaften“.

Festlegen der Algorithmeigenschaften

Für die Knoten vom Typ "Automatisches Klassifikationsmerkmal", "Auto-Numerisch" und "Autom. Cluster" können Eigenschaften für bestimmte vom Knoten verwendete Algorithmen mithilfe des folgenden allgemeinen Formats festgelegt werden:

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

Beispiel:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Die Algorithmenamen für den Knoten "Automatisches Klassifikationsmerkmal" lauten cart, chaid, quest, c50, logreg, decisionlist, bayesnet, discriminant, svm und knn.

Die Algorithmusnamen für den Knoten "Auto-Numerisch" lauten cart, chaid, neuralnetwork, genlin, svm, regression, linear und knn.

Algorithmusnamen für den Knoten "Autom. Cluster" sind twostep, k-means und kohonen.

Für die Eigenschaftsnamen wird der jeweils für den Algorithmusknoten dokumentierte Standard verwendet.

Algorithmeigenschaften, die Punkte oder andere Satzzeichen enthalten, müssen in einzelne Anführungsstriche eingebettet sein, z. B.:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

Als Eigenschaft können auch mehrere Werte zugewiesen werden, z. B.:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

So können Sie die Verwendung eines bestimmten Algorithmus aktivieren bzw. inaktivieren:

```
node.setPropertyValue("chaid", True)
```

Anmerkung: In Fällen, in denen bestimmte Algorithmusoptionen nicht im Knoten "Automatisches Klassifikationsmerkmal" verfügbar sind oder in denen nur ein einzelner Wert und kein Wertebereich angegeben werden kann, gelten dieselben Einschränkungen bei der Scripterstellung wie beim standardmäßigen Zugriff auf den Knoten.

Eigenschaften von "autoclusternode"



Mit dem Knoten "Autom. Cluster" können Sie Clustering-Modelle, die Gruppen und Datensätze mit ähnlichen Merkmalen identifizieren, schätzen und vergleichen. Die Funktionsweise des Knotens gleicht der von anderen Knoten für automatisierte Modellierung, und Sie können in einem einzigen Modellierungsdurchgang mit mehreren Optionskombinationen experimentieren. Modelle können mithilfe grundlegender Messwerte für Filterung und Rangfolge der Nützlichkeit von Clustermodellen verglichen werden, um ein Maß auf der Basis der Wichtigkeit von bestimmten Feldern zu liefern.

Beispiel

```
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)
```

Tabelle 103. Eigenschaften von "autoclusternode"

Eigenschaften von autoclusternode	Werte	Eigenschaftsbeschreibung
evaluation	Feld	Anmerkung: Nur Knoten "Autom. Cluster". Kennzeichnet das Feld, für das ein Wichtigkeitswert berechnet wird. Kann auch verwendet werden, um festzustellen, wie gut das Cluster den Wert dieses Felds differenziert, also wie gut das Modell den Wert für dieses Feld vorhersagen kann.

Tabelle 103. Eigenschaften von "autoclusterernode" (Forts.)

Eigenschaften von autoclusterernode	Werte	Eigenschaftsbeschreibung
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Training Test	
summary_limit	Ganzzahl	Anzahl der im Bericht aufzuführenden Modelle. Geben Sie eine Ganzzahl zwischen 1 und 100 an.
enable_silhouette_limit	Flag	
silhouette_limit	Ganzzahl	Ganzzahl zwischen 0 und 100.
enable_number_less_limit	Flag	
number_less_limit	Zahl	Reelle Zahl zwischen 0,0 und 1,0.
enable_number_greater_limit	Flag	
number_greater_limit	Zahl	Ganzzahl größer als 0.
enable_smallest_cluster_limit	Flag	
smallest_cluster_units	Prozent Counts	
smallest_cluster_limit_percentage	Zahl	
smallest_cluster_limit_count	Ganzzahl	Ganzzahl größer als 0.
enable_largest_cluster_limit	Flag	
largest_cluster_units	Prozent Counts	
largest_cluster_limit_percentage	Zahl	
largest_cluster_limit_count	Ganzzahl	
enable_smallest_largest_limit	Flag	
smallest_largest_limit	Zahl	
enable_importance_limit	Flag	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	Zahl	Ganzzahl zwischen 0 und 100.
importance_limit_less_than	Zahl	Ganzzahl zwischen 0 und 100.
<Algorithmus>	Flag	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
<Algorithmus>.<Eigenschaft>	Zeichenfolge	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmeigenschaften“ auf Seite 174.

Eigenschaften von "autonumericnode"



Der Knoten "Auto-Numerisch" schätzt und vergleicht mit einer Reihe verschiedener Methoden Modelle für die Ergebnisse stetiger numerischer Bereiche. Der Knoten arbeitet auf dieselbe Weise wie der Knoten "Automatisches Klassifikationsmerkmal": Sie können die zu verwendenden Algorithmen auswählen und in einem Modellierungsdurchlauf mit mehreren Optionskombinationen experimentieren. Folgende Algorithmen werden unterstützt: C&RT-Baum, CHAID, lineare Regression, verallgemeinerte lineare Regression und Support Vector Machines (SVM). Modelle können anhand von Korrelation, relativem Fehler bzw. Anzahl der verwendeten Variablen verglichen werden.

Beispiel

```
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)
```

Tabelle 104. Eigenschaften von "autonumericnode"

Eigenschaften von autonumericnode	Werte	Eigenschaftsbeschreibung
custom_fields	Flag	Bei "True" (Wahr) werden anstelle der Typknoteneinstellungen Einstellungen aus benutzerdefinierten Feldern verwendet.
target	Feld	Für den Knoten "Auto-Numerisch" sind ein einzelnes Ziel und eines oder mehrere Eingabefelder erforderlich. Außerdem können Gewichtung- und Häufigkeitsfelder angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
inputs	[Feld1 ... Feld2]	
partition	Feld	
use_frequency	Flag	
frequency_field	Feld	
use_weight	Flag	
weight_field	Feld	
use_partitioned_data	Flag	Wenn ein Partitionsfeld definiert ist, werden nur die Trainingsdaten für die Modellerstellung verwendet.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	Ganzzahl	Anzahl der Modelle, die in das Modellnugget aufgenommen werden sollen. Geben Sie eine Ganzzahl zwischen 1 und 100 an.
calculate_variable_importance	Flag	

Tabelle 104. Eigenschaften von "autonumericnode" (Forts.)

Eigenschaften von autonumericnode	Werte	Eigenschaftsbeschreibung
enable_correlation_limit	Flag	
correlation_limit	Ganzzahl	
enable_number_of_fields_limit	Flag	
number_of_fields_limit	Ganzzahl	
enable_relative_error_limit	Flag	
relative_error_limit	Ganzzahl	
enable_model_build_time_limit	Flag	
model_build_time_limit	Ganzzahl	
enable_stop_after_time_limit	Flag	
stop_after_time_limit	Ganzzahl	
stop_if_valid_model	Flag	
<Algorithmus>	Flag	Aktiviert oder inaktiviert die Verwendung eines bestimmten Algorithmus.
<Algorithmus>.<Eigenschaft>	Zeichenfolge	Legt einen Eigenschaftswert für einen bestimmten Algorithmus fest. Weitere Informationen finden Sie im Thema „Festlegen der Algorithmeigenschaften“ auf Seite 174.

Eigenschaften von "bayesnetnode"



Mithilfe des Bayes-Netzknosens können Sie ein Wahrscheinlichkeitsmodell erstellen, indem Sie beobachtete und aufgezeichnete Hinweise mit Weltwissen kombinieren, um die Wahrscheinlichkeit ihres Vorkommens zu ermitteln. Der Knoten ist speziell für Netze vom Typ "Tree Augmented Naïve Bayes" (TAN) und "Markov-Decke" gedacht, die in erster Linie zur Klassifizierung verwendet werden.

Beispiel

```
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")
```

Tabelle 105. Eigenschaften von "bayesnetnode"

Eigenschaften von bayesnetnode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	Bayes-Netzmodelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Stetige Felder werden automatisch klassiert. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue_training_existing_model	Flag	

Tabelle 105. Eigenschaften von "bayesnetnode" (Forts.)

Eigenschaften von bayesnetnode	Werte	Eigenschaftsbeschreibung
structure_type	TAN MarkovBlanket	Dient zur Auswahl der beim Erstellen des Bayes-Netztes zu verwendenden Struktur.
use_feature_selection	Flag	
parameter_learning_method	Likelihood Bayes	Gibt die Methode an, die zur Schätzung der Tabellen zur bedingten Wahrscheinlichkeit zwischen Knoten verwendet wird, wenn die Werte der übergeordneten Elemente bekannt sind.
mode	Expert Simple	
missing_values	Flag	
all_probabilities	Flag	
independence	Likelihood Pearson	Gibt die Methode an, die zur Einschätzung verwendet wird, ob paarige Beobachtungen bei zwei Variablen voneinander unabhängig sind.
significance_level	Zahl	Gibt den Trennwert für die Bestimmung der Unabhängigkeit an.
maximal_conditioning_set	Zahl	Legt die Maximalzahl der für die Unabhängigkeitstests zu verwendenden Konditionierungsvariablen fest.
inputs_always_selected	[Feld1 ... FeldN]	Gibt an, welche Felder aus dem Dataset immer beim Erstellen des Bayes-Netztes verwendet werden. Anmerkung: Das Zielfeld ist immer ausgewählt.
maximum_number_inputs	Zahl	Gibt die maximale Anzahl an Eingabefeldern an, die beim Erstellen des Bayes-Netztes verwendet werden sollen.
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "buildr"



Der R-Erstellungsknoten ermöglicht es Ihnen, ein benutzerdefiniertes R-Script einzugeben, um die Modellerstellung und das Modellscoring, die in IBM SPSS Modeler implementiert sind, auszuführen.

Beispiel

```
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="")
modellerDataModel<-data.frame(modellerDataModel,var1)""")
```

Tabelle 106. Eigenschaften von "buildr".

Eigenschaften von buildr	Werte	Eigenschaftsbeschreibung
build_syntax	Zeichenfolge	R-Scriptsyntax für die Modellerstellung.
score_syntax	Zeichenfolge	R-Scriptsyntax für das Modellscoring.
convert_flags	StringsAndDoubles LogicalValues	Option zum Konvertieren von Flagfeldern.
convert_datetime	Flag	Option zum Konvertieren von Variablen mit Datums- oder Datums-/Zeitformaten in R-Datums-/Zeitformate.
convert_datetime_class	POSIXct POSIXlt	Optionen, die angeben, in welches Format Variablen mit Datums- oder Datums-/Zeitformaten konvertiert werden.
convert_missing	Flag	Option zum Konvertieren fehlender Werte in R-Werte "NA".
output_html	Flag	Option für die Anzeige von Diagrammen im R-Modellnugget.
output_text	Flag	Option zum Schreiben von R-Konsolentext auf eine Registerkarte des R-Modellnuggets.

Eigenschaften von "c50node"



Der C5.0-Knoten erstellt entweder einen Entscheidungsbaum oder ein Regelset. Das Modell teilt die Stichprobe auf der Basis des Felds auf, das auf der jeweiligen Ebene den maximalen Informationsgewinn liefert. Das Zielfeld muss kategorial sein. Es sind mehrere Aufteilungen in mehr als zwei Untergruppen zulässig.

Beispiel

```
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]])
```

Tabelle 107. Eigenschaften von "c50node"

Eigenschaften von c50node	Werte	Eigenschaftsbeschreibung
target	Feld	C50-Modelle verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
output_type	DecisionTree RuleSet	
group_symbolics	Flag	
use_boost	Flag	
boost_num_trials	Zahl	
use_xval	Flag	
xval_num_folds	Zahl	
mode	Simple Expert	
favor	Accuracy Generality	Genauigkeit oder Allgemeingültigkeit werden vorselektiert.
expected_noise	Zahl	
min_child_records	Zahl	
pruning_severity	Zahl	
use_costs	Flag	
Kosten	strukturiert	Hierbei handelt es sich um eine strukturierte Eigenschaft.
use_winnowing	Flag	
use_global_pruning	Flag	Standardmäßig auf True gesetzt.
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "carmanode"



Beim CARMA-Modell wird ein Regelset aus den Daten extrahiert, ohne dass Sie Eingabe- oder Zielfelder angeben müssen. Im Gegensatz zu Apriori bietet der CARMA-Knoten Erstellungseinstellungen für die Regelunterstützung (Unterstützung für Antezedens und Sukzedens) und nicht nur für die Antezedens-Unterstützung. Die erstellten Regeln können somit für eine größere Palette an Anwendungen verwendet werden, beispielsweise um eine Liste mit Produkten und Dienstleistungen (Antezedenzen) zu finden, deren Nachfolger (Sukzedens) das Element darstellt, das Sie in der Ferienzeit desselben Jahres bewerben möchten.

Beispiel

```

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)
# Expertenoptionen
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)

```

Tabelle 108. Eigenschaften von "carmanode"

Eigenschaften von carmanode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... Feldn]	CARMA-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtung- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
id_field	Feld	Das Feld wird als ID-Feld für die Modellerstellung verwendet.
contiguous	Flag	Dient zur Angabe, ob IDs im ID-Feld zusammenhängend sind.
use_transactional_data	Flag	
content_field	Feld	
min_supp	Zahl (Prozent)	Bezieht sich auf die Regelunterstützung und nicht auf die Antezedens-Unterstützung. Der Standardwert ist 20 %.
min_conf	Zahl (Prozent)	Der Standardwert ist 20 %.
max_size	Zahl	Der Standardwert ist 10.
mode	Simple Expert	Der Standardwert ist Simple.
exclude_multiple	Flag	Schließt Regeln mit mehreren Sukzedenzen aus. Standardmäßig ist dieser Wert False.
use_pruning	Flag	Standardmäßig ist dieser Wert False.
pruning_value	Zahl	Der Standardwert ist 500.
vary_support	Flag	
estimated_transactions	Ganzzahl	
rules_without_antecedents	Flag	

Eigenschaften von "cartnode"



Der Knoten für Klassifizierungs- und Regressions-Bäume (C&RT-Bäume) erstellt einen Entscheidungsbaum, mit dem Sie zukünftige Beobachtungen vorhersagen oder klassifizieren können. Bei dieser Methode wird eine rekursive Partitionierung verwendet, um die Trainingsdatensätze in Segmente aufzuteilen. Dabei wird bei jedem Schritt die Unreinheit verringert und ein Knoten im Baum wird als "rein" betrachtet, wenn 100 % der Fälle in eine bestimmte Kategorie des Zielfelds fallen. Ziel- und Eingabefelder können numerische Bereiche oder kategorial (nominal, ordinal oder Flags) sein. Alle Aufteilungen sind binär (nur zwei Untergruppen).

Beispiel

```
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")
```

Tabelle 109. Eigenschaften von "cartnode"

Eigenschaften von cartnode	Werte	Eigenschaftsbeschreibung
target	Feld	Modelle vom Typ "C&R-Baum" verwenden ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue_training_existing_model	Flag	
objective	Standard Boosting Bagging PSM	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	Flag	

Tabelle 109. Eigenschaften von "cartnode" (Forts.)

Eigenschaften von cartnode	Werte	Eigenschaftsbeschreibung
tree_directives	Zeichenfolge	Geben Sie Aufbauregeln für die Erweiterung des Baums an. Aufbauregeln können in dreifache Anführungszeichen gesetzt werden, um auf Escapezeichen für neue Zeilen oder Anführungszeichen verzichten zu können. Beachten Sie, dass die Anweisungen auf kleinste Änderungen in den Daten- oder Modellierungsoptionen reagieren und nicht für andere Datasets verallgemeinert werden können.
use_max_depth	Default Custom	
max_depth	Ganzzahl	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	Flag	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	Flag	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	Zahl	Maximale Differenz.
max_surrogates	Zahl	Maximale Anzahl Ersatztrenner.
use_percentage	Flag	
min_parent_records_pc	Zahl	
min_child_records_pc	Zahl	
min_parent_records_abs	Zahl	
min_child_records_abs	Zahl	
use_costs	Flag	
Kosten	strukturiert	Strukturierte Eigenschaft.
priors	Data Equal Custom	
custom_priors	strukturiert	Strukturierte Eigenschaft.
adjust_priors	Flag	
trails	Zahl	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	Flag	Boosting auf sehr große Datasets anwenden.
min_impurity	Zahl	

Tabelle 109. Eigenschaften von "cartnode" (Forts.)

Eigenschaften von cartnode	Werte	Eigenschaftsbeschreibung
impurity_measure	Gini Twoing Ordered	
train_pct	Zahl	Set zur Verhinderung übermäßiger Anpassung.
set_random_seed	Flag	Option "Ergebnisse replizieren".
seed	Zahl	
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "chaidnode"



Der CHAID-Knoten erzeugt Entscheidungsbäume unter Verwendung von Chi-Quadrat-Statistiken zur Ermittlung optimaler Aufteilungen. Im Gegensatz zu den Knoten vom Typ "C&RT-Baum" und "QUEST" kann CHAID nicht binäre Bäume generieren, d. h. Bäume mit Aufteilungen mit mehr als zwei Verzweigungen. Ziel- und Eingabefelder können in einem numerischen Bereich (stetig) oder kategorial sein. Exhaustive CHAID ist eine Änderung von CHAID, die noch gründlicher vorgeht, indem sie alle möglichen Aufteilungen untersucht, allerdings mehr Rechenzeit beansprucht.

Beispiel

```

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)

```

Tabelle 110. Eigenschaften von "chaidnode"

Eigenschaften von chaidnode	Werte	Eigenschaftsbeschreibung
target	Feld	CHAID-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue_training_existing_model	Flag	
objective	Standard Boosting Bagging PSM	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	Flag	
tree_directives	Zeichenfolge	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	Ganzzahl	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
use_percentage	Flag	
min_parent_records_pc	Zahl	
min_child_records_pc	Zahl	
min_parent_records_abs	Zahl	
min_child_records_abs	Zahl	
use_costs	Flag	
Kosten	strukturiert	Strukturierte Eigenschaft.
trails	Zahl	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	Flag	Boosting auf sehr große Datasets anwenden.
split_alpha	Zahl	Signifikanzschwelle für Aufteilung.
merge_alpha	Zahl	Signifikanzschwelle für Zusammenführung.
bonferroni_adjustment	Flag	Signifikanzwerte mit der Bonferroni-Methode anpassen.
split_merged_categories	Flag	Erneutes Aufteilen zusammengeführter Kategorien zulassen.

Tabelle 110. Eigenschaften von "chaidnode" (Forts.)

Eigenschaften von chaidnode	Werte	Eigenschaftsbeschreibung
chi_square	Pearson LR	Verwendetes Verfahren für die Berechnung der Chi-Quadrat-Statistik: Pearson oder Likelihood-Quotient
Epsilon	Zahl	Minimale Änderung in der erwarteten Zellhäufigkeit...
max_iterations	Zahl	Maximale Anzahl der Iterationen für Konvergenz.
set_random_seed	Ganzzahl	
seed	Zahl	
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	Ganzzahl	

Eigenschaften von "coxregnode"



Der Knoten vom Typ "Cox-Regression" ermöglicht Ihnen auch bei zensierten Datensätzen die Erstellung eines Überlebensmodells für Daten über die Zeit bis zum Eintreten des Ereignisses. Das Modell erstellt eine Überlebensfunktion, die die Wahrscheinlichkeit vorhersagt, dass das untersuchte Ereignis für bestimmte Werte der Eingabevariablen zu einem bestimmten Zeitpunkt (t) eingetreten ist.

Beispiel

```
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)
```

Tabelle 111. Eigenschaften von "coxregnode"

Eigenschaften von coxregnode	Werte	Eigenschaftsbeschreibung
survival_time	Feld	Cox-Regressionsmodelle erfordern ein einzelnes Feld, das die Überlebenszeiten enthält.
target	Feld	Cox-Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
method	Enter Stepwise BackwardsStepwise	
groups	Feld	

Tabelle 111. Eigenschaften von "coxregnode" (Forts.)

Eigenschaften von coxregnode	Werte	Eigenschaftsbeschreibung
model_type	MainEffects Custom	
custom_terms	["BP*Sex" "BP*Age"]	
mode	Expert Simple	
max_iterations	Zahl	
p_converge	1,0E-4 1,0E-5 1,0E-6 1,0E-7 1,0E-8 0	
p_converge	1,0E-4 1,0E-5 1,0E-6 1,0E-7 1,0E-8 0	
l_converge	1,0E-1 1,0E-2 1,0E-3 1,0E-4 1,0E-5 0	
removal_criterion	LR Wald Bedingung	
probability_entry	Zahl	
probability_removal	Zahl	
output_display	EachStep LastStep	
ci_enable	Flag	
ci_value	90 95 99	
correlation	Flag	
display_baseline	Flag	
survival	Flag	
hazard	Flag	
log_minus_log	Flag	
one_minus_survival	Flag	
separate_line	Feld	
value	Zahl oder Zeichenfolge	Wenn für ein Feld kein Wert angegeben ist, wird die Standardoption "Mittelwert" für das betreffende Feld verwendet.

Eigenschaften von "decisionlistnode"



Der Knoten "Entscheidungsliste" kennzeichnet Untergruppen bzw. Segmente, die eine höhere oder geringere Wahrscheinlichkeit für ein bestimmtes binäres Ergebnis aufweisen als die Gesamtpopulation. Sie könnten beispielsweise nach Kunden suchen, deren Abwanderung unwahrscheinlich ist oder die mit großer Wahrscheinlichkeit positiv auf eine Kampagne reagieren. Sie können Ihr Fachwissen in das Modell integrieren, indem Sie eigene, benutzerdefinierte Segmente hinzufügen und eine Vorschau anzeigen, in der alternative Modelle nebeneinander angezeigt werden, um die Ergebnisse zu vergleichen. Entscheidungslistenmodelle bestehen aus einer Liste von Regeln, bei denen jede Regel eine Bedingung und ein Ergebnis aufweist. Regeln werden in der vorgegebenen Reihenfolge angewendet und die erste Regel, die zutrifft, bestimmt das Ergebnis.

Beispiel

```
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)
```

Tabelle 112. Eigenschaften von "decisionlistnode"

Eigenschaften von decisionlistnode	Werte	Eigenschaftsbeschreibung
target	Feld	Entscheidungslistenmodelle verwenden ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
model_output_type	Modell InteractiveBuilder	
search_direction	Up Down	Bezieht sich auf das Finden von Segmenten; dabei entspricht "Up" einer hohen Wahrscheinlichkeit und "Down" einer geringen Wahrscheinlichkeit.
target_value	Zeichenfolge	Wenn dieser Wert nicht angegeben wird, nimmt er für Flags den Wert "True" (Wahr) an.
max_rules	Ganzzahl	Die maximale Anzahl der Segmente ausschließlich des Rests.
min_group_size	Ganzzahl	Mindestsegmentgröße.
min_group_size_pct	Zahl	Mindestsegmentgröße als Prozentsatz.
confidence_level	Zahl	Mindestschwellenwert, den ein Eingabefeld aufweist, um die Wahrscheinlichkeit eines Treffers zu verbessern (Lift), damit es zu einer Segmentdefinition hinzugefügt werden kann.
max_segments_per_rule	Ganzzahl	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	

Tabelle 112. Eigenschaften von "decisionlistnode" (Forts.)

Eigenschaften von decisionlistnode	Werte	Eigenschaftsbeschreibung
bin_count	Zahl	
max_models_per_cycle	Ganzzahl	Suchbreite für Listen.
max_rules_per_cycle	Ganzzahl	Suchbreite für Segmentregeln.
segment_growth	Zahl	
include_missing	Flag	
final_results_only	Flag	
reuse_fields	Flag	Ermöglicht die Wiederverwendung von Attributen (Eingabefelder, die in Regeln vorkommen).
max_alternatives	Ganzzahl	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "discriminantnode"



Bei der Diskriminanzanalyse werden strengere Annahmen als bei der logistischen Regression verwendet, sie kann jedoch eine wertvolle Alternative oder Ergänzung zu einer logistischen Regressionsanalyse sein, wenn diese Annahmen erfüllt sind.

Beispiel

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tabelle 113. Eigenschaften von "discriminantnode"

Eigenschaften von discriminantnode	Werte	Eigenschaftsbeschreibung
target	Feld	Diskriminanzmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Gewichtungsfelder und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	

Tabelle 113. Eigenschaften von "discriminantnode" (Forts.)

Eigenschaften von discriminantnode	Werte	Eigenschaftsbeschreibung
Mittelwert	Flag	Statistikoptionen im Dialogfeld "Erweiterte Ausgabe".
univariate_anovas	Flag	
box_m	Flag	
within_group_covariance	Flag	
within_groups_correlation	Flag	
separate_groups_covariance	Flag	
total_covariance	Flag	
fishers	Flag	
unstandardized	Flag	
casewise_results	Flag	Klassifizierungsoptionen im Dialogfeld "Erweiterte Ausgabe".
limit_to_first	Zahl	Der Standardwert ist 10.
summary_table	Flag	
leave_one_classification	Flag	
combined_groups	Flag	
separate_groups_covariance	Flag	Matrizenoption Gruppenspezifische Kovarianzmatrix
territorial_map	Flag	
combined_groups	Flag	Plotoption Kombinierte Gruppen.
separate_groups	Flag	Plotoption Gruppenspezifisch.
summary_of_steps	Flag	
F_pairwise	Flag	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	Zahl	
criteria	UseValue UseProbability	
F_value_entry	Zahl	Der Standardwert ist 3,84.
F_value_removal	Zahl	Der Standardwert ist 2,71.
probability_entry	Zahl	Der Standardwert ist 0.05.
probability_removal	Zahl	Der Standardwert ist 0.10.
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "factornode"



Der Faktor/PCA-Knoten bietet leistungsstarke Datenreduktionsverfahren zur Verringerung der Komplexität der Daten. Die Hauptkomponentenanalyse (PCA) findet lineare Kombinationen der Eingabefelder, die die Varianz im gesamten Set der Felder am besten erfassen, wenn die Komponenten orthogonal (senkrecht) zueinander sind. Mit der Faktorenanalyse wird versucht, die zugrunde liegenden Faktoren zu bestimmen, die die Korrelationsmuster innerhalb eines Sets beobachteter Felder erklären. Bei beiden Ansätzen besteht das Ziel darin, eine kleinere Zahl abgeleiteter Felder zu finden, mit denen die Informationen in der ursprünglichen Menge der Felder effektiv zusammengefasst werden können.

Beispiel

```
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)
```

Tabelle 114. Eigenschaften von "factornode"

Eigenschaften von factornode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	PCA-/Faktormodelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtung- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	

Tabelle 114. Eigenschaften von "factornode" (Forts.)

Eigenschaften von factornode	Werte	Eigenschaftsbeschreibung
max_iterations	Zahl	
complete_records	Flag	
Matrix	Correlation Kovarianz	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	Zahl	
max_factor	Zahl	
Rotation	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	Zahl	Wenn Sie DirectOblimin als Rotationsdatentyp auswählen, können Sie einen Wert für delta festlegen. Wenn Sie keinen Wert festlegen, wird für delta der Standardwert verwendet.
kappa	Zahl	Wenn Sie Promax als Rotationsdatentyp auswählen, können Sie einen Wert für kappa festlegen. Wenn Sie keinen Wert festlegen, wird für kappa der Standardwert verwendet.
sort_values	Flag	
hide_values	Flag	
hide_below	Zahl	

Eigenschaften von "featureselectionnode"



Der Merkmalauswahlknoten sichtet die Eingabefelder, um auf der Grundlage einer Reihe von Kriterien (z. B. dem Prozentsatz der fehlenden Werte) zu entscheiden, ob diese entfernt werden sollen. Anschließend erstellt er eine Wichtigkeitsrangfolge der verbleibenden Eingaben in Bezug auf ein angegebenes Ziel. Beispiel: Angenommen, Sie haben ein Dataset mit Hunderten potenzieller Eingaben. Welche davon sind voraussichtlich für die Modellierung von medizinischen Behandlungsergebnissen von Bedeutung?

Beispiel

```
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)

```

Ein detaillierteres Beispiel, mit dem ein Merkmalauswahlmodell erstellt und angewendet wird, finden Sie in „Beispiel für Standalone-Script: Generieren eines Merkmalauswahlmodells“ auf Seite 5.

Tabelle 115. Eigenschaften von "featureselectionnode"

Eigenschaften von featureselectionnode	Werte	Eigenschaftsbeschreibung
target	Feld	Merkmalauswahlmodelle teilen Prädiktoren relativ zum angegebenen Ziel in Ränge ein. Gewichtungs- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
screen_single_category	Flag	Bei True wird ein Screening der Felder durchgeführt, bei denen zu viele Datensätze (im Verhältnis zur Gesamtzahl der Datensätze) in dieselbe Kategorie fallen.
max_single_category	Zahl	Gibt den Schwellenwert an, der verwendet wird, wenn screen_single_category auf True gesetzt ist.
screen_missing_values	Flag	Bei True wird ein Screening der Felder durchgeführt, die zu viele fehlende Werte (ausgedrückt als Prozentsatz der Gesamtzahl an Datensätzen) aufweisen.
max_missing_values	Zahl	
screen_num_categories	Flag	Bei True wird ein Screening der Felder durchgeführt, die zu viele Kategorien im Verhältnis zur Gesamtzahl der Datensätze aufweisen.
max_num_categories	Zahl	
screen_std_dev	Flag	Bei True wird ein Screening der Felder durchgeführt, deren Standardabweichung kleiner oder gleich dem angegebenen Mindestwert ist.
min_std_dev	Zahl	
screen_coeff_of_var	Flag	Bei True wird ein Screening der Felder durchgeführt, deren Varianzkoeffizient kleiner oder gleich dem angegebenen Mindestwert ist.
min_coeff_of_var	Zahl	
criteria	Pearson Likelihood CramersV Lambda	Wenn kategoriale Prädiktoren hinsichtlich eines kategorialen Ziels nach Rängen geordnet werden, wird hier das Maß angegeben, auf dem der Wert für die Wichtigkeit beruht.

Tabelle 115. Eigenschaften von "featureselectionnode" (Forts.)

Eigenschaften von featureselectionnode	Werte	Eigenschaftsbeschreibung
unimportant_below	Zahl	Gibt die p -Schwellenwerte an, die verwendet werden, um Variablen als "bedeutsam", "marginal" bzw. "unbedeutend" eingestuft werden. Zulässig sind Werte von 0,0 bis 1,0.
important_above	Zahl	Zulässig sind Werte von 0,0 bis 1,0.
unimportant_label	Zeichenfolge	Gibt die Beschriftung für die Rangstufe "unbedeutsam" an.
marginal_label	Zeichenfolge	
important_label	Zeichenfolge	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.
select_marginal	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
select_unimportant	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob unbedeutende Felder ausgewählt werden sollen.
importance_value	Zahl	Wenn selection_mode auf ImportanceValue gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 100.
top_n	Ganzzahl	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 1000.

Eigenschaften von "genlinnode"



Das verallgemeinerte lineare Modell erweitert das allgemeine lineare Modell so, dass die abhängige Variable über eine angegebene Verknüpfungsfunktion in linearem Zusammenhang zu den Faktoren und Kovariaten steht. Außerdem ist es mit diesem Modell möglich, dass die abhängige Variable eine von der Normalverteilung abweichende Verteilung aufweist. Es deckt die Funktionen einer großen Bandbreite an Statistikmodellen ab, darunter lineare Regression, logistische Regression, loglineare Modelle für Häufigkeitsdaten und Überlebensmodelle mit Intervallzensurierung.

Beispiel

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabelle 116. Eigenschaften von "genl innode"

Eigenschaften von genl innode	Werte	Eigenschaftsbeschreibung
target	Feld	Verallgemeinerte lineare Modelle erfordern ein einzelnes Zielfeld, bei dem es sich um ein nominales oder ein Flagfeld handeln muss, und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
use_weight	Flag	
weight_field	Feld	Der Feldtyp ist nur stetig.
target_represents_trials	Flag	
trials_type	Variable FixedValue	
trials_field	Feld	Der Feldtyp ist stetig, Flag oder ordinal.
trials_number	Zahl	Der Standardwert ist 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	Feld	Der Feldtyp ist nur stetig.
offset_value	Zahl	Muss eine reelle Zahl sein.
base_category	Last First	
include_intercept	Flag	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Invers normal. NEGBIN: Negativ binomial.
negbin_para_type	Specify Estimate	
negbin_parameter	Zahl	Der Standardwert ist 1. Muss eine nicht negative reelle Zahl enthalten.
tweedie_parameter	Zahl	

Tabelle 116. Eigenschaften von "genlnode" (Forts.)

Eigenschaften von genlnode	Werte	Eigenschaftsbeschreibung
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: Log-Log komplementär. LOGC: Log-Komplement. NEGBIN: Negativ binomial. NLOGLOG: Log-Log negativ. CUMCAUCHIT: Cauchit (kumulativ). CUMCLOGLOG: Log-Log komplementär (kumulativ). CUMLOGIT: Logit (kumulativ). CUMNLOGLOG: Log-Log negativ (kumulativ). CUMPROBIT: Probit (kumulativ).
power	Zahl	Der Wert muss eine reelle Zahl ungleich null sein.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	Zahl	Der Standardwert ist 1; nur positive Ganzzahlen zulässig.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	Zahl	Der Standardwert ist 1; muss größer als 0 sein.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	Zahl	Der Standardwert ist 100; nur nicht negative Ganzzahlen.
max_step_halving	Zahl	Der Standardwert ist 5; nur positive Ganzzahlen.
check_separation	Flag	
start_iteration	Zahl	Der Standardwert ist 20; nur positive Ganzzahlen zulässig.
estimates_change	Flag	
estimates_change_min	Zahl	Der Standardwert ist 1E-006; nur positive Zahlen zulässig.
estimates_change_type	Absolute Relative	
loglikelihood_change	Flag	
loglikelihood_change_min	Zahl	Nur positive Zahlen zulässig.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	Flag	
hessian_convergence_min	Zahl	Nur positive Zahlen zulässig.

Tabelle 116. Eigenschaften von "genl innode" (Forts.)

Eigenschaften von genl innode	Werte	Eigenschaftsbeschreibung
hessian_convergence_type	Absolute Relative	
case_summary	Flag	
contrast_matrices	Flag	
descriptive_statistics	Flag	
estimable_functions	Flag	
model_info	Flag	
iteration_history	Flag	
goodness_of_fit	Flag	
print_interval	Zahl	Der Standardwert ist 1; muss eine positive Ganzzahl sein.
model_summary	Flag	
lagrange_multiplier	Flag	
parameter_estimates	Flag	
include_exponential	Flag	
covariance_estimates	Flag	
correlation_estimates	Flag	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	Zahl	Der Standardwert ist 0,0001.
confidence_interval	Zahl	Der Standardwert ist 95.
loglikelihood_function	Full Kern	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "glmmnode"



Verallgemeinerte lineare gemischte Modelle (Generalized Linear Mixed Models; GLMM) erweitern lineare Modelle so, dass das Ziel nicht normalverteilt zu sein braucht und über eine angegebene Verknüpfungsfunktion in einer linearen Beziehung zu den Faktoren und Kovariaten steht und die Beobachtungen korreliert werden können. Verallgemeinerte lineare gemischte Modelle decken eine breite Palette verschiedener Modelle ab, von einfacher linearer Regression bis hin zu komplexen Mehrebenenmodellen für nicht normalverteilte Longitudinaldaten.

Tabelle 117. Eigenschaften von "glmmnode".

Eigenschaften von glmmnode	Werte	Eigenschaftsbeschreibung
residual_subject_spec	strukturiert	Die Wertekombination der angegebenen kategorialen Felder, die Subjekte innerhalb des Datensets eindeutig definieren.
repeated_measures	strukturiert	Felder zur Ermittlung von wiederholten Beobachtungen.
residual_group_spec	[Feld1 ... FeldN]	Felder, die unabhängige Sätze von Kovarianzparametern für wiederholte Effekte definieren.
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Definiert die Kovarianzstruktur für Residuen.
custom_target	Flag	Gibt an, ob das im vorausgehenden Knoten definierte Ziel (false) oder das im Feld target_field festgelegte benutzerdefinierte Ziel (true) verwendet werden soll.
target_field	Feld	Als Ziel zu verwendendes Feld, wenn custom_target auf true gesetzt ist.
use_trials	Flag	Gibt an, ob zusätzliche Felder oder Werte zur Angabe der Anzahl an Tests verwendet werden sollen, wenn es sich bei der Zielantwort um eine Reihe von Ereignissen handelt, die während Tests auftreten. Die Standardeinstellung ist false.
use_field_or_value	Field Value	Gibt an, ob die Anzahl an Test in einem Feld (Standard) oder als Wert angegeben werden soll.
trials_field	Feld	Feld zur Angabe der Anzahl an Tests.
trials_value	Ganzzahl	Wert zur Angabe der Anzahl an Tests. Wenn angegeben, ist der Minimalwert 1.
use_custom_target_reference	Flag	Gibt an, ob eine benutzerdefinierte Referenzkategorie für ein kategoriales Ziel verwendet werden soll. Die Standardeinstellung ist false.
target_reference_value	Zeichenfolge	Zu verwendende Referenzkategorie, wenn use_custom_target_reference auf true gesetzt ist.

Tabelle 117. Eigenschaften von "glimmnode" (Forts.).

Eigenschaften von glimmnode	Werte	Eigenschaftsbeschreibung
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Allgemeine Modelle für die Verteilung von Werten für das Ziel. Wählen Sie Custom aus, um einen Verteilungstyp aus der von target_distribution bereitgestellten Liste festzulegen.
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson	Verteilung von Werten für das Ziel, wenn dist_link_combination auf Custom gesetzt ist.
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Verknüpfungsfunktion zum Herstellen von Beziehungen zwischen Zielwerten und Prädiktoren. Wenn target_distribution auf Binomial gesetzt ist, können Sie jede der aufgelisteten Verknüpfungsfunktionen verwenden. Wenn target_distribution auf Multinomial gesetzt ist, können Sie CLOGLOG, CAUCHIT, LOGIT, NLOGLOG oder PROBIT verwenden. Wenn target_distribution auf einen anderen Wert als Binomial oder Multinomial gesetzt ist, können Sie IDENTITY, LOG oder POWER verwenden.
link_function_param	Zahl	Für die Verknüpfungsfunktion zu verwendender Parameterwert. Wird nur verwendet, wenn normal_link_function oder link_function_type auf POWER gesetzt ist.
use_predefined_inputs	Flag	Gibt an, ob die Felder, die in einer übergeordneten Ebene als Eingabefelder definiert wurden (true), oder die Felder in fixed_effects_list (false) als Felder für feste Effekte verwendet werden sollen. Die Standardeinstellung ist false.
fixed_effects_list	strukturiert	Wenn use_predefined_inputs auf false gesetzt ist, werden die Eingabefelder als Felder für feste Effekte verwendet.
use_intercept	Flag	Wenn true gesetzt ist (Standardeinstellung), wird der konstante Term in das Modell einbezogen.
random_effects_list	strukturiert	Liste von Feldern, die als zufällige Effekte festgelegt werden.
regression_weight_field	Feld	Zur Analysegewichtung zu verwendendes Feld.
use_offset	None offset_value offset_field	Gibt an, wie der Offset festgelegt wird. Lautet der Wert None, wird kein Offset verwendet.

Tabelle 117. Eigenschaften von "glimmnode" (Forts.).

Eigenschaften von glimmnode	Werte	Eigenschaftsbeschreibung
offset_value	Zahl	Für den Offset zu verwendender Wert, wenn use_offset auf offset_value gesetzt ist.
offset_field	Feld	Für den Offsetwert zu verwendendes Feld, wenn use_offset auf offset_field gesetzt ist.
target_category_order	Ascending Descending Data	Sortierreihenfolge für kategoriale Ziele. Der Wert Data gibt an, dass die Sortierreihenfolge der Daten verwendet wird. Die Standardeinstellung ist Ascending.
inputs_category_order	Ascending Descending Data	Sortierreihenfolge für kategoriale Prädiktoren. Der Wert Data gibt an, dass die Sortierreihenfolge der Daten verwendet wird. Die Standardeinstellung ist Ascending.
max_iterations	Ganzzahl	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden. Eine nicht negative Ganzzahl. Der Standardwert ist 100.
confidence_level	Ganzzahl	Konfidenzniveau für die Berechnung von Intervallschätzungen der Modellkoeffizienten. Eine nicht negative Ganzzahl. Der Maximalwert ist 100 und der Standardwert ist 95.
degrees_of_freedom_method	Fixed Varied	Gibt an, wie Freiheitsgrade für Signifikanztests berechnet werden.
test_fixed_effects_coeffecients	Modell Robust	Methode zur Berechnung der Kovarianzmatrix für Parameterschätzungen.
use_p_converge	Flag	Option für Parameterkonvergenz.
p_converge	Zahl	Leerzeichen oder ein beliebiger positiver Wert.
p_converge_type	Absolute Relative	
use_l_converge	Flag	Option für Log-Likelihood-Konvergenz.
l_converge	Zahl	Leerzeichen oder ein beliebiger positiver Wert.
l_converge_type	Absolute Relative	
use_h_converge	Flag	Option für Konvergenz der Hesse-Matrix.
h_converge	Zahl	Leerzeichen oder ein beliebiger positiver Wert.
h_converge_type	Absolute Relative	
max_fisher_steps	Ganzzahl	
singularity_tolerance	Zahl	
use_model_name	Flag	Gibt an, ob ein benutzerdefinierter Name (true) oder ein vom System generierter Name (false) für das Modell verwendet werden soll. Die Standardeinstellung ist false.

Tabelle 117. Eigenschaften von "glimmnode" (Forts.).

Eigenschaften von glimmnode	Werte	Eigenschaftsbeschreibung
model_name	Zeichenfolge	Gibt den zu verwendenden Modellnamen an, wenn use_model_name auf true gesetzt ist.
confidence	onProbability onIncrease	Grundlage für die Berechnung des Konfidenzwerts für das Scoring: höchste vorhergesagte Wahrscheinlichkeit oder Differenz zwischen der höchsten und zweithöchsten vorhergesagten Wahrscheinlichkeit.
score_category_probabilities	Flag	Auf true gesetzt, werden vorhergesagte Wahrscheinlichkeiten für kategoriale Ziele generiert. Die Standardeinstellung ist false.
max_categories	Ganzzahl	Wenn score_category_probabilities auf true gesetzt ist, wird hier die maximale Anzahl der zu speichernden Kategorien festgelegt.
score_propensity	Flag	Auf true gesetzt, werden Propensity-Scores für Flagzielfelder generiert, die die Wahrscheinlichkeit angeben, mit der das Feld den Wert "true" haben wird.
emeans	structure	Für jedes kategoriale Feld aus der Liste mit festen Effekten wird hier angegeben, ob geschätzte Randmittel generiert werden sollen.
covariance_list	structure	Für jedes stetige Feld aus der Liste mit festen Effekten wird hier angegeben, ob der Mittelwert oder ein benutzerdefinierter Wert für die Berechnung der geschätzten Randmittel verwendet werden soll.
mean_scale	Original Transformed	Gibt an, ob geschätzte Randmittel anhand der ursprünglichen Skala des Ziels (Standard) oder anhand der Transformation der Verknüpfungsfunktion berechnet werden sollen.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Zu verwendende Anpassungsmethode bei Hypothesentests mit mehreren Kontrasten.

Eigenschaften von "kmeansnode"



Der K-Means-Knoten teilt das Dataset in unterschiedliche Gruppen (oder Cluster) auf. Bei diesem Verfahren wird eine festgelegte Anzahl von Clustern definiert, den Clustern werden iterativ Datensätze zugewiesen und die Clusterzentren werden angepasst, bis eine weitere Verfeinerung keine wesentliche Verbesserung des Modells mehr darstellen würde. Statt zu versuchen, ein Ergebnis vorherzusagen, versucht K-Means mithilfe eines als "nicht überwachtetes Lernen" bezeichneten Verfahrens Muster im Set der Eingabefelder zu entdecken.

Beispiel

```
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)

```

Tabelle 118. Eigenschaften von "kmeansnode"

Eigenschaften von kmeansnode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	K-Means-Modelle führen eine Clusteranalyse an einer Menge von Eingabefeldern durch, verwenden jedoch kein Zielfeld. Gewichtung- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
num_clusters	Zahl	
gen_distance	Flag	
cluster_label	String Zahl	
label_prefix	Zeichenfolge	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	Zahl	
tolerance	Zahl	
encoding_value	Zahl	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.

Eigenschaften von "knnnode"



Der Knoten "k-Nächste Nachbarn" (KNN) verknüpft einen neuen Fall mit der Kategorie oder dem Wert der k Objekte, die ihm im Prädiktorraum am nächsten liegen, wobei k eine Ganzzahl ist. Ähnliche Fälle liegen nah beieinander und Fälle mit geringer Ähnlichkeit sind weit voneinander entfernt.

Beispiel

```

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)

```

Tabelle 119. Eigenschaften von "knnnode"

Eigenschaften von knnnode	Werte	Eigenschaftsbeschreibung
Analyse	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	Flag	
use_case_labels	Flag	Kontrollkästchen markieren, um nächste Option zu aktivieren.
case_labels_field	Feld	
identify_focal_cases	Flag	Kontrollkästchen markieren, um nächste Option zu aktivieren.
focal_cases_field	Feld	
automatic_k_selection	Flag	
fixed_k	Ganzzahl	Nur aktiviert, wenn automatic_k_selection auf False eingestellt ist.
minimum_k	Ganzzahl	Nur aktiviert, wenn automatic_k_selection auf True eingestellt ist.
maximum_k	Ganzzahl	
distance_computation	Euclidean CityBlock	
weight_by_importance	Flag	
range_predictions	Mean Median	
perform_feature_selection	Flag	
forced_entry_inputs	[Feld1 ... FeldN]	
stop_on_error_ratio	Flag	
Zahl_to_select	Ganzzahl	
minimum_change	Zahl	
validation_fold_assign_by_field	Flag	
number_of_folds	Ganzzahl	Nur aktiviert, wenn validation_fold_assign_by_field auf False eingestellt ist.
set_random_seed	Flag	
random_seed	Zahl	

Tabelle 119. Eigenschaften von "knnnode" (Forts.)

Eigenschaften von knnnode	Werte	Eigenschaftsbeschreibung
fold_field	Feld	Nur aktiviert, wenn validation_fold_assign_by_field auf True eingestellt ist.
all_probabilities	Flag	
save_distances	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "kohonennode"



Der Kohonen-Knoten erstellt eine Art von neuronalem Netz, das verwendet werden kann, um ein Clustering des Datensets in einzelne Gruppen vorzunehmen. Wenn das Netz voll trainiert ist, sollten ähnliche Datensätze auf der Ausgabekarte eng nebeneinander stehen, während Datensätze, die sich unterscheiden, weit voneinander entfernt sein sollten. Die Zahl der von jeder Einheit im Modellnugget erfassten Beobachtungen gibt Aufschluss über die starken Einheiten. Dadurch wird ein Eindruck von der ungefähren Zahl der Cluster vermittelt.

Beispiel

```
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)
```

Tabelle 120. Eigenschaften von "kohonennode"

Eigenschaften von kohonennode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	Kohonen-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue	Flag	

Tabelle 120. Eigenschaften von "kohonenode" (Forts.)

Eigenschaften von kohonenode	Werte	Eigenschaftsbeschreibung
show_feedback	Flag	
stop_on	Default Zeit	
time	Zahl	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
cluster_label	Flag	
mode	Simple Expert	
width	Zahl	
length	Zahl	
decay_style	Linear Exponentiell	
phase1_neighborhood	Zahl	
phase1_eta	Zahl	
phase1_cycles	Zahl	
phase2_neighborhood	Zahl	
phase2_eta	Zahl	
phase2_cycles	Zahl	

Eigenschaften von "linearnode"



Bei linearen Regressionsmodellen wird ein stetiges Ziel auf der Basis linearer Beziehungen zwischen dem Ziel und einem oder mehreren Prädiktoren vorhergesagt.

Beispiel

```
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")
```

Tabelle 121. Eigenschaften von "linearnode".

Eigenschaften von linearnode	Werte	Eigenschaftsbeschreibung
target	Feld	Gibt ein einzelnes Zielfeld an.
inputs	[Feld1 ... FeldN]	Im Modell verwendete Prädiktorfelder.
continue_training_existing_model	Flag	

Tabelle 121. Eigenschaften von "linearnode" (Forts.).

Eigenschaften von linearnode	Werte	Eigenschaftsbeschreibung
objective	Standard Bagging Boosting psm	PSM wird für sehr umfangreiche Datensets verwendet und erfordert eine Serververbindung.
use_auto_data_preparation	Flag	
confidence_level	Zahl	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	Zahl	
probability_removal	Zahl	
use_max_effects	Flag	
max_effects	Zahl	
use_max_steps	Flag	
max_steps	Zahl	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	Zahl	
use_random_seed	Flag	
random_seed	Zahl	
use_custom_model_name	Flag	
custom_model_name	Zeichenfolge	
use_custom_name	Flag	
custom_name	Zeichenfolge	
tooltip	Zeichenfolge	
keywords	Zeichenfolge	
annotation	Zeichenfolge	

Eigenschaften von "logregnode"



Die logistische Regression ist ein statistisches Verfahren zur Klassifizierung von Datensätzen auf der Grundlage der Werte von Eingabefeldern. Sie ist analog zur linearen Regression, außer dass statt eines numerischen Bereichs ein kategoriales Zielfeld verwendet wird.

Beispiel für ein multinomiales Modell

```

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")

```

Beispiel für ein binomiales Modell

```

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)

```

Tabelle 122. Eigenschaften von "logregnode".

Eigenschaften von logregnode	Werte	Eigenschaftsbeschreibung
target	<i>Feld</i>	Logistische Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
logistic_procedure	Binomial Multinomial	
include_constant	<i>Flag</i>	
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	

Tabelle 122. Eigenschaften von "logregnode" (Forts.).

Eigenschaften von logregnode	Werte	Eigenschaftsbeschreibung
model_type	MainEffects FullFactorial Custom	Wenn als Modelltyp FullFactorial festgelegt ist, werden keine Schrittmethoden ausgeführt, auch wenn diese ebenfalls angegeben wurden. Stattdessen wird die Methode Enter verwendet. Wenn der Modelltyp auf Custom gesetzt ist, jedoch keine benutzerdefinierten Felder angegeben wurden, wird ein Haupteffektmodell erstellt.
custom_terms	[{BD Geschlecht}{BD}{Alter}]	
multinomial_base_category	Zeichenfolge	Gibt an, wie die Referenzkategorie bestimmt wird.
binomial_categorical_input	Zeichenfolge	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Verschlüsselte Eigenschaft für kategoriale Eingaben, die angibt, wie der Kontrast bestimmt wird.
binomial_input_category	First Last	Verschlüsselte Eigenschaft für kategoriale Eingaben, die angibt, wie die Referenzkategorie bestimmt wird.
scale	None UserDefined Pearson Deviance	
scale_value	Zahl	
all_probabilities	Flag	
tolerance	1,0E-5 1,0E-6 1,0E-7 1,0E-8 1,0E-9 1,0E-10	
min_terms	Zahl	
use_max_terms	Flag	
max_terms	Zahl	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	Zahl	
probability_removal	Zahl	
binomial_probability_entry	Zahl	
binomial_probability_removal	Zahl	

Tabelle 122. Eigenschaften von "logregnode" (Forts.).

Eigenschaften von logregnode	Werte	Eigenschaftsbeschreibung
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	Zahl	
max_steps	Zahl	
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	Zahl	
iteration_history	Flag	
history_steps	Zahl	
summary	Flag	
likelihood_ratio	Flag	
asymptotic_correlation	Flag	
goodness_fit	Flag	
parameters	Flag	
confidence_interval	Zahl	
asymptotic_covariance	Flag	
classification_table	Flag	
stepwise_summary	Flag	
info_criteria	Flag	
monotonicity_measures	Flag	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	Flag	
binomial_parameters	Flag	
binomial_iteration_history	Flag	
binomial_classification_plots	Flag	
binomial_ci_enable	Flag	
binomial_ci	Zahl	
binomial_residual	outliers all	
binomial_residual_enable	Flag	
binomial_outlier_threshold	Zahl	
binomial_classification_cutoff	Zahl	

Tabelle 122. Eigenschaften von "logregnode" (Forts.).

Eigenschaften von logregnode	Werte	Eigenschaftsbeschreibung
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	

Eigenschaften von "neuralnetnode"

Vorsicht: In dieser Version ist eine neuere Fassung des Netzmodellierungsknotens mit erweiterten Funktionen verfügbar, die im nächsten Abschnitt beschrieben wird (*neuralnetwork*). Sie können zwar auch weiterhin Modelle mit der Vorgängerversion erstellen und scoren, doch empfehlen wir die Aktualisierung Ihrer Scripts zur Verwendung der neuen Version. Details der vorherigen Version werden hier aus Referenzgründen aufbewahrt.

Beispiel

```
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)
```

Tabelle 123. Eigenschaften von "neuralnetnode"

Eigenschaften von neuralnetnode	Werte	Eigenschaftsbeschreibung
targets	[Feld1 ... FeldN]	Der Netzknoten erwartet mindestens ein Zielfeld und mindestens ein Eingabefeld. Häufigkeits- und Gewichtungsfelder werden ignoriert. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	Flag	
train_pct	Zahl	

Tabelle 123. Eigenschaften von "neuralnetnode" (Forts.)

Eigenschaften von neuralnetnode	Werte	Eigenschaftsbeschreibung
set_random_seed	Flag	
random_seed	Zahl	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Zeit	Stoppmodus.
Genauigkeit	Zahl	Stoppgenauigkeit.
cycles	Zahl	Zu trainierende Zyklen.
time	Zahl	Dauer der Trainingsphase (Minuten)
continue	Flag	
show_feedback	Flag	
binary_encode	Flag	
use_last_model	Flag	
gen_logfile	Flag	
logfile_name	Zeichenfolge	
alpha	Zahl	
initial_eta	Zahl	
high_eta	Zahl	
low_eta	Zahl	
eta_decay_cycles	Zahl	
hid_layers	One Two Three	
h1_units_one	Zahl	
h1_units_two	Zahl	
h1_units_three	Zahl	
Persistenz	Zahl	
m_topologies	Zeichenfolge	
m_non_pyramids	Flag	
m_persistence	Zahl	
p_hid_layers	One Two Three	
p_h1_units_one	Zahl	
p_h1_units_two	Zahl	
p_h1_units_three	Zahl	
p_persistence	Zahl	
p_hid_rate	Zahl	
p_hid_pers	Zahl	
p_inp_rate	Zahl	

Tabelle 123. Eigenschaften von "neuralnetnode" (Forts.)

Eigenschaften von neuralnetnode	Werte	Eigenschaftsbeschreibung
p_inp_pers	Zahl	
p_overall_pers	Zahl	
r_persistence	Zahl	
r_num_clusters	Zahl	
r_eta_auto	Flag	
r_alpha	Zahl	
r_eta	Zahl	
optimize	Speed Memory	Dient zur Angabe, ob die Modellerstellung in Bezug auf die Geschwindigkeit oder den Speicher optimiert werden soll.
calculate_variable_importance	Flag	Hinweis: Die in früheren Versionen verwendete Eigenschaft <code>sensitivity_analysis</code> wird zugunsten dieser Eigenschaft nicht mehr verwendet. Die alte Eigenschaft wird weiterhin unterstützt, es wird jedoch <code>calculate_variable_importance</code> empfohlen.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "neuralnetworknode"



Der Netzknoten verwendet ein vereinfachtes Modell der Art und Weise, wie ein menschliches Gehirn Informationen verarbeitet. Es funktioniert, indem eine große Anzahl miteinander verbundener einfacher Verarbeitungseinheiten simuliert wird, die abstrakten Versionen von Neuronen ähnlich sind. Neuronale Netze sind leistungsstarke Mehrzweckschätzer, für deren Training und Anwendung nur sehr geringe statistische oder mathematische Kenntnisse erforderlich sind.

Beispiel

```
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")
```

Tabelle 124. Eigenschaften von "neuralnetworknode"

Eigenschaften von neuralnetworknode	Werte	Eigenschaftsbeschreibung
targets	[Feld1 ... FeldN]	Gibt die Zielfelder an.
inputs	[Feld1 ... FeldN]	Im Modell verwendete Prädiktorfelder.
splits	[Feld1 ... FeldN]	Gibt das Feld bzw. die Felder für die Aufteilungsmodellierung an.

Tabelle 124. Eigenschaften von "neuralnetworknode" (Forts.)

Eigenschaften von neuralnetworknode	Werte	Eigenschaftsbeschreibung
use_partition	Flag	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
continue	Flag	Training des bestehenden Modells fortsetzen.
objective	Standard Bagging Boosting PSM	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	Flag	
first_layer_units	Zahl	
second_layer_units	Zahl	
use_max_time	Flag	
max_time	Zahl	
use_max_cycles	Flag	
max_cycles	Zahl	
use_min_accuracy	Flag	
min_accuracy	Zahl	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	Zahl	
overfit_prevention_pct	Zahl	
use_random_seed	Flag	
random_seed	Zahl	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	boolesch	
model_name	Zeichenfolge	
Konfidenz	onProbability onIncrease	
score_category_probabilities	Flag	
max_categories	Zahl	
score_propensity	Flag	
use_custom_name	Flag	
custom_name	Zeichenfolge	
tooltip	Zeichenfolge	

Tabelle 124. Eigenschaften von "neuralnetworknode" (Forts.)

Eigenschaften von neuralnetworknode	Werte	Eigenschaftsbeschreibung
Schlüsselwörter	Zeichenfolge	
annotation	Zeichenfolge	

Eigenschaften von "questnode"



Der QUEST-Knoten bietet eine binäre Klassifizierungsmethode zum Erstellen von Entscheidungsbäumen, die dazu dient, die für große C&R-Baumanalysen erforderliche Verarbeitungszeit zu verkürzen. Gleichzeitig soll die in den Klassifizierungsbaummodellen festgestellte Tendenz verringert werden, die darin besteht, dass Eingaben bevorzugt werden, die mehr Aufteilungen erlauben. Eingabefelder können stetig (numerische Bereiche) sein, das Zielfeld muss aber kategorial sein. Alle Aufteilungen sind binär.

Beispiel

```
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)
```

Tabelle 125. Eigenschaften von "questnode"

Eigenschaften von questnode	Werte	Eigenschaftsbeschreibung
target	Feld	QUEST-Modelle erfordern ein einzelnes Ziel und eines oder mehrere Eingabefelder. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue_training_existing_model	Flag	
objective	Standard Boosting Bagging PSM	PSM wird für sehr umfangreiche Datasets verwendet und erfordert eine Serververbindung.
model_output_type	Single InteractiveBuilder	
use_tree_directives	Flag	
tree_directives	Zeichenfolge	
use_max_depth	Default Custom	

Tabelle 125. Eigenschaften von "questnode" (Forts.)

Eigenschaften von questnode	Werte	Eigenschaftsbeschreibung
max_depth	Ganzzahl	Maximale Baumtiefe, von 0 bis 1000. Wird nur verwendet, wenn use_max_depth = Custom.
prune_tree	Flag	Baum reduzieren, um zu große Anpassung zu vermeiden.
use_std_err	Flag	Maximale Risikendifferenz verwenden (in Standardfehler).
std_err_multiplier	Zahl	Maximale Differenz.
max_surrogates	Zahl	Maximale Anzahl Ersatztrenner.
use_percentage	Flag	
min_parent_records_pc	Zahl	
min_child_records_pc	Zahl	
min_parent_records_abs	Zahl	
min_child_records_abs	Zahl	
use_costs	Flag	
Kosten	strukturiert	Strukturierte Eigenschaft.
priors	Data Equal Custom	
custom_priors	strukturiert	Strukturierte Eigenschaft.
adjust_priors	Flag	
trails	Zahl	Anzahl der Komponentenmodelle für Boosting oder Bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Standardkombinationsregel für kategoriale Ziele.
range_ensemble_method	Mean Median	Standardkombinationsregel für stetige Ziele.
large_boost	Flag	Boosting auf sehr große Datasets anwenden.
split_alpha	Zahl	Signifikanzschwelle für Aufteilung.
train_pct	Zahl	Set zur Verhinderung übermäßiger Anpassung.
set_random_seed	Flag	Option "Ergebnisse replizieren".
seed	Zahl	
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "regressionnode"



Die lineare Regression ist ein statistisches Verfahren zur Zusammenfassung von Daten und die Erstellung von Vorhersagen durch Anpassung einer geraden Linie oder Fläche, mit der die Diskrepanzen zwischen den vorhergesagten und den tatsächlichen Ausgabewerten minimiert werden.

Anmerkung: Der Regressionsknoten wird in einer zukünftigen Version durch den Linearknoten ersetzt. Es wird empfohlen, dass Sie von nun an lineare Modelle für lineare Regression verwenden.

Beispiel

```
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)
```

Tabelle 126. Eigenschaften von "regressionnode"

Eigenschaften von regressionnode	Werte	Eigenschaftsbeschreibung
target	Feld	Regressionsmodelle erfordern ein einzelnes Zielfeld und eines oder mehrere Eingabefelder. Außerdem kann ein Gewichtungsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.

Tabelle 126. Eigenschaften von "regressionnode" (Forts.)

Eigenschaften von regressionnode	Werte	Eigenschaftsbeschreibung
method	Enter Stepwise Backwards Forwards	
include_constant	Flag	
use_weight	Flag	
weight_field	Feld	
mode	Simple Expert	
complete_records	Flag	
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	Verwenden Sie für Argumente doppelte Anführungszeichen.
stepping_method	useP useF	useP: F-Wahrscheinlichkeit verwenden useF: F-Wert verwenden
probability_entry	Zahl	
probability_removal	Zahl	
F_value_entry	Zahl	
F_value_removal	Zahl	
selection_criteria	Flag	
confidence_interval	Flag	
covariance_matrix	Flag	
collinearity_diagnostics	Flag	
regression_coefficients	Flag	
exclude_fields	Flag	
durbin_watson	Flag	
model_fit	Flag	
r_squared_change	Flag	
p_correlations	Flag	
deskriptive Statistiken	Flag	
calculate_variable_importance	Flag	

Eigenschaften von "sequencenode"



Der Sequenzknoten erkennt Assoziationsregeln in sequenziellen oder zeitorientierten Daten. Eine Sequenz ist eine Liste mit Elementsets, die in einer vorhersagbaren Reihenfolge auftreten. Beispiel: Ein Kunde, der einen Rasierer und After-Shave-Lotion kauft, kauft möglicherweise beim nächsten Einkauf Rasiercreme. Der Sequenzknoten basiert auf dem CARMA-Assoziationsregelalgorithmus, der eine effiziente bidirektionale Methode zum Suchen von Sequenzen verwendet.

Beispiel

```
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)
```

Tabelle 127. Eigenschaften von "sequencenode"

Eigenschaften von sequencenode	Werte	Eigenschaftsbeschreibung
id_field	Feld	Um ein Sequenzmodell zu erstellen, müssen Sie ein ID-Feld, ein optionales Zeitfeld und mindestens ein Inhaltsfeld angeben. Gewichtung- und Häufigkeitsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
time_field	Feld	
use_time_field	Flag	
content_fields	[Feld1 ... FeldN]	
contiguous	Flag	
min_supp	Zahl	
min_conf	Zahl	

Tabelle 127. Eigenschaften von "sequencenode" (Forts.)

Eigenschaften von sequencenode	Werte	Eigenschaftsbeschreibung
max_size	Zahl	
max_predictions	Zahl	
mode	Simple Expert	
use_max_duration	Flag	
max_duration	Zahl	
use_gaps	Flag	
min_item_gap	Zahl	
max_item_gap	Zahl	
use_pruning	Flag	
pruning_value	Zahl	
set_mem_sequences	Flag	
mem_sequences	Ganzzahl	

Eigenschaften von "slrmnode"



Mithilfe des Knotens für das lernfähige Antwortmodell (Self-Learning Response Model, SLRM) können Sie ein Modell erstellen, in dem das Modell anhand eines einzelnen neuen Falls oder einer kleinen Anzahl neuer Fälle neu eingeschätzt werden kann, ohne dass das Modell mit allen Daten neu trainiert werden muss.

Beispiel

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tabelle 128. Eigenschaften von "slrmnode"

Eigenschaften von slrmnode	Werte	Eigenschaftsbeschreibung
target	Feld	Beim Zielfeld muss es sich um ein nominales oder ein Flagfeld handeln. Außerdem kann ein Häufigkeitsfeld angegeben werden. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
target_response	Feld	Der Typ muss "Flag" sein.
continue_training_existing_model	Flag	
target_field_values	Flag	Alle verwenden: Alle Werte aus der Quelle verwenden. Angaben: Erforderliche Werte auswählen.
target_field_values_specify	[Feld1 ... FeldN]	
include_model_assessment	Flag	
model_assessment_random_seed	Zahl	Muss eine reelle Zahl sein.
model_assessment_sample_size	Zahl	Muss eine reelle Zahl sein.

Tabelle 128. Eigenschaften von "slrmnode" (Forts.)

Eigenschaften von slrmnode	Werte	Eigenschaftsbeschreibung
model_assessment_iterations	Zahl	Anzahl der Iterationen.
display_model_evaluation	Flag	
max_predictions	Zahl	
randomization	Zahl	
scoring_random_seed	Zahl	
sort	Ascending Descending	Gibt an, ob die Angebote mit den höchsten oder die mit den niedrigsten Scores zuerst angezeigt werden.
model_reliability	Flag	
calculate_variable_importance	Flag	

Eigenschaften von "statisticsmodelnode"



Mithilfe des Statistics-Modellknotens können Sie Ihre Daten analysieren und bearbeiten, indem Sie IBM SPSS Statistics-Prozeduren ausführen, die PMML erzeugen. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften von "statisticsmodelnode"“ auf Seite 308.

Eigenschaften von "stpnode"



Der STP-Knoten (Spatio-Temporal Prediction - räumliche temporale Vorhersage) verwendet Daten, die Positionsdaten, Eingabefelder für Vorhersage (Prädiktoren), ein Zeitfeld und ein Zielfeld enthalten. Die Daten enthalten für jede Position zahlreiche Zeilen, die die Werte der einzelnen Prädiktoren zum Zeitpunkt der Messung darstellen. Mit den Daten können nach ihrer Analyse Zielwerte an jeder Position in den Shapedaten, die in der Analyse verwendet werden, vorhergesagt werden.

Tabelle 129. Eigenschaften von "stpnode"

Eigenschaften von stpnode	Datentyp	Eigenschaftsbeschreibung
Registerkarte Felder		
target	Feld	Das Zielfeld.
location	Feld	Das Feld für die Position des Modells. Nur georäumliche Felder sind zulässig.
location_label	Feld	Das in der Ausgabe zu verwendende kategoriale Feld, um die in location ausgewählten Positionen zu beschriften.
time_field	Feld	Das Zeitfeld für das Modell. Nur Felder mit fortlaufender Messung sind zulässig und der Speichertyp muss 'Zeit', 'Datum', 'Zeitmarke' oder 'Ganzzahl' sein.

Tabelle 129. Eigenschaften von "stptime" (Forts.)

Eigenschaften von stptime	Datentyp	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	Eine Liste von Eingabefeldern.
Registerkarte Zeitintervalle		
interval_type_timestamp	Years Quarters Months Weeks Days Hours Minutes Seconds	
interval_type_date	Years Quarters Months Weeks Days	
interval_type_time	Hours Minutes Seconds	Begrenzt die Anzahl Tage pro Woche, die berücksichtigt werden, wenn der Zeitindex erstellt wird, den STP für die Berechnung verwendet.
interval_type_integer	Periods (nur Zeitindexfelder, Ganzzahlspeicherung)	Das Intervall, in das das Dataset umgewandelt wird. Die verfügbare Auswahl hängt vom Speichertyp des Felds ab, das als time_field für das Modell ausgewählt wird.
period_start	Ganzzahl	
start_month	Januar Februar März April Mai Juni Juli August September Oktober November Dezember	Der Monat, ab dem das Modell die Indexierung startet. Wenn z. B. März angegeben ist, der erste Datensatz im Dataset jedoch Januar ist, überspringt das Modell die ersten beiden Datensätze und startet die Indexierung im März.
week_begins_on	Sonntag Montag Dienstag Mittwoch Donnerstag Freitag Samstag	Der Startpunkt für den Zeitindex, der von STP aus den Daten erstellt wird.
days_per_week	Ganzzahl	Minimum: 1, Maximum: 7, in Inkrementen von 1.

Tabelle 129. Eigenschaften von "stpnode" (Forts.)

Eigenschaften von stpnode	Datentyp	Eigenschaftsbeschreibung
hours_per_day	Ganzzahl	Die Anzahl Stunden, die das Modell in einem Tag berücksichtigt. Bei der Angabe 10 startet das Modell die Indexierung zur durch day_begins_at angegebenen Uhrzeit und setzt die Indexierung 10 Stunden fort, springt dann zum nächsten Wert, der mit dem Wert von day_begins_at übereinstimmt, usw.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Legt den den Stundenwert fest, ab dem das Modell die Indexierung startet.
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Diese Inkrementeneinstellung gilt für Minuten oder Sekunden. Sie legt fest, wo das Modell aus den Daten Indizes erstellt. Bei einem Inkrement von 30 und beim Intervalltyp seconds erstellt das Modell alle 30 Sekunden aus den Daten einen Index.
data_matches_interval	boolesch	<p>Wird diese Eigenschaft auf N gesetzt, tritt die Konvertierung der Daten in den regulären Intervalltyp (interval_type) auf, bevor das Modell erstellt wird.</p> <p>Wenn Ihre Daten bereits im richtigen Format vorliegen und wenn interval_type und zugehörige Einstellungen mit Ihren Daten übereinstimmen, setzen Sie diese Einstellung auf Y, um die Konvertierung oder Aggregation Ihrer Daten zu verhindern.</p> <p>Durch das Setzen dieser Einstellung auf Y werden alle Aggregationssteuerelemente inaktiviert.</p>
agg_range_default	Sum Mean Min Max Median 1stQuartile 3rdQuartile	Legt die Standardaggregationsmethode fest, die für stetige Felder verwendet wird. Alle stetigen Felder, die nicht spezifisch in die angepasste Aggregation eingeschlossen werden, werden mit der hier angegebenen Methode aggregiert.

Tabelle 129. Eigenschaften von "stpnode" (Forts.)

Eigenschaften von stpnode	Datentyp	Eigenschaftsbeschreibung
agg_set_default	Mode Min Max	Entspricht agg_range_default, jedoch für nominale/kategoriale Felder.
agg_flag_default	TrueIfAnyTrue FalseIfAnyFalse	Die Standardaggregationsmethode, die auf alle stetigen Felder angewendet werden soll, die nicht einzeln angegeben werden.
custom_agg	<pre>[{field, aggregation method},{}.]</pre> <p>Demo: <pre>[{'x5' 'FirstQuartile'}{'x4' 'Sum'}]</pre> </p>	<p>Strukturierte Eigenschaft:</p> <p>Scriptparameter: custom_agg</p> <p>Beispiel: <pre>set :stpnode.custom_agg = [{field1 function} {field2 function}]</pre> </p> <p>Dabei ist function die Aggregationsfunktion, die mit diesem Feld verwendet werden soll.</p>
Registerkarte Basis		
include_intercept	Flag	
max_autoregressive_lag	Ganzzahl	Minimum: 1, Maximum: 5, in Inkrementen von 1. Die Anzahl vorheriger Datensätze, die für eine Vorhersage erforderlich sind. Wird z. B. 5 festgelegt, wird aus den vorherigen 5 Datensätzen eine neue Vorhersage erstellt. Die Anzahl Datensätze, die hier aus den Erstellungsdaten angegeben werden, werden in das Modell aufgenommen und daher braucht der Benutzer die Daten nicht erneut anzugeben, wenn für das Modell Scoring durchgeführt wird.
estimation_method	Parametric Nonparametric	Die Methode für das Modellieren der räumlichen Kovarianzmatrix.
parametric_model	Gaussian Exponential PoweredExponential	Reihenfolgeparameter für das räumliche Kovarianzmodell Parametric.
exponential_power	Zahl	Potenzstufe für das Modell PoweredExponential. Minimum: 1, Maximum: 2.
Registerkarte Erweitert		
max_missing_values	Ganzzahl	Der maximale Prozentsatz von Datensätzen mit fehlenden Werten, die im Modell zulässig sind.

Tabelle 129. Eigenschaften von "stptime" (Forts.)

Eigenschaften von stptime	Datentyp	Eigenschaftsbeschreibung
significance	Zahl	Das Signifikanzniveau für Hypothesentests in der Modellerstellung. Gibt den Signifikanzwert für alle Tests in der STP-Modellschätzung an, einschließlich zwei Anpassungsgütetests, F-Tests für Effekte und T-Tests für Koeffizienten.
Registerkarte Ausgabe		
model_specifications	Flag	
temporal_summary	Flag	
location_summary	Flag	Legt fest, ob die Positionszusammenfassungstabelle in die Modellausgabe aufgenommen wird.
model_quality	Flag	
test_mean_structure	Flag	
mean_structure_coefficients	Flag	
autoregressive_coefficients	Flag	
test_decay_space	Flag	
parametric_spatial_covariance	Flag	
correlations_heat_map	Flag	
correlations_map	Flag	
location_clusters	Flag	
similarity_threshold	Zahl	Der Schwellenwert, an dem Ausgabecluster als ähnlich genug betrachtet werden, um in einen einzelnen Cluster zusammengeführt werden zu können.
max_number_clusters	Ganzzahl	Die Obergrenze für die Anzahl Cluster, die in die Modellausgabe eingeschlossen werden können.
Registerkarte Modelloptionen		
use_model_name	Flag	
model_name	Zeichenfolge	
uncertainty_factor	Zahl	Minimum: 0, Maximum: 100. Legt die Unsicherheitszunahme (Fehlerzunahme) fest, die zukünftig auf Vorhersagen angewendet wird. Es handelt sich um die Ober- und Untergrenze für die Vorhersagen.

Eigenschaften von "svmnode"



Der Knoten "Support Vector Machine" (SVM) ermöglicht die Klassifizierung von Daten in eine von zwei Gruppen ohne Überanpassung. SVM eignet sich gut für umfangreiche Datensets, beispielsweise solche mit einer großen Anzahl an Eingabefeldern.

Beispiel

```
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)
```

Tabelle 130. Eigenschaften von "svmnode".

Eigenschaften von svmnode	Werte	Eigenschaftsbeschreibung
all_probabilities	Flag	
stopping_criteria	1,0E-1 1,0E-2 1,0E-3 (Standard) 1,0E-4 1,0E-5 1,0E-6	Bestimmt, wann der Optimierungsalgorithmus gestoppt werden soll.
regularization	Zahl	Auch als C-Parameter bekannt.
precision	Zahl	Nur verwendet, wenn es sich beim Messniveau des Zielfelds um Continuous (Stetig) handelt.
kernel	RBF(Standard) Polynomial Sigmoid Linear	Typ der für die Transformation verwendeten Kernfunktion.
rbf_gamma	Zahl	Nur verwendet, wenn für kernel der Typ RBF gilt.
gamma	Zahl	Nur verwendet, wenn für kernel der Typ Polynomial oder Sigmoid gilt.
bias	Zahl	
degree	Zahl	Nur verwendet, wenn für kernel der Typ Polynomial gilt.
calculate_variable_importance	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	
adjusted_propensity_partition	Test Validation	

Eigenschaften von "tcmnode"

Temporale kausale Modellierung versucht, kausale Beziehungen in Zeitreihendaten aufzuspüren. Bei temporaler kausaler Modellierung geben Sie ein Set von Zielzeitreihen und ein Set von Kandidateneingaben in diese Ziele an. Die Prozedur erstellt dann ein autoregressives Zeitreihenmodell für jedes Ziel und schließt nur jene Eingaben ein, die die signifikantesten kausalen Beziehungen mit dem Ziel haben.

Tabelle 131. Eigenschaften von "tcmnode"

Eigenschaften von tcmnode	Werte	Eigenschaftsbeschreibung
custom_fields	<i>boolesch</i>	
dimensionlist	[<i>Dimension1 ... DimensionN</i>]	
data_struct	Multiple Single	
metric_fields	<i>Felder</i>	
both_target_and_input	[<i>f1 ... fN</i>]	
targets	[<i>f1 ... fN</i>]	
candidate_inputs	[<i>f1 ... fN</i>]	
forced_inputs	[<i>f1 ... fN</i>]	
use_timestamp	Timestamp Period	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	<i>Zeichenfolge</i>	
period_start_value	<i>Ganzzahl</i>	
num_days_per_week	<i>Ganzzahl</i>	
start_day_of_week	Sunday Monday Tuesday Wednesday Donnerstag Friday Saturday	
num_hours_per_day	<i>Ganzzahl</i>	
start_hour_of_day	<i>Ganzzahl</i>	
timestamp_increments	<i>Ganzzahl</i>	
cyclic_increments	<i>Ganzzahl</i>	
cyclic_periods	<i>Liste</i>	

Tabelle 131. Eigenschaften von "tcmnode" (Forts.)

Eigenschaften von tcmnode	Werte	Eigenschaftsbeschreibung
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	Same Notsame	
cross_hour	<i>boolesch</i>	
aggregate_and_distribute	<i>Liste</i>	
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_meridian Linear_trend None	
k_mean_param	<i>Ganzzahl</i>	
k_median_param	<i>Ganzzahl</i>	
missing_value_threshold	<i>Ganzzahl</i>	
conf_level	<i>Ganzzahl</i>	
max_num_predictor	<i>Ganzzahl</i>	
max_lag	<i>Ganzzahl</i>	
Epsilon	<i>Zahl</i>	
threshold	<i>Ganzzahl</i>	
is_re_est	<i>boolesch</i>	
num_targets	<i>Ganzzahl</i>	
percent_targets	Rootmean Bic Rsquare	
fields_display	<i>Liste</i>	
series_display	<i>Liste</i>	
network_graph_for_target	<i>boolesch</i>	
sign_level_for_target	<i>Zahl</i>	

Tabelle 131. Eigenschaften von "tcmnode" (Forts.)

Eigenschaften von tcmnode	Werte	Eigenschaftsbeschreibung
fit_and_outlier_for_target	boolesch	
sum_and_para_for_target	boolesch	
impact_diag_for_target	boolesch	
impact_diag_type_for_target	Effect Cause Both	
impact_diag_level_for_target	Ganzzahl	
series_plot_for_target	boolesch	
res_plot_for_target	boolesch	
top_input_for_target	boolesch	
forecast_table_for_target	boolesch	
same_as_for_target	boolesch	
network_graph_for_series	boolesch	
sign_level_for_series	Zahl	
fit_and_outlier_for_series	boolesch	
sum_and_para_for_series	boolesch	
impact_diagram_for_series	boolesch	
impact_diagram_type_for_series	Effect Cause Both	
impact_diagram_level_for_series	Ganzzahl	
series_plot_for_series	boolesch	
residual_plot_for_series	boolesch	
forecast_table_for_series	boolesch	
outlier_root_cause_analysis	boolesch	
causal_levels	Ganzzahl	
root_cause_analysis_method	First All	
outlier_table	Interactive Pivot Both	
rmsep_error	boolesch	
norm_bic	boolesch	
r_square	boolesch	
outliers_over_time	boolesch	
series_transormation	boolesch	
use_estimation_period	boolesch	
estimation_period	Times Observation	
observations	Liste	
observations_type	Latest Earliest	
observations_num	Ganzzahl	

Tabelle 131. Eigenschaften von "tcmnode" (Forts.)

Eigenschaften von tcmnode	Werte	Eigenschaftsbeschreibung
observations_exclude	Ganzzahl	
extend_records_into_future	boolesch	
forecastperiods	Ganzzahl	

Eigenschaften von "timeseriesnode"



Der Zeitreihenknoten berechnet Schätzungen für die exponentielle Glättung sowie univariate und multivariate ARIMA-Modelle (ARIMA steht für Autoregressive Integrated Moving Average (autoregressiver integrierter gleitender Durchschnitt)) für Zeitreihendaten und erstellt Vorhersagen über die zukünftige Leistung. Einem Zeitreihenknoten muss stets ein Zeitintervallknoten vorangehen.

Beispiel

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tabelle 132. Eigenschaften von "timeseriesnode"

Eigenschaften von timeseriesnode	Werte	Eigenschaftsbeschreibung
targets	Feld	Der Zeitreihenknoten sagt eines oder mehrere Ziele voraus; optional können dabei ein oder mehrere Eingabefelder als Prädiktoren verwendet werden. Häufigkeits- und Gewichtungsfelder werden nicht verwendet. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
continue	Flag	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	Flag	
consider_seasonal	Flag	
detect_outliers	Flag	
expert_outlier_additive	Flag	
expert_outlier_level_shift	Flag	
expert_outlier_innovational	Flag	
expert_outlier_level_shift	Flag	
expert_outlier_transient	Flag	
expert_outlier_seasonal_additive	Flag	

Tabelle 132. Eigenschaften von "timeseriesnode" (Forts.)

Eigenschaften von timeseriesnode	Werte	Eigenschaftsbeschreibung
expert_outlier_local_trend	Flag	
expert_outlier_additive_patch	Flag	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	Ganzzahl	
arima_d	Ganzzahl	
arima_q	Ganzzahl	
arima_sp	Ganzzahl	
arima_sd	Ganzzahl	
arima_sq	Ganzzahl	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	Flag	
tf_arima_p. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_d. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_q. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_sp. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_sd. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_sq. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_delay. fieldname	Ganzzahl	Für Transferfunktionen.
tf_arima_transformation_type. fieldname	None SquareRoot NaturalLog	Für Transferfunktionen.
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	Flag	
arima_outlier_level_shift	Flag	
arima_outlier_innovational	Flag	
arima_outlier_transient	Flag	
arima_outlier_seasonal_additive	Flag	
arima_outlier_local_trend	Flag	
arima_outlier_additive_patch	Flag	
conf_limit_pct	reell	
max_lags	Ganzzahl	
Ereignisse	Felder	

Tabelle 132. Eigenschaften von "timeseriesnode" (Forts.)

Eigenschaften von timeseriesnode	Werte	Eigenschaftsbeschreibung
scoring_model_only	Flag	Für Modelle mit sehr großen Zahlen (Zehntausende) von Zeitreihen.

Eigenschaften von "twostepnode"



Der TwoStep-Knoten verwendet eine aus zwei Schritten bestehende Clusterbildungsmethode. Im ersten Schritt wird ein einzelner Durchlauf durch die Daten vorgenommen, bei dem die Eingangsrohdaten zu einem verwaltbaren Set von Subclustern komprimiert werden. Im zweiten Schritt werden die Subcluster mithilfe einer hierarchischen Methode zur Clusterbildung nach und nach in immer größere Cluster zusammengeführt. TwoStep hat den Vorteil, dass die optimale Anzahl an Clustern für die Trainingsdaten automatisch geschätzt wird. Mit dem Verfahren können gemischte Feldtypen und große Datasets effizient verarbeitet werden.

Beispiel

```
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)
```

Tabelle 133. Eigenschaften von "twostepnode"

Eigenschaften von twostepnode	Werte	Eigenschaftsbeschreibung
inputs	[Feld1 ... FeldN]	TwoStep-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtung- und Häufigkeitsfelder werden nicht erkannt. Weitere Informationen finden Sie im Thema „Allgemeine Eigenschaften von Modellierungsknoten“ auf Seite 167.
standardize	Flag	
exclude_outliers	Flag	
percentage	Zahl	
cluster_num_auto	Flag	
min_num_clusters	Zahl	
max_num_clusters	Zahl	
num_clusters	Zahl	
cluster_label	Zeichenfolge Zahl	
label_prefix	Zeichenfolge	

Tabelle 133. Eigenschaften von "twostepnode" (Forts.)

Eigenschaften von twostepnode	Werte	Eigenschaftsbeschreibung
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Eigenschaften von "twostepAS"

TwoStep-Cluster ist ein exploratives Tool, mit dem natürliche Gruppierungen (d. h. Cluster) in einem Datensatz sichtbar gemacht werden können, die ansonsten nicht sichtbar sind. Der von dieser Prozedur verwendete Algorithmus hat mehrere wünschenswerte Funktionen, die ihn von konventionellen Clustering-Verfahren wie Handhabung von kategorialen und stetigen Variablen, automatische Auswahl von Clustern und Skalierbarkeit unterscheiden.

Tabelle 134. Eigenschaften von "twostepAS"

Eigenschaften von twostepAS	Werte	Eigenschaftsbeschreibung
inputs	[f1 ... fN]	TwoStepAS-Modelle verwenden eine Liste mit Eingabefeldern, jedoch kein Ziel. Gewichtung- und Häufigkeitsfelder werden nicht erkannt.
use_predefined_roles	boolesch	Standard=True
use_custom_field_assignments	boolesch	Standard=False
cluster_num_auto	boolesch	Standard=True
min_num_clusters	Ganzzahl	Standard=2
max_num_clusters	Ganzzahl	Standard=15
num_clusters	Ganzzahl	Standard=5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Minimum Maximum	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	boolesch	
random_seed	Ganzzahl	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	boolesch	Standard=True
num_cases_in_feature_tree_leaf_is_less_than	Ganzzahl	Standard=10
top_perc_outliers	Ganzzahl	Standard=5
initial_dist_change_threshold	Ganzzahl	Standard=0
leaf_node_maximum_branches	Ganzzahl	Standard=8
non_leaf_node_maximum_branches	Ganzzahl	Standard=8

Tabelle 134. Eigenschaften von "twostepAS" (Forts.)

Eigenschaften von twostepAS	Werte	Eigenschaftsbeschreibung
max_tree_depth	Ganzzahl	Standard=3
adjustment_weight_on_measurement_level	Ganzzahl	Standard=6
memory_allocation_mb	Zahl	Standard=512
delayed_split	boolesch	Standard=True
fields_to_standardize	[f1 ... fN]	
adaptive_feature_selection	boolesch	Standard=True
featureMisPercent	Ganzzahl	Standard=70
coefRange	Zahl	Standard=0.05
percCasesSingleCategory	Ganzzahl	Standard=95
numCases	Ganzzahl	Standard=24
include_model_specifications	boolesch	Standard=True
include_record_summary	boolesch	Standard=True
include_field_transformations	boolesch	Standard=True
excluded_inputs	boolesch	Standard=True
evaluate_model_quality	boolesch	Standard=True
show_feature_importance_bar_chart	boolesch	Standard=True
show_feature_importance_word_cloud	boolesch	Standard=True
show_outlier_clusters_interactive_table_and_chart	boolesch	Standard=True
show_outlier_clusters_pivot_table	boolesch	Standard=True
across_cluster_feature_importance	boolesch	Standard=True
across_cluster_profiles_pivot_table	boolesch	Standard=True
withinprofiles	boolesch	Standard=True
cluster_distances	boolesch	Standard=True
cluster_label	Zeichenfolge Zahl	
label_prefix	Zeichenfolge	

Kapitel 14. Eigenschaften von Modellnuggetknoten

Modellnuggetknoten besitzen dieselben allgemeinen Eigenschaften wie andere Knoten. Weitere Informationen finden Sie im Thema „Allgemeine Knoteneigenschaften“ auf Seite 74.

Eigenschaften von "applyanomalydetectionnode"

Mithilfe von Modellierungsknoten vom Typ "Anomalieerkennung" kann ein Modellnugget vom Typ "Anomalieerkennung" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyanomalydetectionnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "anomalydetectionnode"“ auf Seite 167.

Tabelle 135. Eigenschaften von "applyanomalydetectionnode".

Eigenschaften von applyanomalydetectionnode	Werte	Eigenschaftsbeschreibung
<code>anomaly_score_method</code>	FlagAndScore FlagOnly ScoreOnly	Bestimmt, welche Ausgaben für das Scoring erstellt werden.
<code>num_fields</code>	<i>Ganzzahl</i>	Zu meldende Felder.
<code>discard_records</code>	<i>Flag</i>	Gibt an, ob Datensätze aus der Ausgabe verworfen werden sollen oder nicht.
<code>discard_anomalous_records</code>	<i>Flag</i>	Gibt an, ob die anomalen oder die <i>nicht</i> anomalen Datensätze verworfen werden sollen. Der Standardwert ist <i>off</i> , was bedeutet, dass die <i>nicht</i> anomalen Datensätze verworfen werden. Bei <i>on</i> werden die anomalen Datensätze verworfen. Diese Eigenschaft ist nur aktiviert, wenn die Eigenschaft <code>discard_records</code> aktiviert ist.

Eigenschaften von "applyapriorinode"

Mithilfe von Apriori-Modellierungsknoten kann ein Modellnugget vom Typ "Apriori" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyapriorinode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "apriorinode"“ auf Seite 169.

Tabelle 136. Eigenschaften von "applyapriorinode".

Eigenschaften von applyapriorinode	Werte	Eigenschaftsbeschreibung
<code>max_predictions</code>	<i>Anzahl (Ganzzahl)</i>	
<code>ignore_unmattached</code>	<i>Flag</i>	
<code>allow_repeats</code>	<i>Flag</i>	
<code>check_basket</code>	NoPredictions Predictions NoCheck	
<code>criterion</code>	Confidence Support RuleSupport Lift Deployability	

Eigenschaften von "applyautoclassifiernode"

Mithilfe von Modellierungsknoten des Typs "Automatisches Klassifikationsmerkmal" kann ein Modellnugget vom Typ "Automatisches Klassifikationsmerkmal" generiert werden. Der Scriptname dieses Modellnuggets ist *applyautoclassifiernode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "autoclassifiernode"“ auf Seite 173.

Tabelle 137. Eigenschaften von "applyautoclassifiernode".

Eigenschaften von applyautoclassifiernode	Werte	Eigenschaftsbeschreibung
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Flagfeld ist.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Gibt an, welche Methode für die Bestimmung des Ensemble-Score verwendet werden soll. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein Setfeld ist.
set_voting_tie_selection	Random HighestConfidence	Wenn eine Voting-Methode ausgewählt ist, gibt diese Einstellung an, wie Gleichstände aufgelöst werden sollen. Diese Einstellung gilt nur, wenn das ausgewählte Ziel ein nominales Feld ist.

Eigenschaften von "applyautoclusternode"

Mithilfe von Modellierungsknoten des Typs "Autom. Cluster" kann ein Modellnugget vom Typ "Autom. Cluster" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyautoclusternode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "autoclusternode"“ auf Seite 175.

Eigenschaften von "applyautonumericnode"

Mithilfe von Modellierungsknoten des Typs "Auto-Numerisch" kann ein Modellnugget vom Typ "Auto-Numerisch" generiert werden. Der Scriptname dieses Modellnuggets ist *applyautonumericnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "autonumericnode"“ auf Seite 177.

Tabelle 138. Eigenschaften von "applyautonumericnode".

Eigenschaften von applyautonumericnode	Werte	Eigenschaftsbeschreibung
calculate_standard_error	Flag	

Eigenschaften von "applybayesnetnode"

Mithilfe von Bayes-Netzmodellierungsknoten kann ein Modellnugget vom Typ "Bayes-Netz" generiert werden. Der Scriptname dieses Modellnuggets lautet *applybayesnetnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "bayesnetnode"“ auf Seite 178.

Tabelle 139. Eigenschaften von "applybayesnetnode".

Eigenschaften von applybayesnetnode	Werte	Eigenschaftsbeschreibung
all_probabilities	Flag	
raw_propensity	Flag	
adjusted_propensity	Flag	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applyc50node"

Mithilfe von C5.0-Modellierungsknoten kann ein C5.0-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyc50node*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "c50node"“ auf Seite 180.

Tabelle 140. Eigenschaften von "applyc50node".

Eigenschaften von applyc50node	Werte	Eigenschaftsbeschreibung
sql_generate	Never NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen des Regelsets.
calculate_conf	Flag	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applycarmanode"

Mithilfe von CARMA-Modellierungsknoten kann ein CARMA-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applycarmanode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "carmanode"“ auf Seite 181.

Eigenschaften von "applycartnode"

Mithilfe von Modellierungsknoten vom Typ "C&R-Baum" kann ein Modellnugget vom Typ "C&R-Baum" generiert werden. Der Scriptname dieses Modellnuggets lautet *applycartnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "cartnode"“ auf Seite 183.

Tabelle 141. Eigenschaften von "applycartnode".

Eigenschaften von applycartnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues NoMissingValues	Dient zur Festlegung von SQL-Erzeugungsoptionen beim Ausführen des Regelsets.

Tabelle 141. Eigenschaften von "applycartnode" (Forts.).

Eigenschaften von applycartnode	Werte	Eigenschaftsbeschreibung
calculate_conf	Flag	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
display_rule_id	Flag	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applychaidnode"

Mithilfe von CHAID-Modellierungsknoten kann ein CHAID-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applychaidnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "chaidnode"“ auf Seite 185.

Tabelle 142. Eigenschaften von "applychaidnode".

Eigenschaften von applychaidnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues	
calculate_conf	Flag	
display_rule_id	Flag	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applycoxregnode"

Mithilfe von Cox-Modellierungsknoten kann ein Cox-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applycoxregnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "coxregnode"“ auf Seite 187.

Tabelle 143. Eigenschaften von "applycoxregnode".

Eigenschaften von applycoxregnode	Werte	Eigenschaftsbeschreibung
future_time_as	Intervals Fields	
time_interval	Zahl	
num_future_times	Ganzzahl	
time_field	Feld	
past_survival_time	Feld	
all_probabilities	Flag	
cumulative_hazard	Flag	

Eigenschaften von "applydecisionlistnode"

Mithilfe von Entscheidungslistenmodellierungsknoten kann ein Modellnugget vom Typ "Entscheidungsliste" generiert werden. Der Scriptname dieses Modellnuggets lautet *applydecisionlistnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "decisionlistnode"“ auf Seite 189.

Table 144. Eigenschaften von "applydecisionlistnode".

Eigenschaften von applydecisionlistnode	Werte	Eigenschaftsbeschreibung
enable_sql_generation	Flag	Wenn dieser Wert wahr ist, versucht IBM SPSS Modeler, das Entscheidungslistenmodell über einen Pushback-Vorgang an SQL zurückzuführen.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applydiscriminantnode"

Mithilfe von Diskriminanzmodellierungsknoten kann ein Diskriminanzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applydiscriminantnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "discriminantnode"“ auf Seite 190.

Table 145. Eigenschaften von "applydiscriminantnode".

Eigenschaften von applydiscriminantnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applyfactornode"

Mithilfe von Faktormodellierungsknoten kann ein Faktormodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyfactornode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "factornode"“ auf Seite 192.

Eigenschaften von "applyfeatureselectionnode"

Mithilfe von Modellierungsknoten vom Typ "Merkmalauswahl" kann ein Modellnugget vom Typ "Merkmalauswahl" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyfeatureselectionnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "featureselectionnode"“ auf Seite 193.

Table 146. Eigenschaften von "applyfeatureselectionnode".

Eigenschaften von applyfeatureselectionnode	Werte	Eigenschaftsbeschreibung
selected_ranked_fields		Gibt an, welche Felder mit Rangzahlen im Modell-Browser markiert werden.
selected_screened_fields		Gibt an, welche im Screening untersuchten Felder im Modell-Browser markiert werden.

Eigenschaften von "applygeneralizedlinearnode"

Mithilfe von Modellierungsknoten vom Typ "Verallgemeinert linear (genlin)" kann ein Modellnugget vom Typ "Verallgemeinert linear" generiert werden. Der Scriptname dieses Modellnuggets lautet *applygeneralizedlinearnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "genlinnode"“ auf Seite 195.

Tabelle 147. Eigenschaften von "applygeneralizedlinearnode".

Eigenschaften von applygeneralizedlinearnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applyglmnode"

Mithilfe von GLMM-Modellierungsknoten kann ein GLMM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyglmnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "glmnode"“ auf Seite 199.

Tabelle 148. Eigenschaften von "applyglmnode".

Eigenschaften von applyglmnode	Werte	Eigenschaftsbeschreibung
confidence	onProbability onIncrease	Grundlage für die Berechnung des Konfidenzwerts für das Scoring: höchste vorhergesagte Wahrscheinlichkeit oder Differenz zwischen der höchsten und zweithöchsten vorhergesagten Wahrscheinlichkeit.
score_category_probabilities	Flag	Auf True gesetzt, werden die vorhergesagten Wahrscheinlichkeiten für kategoriale Ziele generiert. Für jede Kategorie wird ein Feld erstellt. Die Standardeinstellung ist False.
max_categories	Ganzzahl	Maximal zu erstellende Anzahl an Kategorien, für die Wahrscheinlichkeiten vorhergesagt werden sollen. Wird nur verwendet, wenn <i>score_category_probabilities</i> auf True gesetzt ist.
score_propensity	Flag	Auf True gesetzt, werden Raw-Propensity-Scores (die die Wahrscheinlichkeit für "True" angeben) für Modelle mit Flagzielen erzeugt. Bei aktiven Partitionen werden außerdem Adjusted-Propensity-Scores anhand der Testpartition erzeugt. Die Standardeinstellung ist False.

Eigenschaften von "applyassociationrulesnode"

Der Assoziationsregelmodellierungsknoten kann zum Generieren eines Assoziationsregelmodellnuggets verwendet werden. Der Scriptname dieses Modellnuggets lautet *applyassociationrulesnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "associationrulesnode"“ auf Seite 170.

Tabelle 149. Eigenschaften von "applyassociationrulesnode"

Eigenschaften von applyassociationrulesnode	Datentyp	Eigenschaftsbeschreibung
max_predictions	Ganzzahl	Die maximale Anzahl Regeln, die auf jede Eingabe mit dem Score angewendet werden können.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Wählen Sie das Maß aus, das zum Festlegen der Stärke von Regeln verwendet wird.
allow_repeats	boolesch	Legt fest, ob Regeln mit derselben Vorhersage in den Score eingeschlossen werden.
check_input	NoPredictions Predictions NoCheck	

Eigenschaften von "applykmeansnode"

Mithilfe von K-Means-Modellierungsknoten kann ein K-Means-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applykmeansnode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "kmeansnode"“ auf Seite 202.

Eigenschaften von "applyknnnode"

Mithilfe von KNN-Modellierungsknoten kann ein KNN-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyknnnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "knnnode"“ auf Seite 203.

Tabelle 150. Eigenschaften von "applyknnnode".

Eigenschaften von applyknnnode	Werte	Eigenschaftsbeschreibung
all_probabilities	Flag	
save_distances	Flag	

Eigenschaften von "applykohonennode"

Mithilfe von Kohonen-Modellierungsknoten kann ein Kohonen-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applykohonennode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "c50node"“ auf Seite 180.

Eigenschaften von "applylinearnode"

Mithilfe von Cox-Modellierungsknoten kann ein Nugget für ein lineares Modell generiert werden. Der Scriptname dieses Modellnuggets lautet *applylinearnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "linearnode"“ auf Seite 206.

Tabelle 151. Eigenschaften von "applylinearnode".

Eigenschaften von linear	Werte	Eigenschaftsbeschreibung
use_custom_name	Flag	
custom_name	Zeichenfolge	

Table 151. Eigenschaften von "applylinearnode" (Forts.).

Eigenschaften von linear	Werte	Eigenschaftsbeschreibung
enable_sql_generation	Flag	

Eigenschaften von "applylogregnode"

Mithilfe von Modellierungsknoten vom Typ "Logistische Regression" kann ein Modellnugget vom Typ "Logistische Regression" generiert werden. Der Scriptname dieses Modellnuggets lautet *applylogregnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "logregnode"“ auf Seite 207.

Table 152. Eigenschaften von "applylogregnode".

Eigenschaften von applylogregnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	Flag	
calculate_conf	Flag	
enable_sql_generation	Flag	

Eigenschaften von "applyneuralnetnode"

Mithilfe von Netzmodellierungsknoten kann ein Netzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyneuralnetnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "neuralnetnode"“ auf Seite 212.

Vorsicht: In dieser Version ist eine neuere Fassung des Netznuggets mit erweiterten Funktionen verfügbar, die im nächsten Abschnitt beschrieben wird (*applyneuralnetwork*). Die Vorgängerversion ist zwar weiterhin verfügbar, wir empfehlen jedoch die Aktualisierung Ihrer Scripts zur Verwendung der neuen Version. Zur Referenz sind hier Details zur Vorgängerversion enthalten, doch wird die Unterstützung dafür in zukünftigen Versionen wegfallen.

Table 153. Eigenschaften von "applyneuralnetnode".

Eigenschaften von applyneuralnetnode	Werte	Eigenschaftsbeschreibung
calculate_conf	Flag	Diese Eigenschaft ist verfügbar, wenn die SQL-Erzeugung aktiviert ist; sie enthält Konfidenzberechnungen im erzeugten Baum.
enable_sql_generation	Flag	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applyneuralnetworknode"

Mithilfe von Netzmodellierungsknoten kann ein Netzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyneuralnetworknode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "neuralnetworknode"“ auf Seite 214.

Tabelle 154. Eigenschaften von "applyneuralnetworknode"

Eigenschaften von applyneuralnetworknode	Werte	Eigenschaftsbeschreibung
use_custom_name	Flag	
custom_name	Zeichenfolge	
Konfidenz	onProbability onIncrease	
score_category_probabilities	Flag	
max_categories	Zahl	
score_propensity	Flag	

Eigenschaften von "applyquestnode"

Mithilfe von QUEST-Modellierungsknoten kann ein QUEST-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyquestnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "questnode"“ auf Seite 216.

Tabelle 155. Eigenschaften von "applyquestnode".

Eigenschaften von applyquestnode	Werte	Eigenschaftsbeschreibung
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	Flag	
display_rule_id	Flag	Fügt ein Feld zur Scoring-Ausgabe hinzu, das die ID des Endknotens angibt, dem der jeweilige Datensatz zugewiesen ist.
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applyr"

Mithilfe von R-Erstellungsknoten kann ein R-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyr*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "buildr"“ auf Seite 179.

Tabelle 156. Eigenschaften von "applyr"

Eigenschaften von applyr	Werte	Eigenschaftsbeschreibung
score_syntax	Zeichenfolge	R-Scriptsyntax für das Modellscoring.
convert_flags	StringsAndDoubles LogicalValues	Option zum Konvertieren von Flagfeldern.
convert_datetime	Flag	Option zum Konvertieren von Variablen mit Datums- oder Datums-/Zeitformaten in R-Datums-/Zeitformate.

Tabelle 156. Eigenschaften von "applyr" (Forts.)

Eigenschaften von applyr	Werte	Eigenschaftsbeschreibung
convert_datetime_class	POSIXct POSIXlt	Optionen, die angeben, in welches Format Variablen mit Datums- oder Datums-/Zeitformaten konvertiert werden.
convert_missing	Flag	Option zum Konvertieren fehlender Werte in R-Werte "NA".

Eigenschaften von "applyregressionnode"

Mithilfe von Modellierungsknoten vom Typ "Lineare Regression" kann ein Modellnugget vom Typ "Lineare Regression" generiert werden. Der Scriptname dieses Modellnuggets lautet *applyregressionnode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "regressionnode"“ auf Seite 218.

Eigenschaften von "applyselflearningnode"

Mithilfe von Modellierungsknoten vom Typ (Self-Learning Response Model (SLRM) kann ein SLRM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applyselflearningnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "slrmnode"“ auf Seite 221.

Tabelle 157. Eigenschaften von "applyselflearningnode".

Eigenschaften von applyselflearningnode	Werte	Eigenschaftsbeschreibung
max_predictions	Zahl	
randomization	Zahl	
scoring_random_seed	Zahl	
sort	ascending descending	Gibt an, ob die Angebote mit den höchsten oder die mit den niedrigsten Scores zuerst angezeigt werden.
model_reliability	Flag	Berücksichtigt die Option für die Reliabilität auf der Registerkarte "Einstellungen".

Eigenschaften von "applysequencenode"

Mithilfe von Sequenzmodellierungsknoten kann ein Sequenzmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applysequencenode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "sequencenode"“ auf Seite 220.

Eigenschaften von "applysvmnode"

Mithilfe von SVM-Modellierungsknoten kann ein SVM-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applysvmnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "svmnode"“ auf Seite 227.

Tabelle 158. Eigenschaften von "applysvmnode".

Eigenschaften von applysvmnode	Werte	Eigenschaftsbeschreibung
all_probabilities	Flag	

Tabelle 158. Eigenschaften von "applysvmnode" (Forts.).

Eigenschaften von applysvmnode	Werte	Eigenschaftsbeschreibung
calculate_raw_propensities	Flag	
calculate_adjusted_propensities	Flag	

Eigenschaften von "applystpnode"

Der STP-Modellierungsknoten kann zum Generieren eines zugehörigen Modellnuggets verwendet werden, das die Modellausgabe im Ausgabeviewer anzeigt. Der Scriptname dieses Modellnuggets ist *applystpnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "stpnode"“ auf Seite 222.

Tabelle 159. Eigenschaften von "applystpnode"

Eigenschaften von applystpnode	Datentyp	Eigenschaftsbeschreibung
uncertainty_factor	boolesch	Minimum: 0, Maximum: 100.

Eigenschaften von "applytimeseriesnode"

Mithilfe von Zeitreihenmodellierungsknoten kann ein Zeitreihenmodellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applytimeseriesnode*. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "timeseriesnode"“ auf Seite 231.

Tabelle 160. Eigenschaften von "applytimeseriesnode".

Eigenschaften von applytimeseriesnode	Werte	Eigenschaftsbeschreibung
calculate_conf	Flag	
calculate_residuals	Flag	

Eigenschaften von "applytwostepnode"

Mithilfe von TwoStep-Modellierungsknoten kann ein TwoStep-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets lautet *applytwostepnode*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "twostepnode"“ auf Seite 233.

Eigenschaften von "applytwostepAS"

Mithilfe von TwoStep AS-Modellierungsknoten kann ein TwoStep AS-Modellnugget generiert werden. Der Scriptname dieses Modellnuggets ist *applytwostepAS*. Für dieses Modellnugget gelten keine weiteren Eigenschaften. Weitere Informationen zu Scripts für den Modellierungsknoten selbst finden Sie in „Eigenschaften von "twostepAS"“ auf Seite 234.

Kapitel 15. Eigenschaften von Datenbankmodellierungsknoten

IBM SPSS Modeler unterstützt die Integration mit Data-Mining-Tools und Datenmodellierungstools von Datenbank Anbietern, z. B. Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM DB2 InfoSphere Warehouse und IBM Netezza Analytics. Sie können mithilfe von datenbankeigenen Algorithmen Modelle erstellen und scores, ohne dazu die IBM SPSS Modeler-Anwendung verlassen zu müssen. Datenbankmodelle können außerdem mithilfe von Scripterstellung unter Verwendung der in diesem Abschnitt beschriebenen Eigenschaften erstellt und bearbeitet werden.

Das folgende Script-Exzerpt veranschaulicht z. B. die Erstellung eines Microsoft Decision Trees-Modells über die IBM SPSS Modeler-Scriptschnittstelle:

```
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([])
```

Knoteneigenschaften für Microsoft-Modellierung

Eigenschaften von Microsoft-Modellierungsknoten

Allgemeine Eigenschaften

Folgende Eigenschaften haben alle Microsoft-Datenbankmodellierungsknoten gemeinsam.

Tabelle 161. Allgemeine Eigenschaften von Microsoft-Knoten

Allgemeine Eigenschaften von Microsoft-Knoten	Werte	Eigenschaftsbeschreibung
analysis_database_name	<i>Zeichenfolge</i>	Name der Analysis Services-Datenbank.
analysis_server_name	<i>Zeichenfolge</i>	Name des Analysis Services-Hosts.
use_transactional_data	<i>Flag</i>	Gibt an, ob die Eingabedaten in Tabellen- oder Transaktionsformat vorliegen.
inputs	<i>list</i>	Eingabefelder für Tabellendaten.
target	<i>Feld</i>	Vorhergesagtes Feld (gilt nicht für MS-Clustering- oder Sequenz-Clustering-Knoten).

Table 161. Allgemeine Eigenschaften von Microsoft-Knoten (Forts.)

Allgemeine Eigenschaften von Microsoft-Knoten	Werte	Eigenschaftsbeschreibung
unique_field	<i>Feld</i>	Schlüsselfeld.
msas_parameters	<i>strukturiert</i>	Algorithmusparameter. Weitere Informationen finden Sie im Thema „Algorithmusparameter“ auf Seite 251.
with_drillthrough	<i>Flag</i>	Mit Drillthrough-Option.

MS-Entscheidungsbaum

Für Knoten vom Typ `mstreenode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Clustering

Für Knoten vom Typ `msclusternode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS-Assoziationsregeln

Für Knoten vom Typ `msassocnode` sind die folgenden speziellen Eigenschaften verfügbar.

Table 162. Eigenschaften von "msassocnode"

Eigenschaften von msassocnode	Werte	Eigenschaftsbeschreibung
id_field	<i>Feld</i>	Identifiziert jede Transaktion in den Daten.
trans_inputs	<i>list</i>	Eingabefelder für Transaktionsdaten.
transactional_target	<i>Feld</i>	Prädiktorfeld (Transaktionsdaten).

MS Naive Bayes

Für Knoten vom Typ `msbayesnode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Lineare Regression

Für Knoten vom Typ `msregressionnode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Neuronales Netz

Für Knoten vom Typ `msneuralnetworknode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS - Logistische Regression

Für Knoten vom Typ `mslogisticnode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS Time Series

Für Knoten vom Typ `mstimeseriesnode` sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Microsoft-Eigenschaften am Anfang dieses Abschnitts.

MS Sequenz-Clustering

Für Knoten vom Typ `mssequenceclusternode` sind die folgenden speziellen Eigenschaften verfügbar.

Tabelle 163. Eigenschaften von "mssequenceclusternode"

Eigenschaften von <code>mssequenceclusternode</code>	Werte	Eigenschaftsbeschreibung
<code>id_field</code>	<i>Feld</i>	Identifiziert jede Transaktion in den Daten.
<code>input_fields</code>	<i>list</i>	Eingabefelder für Transaktionsdaten.
<code>sequence_field</code>	<i>Feld</i>	Sequenz-ID.
<code>target_field</code>	<i>Feld</i>	Prädiktorfeld (Tabellendaten).

Algorithmusparameter

Jeder Microsoft-Datenbankmodelltyp weist spezifische Parameter auf, die mithilfe der Eigenschaft `msas_parameters` festgelegt werden können. Beispiel:

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters", [{"MAXIMUM_INPUT_ATTRIBUTES", 255},
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Diese Parameter werden vom SQL-Server abgeleitet. Gehen Sie wie folgt vor, um die relevanten Parameter für die einzelnen Knoten anzuzeigen:

1. Platzieren Sie einen Datenbankquellenknoten im Erstellungsbereich.
2. Öffnen Sie den Datenbankquellenknoten.
3. Wählen Sie eine gültige Quelle in der Dropdown-Liste **Datenquelle** aus.
4. Wählen Sie eine gültige Tabelle in der Liste **Tabellename** aus.
5. Klicken Sie auf **OK**, um den Datenbankquellenknoten zu schließen.
6. Fügen Sie den Microsoft-Datenbankmodellierungsknoten ein, dessen Eigenschaften aufgelistet werden sollen.
7. Öffnen Sie den Datenbankmodellierungsknoten.
8. Wählen Sie die Registerkarte **Experten**.

Die verfügbaren `msas_parameters`-Eigenschaften für diesen Knoten werden angezeigt.

Eigenschaften von Microsoft-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der Microsoft-Datenbankmodellierungsknoten erstellt wurden.

MS-Entscheidungsbaum

Tabelle 164. Eigenschaften des MS-Entscheidungsbaums.

Eigenschaften von <code>appliedstreamnode</code>	Werte	Beschreibung
<code>analysis_database_name</code>	<i>Zeichenfolge</i>	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
<code>analysis_server_name</code>	<i>Zeichenfolge</i>	Name des Analyseserver-Hosts.
<code>datasource</code>	<i>Zeichenfolge</i>	Name der SQL Server ODBC-Datenquelle (DSN).
<code>sql_generate</code>	<i>Flag</i>	Aktiviert die SQL-Erzeugung.

MS - Lineare Regression

Table 165. MS Lineare Regression - Eigenschaften.

Eigenschaften von appliesregressionnode	Werte	Beschreibung
analysis_database_name	Zeichenfolge	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	Zeichenfolge	Name des Analyseserver-Hosts.

MS - Neuronales Netz

Table 166. MS Neuronales Netz - Eigenschaften.

Eigenschaften von appliesneuralnetworknode	Werte	Beschreibung
analysis_database_name	Zeichenfolge	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	Zeichenfolge	Name des Analyseserver-Hosts.

MS - Logistische Regression

Table 167. MS Logistische Regression - Eigenschaften.

Eigenschaften von applieslogisticnode	Werte	Beschreibung
analysis_database_name	Zeichenfolge	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	Zeichenfolge	Name des Analyseserver-Hosts.

MS Time Series

Table 168. MS Time Series-Eigenschaften.

Eigenschaften von aplymstimeseriesnode	Werte	Beschreibung
analysis_database_name	Zeichenfolge	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	Zeichenfolge	Name des Analyseserver-Hosts.
start_from	new_prediction historical_prediction	Gibt an, ob Zukunfts- oder historische Vorhersagen getroffen werden.
new_step	Zahl	Definiert die Startzeit für Zukunftsvorhersagen.

Tabelle 168. MS Time Series-Eigenschaften (Forts.).

Eigenschaften von aplymstimeseriesnode	Werte	Beschreibung
historical_step	Zahl	Definiert die Startzeit für historische Vorhersagen.
end_step	Zahl	Definiert die Endzeit für Vorhersagen.

MS Sequenz-Clustering

Tabelle 169. MS-Sequenz-Clustering-Eigenschaften.

Eigenschaften von aplymssequenceclusternode	Werte	Beschreibung
analysis_database_name	Zeichenfolge	Dieser Knoten kann direkt in einem Stream gescort werden. Anhand dieser Eigenschaft wird der Name der Analysis Services-Datenbank identifiziert.
analysis_server_name	Zeichenfolge	Name des Analyseserver-Hosts.

Knoteneigenschaften für Oracle-Modellierung

Eigenschaften von Oracle-Modellierungsknoten

Folgende Eigenschaften haben alle Oracle-Datenbankmodellierungsknoten gemeinsam.

Tabelle 170. Allgemeine Eigenschaften von Oracle-Knoten.

Allgemeine Eigenschaften von Oracle-Knoten	Werte	Eigenschaftsbeschreibung
target	Feld	
inputs	Liste mit Feldern	
partition	Feld	Feld wird verwendet, um die Daten in getrennte Stichproben für die Trainings-, Test- und Validierungsphase der Modellbildung aufzuteilen.
datasource		
username		
password		
epassword		
use_model_name	Flag	
model_name	Zeichenfolge	Benutzerdefinierter Name für neues Modell.
use_partitioned_data	Flag	Wenn ein Partitionsfeld definiert ist, gewährleistet diese Option, dass nur Daten aus der Trainingspartition für die Modellerstellung verwendet werden.
unique_field	Feld	
auto_data_prep	Flag	Aktiviert oder inaktiviert die automatische Datenvorbereitungsfunktion von Oracle (nur 11g-Datenbanken).

Table 170. Allgemeine Eigenschaften von Oracle-Knoten (Forts.).

Allgemeine Eigenschaften von Oracle-Knoten	Werte	Eigenschaftsbeschreibung
costs	strukturiert	Strukturierte Eigenschaft im Format: [{drugA drugB 1.5} {drugA drugC 2.1}], wobei die Argumente in {} tatsächlich vorausgesagte Kosten sind.
mode	Simple Expert	Wenn diese Einstellung auf Simple (Einfach) gesetzt ist, werden bestimmte Eigenschaften ignoriert (vgl. die Eigenschaften der einzelnen Knoten).
use_prediction_probability	Flag	
prediction_probability	Zeichenfolge	
use_prediction_set	Flag	

Oracle Naive Bayes

Für Knoten vom Typ oranbnode sind folgende Eigenschaften verfügbar.

Table 171. Eigenschaften von "oranbnode".

Eigenschaften von oranbnode	Werte	Eigenschaftsbeschreibung
singleton_threshold	Zahl	0,0-1,0.*
pairwise_threshold	Zahl	0,0-1,0.*
priors	Data Equal Custom	
custom_priors	strukturiert	Strukturierte Eigenschaft im Format: set :oranbnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Adaptive Bayes

Für Knoten vom Typ oraabnnode sind folgende Eigenschaften verfügbar.

Table 172. Eigenschaften von "oraabnnode".

Eigenschaften von oraabnnode	Werte	Eigenschaftsbeschreibung
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	Flag	*
execution_time_limit	Ganzzahl	Der Wert muss größer als 0 sein.*
max_naive_bayes_predictors	Ganzzahl	Der Wert muss größer als 0 sein.*
max_predictors	Ganzzahl	Der Wert muss größer als 0 sein.*
priors	Data Equal Custom	
custom_priors	strukturiert	Strukturierte Eigenschaft im Format: set :oraabnnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Support Vector Machines

Für Knoten vom Typ orasvmnode sind folgende Eigenschaften verfügbar.

Tabelle 173. Eigenschaften von "orasvmnode".

Eigenschaften von orasvmnode	Werte	Eigenschaftsbeschreibung
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	Ganzzahl	Nur gaußscher Kern. Der Wert muss größer als 0 sein.*
convergence_tolerance	Zahl	Der Wert muss größer als 0 sein.*
use_standard_deviation	Flag	Nur gaußscher Kern.*
standard_deviation	Zahl	Der Wert muss größer als 0 sein.*
use_epsilon	Flag	Nur Regressionsmodelle.*
epsilon	Zahl	Der Wert muss größer als 0 sein.*
use_complexity_factor	Flag	*
complexity_factor	Zahl	*
use_outlier_rate	Flag	Nur Ein-Klassen-Variante.*
outlier_rate	Zahl	Nur Ein-Klassen-Variante. 0,0-1,0.*
weights	Data Equal Custom	
custom_weights	strukturiert	Strukturierte Eigenschaft im Format: set :orasvmnode.custom_weights = [{drugA 1} {drugB 2} {drugC 3} {drugX 4} {drugY 5}]

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Verallgemeinerte lineare Modelle von Oracle

Die folgenden Eigenschaften sind für Knoten des Typs oraglmnode verfügbar.

Tabelle 174. Eigenschaften von "oraglmnode".

Eigenschaften von oraglmnode	Werte	Eigenschaftsbeschreibung
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	

Tabelle 174. Eigenschaften von "oraglmnode" (Forts.).

Eigenschaften von oraglmnode	Werte	Eigenschaftsbeschreibung
use_row_weights	Flag	*
row_weights_field	Feld	*
save_row_diagnostics	Flag	*
row_diagnostics_table	Zeichenfolge	*
coefficient_confidence	Zahl	*
use_reference_category	Flag	*
reference_category	Zeichenfolge	*
ridge_regression	Auto Off On	*
parameter_value	Zahl	*
vif_for_ridge	Flag	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Decision Tree

Die folgenden Eigenschaften sind für Knoten des Typs oradecisiontreenode verfügbar.

Tabelle 175. Eigenschaften von "oradecisiontreenode".

Eigenschaften von oradecisiontreenode	Werte	Eigenschaftsbeschreibung
use_costs	Flag	
impurity_metric	Entropy Gini	
term_max_depth	Ganzzahl	2-20.*
term_minpct_node	Zahl	0,0-10,0.*
term_minpct_split	Zahl	0,0-20,0.*
term_minrec_node	Ganzzahl	Der Wert muss größer als 0 sein.*
term_minrec_split	Ganzzahl	Der Wert muss größer als 0 sein.*
display_rule_ids	Flag	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle O-Cluster

Die folgenden Eigenschaften sind für Knoten des Typs oraoclusternode verfügbar.

Tabelle 176. Eigenschaften von "oraoclusternode".

Eigenschaften von oraoclusternode	Werte	Eigenschaftsbeschreibung
max_num_clusters	Ganzzahl	Der Wert muss größer als 0 sein.
max_buffer	Ganzzahl	Der Wert muss größer als 0 sein.*
sensitivity	Zahl	0,0-1,0.*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle KMeans

Die folgenden Eigenschaften sind für Knoten des Typs orakmeansnode verfügbar.

Tabelle 177. Eigenschaften von "orakmeansnode".

Eigenschaften von orakmeansnode	Werte	Eigenschaftsbeschreibung
num_clusters	Ganzzahl	Der Wert muss größer als 0 sein.
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
iterations	Ganzzahl	0-20.*
conv_tolerance	Zahl	0,0-0,5.*
split_criterion	Variance Size	Die Standardeinstellung ist "Variance".
num_bins	Ganzzahl	Der Wert muss größer als 0 sein.*
block_growth	Ganzzahl	1-5.*
min_pct_attr_support	Zahl	0,0-1,0.*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle NMF

Die folgenden Eigenschaften sind für Knoten des Typs oranmfnode verfügbar.

Tabelle 178. Eigenschaften von "oranmfnode".

Eigenschaften von oranmfnode	Werte	Eigenschaftsbeschreibung
normalization_method	minmax none	
use_num_features	Flag	*
num_features	Ganzzahl	0-1. Der Standardwert wird vom Algorithmus durch Schätzung aus den Daten ermittelt.*
random_seed	Zahl	*
num_iterations	Ganzzahl	0-500.*
conv_tolerance	Zahl	0,0-0,5.*
display_all_features	Flag	*

* Eigenschaft wird ignoriert, wenn mode auf Simple gesetzt ist.

Oracle Apriori

Die folgenden Eigenschaften sind für Knoten des Typs oraapriorinode verfügbar.

Tabelle 179. Eigenschaften von "oraapriorinode".

Eigenschaften von oraapriorinode	Werte	Eigenschaftsbeschreibung
content_field	Feld	
id_field	Feld	
max_rule_length	Ganzzahl	2-20.
min_confidence	Zahl	0,0-1,0.
min_support	Zahl	0,0-1,0.
use_transactional_data	Flag	

Oracle Minimum Description Length (MDL)

Für Knoten vom Typ oramdlnode sind keine speziellen Eigenschaften definiert. Informationen finden Sie unter den allgemeinen Oracle-Eigenschaften am Anfang dieses Abschnitts.

Oracle Attribute Importance (AI)

Die folgenden Eigenschaften sind für Knoten des Typs oraainode verfügbar.

Tabelle 180. Eigenschaften von "oraainode".

Eigenschaften von oraainode	Werte	Eigenschaftsbeschreibung
custom_fields	Flag	Bei "true" (wahr) können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob bedeutsame Felder ausgewählt werden sollen.
important_label	Zeichenfolge	Gibt die Beschriftung für die Rangstufe "bedeutsam" an.
select_marginal	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob marginale Felder ausgewählt werden sollen.
marginal_label	Zeichenfolge	Gibt die Beschriftung für die Rangstufe "marginal" an.
important_above	Zahl	0,0-1,0.
select_unimportant	Flag	Wenn selection_mode auf ImportanceLevel gesetzt ist, wird hier angegeben, ob unbedeutende Felder ausgewählt werden sollen.
unimportant_label	Zeichenfolge	Gibt die Beschriftung für die Rangstufe "unbedeutend" an.
unimportant_below	Zahl	0,0-1,0.
importance_value	Zahl	Wenn selection_mode auf ImportanceValue gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 100.
top_n	Zahl	Wenn selection_mode auf TopN gesetzt ist, wird hier der zu verwendende Trennwert angegeben. Zulässig sind Werte von 0 bis 1000.

Eigenschaften von Oracle-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der Oracle-Modelle erstellt wurden.

Oracle Naive Bayes

Für Knoten vom Typ `applyoranbnode` sind keine speziellen Eigenschaften definiert.

Oracle Adaptive Bayes

Für Knoten vom Typ `applyoraabnode` sind keine speziellen Eigenschaften definiert.

Oracle Support Vector Machines

Für Knoten vom Typ `applyorasvmnode` sind keine speziellen Eigenschaften definiert.

Oracle Decision Tree

Die folgenden Eigenschaften sind für Knoten des Typs `applyoradecisiontreenode` verfügbar.

Tabelle 181. Eigenschaften von "applyoradecisiontreenode"

Eigenschaften von <code>applyoradecisiontreenode</code>	Werte	Eigenschaftsbeschreibung
<code>use_costs</code>	<i>Flag</i>	
<code>display_rule_ids</code>	<i>Flag</i>	

Oracle O-Cluster

Für Knoten vom Typ `applyoraoclusternode` sind keine speziellen Eigenschaften definiert.

Oracle KMeans

Für Knoten vom Typ `applyorakmeansnode` sind keine speziellen Eigenschaften definiert.

Oracle NMF

Für Knoten vom Typ `applyoranmfnode` ist die folgende Eigenschaft verfügbar.

Tabelle 182. Eigenschaften von "applyoranmfnode"

Eigenschaften von <code>applyoranmfnode</code>	Werte	Eigenschaftsbeschreibung
<code>display_all_features</code>	<i>Flag</i>	

Oracle Apriori

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Oracle MDL

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Knoteneigenschaften für IBM DB2-Modellierung

Eigenschaften von IBM DB2-Modellierungsknoten

Folgende Eigenschaften haben alle IBM InfoSphere Warehouse-Datenbankmodellierungsknoten (ISW-Datenbankmodellierungsknoten) gemeinsam:

Table 183. Allgemeine Eigenschaften von ISW-Knoten.

Allgemeine Eigenschaften von ISW-Knoten	Werte	Eigenschaftsbeschreibung
inputs	Liste mit Feldern	
datasource		
username		
password		
epassword		
enable_power_options	Flag	
power_options_max_memory	Ganzzahl	Der Wert muss größer als 32 sein.
power_options_cmdline	Zeichenfolge	
mining_data_custom_sql	Zeichenfolge	
logical_data_custom_sql	Zeichenfolge	
mining_settings_custom_sql		

ISW-Entscheidungsbaum

Für Knoten vom Typ `db2imtreenode` sind folgende Eigenschaften verfügbar.

Table 184. Eigenschaften von "db2imtreenode".

Eigenschaften von <code>db2imtreenode</code>	Werte	Eigenschaftsbeschreibung
target	Feld	
perform_test_run	Flag	
use_max_tree_depth	Flag	
max_tree_depth	Ganzzahl	Der Wert muss größer als 0 sein.
use_maximum_purity	Flag	
maximum_purity	Zahl	Eine Zahl zwischen 0 und 100.
use_minimum_internal_cases	Flag	
minimum_internal_cases	Ganzzahl	Wert größer als 1.
use_costs	Flag	
costs	strukturiert	Strukturierte Eigenschaft im Format: <code>[[{drugA drugB 1.5} {drugA drugC 2.1}]]</code> , wobei die Argumente in <code>{}</code> tatsächlich vorausgesagte Kosten sind.

ISW-Assoziation

Die folgenden Eigenschaften sind für Knoten des Typs `db2imassocnode` verfügbar.

Table 185. Eigenschaften von "db2imassocnode".

Eigenschaften von db2imassocnode	Werte	Eigenschaftsbeschreibung
use_transactional_data	Flag	
id_field	Feld	
content_field	Feld	
data_table_layout	basic limited_length	
max_rule_size	Ganzzahl	Der Wert muss größer als 2 sein.
min_rule_support	Zahl	0-100 %
min_rule_confidence	Zahl	0-100 %
use_item_constraints	Flag	
item_constraints_type	Include Exclude	
use_taxonomy	Flag	
taxonomy_table_name	Zeichenfolge	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	Zeichenfolge	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	Zeichenfolge	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	Flag	Steuert, ob in IBM SPSS Modeler gespeicherte Taxonomieinformationen bei der Modellerstellung hochgeladen werden sollen. Die Taxonomietabelle wird verworfen, wenn sie bereits vorhanden ist. Die Taxonomieinformationen werden mit dem Modellknoten gespeichert und können über die Schaltflächen Kategorien bearbeiten und Taxonomie bearbeiten bearbeitet werden.

ISW-Sequenz

Die folgenden Eigenschaften sind für Knoten des Typs db2imsequencenode verfügbar.

Table 186. Eigenschaften von "db2imsequencenode".

Eigenschaften von db2imsequencenode	Werte	Eigenschaftsbeschreibung
id_field	Feld	
group_field	Feld	
content_field	Feld	
max_rule_size	Ganzzahl	Der Wert muss größer als 2 sein.
min_rule_support	Zahl	0-100 %
min_rule_confidence	Zahl	0-100 %
use_item_constraints	Flag	
item_constraints_type	Include Exclude	
use_taxonomy	Flag	

Tabelle 186. Eigenschaften von "db2imsequencenode" (Forts.).

Eigenschaften von db2imsequencenode	Werte	Eigenschaftsbeschreibung
taxonomy_table_name	Zeichenfolge	Der Name der DB2-Tabelle, in der Taxonomiedetails gespeichert werden.
taxonomy_child_column_name	Zeichenfolge	Der Name der untergeordneten Spalte in der Taxonomietabelle. Die untergeordnete Spalte enthält die Element- oder Kategorienamen.
taxonomy_parent_column_name	Zeichenfolge	Der Name der übergeordneten Spalte in der Taxonomietabelle. Die übergeordnete Spalte enthält die Kategorienamen.
load_taxonomy_to_table	Flag	Steuert, ob in IBM SPSS Modeler gespeicherte Taxonomieinformationen bei der Modellerstellung hochgeladen werden sollen. Die Taxonomietabelle wird verworfen, wenn sie bereits vorhanden ist. Die Taxonomieinformationen werden mit dem Modellknoten gespeichert und können über die Schaltflächen Kategorien bearbeiten und Taxonomie bearbeiten bearbeitet werden.

ISW-Regression

Die folgenden Eigenschaften sind für Knoten des Typs db2imregnode verfügbar.

Tabelle 187. Eigenschaften von "db2imregnode".

Eigenschaften von db2imregnode	Werte	Eigenschaftsbeschreibung
target	Feld	
regression_method	transform linear polynomial rbf	In der nächsten Tabelle finden Sie Eigenschaften, die nur gelten, wenn regression_method auf rbf gesetzt ist.
perform_test_run	Feld	
limit_rsquared_value	Flag	
max_rsquared_value	Zahl	Der Wert muss zwischen 0,0 und 1,0 liegen.
use_execution_time_limit	Flag	
execution_time_limit_mins	Ganzzahl	Der Wert muss größer als 0 sein.
use_max_degree_polynomial	Flag	
max_degree_polynomial	Ganzzahl	
use_intercept	Flag	
use_auto_feature_selection_method	Flag	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	Flag	
min_significance_level	Zahl	
use_min_significance_level	Flag	

Die folgenden Eigenschaften gelten nur, wenn regression_method auf rbf gesetzt ist.

Tabelle 188. Eigenschaften von "db2imregnode", wenn "regression_method" auf "rbf" gesetzt ist.

Eigenschaften von db2imregnode	Werte	Eigenschaftsbeschreibung
use_output_sample_size	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
output_sample_size	Ganzzahl	Die Standardeinstellung ist 2. Minimum ist 1.
use_input_sample_size	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
input_sample_size	Ganzzahl	Die Standardeinstellung ist 2. Minimum ist 1.
use_max_num_centers	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
max_num_centers	Ganzzahl	Die Standardeinstellung ist 20. Minimum ist 1.
use_min_region_size	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
min_region_size	Ganzzahl	Die Standardeinstellung ist 15. Minimum ist 1.
use_max_data_passes	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
max_data_passes	Ganzzahl	Die Standardeinstellung ist 5. Minimum ist 2.
use_min_data_passes	Flag	Wenn wahr, den Wert automatisch auf Standard setzen.
min_data_passes	Ganzzahl	Die Standardeinstellung ist 5. Minimum ist 2.

ISW Clustering

Die folgenden Eigenschaften sind für Knoten des Typs db2imclusternode verfügbar.

Tabelle 189. Eigenschaften von "db2imclusternode".

Eigenschaften von db2imclusternode	Werte	Eigenschaftsbeschreibung
cluster_method	demographic kohonen birch	
kohonen_num_rows	Ganzzahl	
kohonen_num_columns	Ganzzahl	
kohonen_passes	Ganzzahl	
use_num_passes_limit	Flag	
use_num_clusters_limit	Flag	
max_num_clusters	Ganzzahl	Wert größer als 1.
birch_dist_measure	log_likelihood euclidean	Die Standardeinstellung ist log_likelihood.
birch_num_cfleaves	Ganzzahl	Die Standardeinstellung ist 1000.
birch_num_refine_passes	Ganzzahl	Die Standardeinstellung lautet 3; Minimum ist 1.
use_execution_time_limit	Flag	

Tabelle 189. Eigenschaften von "db2imclusternode" (Forts.).

Eigenschaften von db2imclusternode	Werte	Eigenschaftsbeschreibung
execution_time_limit_mins	Ganzzahl	Der Wert muss größer als 0 sein.
min_data_percentage	Zahl	0-100 %
use_similarity_threshold	Flag	
similarity_threshold	Zahl	Der Wert muss zwischen 0,0 und 1,0 liegen.

ISW Naive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs db2imnbsnode verfügbar.

Tabelle 190. Eigenschaften von "db2imnbnode".

Eigenschaften von db2imnbnode	Werte	Eigenschaftsbeschreibung
perform_test_run	Flag	
probability_threshold	Zahl	Die Standardeinstellung ist 0,001. Minimalwert ist 0; Maximalwert ist 1.000
use_costs	Flag	
costs	strukturiert	Strukturierte Eigenschaft im Format: <code>[[drugA drugB 1.5] {drugA drugC 2.1}]</code> , wobei die Argumente in {} tatsächlich vorausgesagte Kosten sind.

ISW Logistische Regression

Die folgenden Eigenschaften sind für Knoten des Typs db2imlognode verfügbar.

Tabelle 191. Eigenschaften von "db2imlognode".

Eigenschaften von db2imlognode	Werte	Eigenschaftsbeschreibung
perform_test_run	Flag	
use_costs	Flag	
costs	strukturiert	Strukturierte Eigenschaft im Format: <code>[[drugA drugB 1.5] {drugA drugC 2.1}]</code> , wobei die Argumente in {} tatsächlich vorausgesagte Kosten sind.

ISW Time Series

Hinweis: der Eingabefeldparameter wird für diesen Knoten nicht verwendet. Wenn der Eingabefeldparameter in dem Script gefunden wird, erscheint eine Warnung, dass der Knoten als eingehende Felder *time* und *targets*, aber keine Eingabefelder hat.

Die folgenden Eigenschaften sind für Knoten des Typs db2imtimeseriesnode verfügbar.

Tabelle 192. Eigenschaften von "db2imtimeseriesnode".

Eigenschaften von db2imtimeseriesnode	Werte	Eigenschaftsbeschreibung
time	Feld	"Integer", "time" oder "date" sind zulässig.
targets	Liste mit Feldern	

Tabelle 192. Eigenschaften von "db2imtimeseriesnode" (Forts.).

Eigenschaften von db2imtimeseriesnode	Werte	Eigenschaftsbeschreibung
forecasting_algorithm	arima exponential_ smoothing seasonal_trend_ decomposition	
forecasting_end_time	auto Ganzzahl Datum time	
use_records_all	<i>boolesch</i>	Wenn falsch, müssen use_records_start und use_records_end festgelegt werden.
use_records_start	<i>Ganzzahl / Zeit / Datum</i>	Abhängig vom Zeitfeldtyp
use_records_end	<i>Ganzzahl / Zeit / Datum</i>	Abhängig vom Zeitfeldtyp
interpolation_method	none linear exponential_splines cubic_splines	

Eigenschaften von IBM DB2-Modellnuggets

Folgende Eigenschaften gelten für die Modellnuggets, die mithilfe der IBM DB2 ISW-Modelle erstellt wurden.

ISW-Entscheidungsbaum

Für Knoten vom Typ applydb2imtreenode sind keine speziellen Eigenschaften definiert.

ISW-Assoziation

Dieses Modellnugget kann nicht in Scripts verwendet werden.

ISW-Sequenz

Dieses Modellnugget kann nicht in Scripts verwendet werden.

ISW-Regression

Für Knoten vom Typ applydb2imregnode sind keine speziellen Eigenschaften definiert.

ISW-Clustering

Für Knoten vom Typ applydb2imclusternode sind keine speziellen Eigenschaften definiert.

ISW Naive Bayes

Für Knoten vom Typ applydb2imnbnode sind keine speziellen Eigenschaften definiert.

ISW Logistische Regression

Für Knoten vom Typ applydb2imlognode sind keine speziellen Eigenschaften definiert.

Dieses Modellnugget kann nicht in Scripts verwendet werden.

Knoteneigenschaften für IBM Netezza Analytics-Modellierung

Eigenschaften von Netezza-Modellierungsknoten

Folgende Eigenschaften haben alle IBM Netezza-Datenbankmodellierungsknoten gemeinsam.

Table 193. Allgemeine Eigenschaften von Netezza-Knoten.

Allgemeine Eigenschaften von Netezza-Knoten	Werte	Eigenschaftsbeschreibung
custom_fields	Flag	Bei "true" können Sie Ziel-, Eingabe- und andere Felder für den aktuellen Knoten angeben. Bei "false" (falsch) werden die aktuellen Einstellungen aus einem vorausgehenden Typknoten verwendet.
inputs	[Feld1 ... FeldN]	Im Modell verwendete Eingabe- bzw. Prädiktorfelder.
target	Feld	Zielfeld (stetig oder kategorial).
record_id	Feld	Das als eindeutige ID für einen Datensatz zu verwendende Feld.
use_upstream_connection	Flag	Falls "True" (Standard), die in einem vorausgehenden Knoten angegebenen Verbindungsdetails. Wird bei Angabe von move_data_to_connection nicht verwendet.
move_data_connection	Flag	Falls "True", wird der Wert in die durch connection angegebene Datenbank verschoben. Wird bei Angabe von use_upstream_connection nicht verwendet.
connection	strukturiert	Die Verbindungszeichenfolge für die Netezza-Datenbank, in der das Modell gespeichert ist. Strukturierte Eigenschaft im Format: <code>['odbc' '<DSN>' '<Benutzername>' '<KW>' '<K>' '<Verbattribs>' {true false}]</code> Dabei gilt: <DSN> ist der Datenquellenname <Benutzername> und <KW> sind der Benutzername und das Kennwort für die Datenbank <K> ist der Katalogname <Verbattribs> sind die Verbindungsattribute true false gibt an, ob das Kennwort erforderlich ist.
table_name	Zeichenfolge	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.
use_model_name	Flag	Falls "True", wird der von model_name angegebene Name als Name des Modells verwendet. Andernfalls wird der Modellname vom System erstellt.
model_name	Zeichenfolge	Benutzerdefinierter Name für neues Modell.
include_input_fields	Flag	Falls "True", werden alle Eingabefelder nach unten weitergegeben. Andernfalls werden nur record_id und vom Modell erstellte Felder weitergegeben.

Netezza-Entscheidungsbaum

Die folgenden Eigenschaften sind für Knoten des Typs netezzadectreenode verfügbar.

Tabelle 194. Eigenschaften von "netezzadectreenode".

Eigenschaften von netezzadectreenode	Werte	Eigenschaftsbeschreibung
impurity_measure	Entropy Gini	Das Maß der Unreinheit, das verwendet wird, um die beste Position für eine Baumteilung zu ermitteln.
max_tree_depth	Ganzzahl	Maximale Anzahl der Ebenen, auf die der Baum erweitert werden kann. Der Standardwert ist 63 (größter zulässiger Wert).
min_improvement_splits	Zahl	Mindestverbesserung in Unreinheit, damit eine Aufteilung stattfinden kann. Die Standardeinstellung ist 0.01.
min_instances_split	Ganzzahl	Mindestanzahl der nicht aufgeteilten Datensätze, die verbleiben müssen, bevor eine Aufteilung stattfinden kann. Der Standardwert ist 2 (kleinster zulässiger Wert).
weights	strukturiert	Relative Gewichtungen für Klassen. Strukturierte Eigenschaft im Format: set :netezza_dectree.weights = [<code>{drugA 0.3}{drugB 0.6}</code>] StandardGewichtung ist für alle Klassen 1.
pruning_measure	Acc wAcc	Die Standardeinstellung ist Acc (Genauigkeit). Bei der alternativen Einstellung wAcc (gewichtete Genauigkeit) werden Klassengewichtungen in die Reduzierung/Beschneidung mit einbezogen.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	In der Standardeinstellung wird allTrainingData zur Schätzung der Modellgenauigkeit verwendet. Verwenden Sie partitionTrainingData, um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdataset aus einer angegebenen Datenbanktabelle zu verwenden.
perc_training_data	Zahl	Wenn prune_tree_options auf partitionTrainingData gesetzt ist, wird der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	Ganzzahl	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wenn prune_tree_options auf partitionTrainingData gesetzt ist. Die Standardeinstellung ist 1.

Tabelle 194. Eigenschaften von "netezzadectreenode" (Forts.).

Eigenschaften von netezzadectreenode	Werte	Eigenschaftsbeschreibung
pruning_table	Zeichenfolge	Tabellenname eines separaten Datasets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	Flag	Wenn "True"; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.

Netezza-K-Means

Die folgenden Eigenschaften sind für Knoten des Typs netezzakmeansnode verfügbar.

Tabelle 195. Eigenschaften von "netezzakmeansnode".

Eigenschaften von netezzakmeansnode	Werte	Eigenschaftsbeschreibung
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
num_clusters	Ganzzahl	Anzahl der zu erstellenden Cluster; Standardwert ist 3.
max_iterations	Ganzzahl	Anzahl der Algorithmusiterationen, nach der das Modelltrainig beendet werden soll; Standardwert ist 5.
rand_seed	Ganzzahl	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert; Standardwert ist 12345.

Netezza-Bayes-Netz

Die folgenden Eigenschaften sind für Knoten des Typs netezzabayesnode verfügbar.

Tabelle 196. Eigenschaften von "netezzabayesnode".

Eigenschaften von netezzabayesnode	Werte	Eigenschaftsbeschreibung
base_index	Ganzzahl	Die numerische Kennung, die zur internen Verwaltung dem ersten Eingabefeld zugewiesen wird; Standardwert ist 777.
sample_size	Ganzzahl	Umfang der zu ziehenden Stichprobe, wenn die Anzahl der Attribute sehr groß ist; Standardwert ist 10.000.
display_additional_information	Flag	Wenn "True", werden weitere Fortschrittsinformationen in einem Nachrichtendialogfeld angezeigt.
type_of_prediction	best neighbors nn-neighbors	Typ des zu verwendenden Vorhersagealgorithmus: best (Nachbar mit höchster Korrelation), neighbors (gewichtete Vorhersage von Nachbarn) oder nn-neighbors (Nicht-NULL-Nachbarn).

Netezza - Naive Bayes

Die folgenden Eigenschaften sind für Knoten des Typs netezzanaiveyesnode verfügbar.

Tabelle 197. Eigenschaften von "netezzanaivebayesnode".

Eigenschaften von netezzanaivebayesnode	Werte	Eigenschaftsbeschreibung
compute_probabilities	Flag	Wenn "True"; wird zusätzlich zum Vorhersagefeld auch ein Feld für das Konfidenzniveau (Wahrscheinlichkeit) erstellt.
use_m_estimation	Flag	Wenn "True", wird das m-Schätzverfahren zur Vermeidung der Wahrscheinlichkeit null während der Schätzung verwendet.

Netezza-KNN

Die folgenden Eigenschaften sind für Knoten des Typs netezzaknnnode verfügbar.

Tabelle 198. Eigenschaften von "netezzaknnnode".

Eigenschaften von netezzaknnnode	Werte	Eigenschaftsbeschreibung
weights	strukturiert	Strukturierte Eigenschaft, die zur Zuweisung von Gewichtungen für die einzelnen Klassen verwendet wird. Beispiel: set :netezzaknnnode.weights = [{drugA 0.3}{drugB 0.6}]
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
num_nearest_neighbors	Ganzzahl	Anzahl der nächsten Nachbarn für einen bestimmten Fall; Standardwert ist 3
standardize_measurements	Flag	Wenn "True", werden vor der Berechnung der Abstandswerte die Messungen für stetige Eingabefelder standardisiert.
use_coresets	Flag	Wenn "True", wird Stichprobennahme mit Core-Sets verwendet, um die Berechnung bei großen Datensets zu beschleunigen.

Netezza - Divisives Clustering

Die folgenden Eigenschaften sind für Knoten des Typs netez zadivclusternode verfügbar.

Tabelle 199. Eigenschaften von "netez zadivclusternode".

Eigenschaften von netez zadivclusternode	Werte	Eigenschaftsbeschreibung
distance_measure	Euclidean Manhattan Canberra Maximum	Methode zur Messung des Abstands zwischen Datenpunkten.
max_iterations	Ganzzahl	Maximale Anzahl an Algorithmusiterationen, die durchgeführt werden sollen, bevor das Modelltraining beendet wird; Standardwert ist 5.
max_tree_depth	Ganzzahl	Maximale Anzahl an Ebenen, in die das Dataset unterteilt werden kann; Standardwert ist 3.
rand_seed	Ganzzahl	Für die Reproduktion von Analysen verwendeter Zufallsstartwert; Standardwert ist 12345.

Tabelle 199. Eigenschaften von "netezadivclusternode" (Forts.).

Eigenschaften von netezadivclusternode	Werte	Eigenschaftsbeschreibung
min_instances_split	Ganzzahl	Mindestanzahl von Datensätzen, die aufgeteilt werden können; Standardwert ist 5.
level	Ganzzahl	Hierarchieebene, auf der die Datensätze gesort werden; Standardwert ist -1.

Netezza-PCA

Die folgenden Eigenschaften sind für Knoten des Typs netezapcanode verfügbar.

Tabelle 200. Eigenschaften von "netezapcanode".

Eigenschaften von netezapcanode	Werte	Eigenschaftsbeschreibung
center_data	Flag	Wenn "True" (Standard), wird vor der Analyse Datenzentrierung (auch als Mittelwertsubtraktion bezeichnet) durchgeführt.
perform_data_scaling	Flag	Wenn "True", wird vor der Analyse eine Datenskalierung durchgeführt. Auf diese Weise wird die Analyse eventuell weniger arbiträr, wenn verschiedene Variablen in verschiedenen Einheiten gemessen werden.
force_eigensolve	Flag	Wenn "True", wird eine weniger genaue, jedoch schnellere Methode zur Ermittlung der Hauptkomponenten verwendet.
pc_number	Ganzzahl	Anzahl an Hauptkomponenten, auf die das Dataset reduziert werden soll; Standardwert ist 1.

Netezza-Regressionsbaum

Die folgenden Eigenschaften sind für Knoten des Typs netezaregtreenode verfügbar.

Tabelle 201. Eigenschaften von "netezaregtreenod".

Eigenschaften von netezaregtreenode	Werte	Eigenschaftsbeschreibung
max_tree_depth	Ganzzahl	Maximale Anzahl an Ebenen, auf die ein Baum unterhalb des Stammknotens erweitert werden kann; Standardwert ist 10.
split_evaluation_measure	Variance	Unreinheitsmaß für die Klasse, das verwendet wird, um die beste Position für eine Baumteilung zu ermitteln; Standardwert (und einzige derzeit mögliche Option) ist Variance.
min_improvement_splits	Zahl	Mindestwert der Unreinheitsreduzierung, bevor eine neue Aufteilung des Baums erfolgt.
min_instances_split	Ganzzahl	Mindestanzahl an Datensätzen, die aufgeteilt werden kann.
pruning_measure	mse r2 pearson spearman	Für die Reduzierung zu verwendende Methode.

Tabelle 201. Eigenschaften von "netezzaregtreenod" (Forts.).

Eigenschaften von netezzaregtreenode	Werte	Eigenschaftsbeschreibung
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	In der Standardeinstellung wird allTrainingData zur Schätzung der Modellgenauigkeit verwendet. Verwenden Sie partitionTrainingData, um den Prozentsatz der zu verwendenden Trainingsdaten festzulegen, oder useOtherTable, um ein Trainingsdataset aus einer angegebenen Datenbanktabelle zu verwenden.
perc_training_data	Zahl	Wenn prune_tree_options auf PercTrainingData gesetzt ist, wird der für das Training zu verwendende Prozentsatz angegeben.
prune_seed	Ganzzahl	Für die Reproduktion der Analyseergebnisse zu verwendender Zufallsstartwert, wenn prune_tree_options auf PercTrainingData gesetzt ist. Die Standardeinstellung ist 1.
pruning_table	Zeichenfolge	Tabellenname eines separaten Datasets für die Reduzierung, anhand dessen die Modellgenauigkeit geschätzt wird.
compute_probabilities	Flag	Wenn "True", wird angegeben, ob die Varianz zugewiesener Klassen in die Ausgabe aufgenommen werden soll.

Netezza - Lineare Regression

Die folgenden Eigenschaften sind für Knoten des Typs netezzalinieregressionnode verfügbar.

Tabelle 202. Eigenschaften von "netezzalinieregressionnode".

Eigenschaften von netezzalinieregressionnode	Werte	Eigenschaftsbeschreibung
use_svd	Flag	Wenn "True", wird anstelle der ursprünglichen Matrix die Matrix zur Einzelwertzerlegung verwendet, um eine höhere Geschwindigkeit und numerische Genauigkeit zu erreichen.
include_intercept	Flag	Wenn "True" (Standard), wird die Gesamtgenauigkeit der Lösung erhöht.
calculate_model_diagnostics	Flag	Wenn "True", werden Diagnosedaten für das Modell berechnet.

Netezza-Zeitreihe

Die folgenden Eigenschaften sind für Knoten des Typs netezzatimeseriesnode verfügbar.

Tabelle 203. Eigenschaften von "netezzatimeseriesnode".

Eigenschaften von netezzatimeseriesnode	Werte	Eigenschaftsbeschreibung
time_points	Feld	Das Eingabefeld, das die Datums- bzw. Zeitwerte für die Zeitreihe enthält.
time_series_ids	Feld	Eingabefeld mit Zeitreihen-IDs. Verwenden Sie das Feld, wenn die Eingabe mehrere Zeitreihen enthält.
model_table	Feld	Der Name der Datenbanktabelle, in der das Netezza-Zeitreihenmodell gespeichert werden soll.
description_table	Feld	Name der Eingabetabelle mit Zeitreihennamen und Beschreibungen.
seasonal_adjustment_table	Feld	Name der Ausgabetabelle, in der saisonal angepasste Werte gespeichert werden, die durch exponentielles Glätten oder Algorithmen zur saisonalen Zerlegung in Trends berechnet werden.
algorithm_name	SpectralAnalysis oder spectral ExponentialSmoothing oder esoothing ARIMA (X11 ARIMA) SeasonalTrendDecomposition oder std	Für die Modellierung von Zeitreihen zu verwendender Algorithmus.
trend_name	N A DA M DM	Trendtyp für exponentielles Glätten: N - keiner A - additiv DA - gedämpft additiv M - multiplikativ DM - gedämpft multiplikativ
seasonality_type	N A M	Saisonalitätstyp für exponentielles Glätten: N - keiner A - additiv M - multiplikativ
interpolation_method	linear cubicspline exponentialspline	Zu verwendende Interpolationsmethode.
timerange_setting	SD SP	Einstellung für den zu verwendenden Zeitbereich: SD - systembestimmt (verwendet den vollständigen Bereich der Zeitreihendaten) SP - benutzerdefiniert über earliest_time und latest_time

Tabelle 203. Eigenschaften von "netezatimeseriesnode" (Forts.).

Eigenschaften von netezatimeseriesnode	Werte	Eigenschaftsbeschreibung
earliest_time	Ganzzahl	Start- und Endwerte, wenn timerange_setting auf SP gesetzt ist. Format wie beim time_points-Wert. Wenn das Feld time_points beispielsweise ein Datum enthält, sollte hier auch ein Datum enthalten sein. Beispiel: set NZ_DT1.timerange_setting = 'SP' set NZ_DT1.earliest_time = '1921-01-01' set NZ_DT1.latest_time = '2121-01-01'
latest_time	Datum Zeit Zeitmarke	
arima_setting	SD SP	Einstellung für den ARIMA-Algorithmus (nur verwendet, wenn algorithm_name auf ARIMA gesetzt ist): SD - systembestimmt SP - benutzerdefiniert Wenn arima_setting = SP angegeben ist, verwenden Sie die folgenden Parameter, um die saisonalen und nicht saisonalen Werte festzulegen. Beispiel (nur nicht saisonal): set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol	less	ARIMA - Operator für die Parameter p, d, q, sp, sd und sq: less - kleiner als eq - gleich lesseq - kleiner-gleich
d_symbol	eq	
q_symbol	lesseq	
sp_symbol		
sd_symbol		
sq_symbol		
p (Missing Values)	Ganzzahl	ARIMA - nicht saisonale Autokorrelationsmaße.
q	Ganzzahl	ARIMA - nicht saisonaler Ableitungswert.
d	Ganzzahl	ARIMA - nicht saisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.
sp	Ganzzahl	ARIMA - saisonale Autokorrelationsmaße.

Tabelle 203. Eigenschaften von "netezatimeseriesnode" (Forts.).

Eigenschaften von netezatimeseriesnode	Werte	Eigenschaftsbeschreibung
sq	Ganzzahl	ARIMA - saisonaler Ableitungswert.
sd	Ganzzahl	ARIMA - saisonale Anzahl von Ordnungen des gleitenden Durchschnitts im Modell.
advanced_setting	SD SP	Legt fest, wie erweiterte Einstellungen behandelt werden: SD - systembestimmt SP - benutzerdefiniert über period, units_period und forecast_setting. Beispiel: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
period	Ganzzahl	Länge des saisonalen Zyklus, die in Verbindung mit units_period angegeben wird. Wird nicht für Spektralanalyse verwendet.
units_period	ms s min h d wk q y	Einheiten für period: ms - Millisekunden s - Sekunden min - Minuten h - Stunden d - Tage wk - Wochen q - Quartale y - Jahre Beispiel: Verwenden Sie für einen wöchentliche Zeitreihe 1 für period und wk für units_period.
forecast_setting	forecasthorizon forecasttimes	Gibt an, wie Vorhersagen gemacht werden.
forecast_horizon	Ganzzahl Datum Zeit Zeitmarke	Wenn forecast_setting = forecasthorizon angegeben ist, wird ein Endpunktwert für die Vorhersage angegeben. Format wie beim time_points-Wert. Wenn das Feld time_points beispielsweise ein Datum enthält, sollte hier auch ein Datum enthalten sein.
forecast_times	Ganzzahl Datum Zeit Zeitmarke	Wenn forecast_setting = forecasttimes angegeben ist, werden die für die Vorhersagen zu verwendenden Werte angegeben. Format wie beim time_points-Wert. Wenn das Feld time_points beispielsweise ein Datum enthält, sollte hier auch ein Datum enthalten sein.

Tabelle 203. Eigenschaften von "netezatimeseriesnode" (Forts.).

Eigenschaften von netezatimeseriesnode	Werte	Eigenschaftsbeschreibung
include_history	Flag	Gibt an, ob historische Werte bei der Ausgabe berücksichtigt werden sollen.
include_interpolated_values	Flag	Gibt an, ob interpolierte Werte bei der Ausgabe berücksichtigt werden sollen. Wird nicht verwendet, wenn include_history auf false gesetzt ist.

Verallgemeinertes lineares Netezza-Modell

Die folgenden Eigenschaften sind für Knoten des Typs netezzaglmnode verfügbar.

Tabelle 204. Eigenschaften von "netezzaglmnode".

Eigenschaften von netezzaglmnode	Werte	Eigenschaftsbeschreibung
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	Verteilungstyp. Die Standardeinstellung ist bernoulli.
dist_params	Zahl	Zu verwendender Wert für Verteilungsparameter. Wird nur verwendet, wenn distribution auf Negativebinomial gesetzt ist.
trials	Ganzzahl	Wird nur verwendet, wenn distribution auf Binomial gesetzt ist. Wenn es sich bei der Zielantwort um eine Reihe von Ereignissen handelt, die während Tests auftreten, enthält das Feld target die Anzahl der Ereignisse und das Feld trials die Anzahl der Tests.
model_table	Feld	Der Name der Datenbanktabelle, in der das verallgemeinerte lineare Netezza-Modell gespeichert werden soll.
maxit	Ganzzahl	Die maximale Anzahl der Iterationen, die im Algorithmus vorgenommen werden sollen. Der Standardwert ist 20.
eps	Zahl	Der maximale Fehlerwert (in wissenschaftlicher Notation), bei dem der Algorithmus die Suche nach dem am besten passenden Modell beenden soll. Der Standardwert ist -3, d. h. 1E-3 bzw. 0,001.

Tabelle 204. Eigenschaften von "netezzaglmnode" (Forts.).

Eigenschaften von netezzaglmnode	Werte	Eigenschaftsbeschreibung
tol	Zahl	Der Wert (in wissenschaftlicher Notation), unterhalb dessen Fehler so behandelt werden, als hätten sie den Wert 0. Der Standardwert ist -7, es werden also Fehlerwerte unter 1E-7 (bzw. 0,0000001) als nicht signifikant gewertet.
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Zu verwendende Verknüpfungsfunktion. Die Standardeinstellung ist logit.
link_params	Zahl	Für die Verknüpfungsfunktion zu verwendender Parameterwert. Wird nur verwendet, wenn link_funktion auf power oder oddspower gesetzt ist.
interaction	[[{Spaltennamen1],[Niveaus1}], {[Spaltennamen2],[Niveaus2]}, ...,{[SpaltennamenN],[NiveausN]}],]	Gibt die Interaktionen zwischen Feldern an. Spaltennamen ist eine Liste von Eingabefeldern und Niveau ist für jedes Feld immer 0. Beispiel: [[{"K", "BP", "Sex", "K"}, [0,0,0,0] }, { "Age", "Na" }, [0,0]]]
intercept	Flag	Wenn true gesetzt ist, wird konstanter Term in das Modell einbezogen.

Eigenschaften von Netezza-Modellnuggets

Folgende Eigenschaften haben alle Modellnuggets von Netezza-Datenbanken gemeinsam.

Tabelle 205. Allgemeine Eigenschaften von Netezza-Modellnuggets

Allgemeine Eigenschaften von Netezza-Modellnuggets	Werte	Eigenschaftsbeschreibung
Verbindung	Zeichenfolge	Die Verbindungszeichenfolge für die Netezza-Datenbank, in der das Modell gespeichert ist.
table_name	Zeichenfolge	Der Name der Datenbanktabelle, in der das Modell gespeichert werden soll.

Die anderen Eigenschaften des Modellnuggets stimmen mit denen für den zugehörigen Modellierungsknoten überein.

Die Scriptnamen des Modellnuggets lauten wie folgt.

Tabelle 206. Scriptnamen von Netezza-Modellnuggets

Modellnugget	Scriptname
Entscheidungsbaum	applynetezadectreenode
K-Means	applynetezzakmeansnode
Bayes-Netz	applynetezabayesnode
Naive Bayes	applynetezanaivebayesnode
KNN	applynetezzaknnnode
Divisives Clustering	applynetezadivclusternode
PCA	applynetezzapcanode
Regressionsbaum	applynetezzaregtreenode
Lineare Regression	applynetezzalinieregressionnode
Zeitreihen	applynetezzatimeseriesnode
Verallgemeinert linear	applynetezzaglmnode

Kapitel 16. Eigenschaften von Ausgabeknoten

Die Eigenschaften von Ausgabeknoten unterscheiden sich von denen anderer Knotentypen. Statt auf eine bestimmte Knotenoption zu verweisen, speichern Ausgabeknoteneigenschaften eine Referenz zum Ausgabeobjekt. Dies ist nützlich, wenn ein Wert aus einer Tabelle als Streamparameter festgelegt wird.

In diesem Abschnitt werden die für Ausgabeknoten verfügbaren Scripteigenschaften beschrieben.

Eigenschaften von "analysisnode"



Der Analyseknoten evaluiert die Fähigkeit von Vorhersagemodellen, genaue Vorhersagen zu generieren. Mit Analyseknoten werden verschiedene Vergleiche zwischen den vorhergesagten Werten und den tatsächlichen Werten für ein oder mehrere Modellnuggets angestellt. Sie können außerdem Vorhersagemodelle miteinander vergleichen.

Beispiel

```
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")
```

Tabelle 207. Eigenschaften von "analysisnode".

Eigenschaften von analysisnode	Datentyp	Eigenschaftsbeschreibung
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_format	Text (.txt) HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps.
by_fields	list	

Tabelle 207. Eigenschaften von "analysisnode" (Forts.).

Eigenschaften von analysisnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
coincidence	Flag	
performance	Flag	
evaluation_binary	Flag	
confidence	Flag	
threshold	Zahl	
improve_accuracy	Zahl	
inc_user_measure	Flag	
user_if	Ausdr	
user_then	Ausdr	
user_else	Ausdr	
user_compute	[Mean Sum Min Max SDev]	

Eigenschaften von "dataauditnode"



Der Data Audit-Knoten bietet einen umfassenden ersten Einblick in die Daten mit statistischen Funktionen, Histogrammen und der Verteilung für die einzelnen Felder sowie Informationen zu Ausreißern, fehlenden Werten und Extremwerten. Die Ergebnisse werden in einer übersichtlichen Matrix dargestellt, die sortiert werden kann und als Grundlage für die Erzeugung normal großer Diagramme und Datenvorbereitungsknoten dient.

Beispiel

```

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")

```

Tabelle 208. Eigenschaften von "dataauditnode".

Eigenschaften von dataauditnode	Datentyp	Eigenschaftsbeschreibung
custom_fields	Flag	
fields	[Feld1 ... FeldN]	
overlay	Feld	

Tabelle 208. Eigenschaften von "dataauditnode" (Forts.).

Eigenschaften von dataauditnode	Datentyp	Eigenschaftsbeschreibung
display_graphs	Flag	Dient zur Aktivierung bzw. Inaktivierung der Anzeige von Diagrammen in der Ausgabematrix.
basic_stats	Flag	
advanced_stats	Flag	
median_stats	Flag	
calculate	Anzahl Breakdown	Dient zur Berechnung fehlender Werte. Sie können eine der beiden Berechnungsmethoden, beide Methoden oder auch keine der Methoden auswählen.
outlier_detection_method	std iqr	Dient zur Angabe der Erkennungsmethode für Ausreißer und Extremwerte.
outlier_detection_std_outlier	Zahl	Wenn für outlier_detection_method die Option std verwendet wird, wird die Zahl angegeben, die für die Definition der Ausreißer verwendet werden soll.
outlier_detection_std_extreme	Zahl	Wenn für outlier_detection_method die Option std verwendet wird, wird die Zahl angegeben, die für die Definition der Extremwerte verwendet werden soll.
outlier_detection_iqr_outlier	Zahl	Wenn für outlier_detection_method die Option iqr verwendet wird, wird die Zahl angegeben, die für die Definition der Ausreißer verwendet werden soll.
outlier_detection_iqr_extreme	Zahl	Wenn für outlier_detection_method die Option iqr verwendet wird, wird die Zahl angegeben, die für die Definition der Extremwerte verwendet werden soll.
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps.

Tabelle 208. Eigenschaften von "dataauditnode" (Forts.).

Eigenschaften von dataauditnode	Datentyp	Eigenschaftsbeschreibung
paginate_output	Flag	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	Zahl	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	Zeichenfolge	

Eigenschaften von "matrixnode"



Der Matrixknoten erstellt eine Tabelle, die die Beziehungen zwischen den Feldern aufzeigt. Dieser Knoten dient am häufigsten zur Darstellung der Beziehung zwischen zwei symbolischen Feldern, kann jedoch auch zum Aufzeigen der Beziehungen zwischen Flagfeldern oder numerischen Feldern herangezogen werden.

Beispiel

```
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)
```

Tabelle 209. Eigenschaften von "matrixnode".

Eigenschaften von matrixnode	Datentyp	Eigenschaftsbeschreibung
fields	Ausgewählt Flags Numerics	
row	Feld	
column	Feld	
include_missing_values	Flag	Gibt an, ob benutzerdefiniert fehlende Werte (leer) und systemdefiniert fehlende Werte (null) in die Zeilen- und Spaltenausgabe eingeschlossen werden sollen.

Tabelle 209. Eigenschaften von "matrixnode" (Forts.).

Eigenschaften von matrixnode	Datentyp	Eigenschaftsbeschreibung
cell_contents	CrossTabs Function	
function_field	Zeichenfolge	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Ascending Descending	
highlight_top	Zahl	Wenn ungleich 0, dann ist die Eigenschaft wahr.
highlight_bottom	Zahl	Wenn ungleich 0, dann ist die Eigenschaft wahr.
display	[Counts Expected Residuen RowPct ColumnPct TotalPct]	
include_totals	Flag	
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps. Sowohl für das Format Formatted als auch für das Format Delimited kann der Modifikator transposed verwendet werden, der die Zeilen und Spalten in der Tabelle transponiert.
paginate_output	Flag	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	Zahl	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
full_filename	Zeichenfolge	

Eigenschaften von "meansnode"



Der Mittelwertknoten vergleicht die Mittelwerte zwischen unabhängigen Gruppen oder zwischen Paaren von in Bezug stehenden Feldern, um zu testen, ob ein signifikanter Unterschied vorliegt. So können Sie beispielsweise die Einnahmen vor und nach der Durchführung einer Werbeaktion vergleichen oder die Einnahmen, die von Kunden stammen, die keine Werbebetriebe erhielten, mit den Einnahmen von Kunden vergleichen, die von der Werbeaktion erreicht wurden.

Beispiel

```
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")
```

Tabelle 210. Eigenschaften von "meansnode".

Eigenschaften von meansnode	Datentyp	Eigenschaftsbeschreibung
means_mode	BetweenGroups BetweenFields	Gibt den Typ der Mittelwertstatistik an, die für die Daten ausgeführt werden soll.
test_fields	[Feld1 ... Feldn]	Gibt das Testfeld an, wenn means_mode auf BetweenGroups gesetzt ist.
grouping_field	Feld	Gibt das Gruppierungsfeld an.
paired_fields	[{Feld1 Feld2} {Feld3 Feld4} ...]	Gibt die zu verwendenden Feldpaare an, wenn means_mode auf BetweenFields gesetzt ist.
label_correlations	Flag	Gibt an, ob Korrelationsbeschriftungen in der Ausgabe angezeigt werden sollen. Diese Einstellung gilt nur, wenn means_mode auf BetweenFields gesetzt ist.
correlation_mode	Wahrscheinlichkeitsfunktionen Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
weak_label	Zeichenfolge	
medium_label	Zeichenfolge	
strong_label	Zeichenfolge	
weak_below_probability	Zahl	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	Zahl	Trennwert für starke Korrelationen.

Tabelle 210. Eigenschaften von "meansnode" (Forts.).

Eigenschaften von meansnode	Datentyp	Eigenschaftsbeschreibung
weak_below_absolute	Zahl	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	Zahl	Trennwert für starke Korrelationen.
unimportant_label	Zeichenfolge	
marginal_label	Zeichenfolge	
important_label	Zeichenfolge	
unimportant_below	Zahl	Trennwert für niedrige Feldwichtigkeit. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
important_above	Zahl	
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Zu verwendender Name.
output_mode	Screen File	Gibt den Zielort für die vom Ausgabeknoten erstellte Ausgabe an.
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	Gibt den Ausgabebetyp an.
full_filename	Zeichenfolge	
output_view	Simple Advanced	Gibt an, ob die einfache oder die erweiterte Ansicht in der Ausgabe angezeigt werden soll.

Eigenschaften von "reportnode"



Der Berichtsknoten erstellt formatierte Berichte, die sowohl festen Text als auch Daten und andere aus den Daten abgeleitete Ausdrücke enthalten. Das Format des Berichts wird mithilfe von Textvorlagen festgelegt, mit denen der feste Text und die Datenausgabekonstruktionen definiert werden. Sie können eine benutzerdefinierte Textformatierung angeben; hierzu stehen HTML-Tags in der Vorlage sowie Optionen auf der Registerkarte "Ausgabe" zur Verfügung. Sie können Datenwerte und andere bedingte Ausgaben mithilfe von CLEM-Ausdrücken in der Vorlage aufnehmen.

Beispiel

```
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)
```

Tabelle 211. Eigenschaften von "reportnode".

Eigenschaften von reportnode	Datentyp	Eigenschaftsbeschreibung
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	HTML (.html) Text (.txt) Output (.cou)	Dient zur Angabe des Ausgabetyps.
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
text	Zeichenfolge	
full_filename	Zeichenfolge	
highlights	Flag	
title	Zeichenfolge	
lines_per_page	Zahl	

Eigenschaften von "routputnode"



Mit dem Knoten "Routput" können Sie Daten und die Ergebnisse des Modellscorings mithilfe Ihres eigenen benutzerdefinierten R-Scripts analysieren. Die Ausgabe von der Analyse kann Text oder grafisch sein. Die Ausgabe wird der Registerkarte **Ausgabe** des Managerbereichs hinzugefügt. Alternativ kann die Ausgabe in eine Datei umgeleitet werden.

Tabelle 212. Eigenschaften von "routputnode"

Eigenschaften von routputnode	Datentyp	Eigenschaftsbeschreibung
Syntax	Zeichenfolge	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	Flag	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	Flag	
output_name	Auto Custom	
custom_name	Zeichenfolge	
output_to	Screen File	
output_type	Graph Text	
full_filename	Zeichenfolge	
graph_file_type	HTML COU	

Tabelle 212. Eigenschaften von "routputnode" (Forts.)

Eigenschaften von routputnode	Datentyp	Eigenschaftsbeschreibung
text_file_type	HTML TEXT COU	

Eigenschaften von "setglobalsnode"



Mit dem Globalwerteknoten werden die Daten gescannt und Übersichtswerte berechnet, die in CLEM-Ausdrücken herangezogen werden können. Mit diesem Knoten können Sie beispielsweise die Statistiken für das Feld *Alter* berechnen und dann den Gesamtmittelwert für *Alter* in CLEM-Ausdrücken verwenden. Fügen Sie hierzu die Funktion @GLOBAL_MEAN(alter) ein.

Beispiel

```
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)
```

Tabelle 213. Eigenschaften von "setglobalsnode".

Eigenschaften von setglobalsnode	Datentyp	Eigenschaftsbeschreibung
globals	[Sum Mean Min Max SDev]	Strukturierte Eigenschaft, bei der festzulegende Felder mit folgender Syntax referenziert werden müssen: node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	Flag	
show_preview	Flag	

Eigenschaften von "simevalnode"



Der Simulationsevaluierungsknoten wertet ein angegebenes vorausgesagtes Zielfeld aus und stellt Verteilungs- und Korrelationsinformationen zu dem Zielfeld dar.

Tabelle 214. Eigenschaften von "simevalnode".

Eigenschaften von simevalnode	Datentyp	Eigenschaftsbeschreibung
target	Feld	
iteration	Feld	
presorted_by_iteration	boolesch	
max_iterations	Zahl	
tornado_fields	[Feld1...FeldN]	
plot_pdf	boolesch	
plot_cdf	boolesch	

Tabelle 214. Eigenschaften von "simevalnode" (Forts.).

Eigenschaften von simevalnode	Datentyp	Eigenschaftsbeschreibung
show_ref_mean	boolesch	
show_ref_median	boolesch	
show_ref_sigma	boolesch	
num_ref_sigma	Zahl	
show_ref_pct	boolesch	
ref_pct_bottom	Zahl	
ref_pct_top	Zahl	
show_ref_custom	boolesch	
ref_custom_values	[Zahl1...ZahlN]	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	boolesch	
pct_table	Quartiles Intervals Custom	
pct_intervals_num	Zahl	
pct_custom_values	[Zahl1...ZahlN]	

Eigenschaften von "simfitnode"



Der Simulationsanpassungsknoten untersucht die statistische Verteilung der Daten in jedem Feld und generiert (oder aktualisiert) einen Simulationsgenerierungsknoten, wobei jedem Feld die am besten passende Verteilung zugewiesen wird. Der Simulationsgenerierungsknoten kann dann zum Generieren simulierter Daten verwendet werden.

Tabelle 215. Eigenschaften von "simfitnode".

Eigenschaften von simfitnode	Datentyp	Eigenschaftsbeschreibung
build	Node XMLExport Both	
use_source_node_name	boolesch	
source_node_name	Zeichenfolge	Der benutzerdefinierte Name des Quellenknotens, der generiert oder aktualisiert wird.
use_cases	All LimitFirstN	
use_case_limit	Ganzzahl	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	Ganzzahl	
parameter_xml_filename	Zeichenfolge	
generate_parameter_import	boolesch	

Eigenschaften von "statisticsnode"



Der Statistikknoten liefert grundlegende Übersichtsdaten zu numerischen Feldern. Er berechnet Übersichtsstatistiken für einzelne Felder und für die Korrelationen zwischen den Feldern.

Beispiel

```
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")
```

Tabelle 216. Eigenschaften von "statisticsnode".

Eigenschaften von statisticsnode	Datentyp	Eigenschaftsbeschreibung
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Text (.txt) HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps.
full_filename	Zeichenfolge	
examine	list	
correlate	list	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Wahrscheinlichkeits- funktionen Absolute	Gibt an, ob die Korrelationen nach Wahrscheinlichkeit oder anhand des absoluten Werts beschriftet werden sollen.
label_correlations	Flag	
weak_label	Zeichenfolge	
medium_label	Zeichenfolge	
strong_label	Zeichenfolge	

Tabelle 216. Eigenschaften von "statisticsnode" (Forts.).

Eigenschaften von statisticsnode	Datentyp	Eigenschaftsbeschreibung
weak_below_probability	Zahl	Wenn correlation_mode auf Probability gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_probability	Zahl	Trennwert für starke Korrelationen.
weak_below_absolute	Zahl	Wenn correlation_mode auf Absolute gesetzt ist, wird hier der Trennwert für schwache Korrelationen angegeben. Hierbei muss es sich um einen Wert zwischen 0 und 1 handeln, beispielsweise 0,90.
strong_above_absolute	Zahl	Trennwert für starke Korrelationen.

Eigenschaften von "statisticsoutputnode"



Mit dem Statistics-Ausgabeknoten können Sie eine IBM SPSS Statistics-Prozedur aufrufen, um Ihre IBM SPSS Modeler-Daten zu analysieren. Es stehen zahlreiche IBM SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften von "statisticsoutputnode"“ auf Seite 308.

Eigenschaften von "tablenode"



Der Tabellenknoten zeigt die Daten in Tabellenform an, die auch in eine Datei geschrieben werden kann. Diese Vorgehensweise empfiehlt sich immer dann, wenn die Datenwerte überprüft oder in leicht lesbarer Form exportiert werden sollen.

Beispiel

```
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)
```

Tabelle 217. Eigenschaften von "tablenode".

Eigenschaften von tablenode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Bei Datenträger-, Daten- oder HTML-Ausgabe gibt diese Eigenschaft den Namen der Ausgabedatei an.
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.

Tabelle 217. Eigenschaften von "tablenode" (Forts.).

Eigenschaften von tablenode	Datentyp	Eigenschaftsbeschreibung
output_name	Zeichenfolge	Wenn use_output_name wahr ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	Formatted (.tab) Delimited (.csv) HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps.
transpose_data	Flag	Transponiert die Daten vor dem Export, sodass die Zeilen Felder und die Spalten Datensätze darstellen.
paginate_output	Flag	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	Zahl	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.
highlight_expr	Zeichenfolge	
output	Zeichenfolge	Eine schreibgeschützte Eigenschaft, die eine Referenz zur letzten vom Knoten erstellten Tabelle enthält.
value_labels	[[Wert Beschriftungszeichenfolge] {Wert Beschriftungszeichenfolge} ...]	Gibt Beschriftungen für Wertpaare an.
display_places	Ganzzahl	Legt die Dezimalstellen für das Feld bei der Anzeige fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Mit dem Wert -1 wird der Streamstandard verwendet.
export_places	Ganzzahl	Legt die Dezimalstellen für das Feld beim Exportieren fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL). Mit dem Wert -1 wird der Streamstandard verwendet.
decimal_separator	DEFAULT PERIOD COMMA	Legt das Dezimaltrennzeichen für das Feld fest (gilt nur für Felder mit dem Speichertyp REELLE ZAHL).

Tabelle 217. Eigenschaften von "tablenode" (Forts.).

Eigenschaften von tablenode	Datentyp	Eigenschaftsbeschreibung
date_format	"TTMMJJ" "MMTTJJ" "JJMMTT" "JJJJMMTT" "JJJJTTT" DAY MONTH "TT-MM-JJ" "TT-MM-JJJJ" "MM-TT-JJ" "MM-TT-JJJJ" "TT-MON-JJ" "TT-MON-JJJJ" "JJJJ-MM-TT" "TT.MM.JJ" "TT.MM.JJJJ" "MM.TT.JJJJ" "TT.MON.JJ" "TT.MON.JJJJ" "TT/MM/JJ" "TT/MM/JJJJ" "MM/TT/JJ" "MM/TT/JJJJ" "TT/MON/JJ" "TT/MON/JJJJ" MON JJJJ q Q JJJJ ww WK JJJJ	Legt das Datumsformat für das Feld fest (gilt nur für Felder mit dem Speichertyp DATE oder TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Legt das Zeitformat für das Feld fest (gilt nur für Felder mit dem Speichertyp TIME oder TIMESTAMP).
column_width	Ganzzahl	Legt die Spaltenbreite für das Feld fest. Mit dem Wert -1 wird die Spaltenbreite auf Auto (Automatisch) gesetzt.
justify	AUTO CENTER LEFT RIGHT	Legt die Spaltenausrichtung für das Feld fest.

Eigenschaften von "transformnode"



Mit dem Transformationsknoten können Sie die Ergebnisse von Transformationen auswählen und in einer Vorschau anzeigen, bevor Sie sie auf ausgewählte Felder anwenden.

Beispiel

```
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)
```

Tabelle 218. Eigenschaften von "transformnode".

Eigenschaften von transformnode	Datentyp	Eigenschaftsbeschreibung
fields	[Feld1... FieldN]	Die bei der Transformation zu verwendenden Felder.
formula	All Select	Gibt an, ob alle oder nur die ausgewählten Transformationen berechnet werden sollen.
formula_inverse	Flag	Gibt an, ob die inversen Transformation verwendet werden soll.
formula_inverse_offset	Zahl	Gibt eine relative Datenadresse an, die für die Formel verwendet werden soll. Standardmäßig auf 0 gesetzt, sofern vom Benutzer nicht anders angegeben.
formula_log_n	Flag	Gibt an, ob die \log_n -Transformation verwendet werden soll.
formula_log_n_offset	Zahl	
formula_log_10	Flag	Gibt an, ob die \log_{10} -Transformation verwendet werden soll.
formula_log_10_offset	Zahl	
formula_exponential	Flag	Gibt an, ob die exponentielle Transformation (e^x) verwendet werden soll.
formula_square_root	Flag	Gibt an, ob die Quadratwurzeltransformation verwendet werden soll.
use_output_name	Flag	Gibt an, ob ein benutzerdefinierter Ausgabename verwendet wird.
output_name	Zeichenfolge	Wenn use_output_name true ist, gibt diese Eigenschaft den zu verwendenden Namen an.
output_mode	Screen File	Dient zur Angabe des Zielorts für die vom Ausgabeknoten erstellte Ausgabe.
output_format	HTML (.html) Output (.cou)	Dient zur Angabe des Ausgabetyps.
paginate_output	Flag	Wenn output_format auf HTML gesetzt ist, wird hiermit die Ausgabe in Seiten unterteilt.
lines_per_page	Zahl	Bei Verwendung mit paginate_output wird die Anzahl der Zeilen pro Ausgabeseite angegeben.

Tabelle 218. Eigenschaften von "transformnode" (Forts.).

Eigenschaften von transformnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Gibt den für die Dateiausgabe zu verwendenden Dateinamen an.

Kapitel 17. Eigenschaften von Exportknoten

Allgemeine Eigenschaften von Exportknoten

Folgende Eigenschaften haben alle Exportknoten gemeinsam:

Table 219. Allgemeine Eigenschaften von Exportknoten

Eigenschaft	Werte	Eigenschaftsbeschreibung
publish_path	Zeichenfolge	Geben Sie den Stammnamen für die veröffentlichten Image- und Parameterdateien an.
publish_metadata	Flag	Gibt an, ob eine Metadatendatei erzeugt wird, welche die Ein- und Ausgaben des Bilds und der zugehörigen Datenmodelle beschreibt.
publish_use_parameters	Flag	Gibt an, ob Streamparameter in der *.par-Datei enthalten sind.
publish_parameters	Zeichenfolgeliste	Geben Sie die Parameter an, die eingeschlossen werden sollen.
execute_mode	export_data publish	Gibt an, ob der Knoten ohne Veröffentlichung des Streams ausgeführt wird oder ob der Stream automatisch beim Ausführen des Knotens veröffentlicht wird.

Eigenschaften von "asexport"

Der Analytic Server-Export ermöglicht Ihnen die Ausführung eines Streams unter HDFS (Hadoop Distributed File System).

Beispiel

```
node = stream.create("asexport", "My node")
node.setPropertyValue("data_source", "Drug1n")
node.setPropertyValue("export_mode", "overwrite")
```

Table 220. Eigenschaften von "asexport".

Eigenschaften von asexport	Datentyp	Eigenschaftsbeschreibung
data_source	Zeichenfolge	Der Name der Datenquelle.
export_mode	Zeichenfolge	Gibt an, ob die exportierten Daten an die vorhandene Datenquelle angehängt (append) werden oder die vorhandene Datenquelle überschreiben (overwrite).

Eigenschaften von "cognosexportnode"



Der IBM Cognos BI-Exportknoten exportiert Daten in einem Format, das von Cognos BI-Datenbanken gelesen werden kann.

Für diesen Knoten müssen Sie eine Cognos-Verbindung und eine ODBC-Verbindung definieren.

Cognos-Verbindung

Im Folgenden finden Sie die Eigenschaften für die Cognos-Verbindung.

Tabelle 221. Eigenschaften von cognosexportnode

Eigenschaften von cognosexportnode	Datentyp	Eigenschaftsbeschreibung
cognos_connection	{ <i>"Feld"</i> , <i>"Feld"</i> , ... , <i>"Feld"</i> }	<p>Eine Listeneigenschaft mit den Verbindungsdetails für den Cognos-Server. Format: {"Cognos-Server-URL", login_mode, "Namespace", "Benutzername", "Kennwort"}</p> <p>Dabei gilt: Cognos-Server-URL ist die URL des Cognos-Servers, der die Quelle enthält. login_mode gibt an, ob eine anonyme Anmeldung verwendet wird, und ist entweder true oder false; bei Angabe von true sollten die folgenden Felder auf "" gesetzt werden. Namespace gibt den Sicherheitsanbieter für die Authentifizierung an, mit dem Sie sich beim Server anmelden. Benutzername und Kennwort sind die Daten, die zur Anmeldung beim Cognos-Server verwendet werden.</p> <p>Anstelle von login_mode sind die folgenden Modi verfügbar:</p> <ul style="list-style-type: none"> anonymousMode. Beispiel: {'Cognos-Server-URL', 'anonymousMode', "Namespace", "Benutzername", "Kennwort"} credentialMode. Beispiel: {'Cognos-Server-URL', 'credentialMode', "Namespace", "Benutzername", "Kennwort"} storedCredentialMode. Beispiel: {'Cognos-Server-URL', 'storedCredentialMode', "gespeicherter_Berechtigungs-nachweisname"} <p>Dabei ist gespeicherter_Berechtigungsname der Name eines Cognos-Berechtigungs-nachweises im Repository.</p>
cognos_package_name	<i>Zeichenfolge</i>	Pfad und Name des Cognos-Pakets, an die Sie Daten exportieren, z. B.: /Public Folders/MyPackage
cognos_datasource	<i>Zeichenfolge</i>	
cognos_export_mode	Publish ExportFile	
cognos_filename	<i>Zeichenfolge</i>	

ODBC-Verbindung

Die Eigenschaften für die ODBC-Verbindung sind identisch mit denen, die im nächsten Bereich für `databaseexportnode` aufgelistet sind, mit der Ausnahme, dass die Eigenschaft der Datenquelle nicht gültig ist.

Eigenschaften von "databaseexportnode"



Der Datenbankexportknoten schreibt Daten in eine ODBC-kompatible relationale Datenquelle. Um Daten in eine ODBC-Datenquelle schreiben zu können, muss die betreffende Datenquelle bereits vorhanden sein und Sie benötigen Schreibzugriff dafür.

Beispiel

```
...
```

Use this sample with `fraud.str` from demo folder
Assumes a datasource named "MyDatasource" has been configured

```
...
```

```
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)
```

```
# Export tab
```

```
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)
```

```
# Schema dialog
```

```
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")
```

```
# Indexes dialog
```

```
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP INDEX <index-name>
ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", [{"fields", ["id", "region"]}]
```

Table 222. Eigenschaften von "databaseexportnode".

Eigenschaften von <code>databaseexportnode</code>	Datentyp	Eigenschaftsbeschreibung
<code>datasource</code>	<i>Zeichenfolge</i>	
<code>username</code>	<i>Zeichenfolge</i>	
<code>password</code>	<i>Zeichenfolge</i>	

Tabelle 222. Eigenschaften von "databaseexportnode" (Forts.).

Eigenschaften von databaseexportnode	Datentyp	Eigenschaftsbeschreibung
epassword	Zeichenfolge	Dieser Slot ist während der Ausführung schreibgeschützt. Um ein codiertes Kennwort zu erstellen, verwenden Sie das Kennworttool im Menü "Extras". Weitere Informationen finden Sie im Thema „Erstellen eines verschlüsselten Kennworts“ auf Seite 55.
table_name	Zeichenfolge	
write_mode	Create Append Merge	
map	Zeichenfolge	Ordnet einen Streamfeldnamen zu einer Datenbankspalte zu (nur gültig, wenn write_mode auf Merge eingestellt ist). Für eine Zusammenführung müssen alle Felder zugeordnet sein, damit sie exportiert werden. Feldnamen, die in der Datenbank nicht vorhanden sind, werden als neue Spalten hinzugefügt.
key_fields	list	Gibt an, dass das Streamfeld für "key" verwendet wird. Die map-Eigenschaft zeigt, welche Entsprechung in der Datenbank vorhanden ist.
join	Database Add	
drop_existing_table	Flag	
delete_existing_rows	Flag	
default_string_size	Ganzzahl	
type		Strukturierte Eigenschaft, mit der das Schema festgelegt wird.
generate_import	Flag	
use_custom_create_table_command	Flag	Mit dem Slot <i>custom_create_table</i> ändern Sie den SQL-Standardbefehl CREATE TABLE.
custom_create_table_command	Zeichenfolge	Gibt eine Befehlszeichenfolge an, die statt des SQL-Standardbefehls CREATE TABLE verwendet werden soll.
use_batch	Flag	Bei den folgenden Eigenschaften handelt es sich um erweiterte Optionen für das Massenladen von Datenbanken. Mit dem Wert "True" für Use_batch werden zeilenweise Commits zur Datenbank inaktiviert.
batch_size	Zahl	Gibt Anzahl der Datensätze an, die an die Datenbank gesendet werden sollen, bevor die Übertragung in den Speicher erfolgt.

Tabelle 222. Eigenschaften von "databaseexportnode" (Forts.).

Eigenschaften von databaseexportnode	Datentyp	Eigenschaftsbeschreibung
bulk_loading	Off ODBC External	Gibt den Typ für das Massenladen an. Zusätzliche Optionen für ODBC und External sind unten aufgeführt.
not_logged	Flag	
odbc_binding	Row Column	Geben Sie eine zeilen- oder spaltenweise Bindung für das Massenladen über ODBC an.
loader_delimit_mode	Tab Space Other	Geben Sie für das Massenladen über ein externes Programm das Trennzeichen an. Wählen Sie Other in Verbindung mit der Eigenschaft loader_other_delimiter zur Angabe von Trennzeichen, wie z. B. Komma (,).
loader_other_delimiter	Zeichenfolge	
specify_data_file	Flag	Mit dem Flag "True" wird die unten genannte Eigenschaft data_file aktiviert, mit der Sie den Dateinamen und den Pfad für das Speichern beim Massenladen in die Datenbank angeben können.
data_file	Zeichenfolge	
specify_loader_program	Flag	Mit dem Flag "True" wird die unten genannte Eigenschaft loader_program aktiviert, mit der Sie den Namen und den Speicherort eines externen Ladescripts oder Ladeprogramms angeben können.
loader_program	Zeichenfolge	
gen_logfile	Flag	Mit dem Flag "True" wird die unten genannte Eigenschaft logfile_name aktiviert, mit der Sie den Namen einer Datei zur Erzeugung eines Fehlerprotokolls auf dem Server angeben können.
logfile_name	Zeichenfolge	
check_table_size	Flag	Mit dem Flag "True" wird die Tabellenprüfung ermöglicht, um sicherzustellen, dass die Zunahme der Größe der Datenbanktabelle der Anzahl der aus IBM SPSS Modeler exportierten Zeilen entspricht.
loader_options	Zeichenfolge	Legen Sie für das Ladeprogramm zusätzliche Argumente fest, z. B. -comment und -specialdir.
export_db_primarykey	Flag	Gibt an, ob es sich bei einem bestimmten Feld um einen Primärschlüssel handelt.

Tabelle 222. Eigenschaften von "databaseexportnode" (Forts.).

Eigenschaften von databaseexportnode	Datentyp	Eigenschaftsbeschreibung
use_custom_create_index_command	Flag	Bei true wird hiermit benutzerdefinierte SQL für alle Indizes aktiviert.
custom_create_index_command	Zeichenfolge	Gibt den SQL-Befehl an, der zum Erstellen von Indizes verwendet wird, wenn benutzerdefinierte SQL aktiviert ist. (Dieser Wert kann für bestimmte Indizes außer Kraft gesetzt werden (siehe unten).)
indexes.INDEXNAME.fields		Erstellt bei Bedarf den angegebenen Index und listet die in diesen Index aufzunehmenden Feldnamen auf.
INDEXNAME "use_custom_create_index_command"	Flag	Wird zum Aktivieren bzw. Inaktivieren der benutzerdefinierten SQL für einen bestimmten Index verwendet. Siehe unten stehende Beispiele.
INDEXNAME "custom_create_index_command"	Zeichenfolge	Gibt die für den angegebenen Index verwendete benutzerdefinierte SQL an. Siehe unten stehende Beispiele.
indexes.INDEXNAME.remove	Flag	Bei True wird hiermit der angegebene Index aus der Menge der Indizes entfernt.
table_space	Zeichenfolge	Gibt den zu erstellenden Tabellenbereich an.
use_partition	Flag	Gibt an, dass das Verteilungsfeld verwendet werden soll.
partition_field	Zeichenfolge	Gibt den Inhalt des Verteilungsfelds an.

Anmerkung: Bei einigen Datenbanken können Sie angeben, dass Datenbanktabellen für den Export mit Komprimierung erstellt werden sollen (z. B. die Entsprechung von CREATE TABLE MYTABLE (...) COMPRESS YES; in SQL). Die Eigenschaften use_compression und compression_mode werden zur Unterstützung dieser Funktion wie folgt bereitgestellt.

Tabelle 223. Eigenschaften von "databaseexportnode" mit Komprimierungsfunktionen.

Eigenschaften von databaseexportnode	Datentyp	Eigenschaftsbeschreibung
use_compression	boolesch	Wenn auf True gesetzt, werden Tabellen für den Export mit Komprimierung erstellt.
compression_mode	Row Page	Legt das Komprimierungsniveau für SQL Server-Datenbanken fest.
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	Legt das Komprimierungsniveau für Oracle-Datenbanken fest. Beachten Sie, dass für die Werte OLTP, Query_High, Query_Low, Archive_High und Archive_Low mindestens Oracle 11gR2 erforderlich ist.

Beispiel für das Ändern des Befehls CREATE INDEX für einen bestimmten Index:

```
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)>"])
```

Diese Änderung ist auch über eine Hashtabelle möglich:

```
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)>"})
```

Eigenschaften von "datacollectionexportnode"



Der IBM SPSS Data Collection-Exportknoten gibt Daten in dem von der Marktforschungssoftware IBM SPSS Data Collection verwendeten Format aus. Um diesen Knoten verwenden zu können, muss die IBM SPSS Data Collection Data Library installiert sein.

Beispiel

```
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)
```

Tabelle 224. Eigenschaften von "datacollectionexportnode"

Eigenschaften von datacollectionexportnode	Datentyp	Eigenschaftsbeschreibung
metadata_file	Zeichenfolge	Name der zu exportierenden Metadatendatei.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	Flag	Gibt an, ob die exportierte .mdd-Datei IBM SPSS Data Collection-Systemvariablen enthalten soll.
casedata_file	Zeichenfolge	Der Name der .sav-Datei, in die die Falldaten exportiert werden.
generate_import	Flag	

Eigenschaften von "excelexportnode"



Der Excel-Exportknoten gibt Daten im Microsoft Excel-Format (.xls) aus. Optional können Sie auswählen, dass bei der Ausführung des Knotens Excel automatisch gestartet und die exportierte Datei geöffnet werden soll.

Beispiel

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xls")
```

```

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)

```

Tabelle 225. Eigenschaften von "excelexportnode"

Eigenschaften von excelexportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	
excel_file_type	Excel2003 Excel2007	
export_mode	Create Append	
inc_field_names	Flag	Gibt an, ob Feldnamen in die erste Zeile des Arbeitsblattes eingefügt werden sollen.
start_cell	Zeichenfolge	Gibt die Startzelle für den Export an.
worksheet_name	Zeichenfolge	Name des zu schreibenden Arbeitsblattes.
launch_application	Flag	Gibt an, ob Excel für die resultierende Datei aufgerufen werden soll. Beachten Sie, dass der Pfad für den Start von Excel im Dialogfeld "Hilfsanwendungen" (Menü "Extras", "Hilfsanwendungen") angegeben werden muss.
generate_import	Flag	Gibt an, ob ein Excel-Importknoten generiert werden soll, der die exportierte Datendatei liest.

Eigenschaften von "outputfilenode"



Der Flatfile-Export gibt Daten in einer Textdatei mit Trennzeichen aus. Diese Vorgehensweise eignet sich für das Exportieren von Daten, die von anderen Analyse- oder Tabellenkalkulationsprogrammen gelesen werden sollen.

Beispiel

```

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("delimiter_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)

```

Tabelle 226. Eigenschaften von "outputfilenode"

Eigenschaften von outputfilenode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Name der Ausgabedatei.
write_mode	Overwrite Append	
inc_field_names	Flag	
use_newline_after_records	Flag	
delimit_mode	Comma Tab Space Andere	
other_delimiter	Element	
quote_mode	None Single Double Andere	
other_quote	Flag	
generate_import	Flag	
encoding	StreamDefault SystemDefault "UTF-8"	

Eigenschaften von "sasexportnode"



Mit dem SAS-Exportknoten werden Daten in das SAS-Format ausgegeben, die dann in SAS oder in SAS-kompatible Softwarepakete eingelesen werden können. Drei SAS-Dateiformate sind verfügbar: SAS für Windows/OS2, SAS für UNIX sowie SAS Version 7/8.

Beispiel

```
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)
```

Tabelle 227. Eigenschaften von "sasexportnode"

Eigenschaften von sasexportnode	Datentyp	Eigenschaftsbeschreibung
format	Windows UNIX SAS7 SAS8	Beschriftungsfelder für die Eigenschaft "Variante".
full_filename	Zeichenfolge	
export_names	NamesAndLabels NamesAsLabels	Dient der Zuordnung von Feldnamen von IBM SPSS Modeler zu IBM SPSS Statistics- oder SAS-Variablenamen nach dem Export.
generate_import	Flag	

Eigenschaften von "statisticsexportnode"



Der Statistikexportknoten gibt Daten im IBM SPSS Statistics-Format *.sav* oder *.zsav* aus. Die *.sav*- oder *.zsav*-Dateien können von IBM SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cache-Dateien in IBM SPSS Modeler verwendet.

Eine Beschreibung der Eigenschaften für diesen Knoten finden Sie in „Eigenschaften von "statisticsexportnode"“ auf Seite 309.

Eigenschaften des Knotens "tm1export"



Der IBM Cognos TM1-Exportknoten exportiert Daten in einem Format, das von Cognos TM1-Datenbanken gelesen werden kann.

Tabelle 228. Eigenschaften des Knotens "tm1export".

Eigenschaften des Knotens tm1export	Datentyp	Eigenschaftsbeschreibung
pm_host	Zeichenfolge	Der Hostname. Beispiel: TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	{ "Feld", "Feld", ... ,"Feld" }	Eine Listeneigenschaft mit den Verbindungsdetails für den TM1-Server. Format: ["TM1-Servername", "TM1-Benutzername", "TM1-Kennwort"] Beispiel: TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])
selected_cube	Feld	Der Name des Cubes, in den Sie Daten exportieren. Beispiel: TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")
spssfield_tm1element_mapping	Liste	Das zuzuordnende TM1-Element muss Teil der Spaltendimension für die ausgewählte Cube-Ansicht sein. Format: [{"Parameter1", "Wert"}, ..., {"ParameterN", "Wert"}] Beispiel: TM1_export.setPropertyValue("spssfield_tm1element_mapping", [["plan_version", "plan_version"], ["plan_department", "plan_department"]])

Eigenschaften von "xmlexportnode"



Der XML-Exportknoten gibt Daten an eine Datei im XML-Format aus. Optional können Sie einen XML-Quellenknoten erstellen, um die exportierten Daten wieder in der Stream einzulesen.

Beispiel


```

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"}])

```

Tabella 229. Eigenschaften von "xmlexportnode"

Eigenschaften von xmlexportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	(erforderlich) Vollständiger Pfad und Dateiname der XML-Exportdatei.
use_xml_schema	Flag	Legt fest, ob ein XML-Schema (XSD- oder DTD-Datei) für die Steuerung der Struktur der exportierten Daten verwendet wird.
full_schema_filename	Zeichenfolge	Vollständiger Pfad und Dateiname der zu verwendenden XSD- oder DTD-Datei. Erforderlich, wenn use_xml_schema auf "wahr" gesetzt ist.
generate_import	Flag	Generiert einen XML-Quellenknoten, der die exportierten Daten wieder in den Stream einliest.
Datensätze	Zeichenfolge	XPath-Ausdruck, der die Datensatzgrenze angibt.
Karte	Zeichenfolge	Ordnet den Feldnamen der XML-Struktur zu.

Kapitel 18. IBM SPSS Statistics-Knoteneigenschaften

Eigenschaften von "statisticsimportnode"



Der Statistikdateiknoten liest Daten aus dem Dateiformat *.sav* oder *.zsav* ein, das von IBM SPSS Statistics verwendet wird, sowie in IBM SPSS Modeler gespeicherte Cachedateien, die ebenfalls dasselbe Format verwenden.

Beispiel

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import", 200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drugIn.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

Tabelle 230. Eigenschaften von "statisticsimportnode".

Eigenschaften von statisticsimportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	Der vollständige Dateiname mit Pfad.
password	Zeichenfolge	Das Kennwort. Der Parameter password muss vor dem Parameter file_encrypted festgelegt werden.
file_encrypted	Flag	Ob die Datei kennwortgeschützt ist.
import_names	NamesAndLabels LabelsAsNames	Methode für die Behandlung von Variablennamen und -beschriftungen.
import_data	DataAndLabels LabelsAsData	Methode für die Behandlung von Werten und Beschriftungen.
use_field_format_for_storage	boolesch	Gibt an, ob IBM SPSS Statistics-Feldformatinformationen beim Import verwendet werden.

Eigenschaften von "statistictransformnode"



Der Statistics-Transformationsknoten führt eine Auswahl von IBM SPSS Statistics-Syntaxbefehlen für Datenquellen in IBM SPSS Modeler aus. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Beispiel

```
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)
```

Tabelle 231. Eigenschaften von "statisticstransformnode"

Eigenschaften von statisticstransformnode	Datentyp	Eigenschaftsbeschreibung
Syntax	Zeichenfolge	
check_before_saving	Flag	Überprüft die eingegebene Syntax vor dem Speichern der Einträge. Zeigt eine Fehlermeldung an, wenn die Syntax ungültig ist.
default_include	Flag	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.
include	Flag	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.
new_name	Zeichenfolge	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.

Eigenschaften von "statisticsmodelnode"



Mithilfe des Statistics-Modellknotens können Sie Ihre Daten analysieren und bearbeiten, indem Sie IBM SPSS Statistics-Prozeduren ausführen, die PMML erzeugen. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Beispiel

```
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")
```

Eigenschaften von statisticsmodelnode	Datentyp	Eigenschaftsbeschreibung
Syntax	Zeichenfolge	
default_include	Flag	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.
include	Flag	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.
new_name	Zeichenfolge	Weitere Informationen finden Sie im Thema „Eigenschaften von "filternode"“ auf Seite 132.

Eigenschaften von "statisticsoutputnode"



Mit dem Statistics-Ausgabeknoten können Sie eine IBM SPSS Statistics-Prozedur aufrufen, um Ihre IBM SPSS Modeler-Daten zu analysieren. Es stehen zahlreiche IBM SPSS Statistics-Analyseprozeduren zur Verfügung. Für diesen Knoten ist eine lizenzierte Kopie von IBM SPSS Statistics erforderlich.

Beispiel

```
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")
```

Tabelle 232. Eigenschaften von "statisticsoutputnode"

Eigenschaften von statisticsoutputnode	Datentyp	Eigenschaftsbeschreibung
mode	Dialog Syntax	Wählt die Option "IBM SPSS Statistics-Dialogfeld" oder Syntaxeditor aus
Syntax	Zeichenfolge	
use_output_name	Flag	
output_name	Zeichenfolge	
output_mode	Screen File	
full_filename	Zeichenfolge	
file_type	HTML SPV SPW	

Eigenschaften von "statisticsexportnode"



Der Statistikexportknoten gibt Daten im IBM SPSS Statistics-Format *.sav* oder *.zsav* aus. Die *.sav*- oder *.zsav*-Dateien können von IBM SPSS Statistics Base und anderen Produkten gelesen werden. Dieses Format wird auch für Cache-Dateien in IBM SPSS Modeler verwendet.

Beispiel

```
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)
```

Tabelle 233. Eigenschaften von "statisticsexportnode".

Eigenschaften von statisticsexportnode	Datentyp	Eigenschaftsbeschreibung
full_filename	Zeichenfolge	
file_type	Standard Komprimiert	Datei im <i>sav</i> - oder <i>zsav</i> -Format speichern.
encrypt_file	Flag	Ob die Datei kennwortgeschützt ist.
password	Zeichenfolge	Das Kennwort.
launch_application	Flag	

Tabelle 233. Eigenschaften von "statisticsexportnode" (Forts.).

Eigenschaften von statisticsexportnode	Datentyp	Eigenschaftsbeschreibung
export_names	NamesAndLabels NamesAsLabels	Dient der Zuordnung von Feldnamen von IBM SPSS Modeler zu IBM SPSS Statistics- oder SAS-Variablennamen nach dem Export.
generate_import	Flag	

Kapitel 19. Superknoteneigenschaften

In den folgenden Tabellen werden die für Superknoten spezifischen Eigenschaften beschrieben. Beachten Sie, dass allgemeine Knoteneigenschaften auch für Superknoten gelten.

Table 234. Eigenschaften von Endsuperknoten

Eigenschaftsname	Eigenschaftstyp/Liste der Werte	Eigenschaftsbeschreibung
execute_method	Script Normal	
script	Zeichenfolge	

Superknotenparameter

Mithilfe von Scripts können Sie SuperNode-Parameter mit allgemeinem Format erstellen bzw. festlegen:
`mySuperNode.setParameterValue("minvalue", 30)`

Sie können den Parameterwert wie folgt abrufen:

```
value mySuperNode.getParameterValue("minvalue")
```

Suchen von vorhandenen Superknoten

Sie können Superknoten in Streams mit der Funktion `findByType()` suchen:

```
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)
```

Festlegen von Eigenschaften für gekapselte Knoten

Sie können Eigenschaften für einzelne, in einem Superknoten gekapselte Knoten festlegen, indem Sie auf das untergeordnete Diagramm im Superknoten zugreifen. Nehmen Sie beispielsweise an, dass Sie einen Quellensuperknoten mit einem gekapselten Knoten "Variable Datei" zum Einlesen der Daten haben. Sie können den Namen der zu lesenden Datei (mithilfe der Eigenschaft `full_filename` angegeben) durch Zugreifen auf das untergeordnete Diagramm und Suchen des relevanten Knotens wie folgt weitergeben:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Erstellen von Superknoten

Wenn Sie einen Superknoten und seinen Inhalt völlig neu erstellen wollen, können Sie dies auf ähnliche Weise tun, indem Sie den Superknoten erstellen, auf das untergeordnete Diagramm zugreifen und die gewünschten Knoten erstellen. Sie müssen außerdem sicherstellen, dass die Knoten im Superknotendiagramm auch mit den Eingabe- und/oder Ausgabebestückerknoten verbunden sind. Beispiel für das Erstellen eines Prozesssuperknotens:

```
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)
```

Anhang A. Knotennamenreferenz

In diesem Abschnitt erhalten Sie eine Referenz für die Scriptnamen der Knoten in IBM SPSS Modeler.

Modellnuggetnamen

Modellnuggets (auch als "generierte Modelle" bezeichnet) können ebenso wie Knoten- und Ausgabeobjekte nach Typ referenziert werden. In der folgenden Tabelle werden die Referenznamen für Modellobjekte aufgeführt.

Beachten Sie, dass diese Namen speziell zur Referenzierung von Modellnuggets in der Modellpalette (in der rechten oberen Ecke des IBM SPSS Modeler-Fensters) verwendet werden. Zur Referenzierung von Modellknoten, die zu Scoring-Zwecken zu einem Stream hinzugefügt wurden, wird ein anderes Set von Namen mit dem Präfix `apply...` verwendet. Weitere Informationen finden Sie in Eigenschaften von Modellnuggetknoten.

Hinweis: Unter normalen Umständen wird die Referenzierung von Modellen sowohl anhand des Namens als auch anhand des Typs empfohlen, um Verwirrungen zu vermeiden.

Tabelle 235. Namen von Modellnuggets (Modellierungspalette).

Modellname	Modell
anomalydetection	Anomalie
apriori	Apriori
autoclassifier	Automatisches Klassifikationsmerkmal
autocluster	Automatisches Clustering
autonumeric	Auto-Numerisch
bayesnet	Bayes-Netz
c50	C5.0
carma	Carma
cart	C&R-Baum
chaid	CHAID
coxreg	Cox-Regression
decisionlist	Entscheidungsliste
Diskriminanz	Diskriminanz
Faktor	Faktor/PCA
featureselection	Merkmalauswahl
genlin	Verallgemeinerte lineare Regression
glmm	GLMM
kmeans	K-Means
knn	<i>k</i> -Nächste-Nachbarn
kohonen	Kohonen
linear	Linear
logreg	Logistische Regression
neuralnetwork	Netz

Tabelle 235. Namen von Modellnuggets (Modellierungspalette) (Forts.).

Modellname	Modell
quest	QUEST
regression	Lineare Regression
sequence	Sequenz
slrm	Lernfähiges Antwortmodell
statisticsmodel	IBM SPSS Statistics-Modell
svm	Support Vector Machine
timeseries	Zeitreihen
twostep	Two Step

Tabelle 236. Namen von Modellnuggets (Datenbankmodellierungspalette).

Modellname	Modell
db2imcluster	IBM ISW-Clustering
db2imlog	IBM ISW Logistische Regression
db2imnb	IBM ISW Naive Bayes
db2imreg	IBM ISW-Regression
db2imtree	IBM ISW-Entscheidungsbaum
msassoc	MS-Assoziationsregeln
msbayes	MS Naive Bayes
mscluster	MS-Clustering
mslogistic	MS - Logistische Regression
msneuralnetwork	MS - Neuronales Netz
msregression	MS - Lineare Regression
mssequencecluster	MS-Sequenzclustering
mstimeseries	MS Time Series
mstree	MS-Entscheidungsbaum
netezzabayes	Netezza-Bayes-Netz
netezzadectree	Netezza-Entscheidungsbaum
netezzadivcluster	Netezza - Divisives Clustering
netezzaglm	Verallgemeinertes lineares Netezza-Modell
netezzakmeans	Netezza-K-Means
netezzaknn	Netezza-KNN
netezzalinegression	Netezza - Lineare Regression
netezzanaivebayes	Netezza - Naive Bayes
netezzapca	Netezza-PCA
netezzaregtree	Netezza-Regressionsbaum
netezzatimeseries	Netezza-Zeitreihe
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Oracle Decision Tree
oraglm	Oracle GLM

Tabelle 236. Namen von Modellnuggets (Datenbankmodellierungspalette) (Forts.).

Modellname	Modell
orakmeans	Oracle <i>k</i> -Means
oranb	Oracle Naive Bayes
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

Vermeidung doppelter Modellnamen

Bei der Verwendung von Scripts zur Bearbeitung generierter Modelle sollten Sie sich bewusst sein, dass das Zulassen doppelter Modellnamen zu mehrdeutigen Referenzen führen kann. Um dies zu vermeiden, sollten bei der Scripterstellung eindeutige Namen für die generierten Modelle erforderlich sein.

So legen Sie Optionen für doppelte Modellnamen fest:

1. Wählen Sie die folgenden Befehle aus den Menüs aus:
Extras > Benutzeroptionen
2. Klicken Sie auf die Registerkarte **Benachrichtigungen**.
3. Wählen Sie die Option **Bisheriges Modell ersetzen**, um die Vergabe doppelter Namen für generierte Modelle zu beschränken.

Das Verhalten der Scriptausführung kann zwischen SPSS Modeler und IBM SPSS Collaboration and Deployment Services variieren, wenn mehrdeutige Modellverweise vorliegen. Der SPSS Modeler-Client beinhaltet die Option "Bisheriges Modell ersetzen", bei dem automatisch Modelle mit demselben Namen ersetzt werden (z. B., wenn ein Script in mehreren Iterationen eine Schleife durchläuft und jedes Mal ein anderes Modell erstellt). Diese Option steht jedoch nicht zur Verfügung, wenn dasselbe Script in IBM SPSS Collaboration and Deployment Services ausgeführt wird. Sie können diese Situation vermeiden, indem Sie entweder das in den einzelnen Iterationen generierte Modell umbenennen, um mehrdeutige Verweise auf Modelle zu vermeiden, oder indem Sie das aktuelle Modell vor dem Ende der Schleife löschen (z. B. durch Hinzufügen der Anweisung `clear generated palette`).

Namen der Ausgabetypen

In der folgenden Tabelle werden alle Ausgabeobjekttypen und die Knoten, von denen Sie erstellt werden, aufgelistet. Eine vollständige Liste der für die einzelnen Ausgabeobjekttypen verfügbaren Exportformate finden Sie in der Eigenschaftsbeschreibung für den Knoten, der den Ausgabetypp erstellt. Diese Beschreibung finden Sie in Allgemeine Eigenschaften von Diagrammknoten und Eigenschaften von Ausgabeknoten.

Tabelle 237. Ausgabeobjekttypen und die Knoten, von denen sie erstellt werden.

Ausgabeobjekttyp	Knoten
analysisoutput	Analyse
collectionoutput	Sammlung
dataauditoutput	Data Audit
distributionoutput	Verteilung
evaluationoutput	Evaluation
histogramoutput	Histogramm
matrixoutput	Matrix
meansoutput	Mittelwerte

Tabelle 237. Ausgabeobjekttypen und die Knoten, von denen sie erstellt werden (Forts.).

Ausgabeobjekttyp	Knoten
multiplotoutput	Multiplot
plotoutput	Diagramm
qualityoutput	Qualität
reportdocumentoutput	Dieser Objekttyp stammt nicht aus einem Knoten; es handelt sich um die von einem Projektbericht erstellte Ausgabe.
reportoutput	Bericht
statisticsprocedureoutput	Statistics-Ausgabe
statisticsoutput	Statistics
tableoutput	Tabelle
timeplotoutput	Zeitdiagramm
weboutput	Internet

Anhang B. Migration von traditionellem Scripting zu Python-Scripting

Übersicht über die Migration traditioneller Scripts

In diesem Abschnitt erhalten Sie eine Zusammenfassung der Unterschiede zwischen Python-Scripting und traditionellem Scripting in IBM SPSS Modeler sowie Informationen zur Migration von traditionellen Scripts in Python-Scripts. In diesem Abschnitt finden Sie eine Liste der traditionellen SPSS Modeler-Standardbefehle und der entsprechenden Python-Befehle.

Allgemeine Unterschiede

Traditionelles Scripting beruht stark auf Betriebssystembefehlsscripts. Traditionelles Scripting ist zeilenorientiert. Obwohl es einige Blockstrukturen wie z. B. `if...then...else...endif` und `for...endfor` gibt, hat eine Einrückung in der Regel keine Bedeutung.

Bei Python-Scripting ist die Einrückung von Bedeutung. Zeilen, die zum selben logischen Block gehören, müssen gleich weit eingerückt sein.

Anmerkung: Darauf müssen Sie beim Kopieren und Einfügen von Python-Code achten. Eine Zeile, die mit Tabstopps eingerückt ist, könnte im Editor genauso aussehen wie eine Zeile, die mit Leerzeichen eingerückt ist. Das Python-Script generiert jedoch einen Fehler, da diese Zeilen nicht als gleich weit eingerückt gelten.

Scriptingkontext

Der Scriptingkontext definiert die Umgebung, in der das Script ausgeführt wird, z. B. den Stream oder den Superknoten, der das Script ausführt. Bei traditionellem Scripting ist der Kontext implizit, d. h. es wird angenommen, dass alle Knotenreferenzen in einem Stream-Script in dem Stream liegen, der das Script ausführt.

Bei Python-Scripting wird der Scriptingkontext explizit über das Modul `modeler.script` angegeben. Beispiel: ein Python-Stream-Script kann mit dem folgenden Code auf den Stream zugreifen, der das Script ausführt:

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

Funktionen im Zusammenhang mit Streams können dann über das zurückgegebene Objekt aufgerufen werden.

Befehle und Funktionen

Traditionelles Scripting ist befehlsorientiert. Das heißt, dass jede Zeile des Scripts typischerweise mit dem auszuführenden Befehl beginnt und dann die Parameter folgen. Beispiel:

```
connect 'Type':typenode to :filternode  
rename :derivenode as "Compute Total"
```

Python verwendet Funktionen, die üblicherweise über ein Objekt (Modul, Klasse oder Objekt) aufgerufen werden, das die Funktion definiert. Beispiel:

```
stream = modeler.script.stream()  
typenode = stream.findByType("type", "Type")  
filternode = stream.findByType("filter", None)  
stream.link(typenode, filternode)  
derive.setLabel("Compute Total")
```

Literale und Kommentare

Einige Literal- und Kommentarbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 238. Zuordnung von traditionellem Scripting zu Python-Scripting für Literale und Kommentare.

Traditionelles Scripting	Python-Scripting
Ganzzahl, z. B. 4	Gleich
Gleitkommazahl, z. B. 0.003	Gleich
Zeichenfolgen in einfachen Anführungszeichen, z. B. 'Hallo'	Gleich Anmerkung: Zeichenfolgeliteralen mit Nicht-ASCII-Zeichen muss ein u vorangestellt werden, damit sie als Unicode dargestellt werden.
Zeichenfolgen in doppelten Anführungszeichen, z. B. "Nochmal Hallo"	Gleich Anmerkung: Zeichenfolgeliteralen mit Nicht-ASCII-Zeichen muss ein u vorangestellt werden, damit sie als Unicode dargestellt werden.
Lange Zeichenfolgen, z. B. """Dies ist eine Zeichenfolge, die mehrere Zeilen umfasst"""	Gleich
Listen, z. B. [1 2 3]	[1, 2, 3]
Variablenreferenzen, z. B. set x = 3	x = 3
Zeilenfortsetzung (\), z. B. set x = [1 2 \ 3 4]	x = [1, 2,\n3, 4]
Blockkommentar, z. B. /* Dies ist ein langer Kommentar in einer Zeile. */	""" Dies ist ein langer Kommentar in einer Zeile. """
Zeilenkommentar, z. B. set x = 3 # make x 3	x = 3 # make x 3
undef	None
true	Wahr
false	Falsch

Operatoren

Einige Operatorbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 239. Zuordnung von traditionellem Scripting zu Python-Scripting für Operatoren.

Traditionelles Scripting	Python-Scripting
ZAHL1 + ZAHL2 LISTE + ELEMENT LISTE1 + LISTE2	ZAHL1 + ZAHL2 LIST.append(ELEMENT) LIST1.extend(LISTE2)
ZAHL1 - ZAHL2 LISTE - ELEMENT	ZAHL1 - ZAHL2 LIST.remove(ELEMENT)
ZAHL1 * ZAHL2	ZAHL1 * ZAHL2

Tabelle 239. Zuordnung von traditionellem Scripting zu Python-Scripting für Operatoren (Forts.).

Traditionelles Scripting	Python-Scripting
ZAHL1 / ZAHL2	ZAHL1 / ZAHL2
= ==	==
/= /==	!=
X ** Y	X ** Y
X < Y X <= Y X > Y X >= Y	X < Y X <= Y X > Y X >= Y
X div Y X rem Y X mod Y	X // Y X % Y X % Y
and or not(AUSDR)	and or not AUSDR

Bedingte Befehle und Schleifenbefehle

Für einige bedingte Befehle und Schleifenbefehle, die üblicherweise in IBM SPSS Modeler verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 240. Zuordnung von traditionellem Scripting zu Python-Scripting für bedingte Befehle und Schleifenbefehle.

Traditionelles Scripting	Python-Scripting
for VAR from INT1 to INT2 ... endfor	for VAR in range(INT1, INT2): ... ODER VAR = INT1 while VAR <= INT2: ... VAR += 1
for VAR in LISTE ... endfor	for VAR in LISTE: ...
for VAR in_fields_to KNOTEN ... endfor	for VAR in KNOTEN.getInputDataModel(): ...
for VAR in_fields_at KNOTEN ... endfor	for VAR in KNOTEN.getOutputDataModel(): ...
if...then ... elseif...then ... else ... endif	if ...: ... elif ...: ... else: ...
with TYPE OBJECT ... endwith	Keine Entsprechung

Tabelle 240. Zuordnung von traditionellem Scripting zu Python-Scripting für bedingte Befehle und Schleifenbefehle (Forts.).

Traditionelles Scripting	Python-Scripting
var VAR1	Variablendeklaration nicht erforderlich

Variablen

Bei traditionellem Scripting werden Variablen deklariert, bevor sie referenziert werden. Beispiel:

```
var mynode
set mynode = create typenode at 96 96
```

Bei Python-Scripting werden Variablen bei ihrer ersten Referenzierung erstellt. Beispiel:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

Bei traditionellem Scripting müssen Referenzen auf Variablen explizit mit dem Operator ^ entfernt werden. Beispiel:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Wie bei den meisten Scriptsprachen ist dies bei Python-Scripting nicht erforderlich. Beispiel:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction", "Age", "Input")
```

Knoten-, Ausgabe- und Modelltypen

Bei traditionellem Scripting wird für die unterschiedlichen Objekttypen (Knoten, Ausgabe und Modell) in der Regel der Typ an den Objekttyp angehängt. Beispiel: der Knoten "derive" hat den Typ `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

Die IBM SPSS Modeler-API in Python schließt das Suffix `node` nicht ein, sodass der Knoten "derive" den Typ `derive` hat. Beispiel:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

Typnamen bei traditionellem Scripting und bei Python-Scripting unterscheiden sich nur darin, dass bei Python-Scripting das Typsuffix fehlt.

Eigenschaftsnamen

Eigenschaftsnamen sind bei traditionellem und bei Python-Scripting gleich. Beispiel: im Variablendateiknoten lautet die Eigenschaft, die die Dateiposition definiert, in beiden Scripting-Umgebungen `full_filename`.

Knotenreferenzen

Viele traditionelle Scripts verwenden eine implizite Suche, um den zu ändernden Knoten zu suchen und darauf zuzugreifen. Beispiel: die folgenden Befehle durchsuchen den aktuellen Stream nach einem Typknoten mit der Beschriftung "Type" und legen dann für die Richtung (Modellierungsrolle) das Feld "Age" als Eingabe und das Feld "Drug" als Ziel (vorherzusagender Wert) fest:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

Bei Python-Scripting müssen Knotenobjekte explizit gesucht werden, bevor die Funktion zum Festlegen des Eigenschaftswert aufgerufen wird. Beispiel:


```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Anmerkung: In diesem Fall muss "Target" in Anführungszeichen für Zeichenfolgen stehen.

Python-Scripts können alternativ die Aufzählung `ModelingRole` im Paket `modeler.api` verwenden.

Obwohl Scripts bei Python-Scripting länger sein können, führt dies zu einer besseren Laufzeitleistung, da die Suche nach dem Knoten in der Regel nur einmal erfolgt. Beim Beispiel mit traditionellem Scripting wird die Suche nach dem Knoten bei jedem Befehl ausgeführt.

Die Suche nach Knoten anhand der ID wird ebenfalls unterstützt. (Die Knoten-ID ist auf der Registerkarte "Anmerkungen" des Knotendialogs zu sehen.) Beispiel bei traditionellem Scripting:

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

Das folgende Script zeigt dasselbe Beispiel bei Python-Scripting:

```
typenode = stream.findById("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Abrufen und Festlegen von Eigenschaften

Traditionelles Scripting verwendet den Befehl `set` zum Festlegen eines Wertes. Der Ausdruck nach dem Befehl `set` kann eine Eigenschaftsdefinition sein. Das folgende Script zeigt zwei mögliche Scriptformate zum Festlegen einer Eigenschaft:

```
set <Knotenreferenz>.<Eigenschaft> = <Wert>
set <Knotenreferenz>.<verschlüsselte_Eigenschaft>.<Schlüssel> = <Wert>
```

Bei Python-Scripting wird dasselbe Ergebnis mit den Funktionen `setProperty()` und `setKeyedProperty()` erreicht. Beispiel:

```
Objekt.setProperty(Eigenschaft, Wert)
Objekt.setKeyedProperty(verschlüsselte_Eigenschaft, Schlüssel, Wert)
```

Bei traditionellem Scripting kann mit dem Befehl `get` auf Eigenschaftswerte zugegriffen werden. Beispiel:

```
var n v
set n = get node :filternode
set v = ^n.name
```

Bei Python-Scripting wird dasselbe Ergebnis mit der Funktion `getProperty()` erreicht. Beispiel:

```
n = stream.findByType("filter", None)
v = n.getProperty("name")
```

Bearbeiten von Streams

Bei traditionellen Scripting wird der Befehl `create` zum Erstellen eines neuen Knotens verwendet. Beispiel:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

Bei Python-Scripting gibt es in Streams verschiedene Methoden zur Erstellung von Knoten. Beispiel:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

Bei traditionellen Scripting wird der Befehl `connect` zum Herstellen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
connect ^agg to ^select
```

Bei Python-Scripting wird die Methode `link` zum Herstellen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
stream.link(agg, select)
```

Bei traditionellen Scripting wird der Befehl `disconnect` zum Entfernen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
disconnect ^agg from ^select
```

Bei Python-Scripting wird die Methode `unlink` zum Entfernen von Verbindungen zwischen Knoten verwendet. Beispiel:

```
stream.unlink(agg, select)
```

Bei traditionellen Scripting wird der Befehl `position` zur Positionierung von Knoten im Streamerstellungsbereich oder zwischen anderen Knoten verwendet. Beispiel:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

Bei Python-Scripting wird dasselbe Ergebnis mit zwei separaten Methoden erreicht: `setXYPosition` und `setPositionBetween`. Beispiel:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Knotenoperationen

Einige Befehle für Knotenoperationen, die üblicherweise in IBM SPSS Modeler verwendet werden, haben funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 241. Zuordnung von traditionellem Scripting zu Python-Scripting für Knotenoperationen.

Traditionelles Scripting	Python-Scripting
create <i>Knotenspezifikation</i> at x y	<code>Stream.create(Typ, Name)</code> <code>Stream.createAt(Typ, Name, x, y)</code> <code>Stream.createBetween(Typ, Name, preNode, postNode)</code> <code>Stream.createModelApplier(Modell, Name)</code>
connect <i>Ausgangsknoten</i> to <i>Zielknoten</i>	<code>Stream.link(Ausgangsknoten, Zielknoten)</code>
delete <i>Knoten</i>	<code>Stream.delete(Knoten)</code>
disable <i>Knoten</i>	<code>Stream.setEnabled(Knoten, False)</code>
enable <i>Knoten</i>	<code>Stream.setEnabled(Knoten, True)</code>
disconnect <i>Ausgangsknoten</i> from <i>Zielknoten</i>	<code>Stream.unlink(Ausgangsknoten, Zielknoten)</code> <code>Stream.disconnect(Knoten)</code>
duplicate <i>Knoten</i>	<code>Knoten.duplicate()</code>
execute <i>Knoten</i>	<code>Stream.runSelected(Knoten, Ergebnisse)</code> <code>Stream.runAll(Ergebnisse)</code>
flush <i>Knoten</i>	<code>Knoten.flushCache()</code>
position <i>Knoten</i> at x y	<code>Knoten.setXYPosition(x, y)</code>
position <i>Knoten</i> between <i>Knoten1</i> and <i>Knoten2</i>	<code>Knoten.setPositionBetween(Knoten1, Knoten2)</code>
rename <i>Knoten</i> as <i>Name</i>	<code>Knoten.setLabel(Name)</code>

Verwendung von Schleifen

Bei traditionellem Scripting werden es zwei Hauptschleifenoptionen unterstützt:

- *Gezählte Schleifen*, bei denen eine Indexvariable sich zwischen zwei ganzzahligen Grenzen bewegt.
- *Sequenzschleifen*, die eine Folge von Werten in einer Schleife durchlaufen und den aktuellen Wert an die Schleifenvariable binden.

Das folgende Script ist ein Beispiel für eine gezählte Schleife bei traditionellem Scripting:

```
for i from 1 to 10
  println ^i
endfor
```

Das folgende Script ist ein Beispiel für eine Sequenzschleife bei traditionellem Scripting:

```
var items
set items = [a b c d]

for i in items
  println ^i
endfor
```

Es können auch andere Typen von Schleifen verwendet werden:

- Durchlaufen der Modelle in der Modellpalette oder der Ausgaben in der Ausgabepalette.
- Durchlaufen der Felder, die in einen Knoten eintreten oder aus einem Knoten austreten.

Python-Scripting unterstützt auch unterschiedliche Schleifentypen. Das folgende Script ist ein Beispiel für eine gezählte Schleife bei Python-Scripting:

```
i = 1
while i <= 10:
  print i
  i += 1
```

Das folgende Script ist ein Beispiel für eine Sequenzschleife bei Python-Scripting:

```
items = ["a", "b", "c", "d"]
for i in items:
  print i
```

Die Sequenzschleife ist sehr flexibel. Wenn sie mit API-Methoden von IBM SPSS Modeler kombiniert wird, kann sie die Mehrzahl der Anwendungsfälle bei traditionellem Scripting unterstützen. Das folgende Beispiel zeigt, wie mit einer Sequenzschleife bei Python-Scripting die Felder durchlaufen werden können, die aus einem Knoten austreten:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
  print column.getColumnName()
```

Streams ausführen

Während der Streamausführung werden generierte Modelle oder Ausgabeobjekte zu einem der Objektmanager hinzugefügt. Bei traditionellen Scripting muss das Script entweder die erstellten Objekte im Objektmanager finden oder auf die zuletzt generierte Ausgabe aus dem Knoten zugreifen, der die Ausgabe erstellt hat.

Die Streamausführung in Python unterscheidet sich darin, dass alle von der Ausführung generierten Modell- oder Ausgabeobjekte in einer Liste zurückgegeben werden, die an die Ausführungsfunktion übergeben wird. Dadurch kann leichter auf die Ergebnisse der Streamausführung zugegriffen werden.

Traditionelles Scripting unterstützt drei Befehle zur Streamausführung:

- `execute_all` - führt alle ausführbaren Endknoten im Stream aus.
- `execute_script` - führt das Stream-Script unabhängig von der Einstellung der Scriptausführung aus.
- `execute Knoten` - führt den angegebenen Knoten aus.

Python-Scripting unterstützt eine ähnliche Gruppe von Funktionen:

- `Stream.runAll(Ergebnisliste)` - führt alle ausführbaren Endknoten im Stream aus.
- `Stream.runScript(Ergebnisliste)` - führt das Stream-Script unabhängig von der Einstellung der Scriptausführung aus.
- `Stream.runSelected(Knotenarray, Ergebnisliste)` - führt das angegebene Set von Knoten in der aufgelisteten Reihenfolge auf.
- `Knoten.run(Ergebnisliste)` - führt den angegebenen Knoten aus.

Bei traditionellem Scripting kann eine Streamausführung mit dem Befehl `exit` und einem optionalen ganzzahligen Code beendet werden. Beispiel:

```
exit 1
```

Bei Python-Scripting wird dasselbe Ergebnis mit dem folgenden Script erreicht:

```
modeler.script.exit(1)
```

Zugriff auf Objekte über das Dateisystem und das Repository

Bei traditionellem Scripting können Sie einen vorhandenen Stream, ein vorhandenes Modell oder ein vorhandenes Ausgabeobjekt mit dem Befehl `open` öffnen: Beispiel:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

Bei Python-Scripting gibt es die Klasse `TaskRunner`, auf die von der Sitzung zugegriffen werden kann und mit der ähnliche Tasks ausgeführt werden können. Beispiel:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Wenn Sie bei traditionellen Scripting ein Objekt speichern wollen, können Sie den Befehl `save` verwenden. Beispiel:

```
save stream s as "c:/my streams/new_modeling.str"
```

Die entsprechende Vorgehensweise bei einem Python-Script ist die Verwendung der Klasse `TaskRunner`. Beispiel:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Auf IBM SPSS Collaboration and Deployment Services Repository basierende Operationen werden bei traditionellen Scripting über die Befehle `retrieve` und `store` unterstützt. Beispiel:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

Bei Python-Scripting wird die entsprechende Funktionalität über das der Sitzung zugeordnete Repository-Objekt abgerufen. Beispiel:

```
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)
```

Anmerkung: Für den Repository-Zugriff muss die Sitzung mit einer gültigen Repository-Verbindung konfiguriert worden sein.

Streamoperationen

Für einige Befehle für Streamoperationen, die üblicherweise in IBM SPSS Modeler verwendet werden, gibt es funktional entsprechende Befehle in Python-Scripts. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 242. Zuordnung von traditionellem Scripting zu Python-Scripting für Streamoperationen.

Traditionelles Scripting	Python-Scripting
create stream <i>STANDARDDATEINAME</i>	<i>Task-Runner.createStream(Name, autoVerbindung, autoVerwaltung)</i>
close stream	<i>Stream.close()</i>
clear stream	<i>Stream.clear()</i>
get stream <i>Stream</i>	Keine Entsprechung
load stream <i>Pfad</i>	Keine Entsprechung
open stream <i>Pfad</i>	<i>Task-Runner.openStreamFromFile(Pfad, autoVerwaltung)</i>
save <i>Stream</i> as <i>Pfad</i>	<i>Task-Runner.saveStreamToFile(Stream, Pfad)</i>
retrieive stream <i>Pfad</i>	<i>Repository.retrieveStream(Pfad, Version, Beschriftung, autoVerwaltung)</i>
store <i>Stream</i> as <i>Pfad</i>	<i>Repository.storeStream(Stream, Pfad, Beschriftung)</i>

Modelloperationen

Für einige Befehle für Modelloperationen, die in IBM SPSS Modeler häufig verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 243. Zuordnung von traditionellem Scripting zu Python-Scripting für Modelloperationen.

Traditionelles Scripting	Python-Scripting
open model <i>Pfad</i>	<i>Task-Runner.openModelFromFile(Pfad, autoVerwaltung)</i>
save <i>Modell</i> as <i>Pfad</i>	<i>Task-Runner.saveModelToFile(Modell, Pfad)</i>
retrieve model <i>Pfad</i>	<i>Repository.retrieveModel(Pfad, Version, Beschriftung, autoVerwaltung)</i>
store <i>Modell</i> as <i>Pfad</i>	<i>Repository.storeModel(Modell, Pfad, Beschriftung)</i>

Dokumentausgabeoperationen

Für einige Befehle für Dokumentausgabeoperationen, die in IBM SPSS Modeler häufig verwendet werden, gibt es entsprechende Befehle bei Python-Scripting. Dies könnte Ihnen dabei helfen, Ihre vorhandenen traditionellen SPSS Modeler-Scripts für die Verwendung in IBM SPSS Modeler 17 in Python-Scripts zu konvertieren.

Tabelle 244. Zuordnung von traditionellem Scripting zu Python-Scripting für Dokumentausgabeoperationen.

Traditionelles Scripting	Python-Scripting
open output <i>Pfad</i>	<i>Task-Runner.openDocumentFromFile(Pfad, autoVerwaltung)</i>
save <i>Ausgabe</i> as <i>Pfad</i>	<i>Task-Runner.saveDocumentToFile(Ausgabe, Pfad)</i>
retrieve output <i>Pfad</i>	<i>Repository.retrieveDocument(Pfad, Version, Beschriftung, autoVerwaltung)</i>

Tabelle 244. Zuordnung von traditionellem Scripting zu Python-Scripting für Dokumentausgabeoperationen (Forts.).

Traditionelles Scripting	Python-Scripting
store <i>Ausgabe</i> as <i>Pfad</i>	<code>Repository.storeDocument(<i>Ausgabe</i>, <i>Pfad</i>, <i>Beschriftung</i>)</code>

Weitere Unterschiede zwischen traditionellem Scripting und Python-Scripting

Traditionelle Scripts bieten Unterstützung zur Manipulation von IBM SPSS Modeler-Projekten. Python-Scripting unterstützt diese Funktion derzeit nicht.

Traditionelles Scripting bietet eine gewisse Unterstützung beim Laden von *Statusobjekten* (Kombinationen von Streams und Modellen). Statusobjekte werden seit IBM SPSS Modeler 8.0 nicht weiter unterstützt. Python-Scripting unterstützt keine Statusobjekte.

Python-Scripting bietet die folgenden zusätzlichen Funktionen, die bei traditionellem Scripting nicht verfügbar sind:

- Klassen- und Funktionsdefinitionen
- Fehlerbehandlung
- Fortgeschrittenere Ein-/Ausgabe-Unterstützung
- Externe und Fremdanbieter-Module

Bemerkungen

Diese Informationen wurden für weltweit angebotene Produkte und Dienstleistungen erarbeitet.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
USA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corp in den USA und/oder anderen Ländern. Weitere Produkt- und Servicennamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder der Tochtergesellschaften des Unternehmens in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA, anderen Ländern oder beidem.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Java und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

Weitere Produkt- und Servicennamen können Marken von IBM oder anderen Unternehmen sein.

Index

A

Ableitungsknoten
Eigenschaften 128
aggregatenode, Eigenschaften 105
Aggregatknöten
Eigenschaften 105
Analyseknöten
Eigenschaften 279
analysisnode, Eigenschaften 279
Analytic Server-Quellenknöten
Eigenschaften 83
Anhangknöten
Eigenschaften 105
Anmerkungen 21
Anomalieerkennungsmödelles
Knöten, Scripteigenschaften 167, 237
anomalydetectionnode, Eigenschaften 167
Anonymisierungsknöten
Eigenschaften 121
anonymizenode, Eigenschaften 121
Anweisungen 21
Anzeigen von IBM ISW-Zeitreihenmodellen
Knöten, Scripteigenschaften 260
appendnode, Eigenschaften 105
applyanomalydetectionnode, Eigenschaften 237
applyapriorinode, Eigenschaften 237
applyassociationrulesnode, Eigenschaften 242
applyautoclassifiernode, Eigenschaften 238
applyautoclusternode, Eigenschaften 238
applyautonumericnode, Eigenschaften 238
applybayesnetnode, Eigenschaften 239
applyc50node, Eigenschaften 239
applycarmanode, Eigenschaften 239
applycartnode, Eigenschaften 239
applychaidnode, Eigenschaften 240
applycoxregnode, Eigenschaften 240
applydb2imclusternode, Eigenschaften 265
applydb2imlognode, Eigenschaften 265
applydb2imnbnode, Eigenschaften 265
applydb2imregnode, Eigenschaften 265
applydb2imtreenode, Eigenschaften 265
applydecisionlistnode, Eigenschaften 241
applydiscriminantnode, Eigenschaften 241
applyfactornode, Eigenschaften 241
applyfeatureselectionnode, Eigenschaften 241
applygeneralizedlinearnode, Eigenschaften 242
applyglmnode, Eigenschaften 242
applykmeansnode, Eigenschaften 243
applyknnnode, Eigenschaften 243
applykohonenode, Eigenschaften 243

applylinearnode, Eigenschaften 243
applylogregnode, Eigenschaften 244
applymlslogisticnode, Eigenschaften 251
applymlsneuralnetworknode, Eigenschaften 251
applymlsregressionnode, Eigenschaften 251
applymlssequenceclusternode, Eigenschaften 251
applymlstimeseriesnode, Eigenschaften 251
applymlstreenode, Eigenschaften 251
applynetzezbayesnode, Eigenschaften 276
applynetzezzadectreenode, Eigenschaften 276
applynetzezzadivclusternode, Eigenschaften 276
applynetzezzakmeansnode, Eigenschaften 276
applynetzezzaknnnode, Eigenschaften 276
applynetzezzalineregressionnode, Eigenschaften 276
applynetzezzanaivebayesnode, Eigenschaften 276
applynetzezzapcanode, Eigenschaften 276
applynetzezzaregtreenode, Eigenschaften 276
applyneuralnetnode, Eigenschaften 244
applyneuralnetworknode, Eigenschaften 245
applyoraabnode, Eigenschaften 259
applyoradecisiontreenode, Eigenschaften 259
applyorakmeansnode, Eigenschaften 259
applyoranbnode, Eigenschaften 259
applyoranmfnode, Eigenschaften 259
applyoraoclusternode, Eigenschaften 259
applyorasvmnode, Eigenschaften 259
applyquestnode, Eigenschaften 245
applyregressionnode, Eigenschaften 246
applyselflearningnode, Eigenschaften 246
applysequencenode, Eigenschaften 246
applystpnnode, Eigenschaften 247
applysvmnode, Eigenschaften 246
applytimeseriesnode, Eigenschaften 247
applytwestepAS, Eigenschaften 247
applytwestepnode, Eigenschaften 247
Apriori-Mödelles
Knöten, Scripteigenschaften 169, 237
apriorinode, Eigenschaften 169
Argumente
Befehlsdatei 70
Repository-Verbindung für IBM SPSS
Analytic Server 70
Serververbindung 68
System 66

Argumente (*Forts.*)
Verbindung zu IBM SPSS Collaboration and Deployment Services Repository 69
Argumente übergeben 22
AS-Zeitintervallknöten
Eigenschaften 125
asexport, Eigenschaften 295
asimport, Eigenschaften 83
associationrulesnode, Eigenschaften 170
Assoziationsregelknöten
Eigenschaften 170
Assoziationsregelknötennugget
Eigenschaften 242
astimeintervalnode, Eigenschaften 125
Attribute definieren 27
Attribute hinzufügen 27
Ausführungsreihenfolge
mit Scripts ändern 53
Ausgabeknöten
Scripteigenschaften 279
Ausgabeobjekte
Scriptnamen 315
Ausgeblendete Variablen 28
Auswahlknöten
Eigenschaften 116
Auto-Numerisch, Modelle
Knöten, Scripteigenschaften 177, 238
autoclassifiernode, Eigenschaften 173
autoclusternode, Eigenschaften 175
autodataprepnode, Eigenschaften 122
Autom. Cluster, Knöten
Knöten, Scripteigenschaften 175
Autom. Cluster, Modelle
Knöten, Scripteigenschaften 238
Automatische Datenaufbereitung
Eigenschaften 122
Automatisches Klassifikationsmerkmal, Knöten
Knöten, Scripteigenschaften 173
Automatisches Klassifikationsmerkmal, Modelle
Knöten, Scripteigenschaften 238
autonumericnode, Eigenschaften 177

B

balancenode, Eigenschaften 106
Balancierungsknöten
Eigenschaften 106
Bayes-Netzmodelles
Knöten, Scripteigenschaften 178, 239
bayesnet, Eigenschaften 178
Bedingte Ausführung von Streams 6, 11
Befehlszeile
IBM SPSS Modeler ausführen 65
Liste der Argumente 66, 68, 69, 70
mehrere Argumente 70
Parameter 67
Scripts 56
Beispiele 22

Benutzereingabeknoten
Eigenschaften 98
Berichtknoten
Eigenschaften 285
binningnode, Eigenschaften 125
buildr, Eigenschaften 179

C

C&RT-Baummodelle
Knoten, Scripteigenschaften 183, 239
C5.0-Modelle
Knoten, Scripteigenschaften 180, 239
c50node, Eigenschaften 180
CARMA-Modelle
Knoten, Scripteigenschaften 181, 239
carmanode, Eigenschaften 181
cartnode, Eigenschaften 183
CHAID-Modelle
Knoten, Scripteigenschaften 185, 240
chaidnode, Eigenschaften 185
clear generated palette, Befehl 56
CLEM
Scripts 1
Codeblöcke 22
Codierte Kennwörter
Hinzufügen zu Scripts 55
cognosimport, Knoteneigenschaften 83
collectionnode, Eigenschaften 152
Cox-Regressionsmodelle
Knoten, Scripteigenschaften 187, 240
coxregnode, Eigenschaften 187

D

Data Audit-Knoten
Eigenschaften 280
dataauditnode, Eigenschaften 280
databaseexportnode, Eigenschaften 297
databasenode, Eigenschaften 85
datacollectionexportnode, Eigenschaften 301
datacollectionimportnode, Eigenschaften 87
dataviewimport, Eigenschaften 103
Datei (fest), Knoten
Eigenschaften 91
Datenansichtsquellenknoten
Eigenschaften 103
Datenbankexportknoten
Eigenschaften 297
Datenbankknoten
Eigenschaften 85
Datenbankmodellierung 249
db2imassocnode, Eigenschaften 260
db2imclusternode, Eigenschaften 260
db2imlognode, Eigenschaften 260
db2imnbnnode, Eigenschaften 260
db2imregnode, Eigenschaften 260
db2imsequencenode, Eigenschaften 260
db2imtimeseriesnode, Eigenschaften 260
db2imtreenode, Eigenschaften 260
decisionlist, Eigenschaften 189
derive_stbnode
Eigenschaften 107
derivnode, Eigenschaften 128

Diagramme 29
Diagrammknoten
Scripteigenschaften 151
Diagrammtafelknoten
Eigenschaften 156
Dichotomknoten
Eigenschaften 139
directedwebnode, Eigenschaften 164
discriminantnode, Eigenschaften 190
Diskriminanzmodelle
Knoten, Scripteigenschaften 190, 241
distinctnode, Eigenschaften 109
distributionnode, Eigenschaften 153
Duplikatknoten
Eigenschaften 109

E

Eigenschaften
allgemeine Scripts 74
Datenbankmodellierungsknoten 249
Filterknoten 71
Scripts 71, 73, 167, 237, 295
Stream 75
Superknoten 311
Eigenschaften festlegen 32
Ensemble-Knoten
Eigenschaften 130
ensemblenode, Eigenschaften 130
Enterprise-Ansichtsknoten
Eigenschaften 91
Entscheidungslistenmodelle
Knoten, Scripteigenschaften 189, 241
evaluationnode, Eigenschaften 154
Evaluierungsknoten
Eigenschaften 154
evimportnode, Eigenschaften 91
Excel-Exportknoten
Eigenschaften 301
Excel-Quellenknoten
Eigenschaften 90
excelexportnode, Eigenschaften 301
excelimportnode, Eigenschaften 90
Exportknoten
Knoten, Scripteigenschaften 295

F

factornode, Eigenschaften 192
featureselectionnode, Eigenschaften 5, 193
Fehlerprüfung
Scripts 56
Felder
Inaktivieren in Scripts 151
Felder umordnen, Knoten
Eigenschaften 136
Feldnamen
Ändern der Groß- und Kleinschreibung 53
fillernode, Eigenschaften 131
Filterknoten
Eigenschaften 132
filternode, Eigenschaften 132
fixedfilenode, Eigenschaften 91

Flags
Befehlszeilenargumente 65
mehrere Flags kombinieren 70
Flatfile-Knoten
Eigenschaften 302
flatfilenode, Eigenschaften 302
for, Befehl 53
Füllerknoten
Eigenschaften 131
Funktionen
bedingte Befehle 319
Dokumentausgabeoperationen 325
Knotenoperationen 322
Kommentare 318
Literale 318
Modelloperationen 325
Objektreferenzen 318
Operatoren 318
Schleifenbefehle 319
Streamoperationen 325

G

generated, Schlüsselwort 56
Generierte Modelle
Scriptnamen 313, 315
genlinnode, Eigenschaften 195
Georäumlicher Quellenknoten
Eigenschaften 94
Gerichteter Netzdiagrammknoten
Eigenschaften 164
GLMM-Modelle
Knoten, Scripteigenschaften 199, 242
glmnode, Eigenschaften 199
Globalwerteknoten
Eigenschaften 287
graphboardnode, Eigenschaften 156
gsdata_import, Knoteneigenschaften 94

H

Histogrammknoten
Eigenschaften 158
histogramnode, Eigenschaften 158
historynode, Eigenschaften 133

I

IBM Cognos BI-Quellenknoten
Eigenschaften 83
IBM Cognos TM1-Quellenknoten
Eigenschaften 98
IBM DB2-Modelle
Knoten, Scripteigenschaften 260
IBM ISW-Assoziationsmodelle
Knoten, Scripteigenschaften 260, 265
IBM ISW-Clustering-Modelle
Knoten, Scripteigenschaften 260, 265
IBM ISW-Entscheidungsbaummodelle
Knoten, Scripteigenschaften 260, 265
IBM ISW Logistische Regressionsmodelle
Knoten, Scripteigenschaften 260, 265
IBM ISW Naive Bayes-Modelle
Knoten, Scripteigenschaften 260, 265
IBM ISW-Regressionsmodelle
Knoten, Scripteigenschaften 260, 265

- IBM ISW-Sequenzmodelle
 - Knoten, Scripteigenschaften 260, 265
- IBM SPSS Analytic Server-Repository
 - Befehlszeilenargumente 70
- IBM SPSS Collaboration and Deployment Services Repository
 - Befehlszeilenargumente 69
 - Scripts 54
- IBM SPSS Data Collection, Exportknoten
 - Eigenschaften 301
- IBM SPSS Data Collection, Quellenknoten
 - Eigenschaften 87
- IBM SPSS Modeler
 - über Befehlszeile ausführen 65
- IBM SPSS Statistics-Ausgabeknoten
 - Eigenschaften 308
- IBM SPSS Statistics-Exportknoten
 - Eigenschaften 309
- IBM SPSS Statistics-Modelle
 - Knoten, Scripteigenschaften 308
- IBM SPSS Statistics-Quellenknoten
 - Eigenschaften 307
- IBM SPSS Statistics-Transformationsknoten
 - Eigenschaften 307
- IDs 21
- Iterationsschlüssel
 - Schleifen in Scripts 8
- Iterationsvariable
 - Schleifen in Scripts 9

J

- JSON-Inhaltsmodell 60
- Jython 17

K

- K-Means-Modelle
 - Knoten, Scripteigenschaften 202, 243
- Kennwörter
 - codiert 68
 - Hinzufügen zu Scripts 55
- Klasse definieren 26
- Klasse erstellen 27
- Klassierungsknoten
 - Eigenschaften 125
- kmeansnode, Eigenschaften 202
- KNN-Modelle
 - Knoten, Scripteigenschaften 243
- knnode, Eigenschaften 203
- Knoten
 - Ersetzung 35
 - importieren 35
 - Information 37
 - Links für Knoten aktivieren 34
 - Links für Knoten aufheben 34
 - Löschung 35
 - Namensreferenz 313
 - Schleifen in Scripts 53
- Knoten, Scripteigenschaften 249
 - Exportknoten 295
 - Modellierungsknoten 167
 - Modellnuggets 237
- Knoten erstellen 33, 34, 35
- Knoten suchen 31

- Kohonen-Modelle
 - Knoten, Scripteigenschaften 205, 243
- kohonennode, Eigenschaften 205
- Koordinatensystemreprojizierung
 - Eigenschaften 136

L

- Lernfähige Antwortmodelle
 - Knoten, Scripteigenschaften 221, 246
- linear, Eigenschaften 206
- Lineare Modelle
 - Knoten, Scripteigenschaften 206, 243
- Lineare Regression, Modelle
 - Knoten, Scripteigenschaften 218, 245, 246
- Listen 18
- Logistische Regressionsmodelle
 - Knoten, Scripteigenschaften 207, 244
- logregnode, Eigenschaften 207
- lowertoupper, Funktion 53

M

- Mathematische Methoden 23
- Matrixknoten
 - Eigenschaften 282
- matrixnode, Eigenschaften 282
- meansnode, Eigenschaften 284
- Merge, Knoten
 - Eigenschaften 110
- mergenode, Eigenschaften 110
- Merkmalauswahlmodelle
 - Knoten, Scripteigenschaften 193, 241
 - Scripts 5
 - zuweisen 5
- Methoden definieren 27
- Microsoft-Modelle
 - Knoten, Scripteigenschaften 249, 251
- Migration
 - allgemeine Unterschiede 317
 - auf Objekte zugreifen 324
 - Ausgabetypen 320
 - Befehle 317
 - Dateisystem 324
 - Eigenschaften abrufen 321
 - Eigenschaften festlegen 321
 - Eigenschaftsnamen 320
 - Funktionen 317
 - Knotenreferenzen 320
 - Knotentypen 320
 - Modelltypen 320
 - Repository 324
 - Schleifen verwenden 323
 - Scriptingkontext 317
 - sonstiges 326
 - Streams, Ausgabe und Modellmanager entfernen 37
 - Streams ausführen 323
 - Streams bearbeiten 321
 - Übersicht 317
 - Variablen 320
- Mittelwertknoten
 - Eigenschaften 284
- Modelle
 - Scriptnamen 313, 315

- Modellierungsknoten
 - Knoten, Scripteigenschaften 167
- Modellnuggets
 - Knoten, Scripteigenschaften 237
 - Scriptnamen 313, 315
- Modellobjekte
 - Scriptnamen 313, 315
- MS - Lineare Regression
 - Knoten, Scripteigenschaften 249, 251
- MS - Logistische Regression
 - Knoten, Scripteigenschaften 249, 251
- MS - Neuronales Netz
 - Knoten, Scripteigenschaften 249, 251
- MS-Entscheidungsbaum
 - Knoten, Scripteigenschaften 249, 251
- MS Sequenz-Clustering
 - Knoten, Scripteigenschaften 251
- MS Time Series
 - Knoten, Scripteigenschaften 251
- msassocnode, Eigenschaften 249
- msbayesnode, Eigenschaften 249
- msclusternode, Eigenschaften 249
- mslogisticnode, Eigenschaften 249
- msneuralnetworknode, Eigenschaften 249
- msregressionnode, Eigenschaften 249
- mssequenceclusternode, Eigenschaften 249
- mstimeseriesnode, Eigenschaften 249
- mstreenode, Eigenschaften 249
- Multiplotknoten
 - Eigenschaften 159
- multiplotnode, Eigenschaften 159
- Multiset-Befehl 71

N

- Nächste-Nachbarn-Modelle
 - Knoten, Scripteigenschaften 203
- Netezza - Divisives Clustering, Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza - Lineare Regression, Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza - Naive Bayes, Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-Bayes-Netzmodelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-Entscheidungsbaummodelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-K-Means-Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-KNN-Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-Modelle
 - Knoten, Scripteigenschaften 266
- Netezza-PCA-Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-Regressionsbaum, Modelle
 - Knoten, Scripteigenschaften 266, 276
- Netezza-Zeitreihenmodelle
 - Knoten, Scripteigenschaften 266
- netezzabayesnode, Eigenschaften 266
- netezzadectreenode, Eigenschaften 266
- netezzadivclusternode, Eigenschaften 266
- netezzaglmmnode, Eigenschaften 266
- netezzakmeansnode, Eigenschaften 266

netezzaknnnode, Eigenschaften 266
 netezzalinieregressionnode, Eigenschaften 266
 netezzanaivebayesnode, Eigenschaften 266
 netezzapcanode, Eigenschaften 266
 netezzaregtreenod, Eigenschaften 266
 netezzatimeseriesnode, Eigenschaften 266
 Netzdiagrammknoten
 Eigenschaften 164
 neuralnetnode, Eigenschaften 212
 neuralnetworknode, Eigenschaften 214
 Neuronale Netze
 Knoten, Scripteigenschaften 214, 245
 Neuronale Netzmodelle
 Knoten, Scripteigenschaften 212, 244
 Nicht-ASCII-Zeichen 25
 Nuggets
 Knoten, Scripteigenschaften 237
 numericpredictornode, Eigenschaften 177

O

Objektorientierung 26
 Operationen 18
 oraabnnode, Eigenschaften 253
 oraainode, Eigenschaften 253
 oraapriorinode, Eigenschaften 253
 Oracle Adaptive Bayes-Modelle
 Knoten, Scripteigenschaften 253, 259
 Oracle AI-Modelle
 Knoten, Scripteigenschaften 253
 Oracle Apriori-Modelle
 Knoten, Scripteigenschaften 259
 Oracle-Apriori-Modelle
 Knoten, Scripteigenschaften 253
 Oracle Decision Tree-Modelle
 Knoten, Scripteigenschaften 259
 Oracle-Entscheidungsbaummodelle
 Knoten, Scripteigenschaften 253
 Oracle-K-Means-Modelle
 Knoten, Scripteigenschaften 253
 Oracle KMeans-Modelle
 Knoten, Scripteigenschaften 259
 Oracle-MDL-Modelle
 Knoten, Scripteigenschaften 253, 259
 Oracle-Modelle
 Knoten, Scripteigenschaften 253
 Oracle Naive Bayes-Modelle
 Knoten, Scripteigenschaften 253, 259
 Oracle NMF-Modelle
 Knoten, Scripteigenschaften 259
 Oracle-NMF-Modelle
 Knoten, Scripteigenschaften 253
 Oracle O-Cluster
 Knoten, Scripteigenschaften 253, 259
 Oracle Support Vector Machines-Modelle
 Knoten, Scripteigenschaften 259
 Oracle-SVM-Modelle
 Knoten, Scripteigenschaften 253
 oradecisiontreenode, Eigenschaften 253
 oraglmnode, Eigenschaften 253
 orakmeansnode, Eigenschaften 253
 oramdlnode, Eigenschaften 253
 oranbnnode, Eigenschaften 253

oranmfnode, Eigenschaften 253
 oraoclusternode, Eigenschaften 253
 orasvmnode, Eigenschaften 253
 outputfilenode, Eigenschaften 302

P

Parameter 5, 71, 73, 75
 Scripts 18
 Superknoten 311
 partitionnode, Eigenschaften 134
 Partitionsknoten
 Eigenschaften 134
 PCA-/Faktormodelle
 Knoten, Scripteigenschaften 192, 241
 PCA-Modelle
 Knoten, Scripteigenschaften 192, 241
 Plotknoten
 Eigenschaften 160
 plotnode, Eigenschaften 160
 Python 17
 Scripts 18

Q

Quellenknoten
 Eigenschaften 79
 QUEST-Modelle
 Knoten, Scripteigenschaften 216, 245
 questnode, Eigenschaften 216

R

R-Erstellungsknoten
 Knoten, Scripteigenschaften 179
 R-Prozess, Knoten
 Eigenschaften 114
 Räumliche temporale Vorhersage, Knoten
 Eigenschaften 222
 reclassifynode, Eigenschaften 135
 Referenzieren von Knoten 31
 Eigenschaften festlegen 32
 Knoten suchen 31
 regressionnode, Eigenschaften 218
 reordernode, Eigenschaften 136
 reportnode, Eigenschaften 285
 reprojectnode, Eigenschaften 136
 Reprojizierungsknoten
 Eigenschaften 136
 restructurenode, Eigenschaften 137
 retrieve, Befehl 54
 RFM-Aggregat, Knoten
 Eigenschaften 112
 RFM-Analyse, Knoten
 Eigenschaften 137
 rfmaggregatenode, Eigenschaften 112
 rfmanalysisnode, Eigenschaften 137
 Routput, Knoten
 Eigenschaften 286
 routputnode, Eigenschaften 286
 Rprocessnode, Eigenschaften 114

S

Sammlungsknoten
 Eigenschaften 152
 samplennode, Eigenschaften 114
 SAS-Exportknoten
 Eigenschaften 303
 SAS-Quellenknoten
 Eigenschaften 94
 sasexportnode, Eigenschaften 303
 sasimportnode, Eigenschaften 94
 Schleifen
 Verwendung in Scripts 53
 Schleifen in Streams verwenden 6, 7
 Scripterstellung
 bedingte Ausführung 6, 11
 Benutzerschnittstelle 2, 4, 5
 Fehlerprüfung 56
 Felder auswählen 10
 in Superknoten 5
 Iterationsschlüssel 8
 Iterationsvariable 9
 Schleifen verwenden 6, 7
 Streamausführungsreihenfolge 53
 Übersicht 1
 Scripting
 Diagramme 29
 Python-Scripting 318, 319, 322, 325
 Streams 29
 Superknotenstreams 29
 Syntax 23, 25
 traditionelles Scripting 318, 319, 322, 325
 Übersicht 17
 Scripting-API
 Beispiel 41
 Einführung 41
 Fehlerbehandlung 46
 globale Werte 50
 mehrere Streams 41, 51
 Metadaten 41
 Sitzungsparameter 46
 Standalone-Scripts 51
 Streamparameter 46
 Superknotenparameter 46
 Zugreifen auf generierte Objekte 44
 Scripts
 allgemeine Eigenschaften 74
 Ausführung 12
 Ausgabeknoten 279
 bedingte Ausführung 6, 11
 Diagrammknoten 151
 Felder auswählen 10
 Importieren von Textdateien 2
 in der Befehlszeile 56
 Iterationsschlüssel 8
 Iterationsvariable 9
 Kompatibilität mit früheren Versionen 56
 Kontext 30
 Merkmalauswahlmodelle 5
 Python-Scripting 318
 Schleifen verwenden 6, 7
 speichern 2
 Standalone-Scripts 1, 29
 Streams 1, 29
 Superknotenscripts 1, 29
 Syntax 18, 19, 21, 22, 26, 27, 28

- Scripts (*ForTs*)
 - traditionelles Scripting 318
 - Unterbrechung 12
 - verwendete Abkürzungen 72
- Scripts ausführen 12
- selectnode, Eigenschaften 116
- sequencenode, Eigenschaften 220
- Sequenzmodelle
 - Knoten, Scripteigenschaften 220, 246
- Server
 - Befehlszeilenargumente 68
- setglobalnode, Eigenschaften 287
- settoflagnode, Eigenschaften 139
- Sicherheit
 - codierte Kennwörter 55, 68
- simevalnode, Eigenschaften 287
- simfitnode, Eigenschaften 288
- simgen-Knoten
 - Eigenschaften 95
- simgennode, Eigenschaften 95
- Simulationsanpassungsknoten
 - Eigenschaften 288
- Simulationsevaluierungsknoten
 - Eigenschaften 287
- Simulationsgenerierungsknoten
 - Eigenschaften 95
- Slotparameter 5, 71, 73
- SLRM-Modelle
 - Knoten, Scripteigenschaften 221, 246
- slrmnode, Eigenschaften 221
- Sortierknoten
 - Eigenschaften 116
- sortnode, Eigenschaften 116
- Space-Time-Boxes, Knoteneigenschaften 107
- Standalone-Scripts 1, 4, 29
- statisticsexportnode, Eigenschaften 309
- statisticsimportnode, Eigenschaften 5, 307
- statisticsmodelnode, Eigenschaften 308
- statisticsnode, Eigenschaften 289
- statisticsoutputnode, Eigenschaften 308
- statisticstransformnode, Eigenschaften 307
- Statistikknoden
 - Eigenschaften 289
- STB-Knoten
 - Eigenschaften 107
- Stichprobenknoten
 - Eigenschaften 114
- store, Befehl 54
- STP-Knoten
 - Eigenschaften 222
- STP-Knotennugget
 - Eigenschaften 247
- stpnode, Eigenschaften 222
- stream.nodes, Eigenschaft 53
- Streamausführungsreihenfolge
 - mit Scripts ändern 53
- Streaming-ZR-Knoten
 - Eigenschaften 117
- streamingts, Eigenschaften 117
- Streams
 - ändern 33
 - Ausführung 30
 - bedingte Ausführung 6, 11
 - Eigenschaften 75

- Streams (*ForTs*)
 - Multiset-Befehl 71
 - Schleifen verwenden 6, 7
 - Scripting 29
 - Scripts 1, 2, 29
- Streams ändern 33, 36
- Streams ausführen 30
- Strukturierte Eigenschaften 71
- Superknoten 71
 - Eigenschaften 311
 - Festlegen von Eigenschaften 311
 - Parameter 311
 - Scripts 1, 5, 6, 29, 311
 - Stream 29
 - Streams 29
- Support Vector Machine, Modelle
 - Knoten, Scripteigenschaften 246
- SVM-Modelle
 - Knoten, Scripteigenschaften 227
- svmnnode, Eigenschaften 227
- System
 - Befehlszeilenargumente 66

T

- Tabelleninhaltsmodell 57
- Tabellenknoten
 - Eigenschaften 290
- tablenode, Eigenschaften 290
- tcnnode, Eigenschaften 228
- Temporale kausale Modelle
 - Knoten, Scripteigenschaften 228
- timeintervalnode, Eigenschaften 140
- timeplotnode, Eigenschaften 162
- timeseriesnode, Eigenschaften 231
- tmlimport, Knoteneigenschaften 98
- Transformationsknoten
 - Eigenschaften 292
- transformnode, Eigenschaften 292
- Transponierknoten
 - Eigenschaften 144
- transposenode, Eigenschaften 144
- Traversieren durch Knoten 36
- TwoStep AS-Modelle
 - Knoten, Scripteigenschaften 234, 247
- TwoStep-Modelle
 - Knoten, Scripteigenschaften 233, 247
- twostepAS, Eigenschaften 234
- twostepnode, Eigenschaften 233
- type, Knoten
 - Eigenschaften 145
- typenode, Eigenschaften 5, 145

U

- Umcodierungsknoten
 - Eigenschaften 135
- Umstrukturierungsknoten
 - Eigenschaften 137
- Unterbrechen von Scripts 12
- userinputnode, Eigenschaften 98

V

- Variable Datei, Knoten
 - Eigenschaften 99

- variablefilenode, Eigenschaften 99
- Variablen
 - Scripts 18
- Verallgemeinerte lineare Modelle
 - Knoten, Scripteigenschaften 195, 242
- Verallgemeinerte lineare Modelle von Oracle
 - Knoten, Scripteigenschaften 253
- Verallgemeinerte lineare Netezza-Modelle
 - Knoten, Scripteigenschaften 266
- Vererbung 28
- Verlaufsknoten
 - Eigenschaften 133
- Verteilungsknoten
 - Eigenschaften 153

W

- webnode, Eigenschaften 164

X

- XML-Exportknoten
 - Eigenschaften 304
- XML-Inhaltsmodell 59
- XML-Quellenknoten
 - Eigenschaften 102
- xmlexportnode, Eigenschaften 304
- xmlimportnode, Eigenschaften 102

Z

- Zeichenfolgefunktionen 53
- Zeichenfolgen 19
 - Ändern der Groß- und Kleinschreibung 53
- Zeitdiagrammknoten
 - Eigenschaften 162
- Zeitintervallknoten
 - Eigenschaften 140
- Zeitreihenmodelle
 - Knoten, Scripteigenschaften 231, 247
- Zugriff auf Ergebnisse der Streamausführung 56, 62
 - JSON-Inhaltsmodell 60
 - Tabelleninhaltsmodell 57
 - XML-Inhaltsmodell 59
- Zugriff auf Streamausführungsergebnisse 56, 62
 - JSON-Inhaltsmodell 60
 - Tabelleninhaltsmodell 57
 - XML-Inhaltsmodell 59

