IBM SPSS Modeler 16 Python スクリプトと オートメーション ガイド

お願い

本書および本書で紹介する製品をご使用になる前に、 269 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM SPSS Modeler バージョン 16 リリース 0 モディフィケーション 0 および新しい版で明記されない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM SPSS Modeler 16 Python Scripting and Automation Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

第 1 章 スクリプト

スクリプトの概要

IBM® SPSS® Modeler のスクリプトは、ユーザー・インターフェースのプロセスを自動化する強力なツールです。スクリプトで、マウスやキーボードを使用した場合と同じ種類のアクションを実行できます。また、頻繁に繰り返したり手動で実行するのに時間がかかるタスクを自動化するために使用できます。

次の処理にスクリプトを使用できます。

- ストリームでノードを実行する特定の順序を指定し、実行のための条件が満たされているかどうかに応じて、ノードを条件付きで実行する。
- ストリーム内でノードを繰り返し実行するためのループを作成する。
- 通常はユーザーとの対話によって実行される一連の操作 (例えば、モデルを作成してテストするなど) を自動化する。
- 十分なユーザーとの対話が必要な複雑な処理 (例えば、モデルの生成とテストを繰り返す交差検証手順など) を設定する。
- ストリームを操作する処理 (例えば、モデル学習ストリームの取得や実行、対応するモデル・テスト・ストリームの自動生成など) を設定する。

この 章では、ストリームレベルのスクリプト、スタンドアロン・スクリプト、および IBM SPSS Modeler インターフェースのスーパーノード内のスクリプトに関する高度な説明と例を記述しています。スクリプト言語、シンタックス、コマンドについては、これ以降の 章で説明します。[1]

注 : IBM SPSS Modeler 内の IBM SPSS Statistics で作成されたスクリプトはインポートおよび実行できません。

スクリプトの種類

IBM SPSS Modeler では、次の 3 種類のスクリプトが使用されます。

- **ストリーム・スクリプト** は、ストリーム・プロパティーとして格納されるため、特定のストリームと一緒に保存およびロードされます。例えば、モデル・ナゲットの学習と適用のプロセスを自動化するストリーム・スクリプトを書くことができます。また、特定のストリームが実行されたときは常に、そのストリームのキャンバスの内容ではなく、スクリプトが実行されるように指定することもできます。
- **スタンドアロン・スクリプト**は、どのストリームとも関連付けがなく、外部のテキスト・ファイルに保存されます。スタンドアロン・スクリプトは、例えば、複数のストリームを一緒に操作する場合に使用できます。
- **スーパーノード スクリプト**は、スーパーノード ストリーム・プロパティーとして格納されます。スーパーノード スクリプトは、ターミナル・スーパーノードでのみ使用可能です。スーパーノード スクリプトは、スーパーノードの内容のシーケンスの実行を制御するのに使用できます。ターミナル以外の (ソ

1. IBM SPSS Modeler のレガシー・スクリプト言語は、引き続き IBM SPSS Modeler 16 で使用することができます。詳しくは、資料「*IBM SPSS Modeler 16 スクリプトとオートメーション・ガイド*」を参照してください。IBM SPSS Modeler の既存のレガシー・スクリプトを Python スクリプトにマッピングする方法については、 257 ページの『付録 B. 従来のスクリプトから Python スクリプトへの移行』を参照してください。

ースまたはプロセス) スーパーノードの場合、ストリーム・スクリプト内で直接、スーパーノードまたはスーパーノード内のノードにプロパティーを定義できます。

ストリーム・スクリプト

スクリプトを使用して特定のストリーム内の操作をカスタマイズできます。また、スクリプトをそのストリームとともに保存することができます。ストリーム・スクリプトは、ストリーム内のターミナル・ノードの、特定の実行順序を指示するために使用されます。ストリーム・スクリプト ダイアログ・ボックスを使用して、現在のストリームとともに保存されているスクリプトを編集します。

「ストリームのプロパティー」ダイアログ・ボックスの「ストリーム・スクリプト」タブにアクセスするには

1. 「ツール」メニューから次の各項目を選択します。

   「ストリームのプロパティー」 > 「実行」
2. **「実行」**タブをクリックして、現在のストリームのスクリプトの処理を行います。
3. 実行モード **「デフォルト (オプション・スクリプト)」**を選択します。

ストリーム・スクリプトのダイアログ・ボックスの一番上にあるツールバー・アイコンを使用すると、次のような作業を実行できます。

- ウィンドウに既存のスタンドアロン・スクリプトの内容をインポートする。
- スクリプトをテキスト・ファイルとして保存する。
- スクリプトを印刷する。
- デフォルト スクリプトを追加する。
- スクリプトを編集する (元に戻す、切り取り、コピー、貼り付けなど、標準的な編集機能を使用)。
- 現在のスクリプト全体を実行する。
- スクリプトから選択した行を実行する。
- 実行時にスクリプトを停止する (このアイコンは、スクリプトの実行時のみ使用可能になります)。
- スクリプトのシンタックスをチェックして、エラーが見つかれば、ダイアログ・ボックスの下部パネルにそれを表示する。

さらに、ストリームが実行されたときにこのスクリプトが実行されるべきか、それとも実行されないべきかを指定できます。**「このスクリプトを実行」**を選択すると、ストリームの実行時に常に、スクリプトに指定された実行順序でこのスクリプトが実行されます。この設定により、ストリーム・レベルでの自動化を実現でき、素早いモデル構築が可能になります。ただし、デフォルトでは、ストリーム実行時にこのスクリプトは無視されます。**「このスクリプトを無視」** オプションを選択した場合でも、常にこのダイアログ・ボックスで直接スクリプトを実行できます。

また、スクリプトのタイプを Python スクリプトから従来のスクリプトに変更することもできます。

スクリプト・エディターには、スクリプト・オーサリングを支援する以下の機能が用意されています。

- シンタックスの強調表示。キーワード、リテラル値 (文字列や数値など)、コメントが強調表示されます。
- 行番号付け。
- ブロックの一致。カーソルがプログラム・ブロックの開始位置に置かれると、対応する終了ブロックも強調表示されます。

- 自動入力の候補表示。

シンタックスの強調表示で使用される色とテキストのスタイルは、IBM SPSS Modeler の表示設定を使用してカスタマイズすることができます。この表示設定にアクセスするには、「**ツール**」 > 「**オプション**」 > 「**ユーザー・オプション**」を選択して、「**シンタックス**」タブをクリックします。

候補として表示されるシンタックス入力のリストにアクセスするには、コンテキスト・メニューから「**自動候補提示機能**」を選択するか、Ctrl + スペースを押します。カーソル・キーを使用してリストを上下に移動し、Enter キーを押して、選択したテキストを挿入します。既存のテキストを変更せずに自動候補提示機能モードを終了するには、Esc キーを押します。

「**デバッグ**」タブには、デバッグ・メッセージが表示されます。このタブを使用すると、スクリプトの実行中にスクリプトの状態を評価することができます。「**デバッグ**」タブは、読み取り専用テキスト領域と 1 行の入力テキスト・フィールドから構成されています。テキスト領域には、Python の `print` コマンドなどを使用して標準出力に送信されたテキスト、またはエラー・メッセージ・テキストなどを使用して、スクリプトによる標準エラーに対して送信されたテキストが表示されます。入力テキスト・フィールドは、ユーザーから入力を受け取ります。次に、この入力は、ダイアログ内で最後に実行されたスクリプトのコンテキスト (スクリプト・コンテキスト) 内で評価されます。テキスト領域には、ユーザーがコマンドのトレースを確認できるように、コマンドと結果の出力が表示されます。入力テキスト・フィールドには、コマンド・プロンプト (Python スクリプトの >>>) が常に表示されます。

以下の場合、新しいスクリプト・コンテキストが作成されます。

- 「このスクリプトを実行」ボタンまたは「選択した行を実行」ボタンを使用してスクリプトを実行した場合。
- スクリプト言語を変更した場合。

新しいスクリプト・コンテキストが作成されると、テキスト領域がクリアされます。

注: スクリプト・パネルの外部でストリームを実行しても、スクリプト・パネルのスクリプト・コンテキストは変更されません。この実行の一部として作成された変数の値は、スクリプト・ダイアログ内には表示されません。

スタンドアロン・スクリプト

「スタンドアロン・スクリプト」ダイアログ・ボックスでは、テキスト・ファイルとして保存されるスクリプトを作成したり編集したりします。このダイアログ・ボックスには、ファイル名が表示されます。スクリプトのロード、保存、インポート、および実行の機能が備わっています。

スタンドアロン・スクリプトのダイアログ・ボックスにアクセスするには

メイン・メニューから次の各項目を選択します。

「ツール」 > 「スタンドアロン・スクリプト」

スタンドアロン・スクリプトでは、ストリーム・スクリプトと同じツールバーやスクリプト・シンタックス検査オプションを使用することができます。 詳しくは、トピック 2 ページの『ストリーム・スクリプト』を参照してください。

スーパーノード スクリプト

IBM SPSS Modeler のスクリプト言語を使用して、スクリプトを作成し、任意のターミナル・スーパーノード内に保存できます。これらのスクリプトはテーミナル・スーパーノードにのみ使用でき、テンプレートストリームの作成時、およびスーパーノードの内容に特定の実行順序を指定する際に使用できます。スーパーノード スクリプトを使用すると、ストリーム内で複数のスクリプトを実行することもできます。

例えば、複雑なストリームで実行の順序を指定する必要があり、スーパーノードには、散布図ノードで使用される新しいフィールドを作成する前に実行される必要のあるグローバル・ノードを含む、いくつかのノードがあるとします。この場合、まずグローバル・ノードを実行するスーパーノード スクリプトを作成できます。このノードが計算する平均や標準偏差などの値は、散布図ノードを実行するときに使用します。

スーパーノード スクリプト内では、ほかのスクリプトの場合と同様の方法で、ノード・プロパティーを指定できます。また、ストリーム・スクリプトから直接に、任意のスーパーノードまたはカプセル化されたノードのプロパティーを変更または定義することもできます。詳しくは、トピック 251 ページの『第 19 章 スーパーノードのプロパティー』を参照してください。この手法は、ソース スーパーノード、プロセス スーパーノード、およびターミナル・スーパーノードに適用できます。

注 : 独自のスクリプトを実行することができるのはターミナル・スーパーノードの場合だけなので、「スーパーノード」ダイアログ・ボックスの「スクリプト」タブは、ターミナル・スーパーノードの場合にだけ利用可能です。

メイン キャンバスから「スーパーノード スクリプト」ダイアログ・ボックスを開くには

ストリーム キャンバスでターミナル・スーパーノードを選択して、「スーパーノード」メニューから次の項目を選択します。

「スーパーノード・スクリプト...」

ズーム・インしたスーパーノード・キャンバスから「スーパーノード スクリプト」ダイアログ・ボックスを開くには

スーパーノード・キャンバス上を右クリックして表示されるコンテキスト・メニューから、次の項目を選択します。

「スーパーノード・スクリプト...」

ストリームでのループと条件付き実行

バージョン 16.0 以降、SPSS Modeler では、スクリプト言語で直接指示を作成するのではなく、さまざまなダイアログ・ボックスで値を選択することによって、ストリーム内でいくつかの基本的なスクリプトを作成することができます。この方法で作成できるスクリプトの 2 つの主なタイプは、単純ループと、条件が満たされた場合にノードを実行する方法です。

ストリーム内でループ規則と条件付き実行規則の両方を組み合わせることができます。例えば、世界中の製造業者の自動車の販売に関連したデータがあるとします。ストリーム内のデータを処理するループを設定して、製造業者の国別に詳細を識別し、モデル別の販売数、製造業者別およびエンジン・サイズ別の排気ガスレベルなどの詳細を示すさまざまなグラフにデータを出力することができます。ヨーロッパの情報のみに関心がある場合は、アメリカとアジアの製造業者用のグラフが作成されないようにする条件をループに追加することもできます。

注: ループと条件付き実行は、いずれもバックグラウンドのスクリプトに基づいているため、これらはストリームの実行時にストリーム全体にのみ適用されます。

- **ループ** ループを使用して、反復タスクを自動化できます。例えば、指定の数のノードをストリームに追加し、追加するたびに 1 つのノード・パラメーターを変更することができます。あるいは、以下の例のように、ストリームまたは枝を指定の数だけ繰り返し実行することを制御できます。
  - ストリームを指定の回数実行し、実行するたびにソースを変更する。
  - ストリームを指定の回数実行し、実行するたびに変数の値を変更する。
  - ストリームを指定の回数実行し、実行するたびに 1 つの追加フィールドを入力する。
  - モデルを指定の回数構築し、毎回モデル設定を変更する。
- **条件付き実行** これを使用すると、事前定義した条件に基づいて、どのようにターミナル・ノードを実行するかを制御できます。例えば、以下のようにします。
  - 指定の値が true か false かに基づいて、ノードを実行するかどうかを制御する。
  - ノードのループを並行して実行するのか、順次に実行するのかを定義する。

ループと条件付き実行は両方とも、「ストリームのプロパティー」ダイアログ・ボックス内の「実行」タブで設定します。条件またはループ要件に使用されているノードは、追加の記号が付加された状態でストリーム・キャンバスに表示され、ループおよび条件付き実行に関与していることが示されます。

「実行」タブには、以下の 3 つのいずれかの方法でアクセスできます。

- メイン・ダイアログ・ボックスの上部にあるメニューを使用する。
  1. 「ツール」メニューから次の各項目を選択します。

     「ストリームのプロパティー」 > 「実行」
  2. 「実行」タブをクリックして、現在のストリーム用のスクリプトを処理します。
- ストリーム内から。
  1. ノードを右クリックして**「ループ/条件付き実行」**を選択する。
  2. 該当するサブメニューのオプションを選択します。
- メイン・ダイアログ・ボックスの上部にあるグラフィック・ツールバーで、ストリーム・プロパティーのアイコンをクリックする。

ループまたは条件付き実行の詳細を初めて設定する場合は、「実行」タブで**「ループ/条件付き実行」**実行モードを選択してから、**「条件」**または**「ループ」**サブタブを選択します。

ストリームでのループ

ループを使用すると、ストリーム内の反復タスクを自動化できます。例えば、以下のようにできます。

- ストリームを指定の回数実行し、実行するたびにソースを変更する。
- ストリームを指定の回数実行し、実行するたびに変数の値を変更する。
- ストリームを指定の回数実行し、実行するたびに 1 つの追加フィールドを入力する。
- モデルを指定の回数構築し、毎回モデル設定を変更する。

満たすべき条件は、ストリームの「実行」タブの**「ループ」**サブタブで設定します。サブタブを表示するには、**「ループ/条件付き実行」**実行モードを選択します。

定義するループ要件は、**「ループ/条件付き実行」**実行モードが設定されている場合は、ストリーム実行時に有効になります。オプションで、ループ要件のスクリプト・コードを生成して、スクリプト・エディター

に貼り付けることができます。これを行うには、「ループ」サブタブの右下隅にある「**貼り付け...**」をクリックします。メインの「実行」タブの表示が「**デフォルト (オプション・スクリプト)**」実行モードを表示するように変わり、スクリプトがタブの上部に表示されます。つまり、スクリプト・エディターで詳細にカスタマイズ可能なスクリプトを生成する前に、ダイアログ・ボックスのさまざまなループ・オプションを使用してループ構造を定義できます。「**貼り付け...**」をクリックすると、生成されたスクリプトには、定義した条件付き実行要件も表示されることに注意してください。

ループをセットアップするには、以下を実行します。

1. ストリームで実行するメインのループ構造を定義する反復キーを作成します。詳しくは、反復キーの作成を参照してください。
2. 必要に応じて、1 つ以上の反復変数を定義します。詳しくは、反復変数の作成を参照してください。
3. 作成した反復および変数が、サブタブの本文に表示されます。デフォルトで、反復は表示されている順に実行されます。反復をリスト内で上または下に移動するには、反復をクリックして選択し、サブタブの右側の列にある上矢印または下矢印を使用して順序を変更します。

ストリームでのループのための反復キーの作成

反復キーを使用して、ストリームで実行するメインのループ構造を定義します。例えば、自動車販売を分析している場合は、ストリーム・パラメーター *製造国* を作成し、これを反復キーとして使用できます。ストリームを実行すると、このキーは、反復のたびにデータ内でそれぞれ異なる国の値に設定されます。「反復キーの定義」ダイアログ・ボックスを使用して、キーを設定します。

このダイアログ・ボックスを開くには、「ループ」サブタブの左下隅にある「**反復キー**」ボタンを選択するか、ストリーム内の任意のノードを右クリックして「**ループ/条件付き実行**」 > 「**反復キーの定義 (フィールド)**」または「**ループ/条件付き実行**」 > 「**反復キーの定義 (値)**」を選択します。ストリームからダイアログ・ボックスを開くと、ノードの名前などの一部のフィールドが自動的に入力されます。

反復キーを設定するには、以下のフィールドに入力します。

反復対象。次のいずれかのオプションを選択できます。

- **ストリーム・パラメーター - フィールド**。このオプションは、既存のストリーム・パラメーターの値を、指定した各フィールドに順に設定するループを作成する場合に使用します。
- **ストリーム・パラメーター - 値**。このオプションは、既存のストリーム・パラメーターの値を、指定した各値に順に設定するループを作成する場合に使用します。
- **ノード・プロパティー - フィールド**。このオプションは、ノード・プロパティーの値を、指定した各フィールドに順に設定するループを作成する場合に使用します。
- **ノード・プロパティー - 値**。このオプションは、ノード・プロパティーの値を、指定した各値に順に設定するループを作成する場合に使用します。

設定内容。ループが実行されるたびに値が設定される項目を選択します。次のいずれかのオプションを選択できます。

- **パラメーター**。「**ストリーム・パラメーター - フィールド**」または「**ストリーム・パラメーター - 値**」を選択した場合にのみ使用可能です。使用可能なリストから、必要なパラメーターを選択します。
- **ノード**。「**ノード・プロパティー - フィールド**」または「**ノード・プロパティー - 値**」を選択した場合にのみ使用可能です。ループをセットアップするノードを選択します。参照ボタンをクリックして「ノード選択」ダイアログを開き、目的のノードを選択します。リストされているノードが多すぎる場合

は、ソース・ノード、プロセス・ノード、グラフ・ノード、モデリング・ノード、出力ノード、エクスポート・ノード、またはモデル・ノードの適用のいずれかのカテゴリー別にノードを表示するように、表示をフィルタリングできます。

- **プロパティー**: 「**ノード・プロパティー - フィールド**」または「**ノード・プロパティー - 値**」を選択した場合にのみ使用可能です。使用可能なリストからノードのプロパティーを選択します。

使用するフィールド。「**ストリーム・パラメーター - フィールド**」または「**ノード・プロパティー - フィールド**」を選択した場合にのみ使用可能です。反復値を提供するために使用するノード内のフィールドを選択します。次のいずれかのオプションを選択できます。

- **ノード**。「**ストリーム・パラメーター - フィールド**」を選択した場合にのみ使用可能です。ループを設定する詳細を含むノードを選択します。参照ボタンをクリックして「ノード選択」ダイアログを開き、目的のノードを選択します。リストされているノードが多すぎる場合は、ソース・ノード、プロセス・ノード、グラフ・ノード、モデリング・ノード、出力ノード、エクスポート・ノード、またはモデル・ノードの適用のいずれかのカテゴリー別にノードを表示するように、表示をフィルタリングできます。
- **フィールド・リスト**。右側の列にあるリスト・ボタンをクリックして、「フィールドの選択」ダイアログ・ボックスを表示します。このダイアログ・ボックスで、反復データを提供するノード内のフィールドを選択します。詳しくは、8 ページの『反復のためのフィールドの選択』を参照してください。

使用する値。「**ストリーム・パラメーター - 値**」または「**ノード・プロパティー - 値**」を選択した場合にのみ使用可能です。選択したフィールド内で、反復値として使用する値 (複数可) を選択します。次のいずれかのオプションを選択できます。

- **ノード**。「**ストリーム・パラメーター - 値**」を選択した場合にのみ使用可能です。ループを設定する詳細を含むノードを選択します。参照ボタンをクリックして「ノード選択」ダイアログを開き、目的のノードを選択します。リストされているノードが多すぎる場合は、ソース・ノード、プロセス・ノード、グラフ・ノード、モデリング・ノード、出力ノード、エクスポート・ノード、またはモデル・ノードの適用のいずれかのカテゴリー別にノードを表示するように、表示をフィルタリングできます。
- **フィールド・リスト**。反復データを提供するためのノード内のフィールドを選択します。
- **値リスト**。右側の列にあるリスト・ボタンをクリックして、「値の選択」ダイアログ・ボックスを表示します。このダイアログ・ボックスで、反復データを提供するフィールド内の値を選択します。

ストリームでのループのための反復変数の作成

反復変数を使用して、ループが実行されるたびに、ストリーム内の選択したノードのストリーム・パラメーターまたはプロパティーの値を変更できます。例えば、ストリーム・ループが自動車販売データを分析していて、*製造国* を反復キーとして使用している場合に、モデル別の販売を示すグラフ出力と、排気ガス情報を示すグラフ出力があるとします。このような場合に、結果グラフごとに新しいタイトル (スウェーデンの *自動車排気ガス* や *日本のモデル別自動車販売* など) を作成する反復変数を作成できます。「反復変数の定義」ダイアログ・ボックスを使用して、必要な変数を設定します。

このダイアログ・ボックスを開くには、「ループ」サブタブの左下隅にある「**反復変数**」ボタンを選択するか、ストリーム内の任意のノードを右クリックして「**ループ/条件付き実行**」 > 「**反復変数の定義**」を選択します。

反復変数を設定するには、以下のフィールドに入力します。

変更。修正する属性の種類を選択します。「**ストリーム・パラメーター**」または「**ノード・プロパティー**」を選択します。

- 「**ストリーム・パラメーター**」を選択した場合、必要なパラメーターを選択してから、以下のいずれかのオプションを使用して (ストリームで使用可能な場合)、ループを反復するたびにそのパラメーターに設定する値を定義します。
  - **グローバル変数**。ストリーム・パラメーターを設定するグローバル変数を選択します。
  - **テーブル出力セル**。ストリーム・パラメーターをテーブル出力セルの値に設定するには、リストからテーブルを選択して、使用する「**行**」と「**列**」を入力します。
  - **手動で入力**。このオプションは、このパラメーターが反復のたびに取る値を手動で入力する場合に選択します。「ループ」サブタブに戻ると、必要なテキストを入力する新しい列が作成されます。
- 「**ノード・プロパティー**」を選択した場合は、必要なノードといずれかのプロパティーを選択してから、そのプロパティーに使用する値を設定します。以下のオプションの 1 つを使用して、新しいプロパティー値を設定します。
  - **単独**。プロパティー値は、反復キー値を使用します。詳しくは、6 ページの『ストリームでのループのための反復キーの作成』を参照してください。
  - **語幹の接頭辞として**。「**語幹**」フィールドに入力する内容の接頭辞として反復キー値を使用します。
  - **語幹の接尾辞として**。「**語幹**」フィールドに入力する内容の接尾辞として反復キー値を使用します。

  接頭辞または接尾辞のオプションを選択した場合は、「**語幹**」フィールドに追加テキストを追加するように求められます。例えば、反復キー値が *製造国* で、「**語幹の接頭辞として**」を選択した場合は、このフィールドに *- モデル別の販売* と入力できます。

反復のためのフィールドの選択

反復を作成する場合は、「フィールドの選択」ダイアログ・ボックスを使用して、1 つ以上のフィールドを選択できます。

ソート基準。次のオプションを使って、表示する利用可能フィールドをソートすることができます。

- **ファイル順**。データ・ストリームから現在のノードにデータが渡された順序にフィールドをソートします。
- **名前**: アルファベット順でフィールドをソートします。
- 「**データ型**」。尺度でソートしたフィールドを表示します。特定の尺度のフィールドを選択する場合に役立ちます。

リストからフィールドを 1 回に 1 つずつ選択するか、または Shift キーまたは Ctrl キーを押しながら複数のフィールドを選択します。また、リストの下のボタンを使用して、尺度に基づいて複数のフィールドを選択したり、テーブル中のすべてのフィールドを選択または選択解除することができます。

選択可能なフィールドは、使用しているストリーム・パラメーターまたはノード・プロパティーに適切なフィールドのみが表示されるようにフィルタリングされていることに注意してください。例えば、ストレージ・タイプが文字列のストリーム・パラメーターを使用している場合は、ストレージ・タイプが文字列のフィールドのみが表示されます。

ストリームでの条件付き実行

条件付き実行では、定義するストリーム内容の一致条件に基づいて、ターミナル・ノードを実行する方法を制御できます。例えば、以下のようにできます。

- 指定の値が true か false かに基づいて、ノードを実行するかどうかを制御する。
- ノードのループを並行して実行するのか、順次に実行するのかを定義する。

満たすべき条件は、ストリームの「実行」タブの「**条件**」サブタブで設定します。サブタブを表示するには、「**ループ/条件付き実行**」実行モードを選択します。

定義する条件付き実行要件は、「**ループ/条件付き実行**」実行モードが設定されている場合は、ストリーム実行時に有効になります。オプションで、条件付き実行要件のスクリプト・コードを生成して、スクリプト・エディターに貼り付けることができます。これを行うには、「条件」サブタブの右下隅にある「**貼り付け...**」をクリックします。メインの「実行」タブの表示が「**デフォルト (オプション・スクリプト)**」実行モードを表示するように変わり、スクリプトがタブの上部に表示されます。つまり、スクリプト・エディターで詳細にカスタマイズ可能なスクリプトを生成する前に、ダイアログ・ボックスのさまざまなループ・オプションを使用して条件を定義できます。「**貼り付け...**」をクリックすると、生成されたスクリプトには、定義したループ要件も表示されることに注意してください。

条件をセットアップするには以下を行います。

1. 「条件」サブタブの右側の列で、「実行式の追加」ボタン  をクリックして、「条件実行式 (Conditional Execution Statement)」ダイアログ・ボックスを開きます。このダイアログで、ノードを実行するために満たす必要がある条件を指定します。
2. 「条件実行式 (Conditional Execution Statement)」ダイアログ・ボックスで、以下を指定します。
   a. **ノード**。条件付き実行を設定するノードを選択します。参照ボタンをクリックして「ノード選択」ダイアログを開き、目的のノードを選択します。リストされているノードが多すぎる場合は、エクスポート・ノード、グラフ・ノード、モデリング・ノード、または出力ノードのいずれかのカテゴリー別にノードを表示するように、表示をフィルタリングできます。
   b. **条件の基準**。ノードを実行するために満たす必要がある条件を指定します。「**ストリーム・パラメーター**」、「**グローバル変数**」、「**テーブル出力セル**」、または「**常に True**」の 4 つのオプションのいずれか 1 つを選択できます。ダイアログ・ボックスの下半分に入力する詳細は、選択する条件によって異なります。
      - **ストリーム・パラメーター**。使用可能なリストからパラメーターを選択してから、そのパラメーターの「**演算子**」を選択します。例えば、演算子は「より大きい」、「等しい」、「より小さい」、「間」などです。次に、演算子に応じて「**値**」か、最小値および最大値を入力します。
      - **グローバル変数**。使用可能なリストから変数を選択します。例えば、「平均」、「合計」、「最小値」、「最大値」、または「標準偏差」がリストに含まれている可能性があります。次に、「**演算子**」と必要な値を選択します。
      - **テーブル出力セル**。使用可能なリストからテーブル・ノードを選択して、テーブルの「**行**」と「**列**」を選択します。次に、「**演算子**」と必要な値を選択します。
      - **常に True**。ノードを常に実行する必要がある場合は、このオプションを選択します。このオプションを選択する場合は、さらに選択するパラメーターはありません。
3. 必要なすべての条件を設定するまで、ステップ 1 と 2 を必要なだけ繰り返します。選択したノードと、ノードが実行される前に満たすべき条件が、サブタブの本体部分の「**実行ノード**」列と、「**値の設定条件 (真の場合に値を設定)**」列に表示されます。
4. デフォルトで、ノードと条件は表示されている順に実行されます。ノードと条件をリスト内で上または下に移動するには、ノードと条件をクリックして選択し、サブタブの右側の列にある上矢印または下矢印を使用して順序を変更します。

さらに、「条件」サブタブの下部にある以下のオプションを設定できます。

- **すべてを順番に評価**。このオプションは、各条件をサブタブに表示されている順序で評価する場合に選択します。条件が「True」のすべてのノードは、すべての条件が評価されてから 1 回だけ実行されます。
- **一度に 1 つずつ実行**。「**すべてを順番に評価**」が選択されている場合にのみ使用できます。このオプションを選択すると、条件が「True」と評価される場合、その条件に関連付けられているノードは、次の条件が評価される前に実行されます。
- **最初のヒットまで評価**。このオプションを選択すると、指定した条件から「True」の評価が返される最初のノードのみが実行されます。

スクリプトの実行と中断

その他多くの方法でスクリプトを実行できます。例えば、ストリーム・スクリプトまたはスタンドアロンのスクリプトのダイアログで、「このスクリプトを実行」ボタンをクリックすると、完全なスクリプトを実行します。

図 1. 「このスクリプトを実行」ボタン

「選択した行」ボタンをクリックすると、スクリプト内で選択した 1 行または隣接する行のブロックを実行します。

図 2. 「選択した行を実行」ボタン

スクリプトの実行は、次のいずれかの方法で行います。

- ストリーム・スクリプトまたはスタンドアロン・スクリプトのダイアログ・ボックスの「このスクリプトを実行」または「選択した行を実行」をクリックします。
- デフォルトの実行方法として 「**このスクリプトを実行**」が設定されているストリームを実行する。
- 起動時にインタラクティブ・モードで `-execute` フラグを使用します。詳しくは、トピック 53 ページの『コマンド・ライン引数の使用』を参照してください。

注 : 「スーパーノード」ダイアログ・ボックスで 「**このスクリプトを実行**」を選択しているかぎり、スーパーノード・スクリプトは、スーパーノードの実行時に実行されます。

スクリプト実行の中断

「ストリーム・スクリプト」ダイアログ・ボックスのツールバーにある赤い中止ボタンは、スクリプト実行時に有効になります。このボタンを使用すると、スクリプトおよび現在のストリームの実行を中止することができます。

検索と置換

「検索/置換」ダイアログ・ボックスは、スクリプト・エディターなど、スクリプトまたは式のテキストを編集する場合、またはレポート・ノードでテンプレートを定義する場合に使用できます。これらの領域のいずれかでテキストを編集する場合は、Ctrl+F を押してダイアログ・ボックスにアクセスし、カーソルがテキスト領域にフォーカスしていることを確認します。「置換」ノードを使用している場合、例えば、「設定」タブのテキスト領域から、または CLEM 式ビルダーのテキスト・フィールドからダイアログ・ボックスにアクセスできます。

1. テキスト領域内にカーソルを置いて、Ctrl + F キーを押して「検索/置換」ダイアログ・ボックスにアクセスします。
2. 検索するテキストを入力するか、最近検索した項目のドロップダウン・リストから選択します。
3. 置換テキストがある場合は、入力します。
4. 「**次を検索**」 をクリックして、検索を開始します。
5. 「**置換**」 をクリックして現在の選択内容を置換するか、「**すべて置換**」 をクリックしてすべてまたは選択したインスタンスを更新します。
6. 各操作が終了すると、ダイアログ・ボックスが閉じます。テキスト領域で F3 を押すと最後の検索操作が繰り返され、または Ctrl + F キーを押すとダイアログに再度アクセスします。

検索オプション

大文字と小文字を区別 : 検索操作で、例えば *myvar* が *myVar* と位置するかどうかなど、大文字と小文字を区別するかどうかを指定します。この設定に関係なく、置換テキストは常に入力したとおりに挿入されます。

語全体のみ : 検索操作が語内に埋め込まれたテキストに一致するかどうかを指定します。このオプションを選択すると、*spider* に関する検索は、*spiderman* または *spider-man* に一致しません。

正規表現 : 正規表現のシンタックスを使用するかどうかを指定します (次項参照)。このオプションを選択すると、「**語全体のみ**」 オプションは無効化され、その値は無視されます。

選択されたテキストのみ: 「**すべて置換**」オプションを使用する場合、検索の範囲を制御します。

正規表現シンタックス

正規表現を使用すると、タブまたは改行文字などの特殊文字、*a* から *d* までなど文字のクラスまたは範囲、行の開始または終了などの境界について検索することができます。正規表現パターンは、入力ストリング内で正規表現が見つけようとするストリングの構造を記述します。次の種類の正規表現構造がサポートされています。

表 1. 文字の一致

Characters	**一致**
x	文字 x
¥¥	円記号
¥0n	8 進法の値を持つ文字 0n (0 <= n <= 7)
¥0nn	8 進法の値を持つ文字 0nn (0 <= n <= 7)
¥0mnn	8 進法の値を持つ文字 0mnn (0 <= m <= 3, 0 <= n <= 7)
¥xhh	16 進法の値を持つ文字 0xhh
¥uhhhh	16 進法の値を持つ文字 0xhhhh

表 1. 文字の一致 (続き)

Characters	**一致**
¥t	タブ文字 ('¥u0009')
¥n	改行文字 ('¥u000A')
¥r	復帰文字 ('¥u000D')
¥f	改ページ文字 ('¥u000C')
¥a	アラート (ベル) 文字 ('¥u0007')
¥e	エスケープ文字 ('¥u001B')
¥cx	xに対応する制御文字

表 2. 一致する文字のクラス

文字クラス	**一致**
「abc]	a、b、または c (単純クラス)
「^abc]	a、b、または c 以外の文字 (減法)
「a-zA-Z]	a から z または A から Z の各文字 (範囲)
「a-d[m-p]]	a から d、または m から p (和集合)。または、「a-dm-p」と指定することもできます
「a-z&&[def]]	a から z、および d、e、または f (交差)
「a-z&&[^bc]]	a から z のうち、b と c を除いたもの (差集合)。または、「ad-z」と指定することもできます
「a-z&&[^m-p]]	a から z のうち、m から p までを除いたもの (差集合)。または、「a-lq-z」と指定することもできます

表 3. 事前設定された文字クラス

事前設定された文字クラス	**一致**
.	任意の文字 (行末に一致する場合または一致しない場合があります)
¥d	任意の数字: [0-9]
¥D	数字以外: [^0-9]
¥s	空白文字: [¥t¥n¥x0B¥f¥r]
¥S	空白文字以外: [^¥s]
¥w	ワード文字: [a-zA-Z_0-9]
¥W	ワード文字以外: [^¥w]

表 4. 境界の一致

境界の一致	**一致**
^	行頭
$	行末
¥b	語の境界
¥B	語以外の境界
¥A	入力の開始
¥Z	最後の行末以外の入力の終了
¥z	入力の終了

正規表現の使用法の詳細と、いくつかの例については、http://www.ibm.com/developerworks/java/tutorials/j-introtojava2/section9.html を参照してください。

例

以下のコードは、文字列の先頭にある 3 つの数値を検索して突き合わせます。

```
^[0-9]{3}
```

以下のコードは、文字列の末尾にある 3 つの数値を検索して突き合わせます。

```
[0-9]{3}$
```

第 2 章 スクリプト言語

スクリプト言語の概要

IBM SPSS Modeler のスクリプト機能を使用すると、SPSS Modeler ユーザー・インターフェースで動作し、出力オブジェクトを操作し、コマンド・シンタックスを実行するスクリプトを作成できます。SPSS Modeler 内から直接スクリプトを実行できます。

IBM SPSS Modeler のスクリプトは、スクリプト言語 Python で作成されています。IBM SPSS Modeler で使用される Python の Java ベースの実装を Jython と呼びます。このスクリプト言語は、以下の機能で構成されています。

- ノード、ストリーム、プロジェクト、出力、およびその他の IBM SPSS Modeler オブジェクトを参照する形式
- 上記オブジェクトを操作するのに使用されるスクリプト ステートメントまたはコマンドのセット
- 変数、パラメーター、およびその他のオブジェクトに値を設定するためのスクリプト式の言語
- コメント、行の継続、およびリテラル テキストのブロックのサポート

以下のセクションでは、Python スクリプト言語、Python の Jython 実装、および IBM SPSS Modeler 内でスクリプトを使い始めるための基本シンタックスについて説明します。特定のプロパティーとコマンドについての情報は、以後のセクションにあります。

Python と Jython

Jython は、Python スクリプト言語の実装の 1 つであり、Java 言語で記述され、Java プラットフォームと統合されています。Python は強力なオブジェクト指向スクリプト言語です。Jython は、成熟したスクリプト言語の生産性向上機能を備え、Python とは異なり、Java 仮想マシン (JVM) をサポートするすべての環境で動作します。そのため、プログラムの作成時に JVM の Java ライブラリーを使用することができます。Jython を使用すると、この違いを利用できると同時に、Python 言語の構文とほとんどの機能を使用できます。

スクリプト言語であるため、Python (およびその Jython 実装) は習得が容易で効率的にコーディングできるほか、動作するプログラムの作成に最小限の構造しか必要としません。コードは対話式で (一度に 1 行) 入力することができます。Python はインタープリター式のスクリプト言語であり、Java にあるプリコンパイルの段階がありません。Python プログラムは単なるテキスト・ファイルであり、(構文エラーがないかどうか構文解析された後に) 入力として解釈されます。単純な式 (定義済みの値など) のほか、複雑な操作 (関数定義など) もただちに実行され、使用可能になります。コードに対して行った変更を迅速にテストすることができます。しかし、スクリプトの解釈には不利な点もあります。例えば、未定義の変数を使用してもコンパイラー・エラーにならないため、その変数を使用するステートメントが実行される場合に限り、その実行のときに検出されます。この場合は、プログラムを編集して実行し、エラーをデバッグすることができます。

Python では、データやコードも含め、あらゆるものをオブジェクトとして扱います。したがって、それらのオブジェクトを一連のコードで操作することができます。一部の型 (数値や文字列など) はオブジェクトではなく値と見なすと便利ですが、この扱いは Python でもサポートされています。サポートされているヌル値が 1 つあります。このヌル値には予約名 `None` が割り当てられています。

Python スクリプトおよび Jython スクリプトの概要やスクリプト例については、www.ibm.com/developerworks/java/tutorials/j-jython1およびwww.ibm.com/developerworks/java/tutorials/j-jython2を参照してください。

Python スクリプト

Python スクリプト言語の以下のガイドでは、IBM SPSS Modeler でスクリプトを作成する場合に使用される可能性が高いコンポーネントの概要と、概念やプログラミングの基礎について取り上げます。これにより、IBM SPSS Modeler 内で使用する Python スクリプトの開発を始めるのに十分な知識を得ることができます。

操作

代入は等号 (=) を使用して行います。例えば、値「3」を「x」という変数に代入するには、以下のステートメントを使用します。

```
x = 3
```

等号は、文字列型のデータを変数に代入する場合にも使用されます。例えば、値「a string value」を「y」という変数に代入するには、以下のステートメントを使用します。

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

次の表に、よく使用される比較演算子および数値演算子と、その説明を示します。

表 5. 一般的な比較演算子および数値演算子

演算	説明
x < y	x が y より小さいかどうか
x > y	x が y より大きいかどうか
x <= y	x が y 以下かどうか
x >= y	x が y 以上かどうか
x == y	x が y と等しいかどうか
x != y	x が y と等しくないかどうか
x <> y	x が y と等しくないかどうか
x + y	y を x に加算する
x - y	y を x から減算する
x * y	x に y を乗算する
x / y	x を y で除算する
x ** y	x を y 乗する

リスト

リストは、一連の要素です。リストには任意の数の要素を入れることができ、リストの要素には任意のタイプのオブジェクトを使用できます。リストは配列と考えることもできます。リスト内の要素の数は、要素を追加、削除、または置換する際に増加または減少します。

例

[]	空のリスト。
[1]	単一の要素 (整数) を含むリスト。

`["Mike", 10, "Don", 20]`	4 つの要素 (2 つの文字列要素と、2 つの整数要素) を含むリスト。
`[[],[7],[8,9]]`	リストを含むリスト。各サブリストは、空のリスト、または整数要素のリスト。
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	整数のリスト。この例は、変数と式の使い方を示しています。

リストを変数に割り当てることができます。例えば、以下のようにします。

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

その後、このリストの特定の要素にアクセスできます。例えば、以下のようにします。

```
mylist[0]
```

これは以下のような出力になります。

```
one
```

大括弧 ([]) 内の数値は、*インデックス* と呼ばれ、リストの特定の要素を参照します。リストの各要素には、0 から始まるインデックスが付けられます。

1 つのリストの複数の要素の範囲を選択することもできます。これは*スライス* と呼ばれます。例えば、`x[1:3]` は、`x` の 2 番目の要素と 3 番目の要素を選択します。末尾のインデックスは、選択範囲の 1 つあとのインデックスです。

文字列

文字列 は、値として扱われる一連の不変の文字です。文字列は、新しい文字列になるすべての不変のシーケンス関数および演算子をサポートします。例えば、`"abcdef"[1:4]` は、`"bcd"` という出力になります。

Python では、文字は長さが 1 の文字列として表されます。

文字列リテラルは、単一引用符または三重引用符によって定義されます。単一引用符を使用して定義される文字列は行をまたぐことはできませんが、三重引用符を使用して定義される文字列は行をまたぐことができます。文字列は単一引用符 (') または二重引用符 (") で囲むことができます。引用符の内側には、エスケープされていない他の引用符、または円記号 (¥) が先行するエスケープされた引用符を入れることができます。

例

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

空白文字で区切られた複数の文字列は、Python パーサーによって自動的に連結されます。これにより、長い文字列を入力したり、単一文字列で異なる種類の引用符を混在させたりすることができます。

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

これにより、次のように出力されます。

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

文字列は、いくつかの有用なメソッドをサポートしています。次の表に、これらのメソッドの一部を示します。

表 6. 文字列メソッド

メソッド	使用法
s.capitalize()	s の頭文字を大文字にします。
s.count(ss {,start {,end}})	s[start:end] 内の ss の出現回数をカウントします。
s.startswith(str {, start {, end}}) s.endswith(str {, start {, end}})	s が str で始まっているかどうかをテストします。 s が str で終わっているかどうかをテストします。
s.expandtabs({size})	タブをスペース (デフォルトの size は 8) で置換します。
s.find(str {, start {, end}}) s.rfind(str {, start {, end}})	s の中で str の最初のインデックスを検索します。見つからない場合、結果は -1 になります。rfind は、右から左に検索します。
s.index(str {, start {, end}}) s.rindex(str {, start {, end}})	s の中で str の最初のインデックスを検索します。見つからない場合、ValueError が発生します。rindex は、右から左に検索します。
s.isalnum	文字列が英数字かどうかを確認するためのテスト。
s.isalpha	文字列が英字かどうかを確認するためのテスト。
s.isnum	文字列が数値かどうかを確認するためのテスト。
s.isupper	文字列がすべて大文字かどうかを確認するためのテスト。
s.islower	文字列がすべて小文字かどうかを確認するためのテスト。
s.isspace	文字列がすべて空白文字かどうかを確認するためのテスト。
s.istitle	文字列が頭文字が大文字の一連の英数字かどうかを確認するためのテスト。
s.lower() s.upper() s.swapcase() s.title()	すべて小文字に変換します。 すべて大文字に変換します。 大文字/小文字をすべて逆に変換します。 すべてタイトル・ケースに変換します。
s.join(seq)	seq 内の文字列を s を区切り文字として結合します。
s.splitlines({keep})	s を複数行に分割します。keep が true の場合、改行を保持します。
s.split({sep {, max}})	s を sep (デフォルトの sep は空白文字です) を使用して max 回まで「単語」に分割します。
s.ljust(width) s.rjust(width) s.center(width) s.zfill(width)	幅が width のフィールド内で文字列を左揃えします。 幅が width のフィールド内で文字列を右揃えします。 幅が width のフィールド内で文字列を中央揃えします。 0 で埋めます。
s.lstrip() s.rstrip() s.strip()	先頭の空白文字を削除します。 末尾の空白文字を削除します。 先頭と末尾の空白文字を削除します。
s.translate(str {,delc})	delc の文字を削除した後で、テーブルを使用して s を変換します。str は、長さが == 256 の文字列である必要があります。
s.replace(old, new {, max})	文字列 old をすべて、または max 個の出現箇所を文字列 new で置き換えます。

注釈

注釈は、ポンド (ハッシュ) 記号 (#) で始まるコメントです。ポンド記号に続く同じ行のすべてのテキストは、注釈の一部と見なされて無視されます。注釈は、任意の桁から開始できます。以下の例で、注釈の使用法を示します。

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

ステートメントの構文

Python のステートメントのシンタックスは非常に単純です。一般に、各ソース行は単一ステートメントです。 expression および assignment ステートメントを除いて、各ステートメントはキーワード名 (if や for など) で始まります。空白行または注釈行は、コード内の任意のステートメントの間のどこにでも挿入できます。 1 行に 2 つ以上のステートメントがある場合、各ステートメントをセミコロン (;) で区切る必要があります。

長いステートメントは、複数の行に続けることができます。この場合、次の行に続けるステートメントの末尾に円記号 (¥) を使用する必要があります。例えば、以下のようにします。

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

ある構造が括弧 (())、大括弧 ([])、または中括弧 ({}) で囲まれている場合は、円記号を挿入することなく、ステートメントをカンマの後ろで新しい行に続けることができます。例えば、以下のようにします。

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

識別子

識別子は、変数、関数、クラス、およびキーワードに名前を付けるために使用します。識別子の長さは任意ですが、先頭の文字は英字 (大文字または小文字) または下線 (_) でなければなりません。下線で始まる名前は、一般に内部名またはプライベート名のために予約されています。識別子の先頭文字の後ろに、英字、0 から 9 の数字、および下線文字をいくつでも自由に組み合わせて使用できます。

Jython には、変数、関数、またはクラスの名前に使用できない予約語がいくつかあります。これらの予約語は、以下のカテゴリーに分かれています。

- **ステートメント接頭部:** assert、break、class、continue、def、del、elif、else、except、exec、finally、for、from、global、if、import、pass、print、raise、return、try、および while
- **パラメーター接頭部:** as、import、および in
- **演算子:** and、in、is、lambda、not、および or

不適切なキーワードを使用すると、通常 SyntaxError が発生します。

コードのブロック

コードのブロックは、単一ステートメントが期待される場所に使用されるステートメントのグループです。コードのブロックは、if、elif、else、for、while、try、except、def、および class のいずれのステートメントの後ろにも置くことができます。これらのステートメントの後ろにコロン (:) を使用して、コードのブロックを続けます。例えば、以下のようにします。

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

コード・ブロックを区切るためにインデントが使用されます (Java では中括弧が使用される)。 1 つのブロック内のすべての行を同じ位置にインデントする必要があります。これは、インデントの変更が、コード・ブロックの終了を示すためです。通常は、レベルごとに 4 つのスペースでインデントします。行のインデントには、タブではなくスペースを使用することが推奨されています。スペースとタブを混在させることはできません。モジュールの最外部のブロックの行は、1 桁目から開始する必要があります。そうでないと、SyntaxError が発生します。

1 つのコード・ブロックを構成する複数のステートメント (コロンに続ける) は、セミコロンで区切って 1 行にすることもできます。例えば、以下のようにします。

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

スクリプトへの引数の引き渡し

スクリプトに引数を渡すことは、変更せずにスクリプトを繰り返し使用できるため便利です。コマンド・ライン行で渡される引数は、リスト sys.argv 内の値として渡されます。渡される値の数は、コマンド len(sys.argv) を使用して取得できます。以下に例を示します。

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

この例では、import コマンドは、sys クラス全体をインポートして、このクラスに存在しているメソッド (argv など) を使用できるようにします。

この例のスクリプトは、以下の行を使用して起動できます。

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

結果は以下の出力になります。

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

例

print キーワードは、このキーワードの直後の引数を表示します。ステートメントの後ろにコンマを続けると、改行は出力に含まれません。以下に例を示します。

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

これは以下のような出力になります。

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

for ステートメントは、コードのブロックを反復するために使用します。以下に例を示します。

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

この例では、3 つの文字列がリスト mylist1 に割り当てられます。リストの各要素が 1 行に 1 つずつ出力されます。これは以下のような出力になります。

```
one
two
three
```

この例では、for ループが要素ごとのコード・ブロックを実装するたびに、イテレーター lv がリスト mylist1 の各要素の値を順にとります。イテレーターは、任意の長さの有効な ID にすることができます。

if ステートメントは、条件ステートメントです。条件を評価し、評価の結果に基づいて true または false を返します。以下に例を示します。

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

この例では、イテレーター lv の値が評価されます。 lv の値が two の場合は、lv が two ではない場合に返されるストリングとは異なるストリングが返されます。これにより、次のように出力されます。

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

数学メソッド

math モジュールから、有用な数学メソッドにアクセスできます。次の表に、これらのメソッドの一部を示します。特に指定のない限り、すべての値は浮動小数点として返されます。

表 7. 数学メソッド

メソッド	使用法
math.ceil(x)	x の天井値を浮動小数点として返します。これは、x 以上の最小の整数です。
math.copysign(x, y)	x を y の符号で返します。copysign(1, -0.0) は、-1 を返します。
math.fabs(x)	x の絶対値を返します。
math.factorial(x)	x 階乗を返します。x が負の場合、または整数でない場合、ValueError が発生します。
math.floor(x)	x の床値を浮動小数点として返します。これは、x 以下の最大の整数です。
math.frexp(x)	x の仮数 (m) と指数 (e) を (m, e) の組みとして返します。m は浮動小数点、e は整数で、x == m * 2**e となります。x がゼロの場合は (0.0, 0) を返し、それ以外の場合は 0.5 <= abs(m) < 1 を返します。
math.fsum(iterable)	iterable の中の値の正確な浮動小数点の和を返します。

表 7. 数学メソッド (続き)

メソッド	使用法
math.isinf(x)	浮動小数点 x が正または負の無限大かどうかをチェックします。
math.isnan(x)	浮動小数点 x が NaN (非数値) かどうかをチェックします。
math.ldexp(x, i)	x * (2**i) を返します。これは、本質的に関数 frexp の逆です。
math.modf(x)	x の小数部と整数部を返します。結果は両方とも x の符号を引き継ぎ、浮動小数点です。
math.trunc(x)	Integral に切り捨てられた Real 値 x を返します。
math.exp(x)	e**x を返します。
math.log(x[, base])	指定した値 base に対する x の対数を返します。 base を指定しない場合は、x の自然対数が返されます。
math.log1p(x)	1+x (base e) の自然対数を返します。
math.log10(x)	x の 10 を底とする対数を返します。
math.pow(x, y)	x を y 乗して返します。pow(1.0, x) および pow(x, 0.0) は、x がゼロまたは NaN であるとしても、常に 1 を返します。
math.sqrt(x)	x の平方根を返します。

数学関数に加えて、有用な三角関数メソッドもあります。次の表に、これらのメソッドを示します。

表 8. 三角関数メソッド

メソッド	使用法
math.acos(x)	x の逆余弦をラジアンで返します。
math.asin(x)	x の逆正弦をラジアンで返します。
math.atan(x)	x の逆正接をラジアンで返します。
math.atan2(y, x)	atan(y / x) をラジアンで返します。
math.cos(x)	x の余弦をラジアンで返します。
math.hypot(x, y)	ユークリッドノルム sqrt(x*x + y*y) を返します。これは原点から点 (x, y) へのベクトルの長さです。
math.sin(x)	x の正弦をラジアンで返します。
math.tan(x)	x の正接をラジアンで返します。
math.degrees(x)	角 x をラジアンから度に変換します。
math.radians(x)	角 x を度からラジアンに変換します。
math.acosh(x)	x の逆双曲線余弦を返します。
math.asinh(x)	x の逆双曲線正弦を返します。
math.atanh(x)	x の逆双曲線正接を返します。
math.cosh(x)	x の双曲線余弦を返します。
math.sinh(x)	x の双曲線余弦を返します。
math.tanh(x)	x の双曲線正接を返します。

2 つの数学定数もあります。math.pi の値は、数学定数 pi です。math.e の値は、数学定数 e です。

非 ASCII 文字の使用

非 ASCII 文字を使用するには、Python では、文字列を Unicode に明示的にエンコードまたはデコードする必要があります。IBM SPSS Modeler では、Python スクリプトは UTF-8 (非 ASCII 文字をサポートする標準 Unicode) でエンコードされていると想定されます。以下のスクリプトは、Python コンパイラーが SPSS Modeler によって UTF-8 に設定されているため、コンパイルされます。

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

しかし、結果ノードのラベルは正しくありません。

図 3. 非 ASCII 文字を含むノード・ラベル (正しく表示されていない)

ストリング・リテラル自体が Python によって ASCII 文字列に変換されているため、このラベルは正しくありません。

Python では、文字列リテラルの前に u 文字を追加することによって、Unicode 文字列リテラルを指定できます。

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

これにより、Unicode 文字列が作成され、ラベルが正しく表示されます。

図 4. 非 ASCII 文字を含むノード・ラベル (正しく表示されている)

Python と Unicode の使用は、本書の範囲を超えた大きなトピックです。このトピックを詳細に扱った書籍やオンライン情報源が数多くあります。

オブジェクト指向プログラミング

オブジェクト指向プログラミングは、対象問題のモデルをプログラム内で作成するという概念に基づいています。オブジェクト指向プログラミングにより、プログラミング・エラーが減り、コードの再使用が促進されます。 Python は、オブジェクト指向言語です。Python で定義されるオブジェクトには、以下の特徴があります。

- **同一:** 各オブジェクトは個別であり、これはテスト可能でなければなりません。`is` テストと `is not` テストは、この目的のために存在しています。
- **状態。**各オブジェクトは、状態を格納できる必要があります。フィールドやインスタンス変数などの属性は、この目的のために存在しています。
- **振る舞い。**各オブジェクトは、状態を操作できる必要があります。メソッドは、この目的のために存在します。

Python には、オブジェクト指向プログラミングをサポートするための以下の特徴があります。

- **クラス・ベースのオブジェクト作成。**クラスは、オブジェクトを作成するためのテンプレートです。オブジェクトは、振る舞いが関連づけられているデータ構造です。
- **ポリモアフィズムによる継承。**Python は、単一継承と多重継承をサポートしています。 Python のすべてのインスタンス・メソッドは、ポリモアフィックであり、サブクラスによるオーバーライドが可能です。
- **データ隠蔽によるカプセル化。**Python では、属性を隠すことができます。隠すと、クラスの外側からは、そのクラスのメソッドによってのみ属性にアクセスできるようになります。クラスには、データを変更するためのメソッドを実装します。

クラスの定義

Python クラスの中では、変数とメソッドの両方を定義できます。Java と異なり、Python では、1 つのソース・ファイル (モジュール) で任意の数の公開クラスを定義できます。したがって、Python のモジュールは Java のパッケージに似ていると考えることができます。

Python では `class` ステートメントを使用してクラスを定義します。`class` ステートメントは、次の形式になっています。

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

or

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

クラスを定義するときには、任意の数の代入 ステートメントを記述することができます (記述しなくても構いません)。これにより、クラスのすべてのインスタンスで共有されるクラス属性が作成されます。また、任意の数の*関数* 定義を記述することもできます (記述しなくても構いません)。これらの関数定義により、メソッドが作成されます。スーパークラスのリストはオプションです。

クラス名はスコープの中 (モジュール、関数、またはクラスの中) で固有でなければなりません。複数の変数を定義して同じクラスを参照することができます。

クラス・インスタンスの作成

クラスは、クラス (共有) 属性の保持やクラス・インスタンスの作成に使用します。クラスのインスタンスを作成するには、そのクラスが関数であるかのように呼び出します。たとえば、次のクラスを考慮してください。

```
class MyClass:
    pass
```

クラスを完結させるためにはステートメントが必要ですがプログラムとしては動作が不要であるため、ここでは pass ステートメントを使用しています。

以下のステートメントは、クラス MyClass のインスタンスを作成します。

```
x = MyClass()
```

クラス・インスタンスへの属性の追加

Java と異なり、Python ではクライアントがクラスのインスタンスに属性を追加することができます。変更されるインスタンスは 1 つだけです。例えば、インスタンス x に複数の属性を追加するには、以下のようにしてそのインスタンスに新しい値を設定します。

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

クラス属性およびメソッドの定義

クラスにバインドされた変数はすべてクラス属性 です。クラス内で定義された関数はすべてメソッド です。メソッドは、クラスのインスタンス (慣習として self と呼びます) を第 1 引数として受け取ります。例えば、クラス属性およびメソッドを定義するには、以下のコードを入力します。

```
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
```

クラスの内側では、クラス属性に対するすべての参照をクラス名で修飾する必要があります (MyClass.attr1 など)。インスタンス属性に対する参照は、すべて self 変数で修飾する必要があります (self.text など)。クラスの外側では、クラス属性に対するすべての参照をクラス名で修飾するか (MyClass.attr1 など)、クラスのインスタンスで修飾する (x をクラスのインスタンスとすると x.attr1 などとする) 必要があります。クラスの外側では、インスタンス変数に対するすべての参照をクラスのインスタンスで修飾する必要があります (x.text など)。

非表示変数

プライベート 変数を作成することにより、データを隠蔽することができます。プライベート変数にアクセスできるのはそのクラス自体に限られます。__xxx または __xxx_yyy という形式で (2 個の下線を前に付けて) 名前を宣言すると、Python パーサーは、宣言された名前に自動的にクラス名を追加して隠蔽された変数を作成します。例を示します。

```
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
```

Java と異なり、Python では、インスタンス変数に対する参照はすべて self で修飾する必要があります。暗黙的な this の使用はありません。

継承

クラスを継承する機能は、オブジェクト指向プログラミングの根幹をなします。Python は、単一継承と多重継承の両方をサポートしています。*単一継承* は、スーパークラスが 1 つしか存在できないことを意味します。*多重継承* は、複数のスーパークラスが存在できることを意味します。

継承は、他のクラスのサブクラスを定義することで実装します。任意の数の Python クラスをスーパークラスにすることができます。Python の Jython 実装では、直接または間接に継承できる Java クラスは 1 つだけです。スーパークラスを提供する必要はありません。

スーパークラスのすべての属性やメソッドはいずれのサブクラスにも存在し、そのクラス自体によって使用できるほか、属性やメソッドが隠蔽されていなければ任意のクライアントから使用することもできます。サブクラスのインスタンスは任意の場所で使用でき、スーパークラスのインスタンスも使用できます。これがポリモアフィズム の一例です。これらの機能によって再利用が可能になり、拡張が容易になります。

例

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

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

第 3 章 IBM SPSS Modeler でのスクリプト

スクリプトの種類

IBM SPSS Modeler には、以下の 3 種類のスクリプトがあります。

- *ストリーム・スクリプト* は、単一ストリームの実行を制御するために使用され、ストリーム内に格納されます。
- *スーパーノード・スクリプト* は、スーパーノードの動作を制御するために使用されます。
- *スタンドアロン・スクリプトまたはセッション・スクリプト* は、さまざまなストリームにわたって実行を調整するために使用できます。

さまざまなメソッドを IBM SPSS Modeler のスクリプトで使用することができ、これらメソッドによって SPSS Modeler の広範な機能にアクセスできます。これらのメソッドは、より高度な機能を作成するために 37 ページの『第 4 章 スクリプト API』 でも使用されます。

ストリーム、スーパーノード・ストリーム、およびダイアグラム

多くの場合、*ストリーム* という語は、ファイルからロードされるストリームであれ、スーパーノード内で使用されるストリームであれ、同じ意味を持ちます。一般に、ストリームは、互いに接続された実行可能なノードの集合を意味します。しかし、スクリプトの場合は、あらゆる場所ですべての操作がサポートされるわけではありません。つまり、スクリプト作成者は、どのストリーム・バリアントを使用しているのかを認識している必要があります。

ストリーム

ストリームは、IBM SPSS Modeler の主なドキュメント・タイプです。ストリームは保存、ロード、編集、および実行することができます。ストリームには、パラメーター、グローバル値、スクリプト、およびその他の情報を関連付けることもできます。

スーパーノード・ストリーム

スーパーノード・ストリーム は、スーパーノード内で使用される種類のストリームです。通常のストリームと同様、互いにリンクされているノードが含まれています。スーパーノード・ストリームは、以下のような点で通常のストリームと異なっています。

- パラメーターおよびスクリプトは、スーパーノード・ストリームではなく、スーパーノード・ストリームを所有しているスーパーノードに関連付けられています。
- スーパーノード・ストリームには、スーパーノードの種類に応じて、追加の入力コネクター・ノードや出力コネクター・ノードがあります。これらのコネクター・ノードは、スーパーノード・ストリームに情報を渡したり、スーパーノード・ストリームから情報を取り出したりするために使用され、スーパーノードの作成時に自動的に作成されます。

ダイアグラム

ダイアグラム という用語は、通常のストリームとスーパーノード・ストリームの両方でサポートされる機能 (ノードの追加や削除、ノード間の接続の変更など) を含んでいます。

ストリームの実行

以下の例は、ストリーム内のすべての実行可能ノードを実行する最もシンプルなタイプのストリーム・スクリプトです。

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

以下の例も、ストリーム内のすべての実行可能ノードを実行します。

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

この例では、ストリームを変数 stream に格納しています。通常、スクリプトはストリームまたはストリーム内のノードを変更するために使用されるため、ストリームを変数に格納すると便利です。ストリームを格納する変数を作成することによって、スクリプトはより簡潔になります。

スクリプト・コンテキスト

modeler.script モジュールは、スクリプトが実行されるコンテキストを提供します。このモジュールは、実行時に SPSS Modeler スクリプトに自動的にインポートされます。このモジュールは、スクリプトがその実行環境にアクセスするための方法を提供する 4 つの関数を定義しています。

- session() 関数は、スクリプトのセッションを返します。セッションは、ストリームを実行するために使用されているロケールや、SPSS Modeler バックエンド (ローカル・プロセス、またはネットワーク SPSS Modeler Server) などの情報を定義します。
- stream() 関数は、ストリームとスーパーノード・スクリプトで使用できます。この関数は、実行中のストリーム・スクリプトまたはスーパーノード・スクリプトを所有しているストリームを返します。
- diagram() 関数は、スーパーノード・スクリプトで使用できます。この関数は、スーパーノード内のダイアグラムを返します。その他のスクリプトのタイプの場合、この関数は stream() 関数と同じ内容を返します。
- supernode() 関数は、スーパーノード・スクリプトで使用できます。この関数は、実行中のスクリプトを所有しているスーパーノードを返します。

これら 4 つの関数と出力を次の表に要約します。

表 9. modeler.script 関数の要約

スクリプト・タイプ	session()	stream()	diagram()	supernode()
スタンドアロン	セッションを返します	スクリプト起動時の現在の管理対象ストリーム (例えば、バッチ・モード -stream オプションによって渡されたストリーム) か、None を返します。	stream() と同じ	なし
ストリーム	セッションを返します	ストリームを返します	stream() と同じ	なし
スーパーノード	セッションを返します	ストリームを返します	スーパーノード・ストリームを返します	スーパーノードを返します

modeler.script モジュールは、終了コードでスクリプトを終了する方法も定義します。 exit(*exit-code*) 関数は、スクリプトの実行を停止し、指定された整数の終了コードを返します。

ストリーム用に定義されているメソッドの 1 つに runAll(List) があります。このメソッドは、すべての実行可能ノードを実行します。ノードを実行することで生成されるモデルまたは出力は、指定されたリストに追加されます。

通常、ストリームを実行すると、モデルやグラフなどの出力が生成されます。この出力をキャプチャーするために、スクリプトは、リストに初期化される変数を提供できます。例えば、以下のとおりです。

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

実行が完了すると、実行によって生成されたオブジェクトに results リストからアクセスできます。

既存のノードの参照

多くの場合、ストリームは、ストリームの実行前に変更する必要があるいくつかのパラメーターを使用して事前構築されています。これらのパラメーターを変更するには、以下の作業を行います。

1. 関連するストリーム内のノードを見つける。
2. ノードまたはストリーム (あるいは両方) の設定を変更する。

ノードの検索

ストリームでは、さまざまな方法で既存のノードを見つけることができます。これらのメソッドを次の表に要約します。

表 10. 既存のノードを見つけるためのメソッド

メソッド	戻り値の型	説明
s.findAll(type, label)	集計棒グラフ	指定したデータ型とラベルを持つすべてのノードのリストを返します。データ型またはラベルのいずれかが None の場合は、もう一方のパラメーターが使用されます。
s.findAll(filter, recursive)	集計棒グラフ	指定したフィルターで受け入れられるすべてのノードの集合を返します。recursive フラグが True の場合は、指定したストリーム内のスーパーノードも検索されます。
s.findByID(id)	ノード	指定した ID のノードを返すか、そのようなノードが存在しない場合は None を返します。検索は現行ストリームに限定されます。
s.findByType(type, label)	ノード	指定したデータ型またはラベルを持つノード、あるいはその両方を持つノードを返します。データ型または名前のいずれかが None の場合は、もう一方のパラメーターが使用されます。一致するノードが複数ある場合は、任意のノードが返されます。一致するノードがない場合、戻り値は None です。

表 10. 既存のノードを見つけるためのメソッド (続き)

メソッド	戻り値の型	説明
s.findDownstream(fromNodes)	集計棒グラフ	指定したノードのリストから検索し、指定したノードの下流にある一連のノードを返します。返されるリストには、最初に指定したノードも含まれます。
s.findUpstream(fromNodes)	集計棒グラフ	指定したノードのリストから検索し、指定したノードの上流にある一連のノードを返します。返されるリストには、最初に指定したノードも含まれます。

例えば、スクリプトがアクセスする必要のある単一のフィルター・ノードがストリームに含まれている場合、そのフィルター・ノードは、以下のスクリプトを使用して見つけることができます。

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

あるいは、ノードの ID (ノード・ダイアログ・ボックスの「注釈」タブに示されている) が分かる場合は、その ID を使用してノードを検索できます。例えば、以下のようにします。

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

プロパティを設定する

ノード、ストリーム、モデル、および出力のすべてには、アクセス可能で、ほとんどの場合に設定可能なプロパティーがあります。通常、プロパティーは、オブジェクトの動作および外観を変更するために使用されます。オブジェクトのプロパティーのアクセスおよび設定に使用できるメソッドを次の表に要約します。

表 11. オブジェクトのプロパティーのアクセスおよび設定のためのメソッド

メソッド	戻り値の型	説明
p.getPropertyValue(propertyName)	オブジェクト	指定したプロパティーの値を返すか、そのようなプロパティーが存在しない場合は None を返します。
p.setPropertyValue(propertyName, value)	なし	指定したプロパティーの値を設定します。
p.setPropertyValues(properties)	なし	指定したプロパティーの値を設定します。プロパティー・マップの各項目は、プロパティー名を表すキーと、そのプロパティーに割り当てる必要がある値で構成されています。
p.getKeyedPropertyValue(propertyName, keyName)	オブジェクト	指定したプロパティーの値および関連付けられているキーを返すか、そのようなプロパティーまたはキーが存在しない場合は None を返します。
p.setKeyedPropertyValue(propertyName, keyName, value)	なし	指定したプロパティーおよびキーの値を設定します。

例えば、ストリームの先頭にある可変長ファイル・ノードの値を設定する場合は、以下のスクリプトを使用できます。

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

あるいは、フィルター・ノードからフィールドをフィルタリングできます。この場合は、フィールド名に対して値も入力します。例えば、以下のようにします。

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

ノードの作成とストリームの変更

新しいノードを既存のストリームに追加する場合があります既存のストリームにノードを追加するには、通常以下の作業を行います。

1. ノードを作成する。
2. ノードを既存のストリーム・フローにリンクする。

ノードの作成

ストリームでは、さまざまな方法でノードを作成できます。これらのメソッドを次の表に要約します。

表 12. ノードを作成するためのメソッド

メソッド	戻り値の型	説明
s.create(nodeType, name)	ノード	指定したデータ型のノードを作成して、指定したストリームに追加します。
s.createAt(nodeType, name, x, y)	ノード	指定したデータ型のノードを作成して、指定したストリームの指定した場所に追加します。x < 0 または y < 0 の場合、場所は設定されません。
s.createModelApplier(modelOutput, name)	ノード	提供されたモデル出力オブジェクトから派生したモデル・アプライヤー・ノードを作成します。

例えば、ストリーム内に新しいデータ型ノードを作成するには、以下のスクリプトを使用できます。

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

ノードのリンクとリンク解除

ストリーム内に新しいノードを作成する場合、そのノードを使用するにはノードのシーケンスに接続する必要があります。ストリームには、ノードをリンクおよびリンク解除するための多くのメソッドがあります。これらのメソッドを次の表に要約します。

表 13. ノードをリンクおよびリンク解除するためのメソッド

メソッド	戻り値の型	説明
`s.link(source, target)`	なし	ソース・ノードとターゲット・ノードの間に新しいリンクを作成します。
`s.link(source, targets)`	なし	ソース・ノードと指定されたリスト内の各ターゲットの間に新しいリンクを作成します。
`s.linkBetween(inserted, source, target)`	なし	他の 2 つのノード・インスタンス (ソース・ノートとターゲット・ノード) の間にノードを接続し、挿入したノードの位置がこれらのノードの間になるように設定します。ソース・ノートとターゲット・ノードの間の直接リンクが最初に削除されます。
`s.linkPath(path)`	なし	ノード・インスタンスの間の新しいパスを作成します。最初のノードが 2 番目のノードにリンクされ、2 番目のノードが 3 番のノードにリンクされ、以下同様にリンクされます。
`s.unlink(source, target)`	なし	ソース・ノードとターゲット・ノードの間の直接リンクを削除します。
`s.unlink(source, targets)`	なし	ソース・ノードと指定されたターゲット・リスト内の各オブジェクトの間の直接リンクを削除します。
`s.unlinkPath(path)`	なし	ノード・インスタンスの間に存在するパスをすべて削除します。
`s.disconnect(node)`	なし	指定されたノードと、指定したストリーム内の他のすべてのノードの間のリンクを削除します。
`s.isValidLink(source, target)`	*boolean*	指定したソース・ノードとターゲット・ノードの間にリンクを作成できる場合は `True` を返します。このメソッドは、指定したストリームに両方のオブジェクトが属していること、ソース・ノードがリンクを提供でき、ターゲット・ノードがリンクを受け取れること、このようなリンクを作成してもストリーム内に循環が発生しないことを検査します。

以下に示すサンプル・スクリプトは、以下の 5 つのタスクを実行します。

1. 可変長ファイル入力ノード、フィルター・ノード、およびテーブル出力ノードを作成する。
2. ノード同士を接続する。
3. 可変長ファイル入力ノードにファイル名を設定する。
4. 結果出力から「Drug」フィールドをフィルタリングする。
5. テーブル・ノードを実行する。

```
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)
```

ノードのインポート、置換、および削除

ノードの作成や接続だけでなく、多くの場合にストリームのノードの置換や削除も必要です。ノードのインポート、置換、および削除に使用できるメソッドを次の表に要約します。

表 14. ノードをインポート、置換、および削除するためのメソッド

メソッド	戻り値の型	説明
s.replace(originalNode, replacementNode, discardOriginal)	なし	指定したストリームの指定したノードを置換します。元のノードと置換ノードの両方が、指定したストリームによって所有されている必要があります。
s.insert(source, nodes, newIDs)	一覧	指定されたリスト内のノードのコピーを挿入します。指定されたリスト内のすべてのノードが、指定したストリームに含まれていると想定されます。newIDs フラグは、ノードごとに新しい ID を生成するのか、または既存の ID をコピーして使用するのかを示します。ストリーム内のすべてのノードの ID は固有であると想定されているため、指定したストリームとソース・ストリームが同じである場合、このフラグを True に設定する必要があります。このメソッドは新しく挿入されたノードのリストを返しますが、ノードの順序は定義されていません (つまり、順序は入力リストのノードの順序と必ずしも同じであるとは限りません)。
s.delete(node)	なし	指定したストリームから指定したノードを削除します。ノードは、指定したストリームによって所有されている必要があります。
s.deleteAll(nodes)	なし	指定したストリームから指定したすべてのノードを削除します。集合内のすべてのノードが、指定したストリームに属している必要があります。
s.clear()	なし	指定したストリームからすべてのノードを削除します。

ストリーム内のノードのトラバース

一般的な要件として、特定のノードの上流または下流にあるノードを識別したい場合があります。ストリームには、これらのノードを識別するために使用できる多くのメソッドがあります。これらのメソッドを次の表に要約します。

表 15. 上流または下流のノードを識別するためのメソッド

メソッド	戻り値の型	説明
s.iterator()	イテレーター	指定したストリームに含まれているノード・オブジェクトのイテレーターを返します。 next() 関数の呼び出しの間にストリームが変更される場合、イテレーターの動作は未定義です。
s.predecessorAt(node, index)	ノード	指定したノードの指定された直接の先行ノードを返すか、インデックスが境界を超えている場合は None を返します。
s.predecessorCount(node)	*int*	指定されたノードの直接の先行ノードの数を返します。
s.predecessors(node)	一覧	指定されたノードの直接の先行ノードを返します。
s.successorAt(node, index)	ノード	指定したノードの指定した直接の後続ノードを返すか、インデックスが境界を超えている場合は None を返します。
s.successorCount(node)	*int*	指定されたノードの直接の後続ノードの数を返します。
s.successors(node)	一覧	指定されたノードの直接の後続ノードを返します。

ノードに関する情報の入手

ノードは、データ・インポート・ノードおよびデータ・エクスポート・ノード、モデル構築ノード、その他の種類のノードなど、さまざまなカテゴリーに分類されます。各ノードには、ノードに関する情報を見つけるために使用できる多くのメソッドがあります。

ノードの ID、名前、およびラベルを取得するために使用できるメソッドを次の表に要約します。

表 16. ノードの ID、名前、およびラベルを取得するためのメソッド

メソッド	戻り値の型	説明
n.getLabel()	*文字列*	指定したノードの表示ラベルを返します。ラベルがプロパティー custom_name の値となるのは、このプロパティーが空文字列ではなく、use_custom_name プロパティーが設定されていない場合のみです。これ以外の場合、ラベルは getName() の値になります。

表 16. ノードの ID、名前、およびラベルを取得するためのメソッド (続き)

メソッド	戻り値の型	説明
n.setLabel(label)	なし	指定したノードの表示ラベルを設定します。新しいラベルが空文字列ではない場合、この文字列がプロパティー custom_name に割り当てられ、指定したラベルが優先されるようにプロパティー use_custom_name に False が割り当てられます。これ以外の場合は、空文字列が custom_name に割り当てられ、プロパティー use_custom_name に True が割り当てられます。
n.getName()	*文字列*	指定されたノードの名前を戻します。
n.getID()	*文字列*	指定したノードの ID を戻します。新しいノードが作成されるたびに、新しい ID が作成されます。この ID は、ストリームの一部としてノードが保存されるときに、ノードで永続化され、ストリームを開いたときにノード ID が保持されるようになります。ただし、保存したノードがストリームに挿入される場合、挿入されたノードは新しいオブジェクトと見なされ、新しい ID が割り当てられます。

ノードに関するその他の情報を取得するために使用できるメソッドを次の表に要約します。

表 17. ノードに関する情報を取得するためのメソッド

メソッド	戻り値の型	説明
n.getTypeName()	*文字列*	このノードのスクリプト名を戻します。これは、このノードの新しいインスタンスを作成するために使用できる名前と同じです。
n.isInitial()	*Boolean*	これが*最初の* ノード (ストリームの先頭にあるノード) である場合は、True を返します。
n.isInline()	*Boolean*	これがインライン・ノード (ストリームの中間にあるノード) である場合は、True を返します。
n.isTerminal()	*Boolean*	これが*終端* ノード (ストリームの末尾にあるノード) である場合は、True を返します。
n.getXPosition()	*int*	ストリーム内のノードの x 位置オフセットを返します。
n.getYPosition()	*int*	ストリーム内のノードの y 位置オフセットを返します。
n.setXYPosition(x, y)	なし	ストリーム内のノードの位置を設定します。

表 17. ノードに関する情報を取得するためのメソッド (続き)

メソッド	戻り値の型	説明
`n.setPositionBetween(source, target)`	なし	指定されたノードの間に位置するようにストリーム内のノードの位置を設定します。
`n.isCacheEnabled()`	*Boolean*	キャッシュが有効な場合は `True` を返し、そうでない場合は `False` を返します。
`n.setCacheEnabled(val)`	なし	このオブジェクトのキャッシュを有効または無効にします。キャッシュがいっぱいの場合にキャッシュが無効になると、キャッシュはフラッシュされます。
`n.isCacheFull()`	*Boolean*	キャッシュがいっぱいの場合は `True` を返し、そうでない場合は `False` を返します。
`n.flushCache()`	なし	このノードのキャッシュをフラッシュします。キャッシュが有効でない場合やいっぱいでない場合、影響はありません。

第 4 章 スクリプト API

スクリプト API の概要

スクリプト API により、幅広い SPSS Modeler 機能にアクセスすることができます。ここまで説明してきたメソッドはいずれも API の一部であり、追加でインポートを行わなくてもスクリプト内から暗黙的にアクセスすることができます。ただし、API クラスを参照する必要がある場合は、以下のステートメントで明示的に API をインポートする必要があります。

```
import modeler.api
```

この import ステートメントは、多くのスクリプト API の例で必要になります。

例: カスタム・フィルターを使用したノードの検索

29 ページの『ノードの検索』のセクションでは、検索基準としてノードのタイプ名を使用してストリームのノードを検索する例を示しました。場合によっては、より汎用的な検索が必要になります。そのような検索を実装するには、NodeFilter クラスおよびストリームの findAll() メソッドを使用します。この種の検索は以下の 2 段階で行います。

1. NodeFilter を拡張し、カスタム・バージョンの accept() メソッドを実装する新しいクラスを作成します。
2. この新しいクラスのインスタンスでストリームの findAll() メソッドを呼び出します。これにより、accept() メソッドで定義された基準を満たすすべてのノードが返されます。

ストリームのノードのうち、ノードのキャッシュが有効になっているノードを検索する方法を以下の例に示します。返されたノードのリストを使用して、それらのノードのキャッシュをフラッシュするか無効化することができます。

```
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)
```

メタデータ: データに関する情報

ストリーム内では複数のノードが互いに接続されているため、各ノードで使用可能な列またはフィールドに関する情報を使用できます。これにより、例えば Modeler UI では、ソートまたは集計の基準となるフィールドを選択できます。この情報はデータ・モデルと呼ばれます。

スクリプトは、ノードを出入りするフィールドを調べることによって、データ・モデルにアクセスすることも可能です。一部のノードでは、入力データ・モデルと出力データ・モデルが同じです。例えば、ソート・ノードは、レコードを並べ替えるだけで、データ・モデルを変更することはありません。一部のノード (フィールド作成ノードなど) では、新しいフィールドを追加できます。他のノード (フィルター・ノードなど) は、フィールドの名前を変更したり、フィールドを削除したりすることができます。

以下の例では、スクリプトは標準の IBM SPSS Modeler druglearn.str ストリームを使用し、いずれかの入力フィールドが欠落した状態のモデルがフィールドごとに構築されます。これは、以下のように行われます。

1. データ型ノードから出力データ・モデルにアクセスする。
2. 出力データ・モデルの各フィールドをループする。
3. 各入力フィールドのフィルター・ノードを変更する。
4. 構築中のモデルの名前を変更する。
5. モデル構築ノードを実行する。

注: druglean.str ストリームのスクリプトを実行する前に、スクリプト言語を Python に設定することを忘れないでください (このストリームは IBM SPSS Modeler の旧バージョンで作成されているため、ストリームのスクリプト言語はレガシーに設定されます)。

```
import modeler.api

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

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

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

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

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

DataModel オブジェクトには、データ・モデル内のフィールドまたは列に関する情報にアクセスするための多くのメソッドがあります。これらのメソッドを次の表に要約します。

表 18. フィールドまたは列に関する情報にアクセスするための DataModel オブジェクト・メソッド

メソッド	戻り値の型	説明
d.getColumnCount()	*int*	データ・モデル内の列の数を返します。
d.columnIterator()	イテレーター	各列を「ファイル順」の挿入順序で返すイテレーターを返します。イテレーターは列のインスタンスを返します。
d.nameIterator()	イテレーター	各列の名前を「ファイル順」の挿入順序で返すイテレーターを返します。
d.contains(name)	*Boolean*	指定した名前の列がこの DataModel 内に存在する場合は True を返し、存在しない場合は False を返します。

表 18. フィールドまたは列に関する情報にアクセスするための DataModel オブジェクト・メソッド (続き)

メソッド	戻り値の型	説明
d.getColumn(name)	列(L)	指定された名前の列を戻します。
d.getColumnGroup(name)	ColumnGroup	指定した列グループを返すか、指定した列グループが存在しない場合は None を返します。
d.getColumnGroupCount()	*int*	このデータ・モデル内の列グループの数を返します。
d.columnGroupIterator()	イテレーター	各列グループを順番に返すイテレーターを返します。
d.toArray()	Column[]	データ・モデルを列の配列として返します。列は「ファイル順」の挿入順序になります。

各フィールド (Column オブジェクト) には、列に関する情報にアクセスするための多くのメソッドが含まれています。以下の表に、これらのメソッドを示します。

表 19. 列に関する情報にアクセスするための Column オブジェクト・メソッド

メソッド	戻り値の型	説明
c.getColumnName()	*文字列*	列の名前を戻します。
c.getColumnLabel()	*文字列*	列のラベルを返すか、列にラベルが関連付けられていない場合は空文字列を返します。
c.getMeasureType()	MeasureType	列の測定タイプを返します。
c.getStorageType()	StorageType	列のストレージ・タイプを返します。
c.isMeasureDiscrete()	*Boolean*	列が離散型の場合は True を返します。セット型またはフラグ型の列は、離散型と見なされます。
c.isModelOutputColumn()	*Boolean*	列がモデル出力列の場合は True を返します。
c.isStorageDatetime()	*Boolean*	列のストレージが、時刻、日付、またはタイム・スタンプの値の場合は True を返します。
c.isStorageNumeric()	*Boolean*	列のストレージが整数または実数の場合は True を返します。
c.isValidValue(value)	*Boolean*	指定した値がこのストレージで有効な場合は True を返し、有効な列の値が分かる場合は valid を返します。
c.getModelingRole()	ModelingRole	列のモデル作成の役割を返します。
c.getSetValues()	Object[]	列の有効な値の配列を返すか、値が分からない場合または列がセット型でない場合は None を返します。
c.getValueLabel(value)	*文字列*	列の値のラベルを返すか、値にラベルが関連付けられていない場合は空文字列を返します。

表 19. 列に関する情報にアクセスするための Column オブジェクト・メソッド (続き)

メソッド	戻り値の型	説明
c.getFalseFlag()	オブジェクト	列の「false」標識値を返すか、値が分からない場合または列がフラグ型でない場合は None を返します。
c.getTrueFlag()	オブジェクト	列の「true」標識値を返すか、値が分からない場合または列がフラグ型でない場合は None を返します。
c.getLowerBound()	オブジェクト	列の値の下限値を返すか、値が分からない場合または列が連続型でない場合は None を返します。
c.getUpperBound()	オブジェクト	列の値の上限値を返すか、値が分からない場合または列が連続型でない場合は None を返します。

列に関する情報にアクセスするほとんどのメソッドには、DataModel オブジェクトに定義されている同等のメソッドがあります。たとえば、次の 2 つのステートメントは、同じものを指します。

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

生成されたオブジェクトへのアクセス

ストリームを実行するには、通常、追加の出力オブジェクトを生成する必要があります。これらの追加のオブジェクトは、新規モデル (以降の実行で使用する情報を提供する出力) にすることができます。

下記の例では、ストリームの開始点として druglearn.str ストリームを再度使用しています。この例では、ストリームのすべてのノードを実行し、結果をリストに格納します。次に、スクリプトでは結果全体についてループし、実行の結果として得られたモデル出力を IBM SPSS Modeler モデル (.gm) ファイルとして保存し、モデルを PMML エクスポートします。

```
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)
```

タスク実行クラスは、よく使用するさまざまな処理を実行するのに便利です。このクラスで使用可能なメソッドの要約を以下の表に示します。

表 20. よく使用する処理を実行するためのタスク実行クラスのメソッド

メソッド	戻り値の型	説明
t.createStream(name, autoConnect, autoManage)	ストリーム	新規ストリームを作成して返します。非公開でストリームを作成してユーザーから不可視にする必要があるコードでは、autoManage フラグを False に設定する必要があります。
t.exportDocumentToFile(documentOutput, filename, fileFormat)	なし	指定されたファイル形式を使用してストリームの説明をファイルにエクスポートします。
t.exportModelToFile(modelOutput, filename, fileFormat)	なし	指定されたファイル形式を使用してモデルをファイルにエクスポートします。
t.exportStreamToFile(stream, filename, fileFormat)	なし	指定されたファイル形式を使用してストリームをファイルにエクスポートします。
t.insertNodeFromFile(filename, diagram)	ノード	指定されたファイルからノードを読み込み、指定されたダイアグラムに挿入して返します。ノード・オブジェクトとスーパーノード・オブジェクトの両方の読み込みに使用することができます。
t.openDocumentFromFile(filename, autoManage)	DocumentOutput	指定されたファイルからドキュメントを読み込んで返します。
t.openModelFromFile(filename, autoManage)	ModelOutput	指定されたファイルからモデルを読み込んで返します。
t.openStreamFromFile(filename, autoManage)	ストリーム	指定されたファイルからストリームを読み込んで返します。
t.saveDocumentToFile(documentOutput, filename)	なし	指定されたファイルの場所にドキュメントを保存します。
t.saveModelToFile(modelOutput, filename)	なし	指定されたファイルの場所にモデルを保存します。
t.saveStreamToFile(stream, filename)	なし	指定されたファイルの場所にストリームを保存します。

エラーの処理

Python 言語には、try...except コード・ブロックによるエラー処理が備わっています。スクリプト内でこれを使用すると、例外をトラップし、対処しなければスクリプトが終了してしまう問題を処理することができます。

下記のスクリプト例では、IBM SPSS Collaboration and Deployment Services Repository からモデルを取得しようとしています。この操作では例外が発生する可能性があります (例えば、リポジトリーのログイン資格情報が正しく設定されていない場合や、リポジトリーのパスが誤っている場合が考えられます)。スクリプトでその事態が発生すると、ModelerException がスローされます (IBM SPSS Modeler によって生成される例外は、すべて modeler.api.ModelerException から派生しています)。

```
import modeler.api

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

注: スクリプト操作によっては、標準の Java 例外が発生する場合があります。それらの例外は ModelerException から派生していません。それらの例外をキャッチするために、追加の except ブロックを使用してすべての Java 例外をキャッチすることができます。以下に例を示します。

```
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()
```

ストリーム、セッション、およびスーパーノード・パラメーター

パラメーターは、直接スクリプトの中で値を固定的にコーディングするのではなく、実行時に渡す場合に便利です。パラメーターとその値は、ストリームの場合と同じ方法で定義します。つまり、ストリームまたはスーパーノードのパラメーター・テーブルの項目として、またはコマンド・ラインのパラメーターとして定義します。以下の表に示すように、Stream クラスおよび SuperNode クラスは、ParameterProvider オブジェクトによって定義される一連の関数を実装しています。セッションには getParameters() の呼び出しが用意されており、呼び出すと、それらの関数を定義するオブジェクトが返されます。

表 21. ParameterProvider オブジェクトによって定義されている関数

メソッド	戻り値の型	説明
p.parameterIterator()	反復	このオブジェクトのパラメーター名の反復子を返します。

表 21. *ParameterProvider* オブジェクトによって定義されている関数 (続き)

メソッド	戻り値の型	説明
p.getParameterDefinition(parameterName)	ParameterDefinition	指定された名前を持つパラメーターのパラメーター定義を返します。該当するパラメーターがこのプロバイダーに存在しない場合は None を返します。結果は、メソッドが呼び出された時点での定義のスナップショットである可能性があり、その後にこのプロバイダーを通じてパラメーターに対して行われた変更が反映されているとは限りません。
p.getParameterLabel(parameterName)	*文字列*	指定されたパラメーターのラベルを返します。該当するパラメーターが存在しない場合は None を返します。
p.setParameterLabel(parameterName, label)	なし	指定されたパラメーターのラベルを設定します。
p.getParameterStorage(parameterName)	ParameterStorage	指定されたパラメーターのストレージを返します。該当するパラメーターが存在しない場合は None を返します。
p.setParameterStorage(parameterName, storage)	なし	指定されたパラメーターのストレージを設定します。
p.getParameterType(parameterName)	ParameterType	指定されたパラメーターのデータ型を返します。該当するパラメーターが存在しない場合は None を返します。
p.setParameterType(parameterName, type)	なし	指定されたパラメーターのデータ型を設定します。
p.getParameterValue(parameterName)	オブジェクト	指定されたパラメーターの値を返します。該当するパラメーターが存在しない場合は None を返します。
p.setParameterValue(parameterName, value)	なし	指定されたパラメーターの値を設定します。

以下の例では、スクリプトで通信データを集計して、平均収入データが最も低い領域を探します。次に、その領域でストリーム・パラメーターを設定します。さらに、そのストリーム・パラメーターを条件抽出ノードで使用してその領域をデータから除外した後、残りのデータに対する顧客離れモデルを作成します。

この例では、スクリプトで条件抽出ノード自体を生成するため、正しい値を条件抽出ノードの式に直接生成できたという点で、不自然な例になっています。しかし、通常ストリームは事前に作成されているため、この方法でパラメーターを設定すると便利です。

スクリプト例の最初の部分では、平均収入が最も低い領域を格納するストリーム・パラメーターを作成します。また、スクリプトでは集計ブランチとモデル作成ブランチにノードを作成し、相互に接続します。

```
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)
```

このスクリプト例では以下のストリームを作成します。

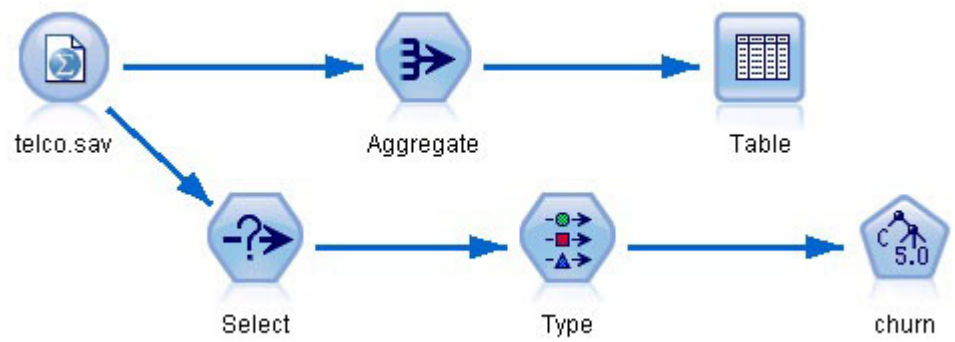

図 5. スクリプト例から得られるストリーム

スクリプト例の以下の部分では、集計ブランチの終端でテーブル・ノードを実行します。

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

スクリプト例の以下の部分では、テーブル・ノードの実行によって生成されたテーブル出力にアクセスします。スクリプトでは次に、テーブルの行全体について反復し、平均収入が最も低い領域を探します。

```
# 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
```

スクリプトの以下の部分では、平均収入が最も低い領域を使用して、以前に作成した「LowestRegion」ストリーム・パラメーターを設定します。スクリプトでは次に、指定の領域を学習データから除外してモデル・ビルダーを実行します。

```
# 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([])
```

スクリプト例全体を以下に示します。

```
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([])
```

グローバル値

グローバル値は、指定したフィールドの各種の要約統計量を計算するために使用します。これらの要約値には、ストリーム内の任意の場所からアクセスできます。グローバル値は、ストリームから名前でアクセスできるという点でストリーム・パラメーターと似ています。ストリーム・パラメーターとの相違点は、スクリプトやコマンド・ラインから代入するのではなく、グローバル値の設定ノードが実行されると関連付けられた値が自動的に更新されることです。ストリームのグローバル値にアクセスするには、ストリームの `getGlobalValues()` メソッドを呼び出します。

GlobalValues オブジェクトは、以下の表に示す関数を定義しています。

表 22. GlobalValues オブジェクトによって定義されている関数

メソッド	戻り値の型	説明
`g.fieldNameIterator()`	反復	グローバル値を 1 つ以上持つ各フィールド名の反復子を返します。
`g.getValue(type, fieldName)`	オブジェクト	指定されたデータ型およびフィールド名のグローバル値を返します。値が見つからない場合は `None` を返します。返される値は一般に数値ですが、将来の実装では別の型の値を返すようになる可能性があります。
`g.getValues(fieldName)`	マップ	指定されたフィールド名の既知のエントリーを含むマップを返します。フィールドに既存のエントリーがない場合は `None` を返します。

`GlobalValues.Type` は、使用可能な要約統計量のタイプを定義します。以下の要約統計量が使用可能です。

- `MAX`: フィールドの最大値。

- MEAN: フィールドの平均値。
- MIN: フィールドの最小値。
- STDDEV: フィールドの標準偏差。
- SUM: フィールドの値の合計。

例えば、以下のスクリプトは「income」フィールドの平均値にアクセスします。このフィールドは、グローバル値の設定ノードによって計算されます。

```
import modeler.api

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

複数のストリームの処理: スタンドアロン・スクリプト

複数のストリームを処理するには、スタンドアロン・スクリプトを使用する必要があります。スタンドアロン・スクリプトは、IBM SPSS Modeler UI 内で編集して実行するか、バッチ・モードでコマンド・ライン・パラメーターとして渡すことができます。

以下のスタンドアロン・スクリプトは 2 つのストリームを開きます。一方のストリームはモデルを作成し、2 番目のストリームは予測値の分布をプロットします。

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/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 histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

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

第 5 章 スクリプトのヒント

このセクションでは、ストリームの実行の修正や、暗号化されたパスワードをスクリプトで使用するなど、スクリプトを使用する場合のヒントと手法について説明します。

ストリーム実行の変更

ストリームを実行すると、ターミナル・ノードがデフォルトの状況に最適化された順番で実行されます。状況に応じて、別の順序で実行させることもできます。ストリームの実行順序を変更するには、「ストリームのプロパティー」ダイアログ・ボックスの「実行」タブで、以下の手順を実行します。

1. 空のスクリプトを用意します。
2. ツールバーの **「デフォルト スクリプトの追加」** ボタンをクリックして、デフォルトのストリーム・スクリプトを追加します。
3. デフォルト ストリーム・スクリプトの文の順序を、実際に実行する順序に変更します。

モデルの処理

モデルの自動置換が IBM SPSS Modeler で使用可能になっており、モデル・ビルダー・ノードが IBM SPSS Modeler ユーザー・インターフェースを使用して実行された場合、そのモデル・ビルダー・ノードにリンクされている既存のモデル・ナゲットが、新しいモデル・ナゲットで置換されます。モデル・ビルダー・ノードがスクリプトを使用して実行された場合、リンクされた既存のモデル・ナゲットは置換されません。既存のモデル・ナゲットを置換するには、スクリプト内でナゲットの置換を明示的に指定する必要があります。

暗号化パスワードの生成

場合によっては、スクリプトにパスワードを記述する必要があるかも知れません。例えば、パスワードで保護されたデータ・ソースにアクセスしたい場合などです。暗号化パスワードは、次の場所で使用することができます。

- データベース入力ノードおよび出力ノードのノード・プロパティー。
- サーバーにログインするためのコマンド・ライン引数。
- エクスポート・ノードの「公開」タブから生成するパラメーター・ファイル *.par* ファイルに保管されるデータベース接続プロパティー。

ユーザー・インターフェースから、Blowfish アルゴリズムに基づいた暗号化パスワードを生成することができます (詳細については、*http://www.schneier.com/blowfish.html* を参照してください)。パスワードを暗号化したら、そのパスワードをコピーしてスクリプト・ファイルやコマンド・ライン引数に指定することができます。`database` および `ddatabaseexport` に使用するノード・プロパティー `epassword` は暗号化パスワードを格納します。

1. 暗号化パスワードを生成するには、「ツール」メニューから次の項目を選択します。

   「パスワードのエンコード...」
2. 「パスワード」ボックスにパスワードを指定します。
3. **「暗号化」** をクリックすると、ランダムに暗号化されたパスワードが生成されます。

4. 「コピー」ボタンをクリックすると、暗号化されたパスワードがクリップボードにコピーされます。
5. パスワードを目的のスクリプトやパラメーターに貼り付けます。

スクリプトの検査

「スタンドアロン・スクリプト」ダイアログ・ボックスのツールバーにある赤い検査ボタンをクリックすれば、すべてのスクリプトのシンタックスを検査することができます。

図 6. ストリーム・スクリプトのツールバー・アイコン

スクリプトの検査時にコードにエラーがあった場合、エラーを警告するメッセージと推奨する修正方法が表示されます。エラーのある行を表示するには、ダイアログ・ボックスの下部にあるフィードバック情報をクリックしてください。エラーが赤で強調表示されます。

コマンド・ラインからのスクリプト

通常はユーザー・インターフェースから行われるような操作を、スクリプトで実行することができます。IBM SPSS Modeler を起動する時には、コマンド・ライン上でスタンドアロン・ストリームを指定して実行してください。

以下に例を示します。

```
client -script scores.py -execute
```

`-script` フラグは指定されたスクリプトをロードすることを、`-execute` フラグはスクリプト・ファイル中のすべてのコマンドを実行することを示しています。

ファイル・パスの指定

ディレクトリーおよびファイルへのファイル・パスを指定する際、ディレクトリーのセパレーターとして、単一のスラッシュ (/) または二重の円記号 (¥¥) を使用できます。例えば、以下のとおりです。

```
c:/demos/druglearn.str
```

or

```
c:¥¥demos¥¥druglearn.str
```

旧リリースとの互換性

IBM SPSS Modeler の以前のリリースで作成された従来のスクリプトは、一般に変更しなくても現在のリリースで動作します。スクリプトが動作するためには、「ストリームのプロパティー」ダイアログ・ボックスまたは「スタンドアロン・スクリプト」ダイアログ・ボックスの「ストリーム・スクリプト」タブで「**レガシー**」を選択する必要があります。モデル・ナゲットがストリームに自動的に挿入され (デフォルト設定)、ストリーム内のその種類の既存ナゲットを置き換えまたは補足する場合があります。これが実際に行われるかどうかは、「**モデルをストリームに追加**」オプションおよび「**前のモデルを置換**」オプション (「**ツール**」>「**オプション**」>「**ユーザー・オプション**」>「**通知**」) の設定によって異なります。例えば、既存のナゲットを削除して新しいナゲットを挿入し、ナゲットの置換を処理する旧リリースからのスクリプトの変更が必要な場合があります。

現在のリリースで作成したスクリプトは、以前のリリースでは動作しないことがあります。

現在のリリースで作成した Python スクリプトは、以前のリリースでは動作しません。

古いリリースで作成されたスクリプトがあるコマンドを使用し、そのコマンドがリリースされてから他のコマンドに置き換えられて (または、廃止されて) いる場合は、古い形が依然としてサポートされますが、同時に警告メッセージも表示されます。例えば、古い `generated` キーワードは `model` に、`clear generated` は `clear generated palette` に置き換えられます。古い形を使うスクリプトは依然として動作しますが、警告も表示されます。

第 6 章 コマンド・ライン引数

ソフトウェアの起動

オペレーティング・システムのコマンド・ラインを使用し、次のようにして IBM SPSS Modeler を起動できます。

1. IBM SPSS Modeler がインストールされているコンピューターで、DOS つまりコマンド・プロンプト・ウィンドウを開きます。
2. IBM SPSS Modeler インターフェースをインタラクティブ・モードで起動するには、`modelerclient` コマンドを入力し、続いて例えば次のような適切な引数を入力します。

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

使用可能な引数 (フラグ) により、サーバーへの接続、ストリームのロード、スクリプトの実行、または必要に応じて他のパラメーターの指定を行うことができます。

コマンド・ライン引数の使用

IBM SPSS Modeler の起動を変更するために、コマンド・ラインの引数 (**フラグ型**とも呼ばれます) を初期の `modelerclient` コマンドに追加できます。

何種類かのコマンド・ライン引数を使用することができます。これらの引数については、このセクションで後述します。

表 23. コマンド・ライン引数の種類 :

引数の種類	参照トピック
システムの引数	詳しくは、トピック 54 ページの『システムの引数』を参照してください。
パラメーターの引数	詳しくは、トピック 56 ページの『パラメーターの引数』を参照してください。
サーバー接続の引数	詳しくは、トピック 56 ページの『サーバー接続の引数』を参照してください。
IBM SPSS Collaboration and Deployment Services Repository の接続の引数	詳しくは、トピック 57 ページの『IBM SPSS Collaboration and Deployment Services Repository の接続の引数』を参照してください。

例えば、以下のように `-server`、`-stream`、`-execute` の各フラグを使用してサーバーに接続し、ストリームをロードして実行することができます。

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

ローカル・クライアントのインストールと競合する場合、サーバー接続の引数は不要です。

スペースを含むパラメーター値は二重引用符で囲むことができます。例えば、次のようになります。

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

この方法で、`-script` フラグを使用して IBM SPSS Modeler スクリプトを実行することもできます。

コマンド・ラインをデバッグするには modelerclient コマンドを使用し、適切な引数を使用して IBM SPSS Modeler を起動します。これにより、コマンドが予期したとおりに実行されるかどうかを検証することができます。また、「セッション・パラメーター］ダイアログ・ボックス（「ツール］メニュー、セッション・パラメーターの設定）のコマンド・ラインから渡されるパラメーターの値を確認することもできます。

システムの引数

ユーザー・インターフェースのコマンド・ラインによる起動で利用できるシステム引数を次の表に示します。

表 24. システムの引数

引数	動作説明
@ <commandFile>	@ 文字に続けてファイル名を記述することにより、コマンド・リストを指定することができます。modelerclient コマンドに @ から始まる引数を指定すると、その引数に指定されたコマンド・ファイル中のコマンドが、コマンド・ラインに指定されているのと同じように処理されます。 詳しくは、トピック 58 ページの『複数の引数の組み合わせ』を参照してください。
-directory <dir>	デフォルトの作業ディレクトリーを設定します。ローカル・モードでは、このディレクトリーはデータと出力の両方で使用されます。例: -directory c:/ や -directory c:¥¥
-server_directory <dir>	デフォルトのデータ用サーバー・ディレクトリーを設定します。-directory フラグで指定された作業ディレクトリーは、出力に使用されます。
-execute	起動後に、起動時にロードされたストリーム、ステート、またはスクリプトを実行します。ストリームやステートではなくスクリプトがロードされた場合は、スクリプトだけが実行されます。
-stream <stream>	起動時に、指定したストリームをロードします。複数のストリームを指定できますが、最後に指定したストリームが現在のストリームに設定されます。
-script <script>	起動時に、指定したスタンドアロン・スクリプトをロードします。下で説明しているストリームやステートに加えてこれも指定できますが、起動時には 1 つのスクリプトしかロードできません。スクリプト・ファイルの接尾辞が .py の場合は、このファイルは Python スクリプトと見なされ、それ以外の場合はレガシー・スクリプトと見なされます。
-model <model>	起動時に、指定の生成モデル (*.gm* 形式ファイル) をロードします。
-state <state>	起動時に、指定した保存済みのステートをロードします。
-project <project>	指定したプロジェクトをロードします。起動時には、プロジェクトを 1 つしかロードできません。
-output <output>	起動時に、保存された出力オブジェクト (*.cou* 形式ファイル) をロードします。
-help	コマンド・ライン引数のリストを表示します。このオプションを指定すると、他の引数はすべて無視されて、ヘルプ画面が表示されます。
-P <name>=<value>	スタートアップ・パラメーターの設定に使用されます。ノードのプロパティー (スロット・パラメーター) の設定に使用することもできます。

表 24. システムの引数 (続き)

引数	動作説明
-scriptlang <python | legacy>	これは、スクリプト・ファイルの接尾辞が何であるかにかかわらず、-script オプションに関連付けられているスクリプト言語を指定するために使用できます。 例 client -scriptlang python -script scores.txt -execute これは、ファイル接尾辞が .py でなかったとしても、提供されたスクリプト・ファイルを Python を使用して実行します。

注：ユーザー・インターフェースでデフォルト・ディレクトリーも設定できます。このオプションにアクセスするには、「ファイル」メニューの「**作業ディレクトリーの設定**」または「**サーバー・ディレクトリーの設定**」を選択します。

複数ファイルのロード

ロードされた各オブジェクトに対応する引数を繰り返し指定して、起動時にコマンド・ラインから、複数のストリーム、ステート、および出力をロードすることができます。例えば、*report.str* と *train.str* の 2 種類のストリームをロード、実行するには、コマンド・ラインに次のコマンドを指定します。

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

IBM SPSS Collaboration and Deployment Services Repository からのオブジェクトのロード

ファイルまたは IBM SPSS Collaboration and Deployment Services Repository (ライセンスがある場合) から特定のオブジェクトをロードすることができるため、ファイル名の接頭辞 spsscr: とオプションの file: (ディスク上のオブジェクトの場合) を使用して、IBM SPSS Modeler に対してオブジェクトの検索場所を指示する必要があります。上記の接頭辞は、次のフラグに適用できます。

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

接頭辞を使用して、オブジェクトの場所を指定する URI を作成します (-stream "spsscr:///folder_1/scoring_stream.str" など)。spsscr: の接頭辞がある場合、IBM SPSS Collaboration and Deployment Services Repository への有効な接続を同じコマンドで指定する必要があります。そのため、例えば、フル・コマンドは次のようになります。

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

コマンド・ラインから URI を使用する「必要がある」ことに注意してください。単純な REPOSITORY_PATH はサポートされていません（その場合は、スクリプト内でのみ作動します）。

パラメーターの引数

IBM SPSS Modeler のコマンド・ライン実行時に、パラメーターをフラグとして使用することができます。コマンド・ラインの引数で -P フラグを使用して、-P <*name*>=<*value*> の形式でパラメーターを指定することができます。

パラメーターは、次のいずれかになります。

- **単純パラメーター**
- **スロット・パラメーター**、**ノードのプロパティー**と呼ばれることもあります。これらのパラメーターは、ストリーム中のノードの設定を変更するために使用されます。 詳しくは、トピック 60 ページの『ノードのプロパティーの概要』を参照してください。
- IBM SPSS Modeler の起動を変更するために用いられる、**コマンド・ライン・パラメーター**。

例えば、データ・ソースのユーザー名とパスワードを、次のようにコマンド・ラインのフラグとして指定することができます。

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

形式は、database ノード・プロパティーの datasource パラメーターの形式と同じです。詳しくは、トピック 68 ページの『database ノードのプロパティー』を参照してください。

サーバー接続の引数

-server フラグは、IBM SPSS Modeler がパブリック・サーバーに接続することを指定し、-hostname 、-use_ssl、-port、-username、-password、-domain の各フラグを使用して、IBM SPSS Modeler によるパブリック・サーバーへの接続方法を指定します。-server 引数が指定されていない場合、デフォルト・サーバーまたはローカル・サーバーが使用されます。

例

パブリック・サーバーに接続するには

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

サーバー・クラスターに接続するには

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

サーバー・クラスターに接続するには、IBM SPSS Collaboration and Deployment Services を使用した Coordinator of Processes が必要です。そのため、-cluster 引数をリポジトリー接続オプション (spsscr_*) とともに使用する必要があります。詳しくは、トピック 57 ページの『IBM SPSS Collaboration and Deployment Services Repository の接続の引数』を参照してください。

表 25. サーバー接続の引数 :

引数	動作説明
-server	IBM SPSS Modeler をサーバー・モードで実行し、-hostname、-port、-username、-password、-domain の各フラグを使用してパブリック・サーバーに接続します。
-hostname <名前>	サーバー・マシンのホスト名を指定します。サーバー・モードでしか利用できません。

表 25. サーバー接続の引数 (続き):

引数	動作説明
-use_ssl	接続で使用する SSL (secure socket layer) を指定します。このフラグはオプションです。SSL 使用時のデフォルト設定は *not* です。
-port <番号>	指定したサーバーのポート番号。サーバー・モードでしか利用できません。
-cluster <名前>	指定されたサーバーではなく、サーバー・クラスターへの接続を指定します。これは、hostname 引数、port 引数、use_ssl 引数の代替引数です。name はクラスター名、または IBM SPSS Collaboration and Deployment Services Repository 内のクラスターを識別する一意の URI です。サーバー・クラスターは、IBM SPSS Collaboration and Deployment Services を使用して Coordinator of Processes で管理されます。 詳しくは、トピック『IBM SPSS Collaboration and Deployment Services Repository の接続の引数』を参照してください。
-username <名前>	サーバーにログオンするためのユーザー名。サーバー・モードでしか利用できません。
-password <パスワード>	サーバーにログオンするためのパスワード。サーバー・モードでしか利用できません。*注*：-password 引数を指定しなかった場合、パスワードを入力するためのプロンプトが表示されます。
-epassword <エンコードされたパスワード文字列>	サーバーにログオンするための暗号化パスワード。サーバー・モードでしか利用できません。*注*：暗号化パスワードは、IBM SPSS Modeler アプリケーションの「ツール」メニューから生成することができます。
-domain <名前>	サーバーにログオンする際に使用するドメイン名。サーバー・モードでしか利用できません。
-P <name>=<value>	スタートアップ・パラメーターの設定に使用されます。ノードのプロパティー (スロット・パラメーター) の設定に使用することもできます。

IBM SPSS Collaboration and Deployment Services Repository の接続の引数

注：IBM SPSS Collaboration and Deployment Services リポジトリーを利用するには、別途ライセンスが必要です。詳しくは、http://www.ibm.com/software/analytics/spss/products/deployment/cds/を参照してください。

コマンド・ラインを経由して IBM SPSS Collaboration and Deployment Services でオブジェクトを保存したり取り出したりするには、IBM SPSS Collaboration and Deployment Services Repositoryに有効な接続を指定する必要があります。以下に例を示します。

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

接続を設定するために使用できる引数の一覧を次の表に示します。

表 26. IBM SPSS Collaboration and Deployment Services Repository の接続の引数

引数	動作説明
-spsscr_hostname <ホスト名または IP アドレス>	IBM SPSS Collaboration and Deployment Services Repository がインストールされているサーバーのホスト名または IP アドレスです。
-spsscr_port <番号>	IBM SPSS Collaboration and Deployment Services Repository が接続を承認したポート番号です（通常、8080 がデフォルト値）。
-spsscr_use_ssl	接続で使用する SSL (secure socket layer) を指定します。このフラグはオプションです。SSL 使用時のデフォルト設定は *not* です。

表 26. *IBM SPSS Collaboration and Deployment Services Repository* の接続の引数 (続き)

引数	動作説明
-spsscr_username <名前>	IBM SPSS Collaboration and Deployment Services Repository にログオンするためのユーザー名。
-spsscr_password <パスワード>	IBM SPSS Collaboration and Deployment Services Repository にログオンするためのパスワード。
-spsscr_epassword <エンコードされたパスワード>	IBM SPSS Collaboration and Deployment Services Repository にログオンするためのエンコードされたパスワード。
-spsscr_domain <名前>	IBM SPSS Collaboration and Deployment Services Repository にログオンする際に使用するドメイン名。このフラグはオプションです。LDAP または Active Directory を使用してログオンしない限り、このフラグは使用しないでください。

複数の引数の組み合わせ

複数の引数を記述したコマンド・ファイルを作成し、起動時に @ 記号に続けてそのファイル名を指定することができます。こうすることによって、コマンド・ラインによる起動を短縮し、OS によるコマンド長の制限に関する問題を解決することができます。例えば以下の起動コマンドは、<commandFileName> が参照するファイル内で指定されている引数を使用します。

```
modelerclient @<commandFileName>
```

ファイル名やコマンド・ファイルへのパスにスペースがある場合は、以下のようにして引用符で囲みます。

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

このコマンド・ファイルには、スタートアップ時に個別に指定していたすべての引数を記述することができます。以下に例を示します。

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

コマンド・ファイルを記述して、コマンド・ファイル名を指定する場合の制限事項を次に示します。

- 1 行につき 1 つの引数またはコマンドを記述する必要があります。
- コマンド・ファイル内に、@CommandFile 引数を組み込まないでください。

第 7 章 プロパティーのリファレンス

プロパティー参照の概要

ノード、ストリーム、スーパーノード、プロジェクトに対して、数多くのさまざまなプロパティーを指定できます。名前、注釈、およびツールヒントなど、すべてのノードに共通のプロパティーもありますが、その一方で、ノードのタイプに固有なプロパティーもあります。キャッシングやスーパーノードの動作などの高レベルなストリーム操作を参照するプロパティーもあります。プロパティーは、標準のユーザー・インターフェースからアクセスでき (ノードのオプションを編集するダイアログ・ボックスをオープンする場合など)、また、多くの標準とは異なる方法でも使用できます。

- プロパティーは、このセクションで説明されているように、スクリプトからアクセスできます。詳しくは、30 ページの『プロパティを設定する』を参照してください。
- ノードのプロパティーは、スーパーノード・パラメーター中で使用することができます。
- ノードのプロパティーは、IBM SPSS Modeler の起動時にコマンド・ライン・オプションの一部として使用することもできます (`-P` フラグを使用)。

IBM SPSS Modeler のスクリプトでは、ノードおよびストリームのプロパティーは、よく**スロット・パラメーター**と呼ばれます。このガイドでは、スロット・パラメーターをノードまたはストリームのプロパティーと記載しています。

スクリプト言語の詳細は、15 ページの『第 2 章 スクリプト言語』を参照してください。

省略形

ノードのプロパティーのシンタックスでは、標準省略形が使用されています。省略形を覚えておけば、スクリプトの作成に役立ちます。

表 27. シンタックスで使用される標準省略形 :

省略形	意味
abs	絶対値
len	長さ
最小	最小値
最大	最大値
correl	相関
covar	共分散
num	数字または数値
pct	パーセントまたは割合
transp	透過度
xval	交差検証
var	分散または変数 (入力ノードで)

ノードとストリームのプロパティーの例

ノードおよびストリームのプロパティーは、IBM SPSS Modeler のさまざまな場面で使用されます。一般的にこれらのプロパティーは、複数のストリームや操作を自動化するために用いられる**スタンドアロン・スクリプト**、または単一のストリーム内のプロセスの自動化に用いられる**ストリーム・スクリプト**など、スクリプトの一部として使われます。スーパーノード内で、ノードのプロパティーを使用してノード・パラメーターを指定することもできます。もっとも基本的なレベルで、IBM SPSS Modeler の起動時にコマンド・ライン・オプションとしてプロパティーを指定することもできます。コマンド・ラインの起動時に、-p 引数を指定すれば、ストリーム・プロパティーを使用してストリームの設定を変更することができます。

スクリプトの例について詳しくは、42 ページの『ストリーム、セッション、およびスーパーノード・パラメーター』と 54 ページの『システムの引数』のトピックを参照してください。

ノードのプロパティーの概要

ノードの種類ごとに、独自の有効なプロパティーのセットが用意されています。また、各プロパティーにはデータ型があります。一般的なデータ型の数値、フラグ、または文字列の場合、プロパティーの設定は強制的に正しいデータ型に設定されます。強制的に設定できない場合はエラーが発生します。それに対し、プロパティー参照が、Discard、PairAndDiscard、および IncludeAsText のような有効な値の範囲を指定していることもあります。この場合、範囲外の値が使われた場合にエラーになります。フラグ型プロパティーは、True および False の値を使用して読み込まれるか、設定される必要があります このガイドにある参照テーブルでは、構造化プロパティーはそのまま「*プロパティーの説明*」欄に、使用形式とともに記載されています。

共通のノード・プロパティー

数多くのプロパティーが、IBM SPSS Modeler 中のすべてのノード (スーパーノードも含む) で共通に使われています。

表 28. 共通のノード・プロパティー:

プロパティー名	データ型	プロパティーの説明
use_custom_name	*boolean*	
name	*文字列*	ストリーム領域上のノード名を対象とする読み込み専用プロパティーです (自動またはユーザー設定)。
custom_name	*文字列*	ノードのカスタム(ユーザー設定)名を指定します。
tooltip	*文字列*	
annotation	*文字列*	
keywords	*文字列*	オブジェクトに関連付けられているキーワードのリストを指定する構造化スロットです。
cache_enabled	*boolean*	
node_type	source_supernode process_supernode terminal_supernode スクリプト用に指定するすべてのノード名	ノードをタイプごとに参照するために使用される読み込み専用プロパティー。例えば、ノードを real_income のような名前だけで参照する代わりに、userinput や filter のようなタイプを指定することもできます。

スーパーノード固有のプロパティーは、他のノードと同様に、個別に説明します。 詳しくは、トピック 251 ページの『第 19 章 スーパーノードのプロパティー』を参照してください。

第 8 章 Stream プロパティー

スクリプトにより、さまざまなストリームのプロパティーを制御することができます。

スクリプトから現在のストリームにアクセスするには、modeler.script モジュールの stream() 関数を使用します。以下に例を示します。

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

ストリームのプロパティーを参照するには、特殊なストリーム変数を使用する必要があります。この変数は、ストリームの先頭に ^ を付けて表されます。

nodes プロパティーは、現在のストリーム中のノードを参照するために使用されます。

ストリームのプロパティーを次の表に示します。

表 29. Stream プロパティー:

プロパティー名	データ型	プロパティーの説明
execute_method	Normal Script	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	*number*	
date_2digit_baseline	*number*	

表 29. *Stream* プロパティー *(続き)*:

プロパティー名	データ型	プロパティーの説明
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	*boolean*	
import_datetime_as_string	*boolean*	
decimal_places	*number*	
decimal_symbol	Default Period Comma	
angles_in_radians	*boolean*	
use_max_set_size	*boolean*	
max_set_size	*number*	
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	*boolean*	ストリーム実行時に、入力ノードを自動的にリフレッシュするために使用します。
script	*文字列*	
script_language	Python レガシー	ストリーム・スクリプトのスクリプト言語を設定します。
annotation	*文字列*	
エンコード	SystemDefault "UTF-8"	
stream_rewriting	*boolean*	
stream_rewriting_maximise_sql	*boolean*	
stream_rewriting_optimise_clem_実行	*boolean*	
stream_rewriting_optimise_syntax_実行	*boolean*	
enable_parallelism	*boolean*	
sql_generation	*boolean*	
database_caching	*boolean*	
sql_logging	*boolean*	
sql_generation_logging	*boolean*	
sql_log_native	*boolean*	

表 29. *Stream* プロパティー *(続き)*:

プロパティー名	データ型	プロパティーの説明
sql_log_prettyprint	*boolean*	
record_count_suppress_input	*boolean*	
record_count_feedback_interval	*integer*	
use_stream_auto_create_node_設定	*boolean*	true の場合はストリーム固有の設定が使用されます。それ以外の場合はユーザー設定が使用されます。
create_model_applier_for_new_モデル	*boolean*	true の場合、モデル・ビルダーが新しいモデルを作成するときにアクティブな更新リンクがなければ、新しいモデル・アプライヤーが追加されます。
create_model_applier_update_links	createEnabled createDisabled doNotCreate	モデル・アプライヤー・ノードの自動追加時に作成するリンクの種類を定義します。
create_source_node_from_builders	*boolean*	true の場合、ソース・ビルダーが新しいソース出力を作成するときにアクティブな更新リンクがなければ、新しい入力ノードが追加されます。
create_source_node_update_links	createEnabled createDisabled doNotCreate	入力ノードの自動追加時に作成するリンクの種類を定義します。

第 9 章 入力ノードのプロパティー

入力ノードの共通プロパティー

すべての入力ノードに共通するプロパティーを次に一覧にします。その後に、特定のノードに関する情報が続きます。

表 30. ソース・ノードの共通プロパティー:

プロパティー名	データ型	プロパティーの説明
direction	Input Target Both None Partition Split 頻度 RecordID	フィールドの役割のキープロパティー。 注: 値 In および Out は 廃止されています。 今後のリリースでは サポートが中断される 場合があります。
type	Range (範囲) Flag Set 型なし Discrete 順序セット Default	フィールドのデータ型。このプロパティーを *Default* に設定すると、values プロパティーに関するすべての値は消去され、value_mode を *Specify* に設定すると、それが *Read* にリセットされます。value_mode が *Pass* または *Read* がすでに設定されている場合、type の設定によって影響を受けることはありません。
storage	Unknown String Integer 実数 Time Date Timestamp	フィールドのストレージ・タイプ用読み込み専用キー・プロパティー。
check	None 無効化 強制 Discard 警告 Abort	フィールド・タイプと範囲の検査用のキー・プロパティー。
values	[*value value*]	連続型 (範囲) フィールドの場合、最初の値が最小値で最後の値が最大値になります。名義型 (セット型) フィールドの場合、すべての値を指定します。フラグ型の場合、最初の値が *false* (偽) を、最後の値が *true* (真) を表します。このプロパティーを設定すると、value_mode プロパティーの値が自動的に *Specify* に設定されます。

表 30. ソース・ノードの共通プロパティー (続き):

プロパティー名	データ型	プロパティーの説明
value_mode	Read Pass Read+ Current Specify	次のデータの受け渡し時にフィールドに値を設定する方法を決定します。このプロパティーに *Specify* を直接には設定できないことに注意してください。特定の値を使用するには、values プロパティーを設定します。
default_value_mode	Read Pass	すべてのフィールドに値を設定するためのデフォルトの方法を指定します。この設定による特定のフィールドの設定は、value_mode プロパティーを使用するとオーバーライドされることがあります。
extend_values	*boolean*	value_mode が *Read* に設定された場合に適用されます。新しく読み込んだ値を、フィールドの既存の値に追加する場合は、*T* を設定します。新しく読み込んだ値を優先して、既存の値を破棄する場合は、*F* を設定します。
value_labels	*文字列*	値ラベルの指定に使用します。数値を先に指定します。
enable_missing	*boolean*	*T* を設定した場合、フィールドの欠損値の追跡が有効になります。
missing_values	[*value value ...*]	欠損データを示すデータ値を指定します。
range_missing	*boolean*	プロパティーが *T* に設定されている場合、フィールドに欠損値 (空白) の範囲が定義されているかどうかを指定します。
missing_lower	*文字列*	range_missing が真 (true) の場合、欠損値範囲の下限値を指定します。
missing_upper	*文字列*	range_missing が真 (true) の場合、欠損値範囲の上限値を指定します。
null_missing	*boolean*	このプロパティーが *T* に設定されていると、ヌル (ソフトウェアでは $null$ として表示される未定義値) は欠損値と見なされます。
whitespace_missing	*boolean*	このプロパティーが *T* に設定されていると、空白値 (スペース、タブ、および改行) だけを含む値は欠損値とみなされます。
description	*文字列*	フィールドのラベルまたは説明の指定に使用します。
default_include	*boolean*	デフォルトの処理としてフィールドを通過させるかフィルターをかけるかの指定をするキー・プロパティー。
include	*boolean*	各フィールドを適用するかフィルターをかけるかを決定するキー・プロパティー:
new_name	*文字列*	

asimport ノードのプロパティー

Analytic Server 入力により、Hadoop 分散ファイル・システム (HDFS) でストリームを実行することができます。

表 31. asimport ノードのプロパティー:

asimport ノードのプロパティー	データ型	プロパティーの説明
data_source	*文字列*	データ・ソースの名前。
host	*文字列*	Analytic Server ホストの名前。
ポート	*整数*	Analytic Server が listen するポート。
tenant	*文字列*	マルチテナント環境における所属先テナントの名前。シングル・テナント環境ではデフォルトで **ibm** になります。
set_credentials	*boolean*	Analytic Server でのユーザー認証が SPSS Modeler Server と同じである場合は、これを **false** に設定してください。それ以外の場合は **true** に設定してください。
user_name	*文字列*	Analytic Server にログインするためのユーザー名。set_credentials が true の場合にのみ必要です。
パスワード	*文字列*	Analytic Server にログインするためのパスワード。set_credentials が true の場合にのみ必要です。

cognosimport ノードのプロパティー

IBM Cognos BI 入力ノードは、Cognos BI データベースからデータをインポートします。

表 32. cognosimport ノードのプロパティー:

cognosimport ノードのプロパティー	データ型	プロパティーの説明
mode	Data Report	Cognos BI データ (デフォルト) またはレポートをインポートするかどうかを指定します。

表 32. *cognosimport* ノードのプロパティー *(続き)*:

cognosimport ノードのプロパティー	データ型	プロパティーの説明
cognos_connection	*{"string",boolean, "string","string", "string"}*	Cognos サーバーの接続の詳細を含むリストのプロパティー。形式は次のとおりです。{"*Cognos_server_URL*", *login_mode*, "*namespace*", "*username*", "*password*"} where: *Cognos_server_URL* は、ソースを含むする Cognos サーバーの URL です。 *login_mode* は匿名ログインが使用されるかどうかを示し、true または false となります。true の場合、次のフィールドは "" に設定されます。*namespace* はサーバーへのログインに使用するセキュリティー認証プロバイダを指定します。 *username* および *password* は Cognos サーバーにログインする際に使用するユーザー名とパスワードです。
cognos_package_name	*文字列*	データ・オブジェクトをインポートしている Cognos データ・ソース (通常はデータベース) のパスおよび名前。次に例を示します。 /Public Folders/GOSALES 注 :フォワード スラッシュのみが有効です。
cognos_items	*{"field","field", ... ,"field"}*	インポートする 1 つまたは複数のデータ・オブジェクトの名前。*field* の形式は、[*namespace*].[*query_subject*].[*query_item*] です。
cognos_filters	*field*	データをインポートする前に適用するフィルターの名前。
cognos_data_parameters	*list*	データのプロンプト・パラメーターの値。名前と値のペアは中括弧で囲み、複数のペアはコンマで区切り、文字列全体は角括弧で囲みます。書式 : [{"*param1*", "*value*"},...,{"*paramN*", "*value*"}]
cognos_report_directory	*field*	レポートをインポートするフォルダーまたはパッケージの Cognos パス。次に例を示します。/Public Folders/GOSALES 注 :フォワード スラッシュのみが有効です。
cognos_report_name	*field*	インポートするレポートのレポートの位置内にあるパスと名前。
cognos_report_parameters	*list*	レポート・パラメーターの値。名前と値のペアはかっこで囲み、複数のペアはコンマで区切り、文字列全体は角かっこで囲みます。書式 : [{"*param1*", "*value*"},...,{"*paramN*", "*value*"}]

database ノードのプロパティー

データベース・ノードは、Microsoft SQL Server、DB2、Oracle など ODBC (開放型データベース接続) を使用するさまざまなパッケージからデータをインポートするのに使用できます。

表 33. database ノードのプロパティー:

database ノードのプロパティー	**データ型**	**プロパティーの説明**
mode	表 Query	ダイアログ・ボックスのコントロールを使用してデータベースに接続するには、*Table* を指定します。SQL を使用して選択されたデータベースにクエリーを行うには、*Query* を指定します。
datasource	*文字列*	データベース名 (下記の注意を参照)。
username	*文字列*	データベース接続の詳細 (下記の注意を参照)。
password	*文字列*	
epassword	*文字列*	スクリプト内でパスワードをハードコード化する代わりに、エンコードされたパスワードを指定します。 詳しくは、トピック 49 ページの『暗号化パスワードの生成』を参照してください。このプロパティーは、実行中は読み取り専用になります。
tablename	*文字列*	アクセスするテーブルの名前。
strip_spaces	None Left Right Both	文字列の前後のスペースを破棄するためのオプションです。
use_quotes	AsNeeded Always Never	クエリーをデータベースに送信するときにテーブル名と列名を引用符で囲むかどうかを指定します (例えば、テーブル名と列名にスペースや句読点が含まれているような場合)。
query	*文字列*	送信するクエリーを表す SQL コードを指定します。

注 : データベース名 (datasource プロパティー) 二スペースがある場合、datasource、username および password の代わりに、次の形式で単一のデータソース・プロパティーを使用します。

表 34. database ノードのプロパティー - データ・ソース固有:

database ノードのプロパティー	**データ型**	**プロパティーの説明**
datasource	*文字列*	形式: [database_name,username,password[,true | false]] 暗号化パスワードと使用しないパラメーターです。true に設定すると、パスワードが使用前に復号化されます。

データ・ソースを変更する場合、この形式を使用します。ただし、ユーザー名またはパスワードを変更する場合、username プロパティーまたは password プロパティーを使用できます。

datacollectionimport ノードのプロパティー

IBM SPSS Data Collection データ・インポート・ノードで、IBM Corp. 市場調査製品によって使用される IBM SPSS Data Collection Data Model に基づいた調査データをインポートします。このノードを使用するには、IBM SPSS Data Collection Data Library がインストールされている必要があります。

図 7. Dimensions データ・インポート・ノード

表 35. datacollectionimport ノードのプロパティー:

datacollectionimport ノードのプロパティー	**データ型**	**プロパティーの説明**
metadata_name	*文字列*	MDSC の名前。特殊な値の DimensionsMDD は、標準的な IBM SPSS Data Collection メタデータ・ドキュメントが使用される必要のあることを示します。ほかに、次の値を指定できます。 mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC 特殊な値の none は、MDSC がないことを示します。
metadata_file	*文字列*	メタデータが格納されるファイルの名前。
casedata_name	*文字列*	CDSC の名前。使用できる値は以下のとおりです。 mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc 特殊な値の none は、CDSC がないことを示します。
casedata_source_type	Unknown File Folder UDL DSN	CDSC のソース・タイプを示します。

表 35. *datacollectionimport* ノードのプロパティー *(続き)*:

datacollectionimport ノードのプロパティー	データ型	プロパティーの説明
casedata_file	*文字列*	casedata_source_type が *File* のときに、ケース・データが含まれるファイルを指定します。
casedata_folder	*文字列*	casedata_source_type が *Folder* のときに、ケース・データが含まれるフォルダーを指定します。
casedata_udl_string	*文字列*	casedata_source_type が *UDL* のときに、ケース・データが含まれるデータ・ソースのための OLD-DB 接続文字列を指定します。
casedata_dsn_string	*文字列*	casedata_source_type が *DSN* のときに、データ・ソースのための ODBC 接続文字列を指定します。
casedata_project	*文字列*	IBM SPSS Data Collection データベースからケース・データを読み込むときに、プロジェクトの名前を入力できます。その他のケース・データのデータ型については、この設定を空白のままにしておく必要があります。
version_import_mode	All 最新 Specify	各バージョンの取り扱い方法を定義します。
specific_version	*文字列*	version_import_mode が *Specify* のときに、インポートされるケース・データのバージョンを定義します。
use_language	*文字列*	特定言語のラベルが使用される必要があるかどうかを定義します。
language	*文字列*	use_language が真 (true) の場合、入力に使用する言語コードを定義します。言語コードは、ケース・データ内で利用できる中の 1 つにする必要があります。
use_context	*文字列*	特定のコンテキストが入力される必要があるかどうかを定義します。コンテキストは、応答に関連する説明を多様化させるために使用されます。
context	*文字列*	use_context が真 (true) の場合、入力するコンテキストを定義します。コンテキストは、ケース・データ内で利用できる中の 1 つにする必要があります。
use_label_type	*文字列*	特定のラベル タイプが入力される必要があるかどうかを定義します。
label_type	*文字列*	use_label_type が真 (true) の場合、入力するラベル・タイプを定義します。ラベル・タイプは、ケース・データ内で利用できる中の 1 つにする必要があります。
user_id	*文字列*	明示的なログインが必要なデータベースの場合、データ・ソースにアクセスするためのユーザー ID とパスワードを提供できます。

表 35. *datacollectionimport* ノードのプロパティー *(続き)*:

datacollectionimport ノードのプロパティー	データ型	プロパティーの説明
password	*文字列*	
import_system_variables	Common None All	インポートされるシステム変数を指定します。
import_codes_variables	*boolean*	
import_sourcefile_variables	*boolean*	
import_multi_response	MultipleFlags Single	

excelimport ノードのプロパティー

Excel インポート・ノードで、Microsoft Excel の各バージョンからデータをインポートします。ODBC データ・ソースは不要です。

表 36. *excelimport* ノードのプロパティー:

excelimport ノードのプロパティー	データ型	プロパティーの説明
excel_file_type	Excel2003 Excel2007	
full_filename	*文字列*	パスを含む、完全なファイル名。
use_named_range	*Boolean*	名前付けられた範囲を使用するかどうかを指定します。真の場合、読み込む範囲を指定するのに named_range プロパティーが使用され、その他のワークシートとデータ範囲の設定は無視されます。
named_range	*文字列*	
worksheet_mode	Index Name	ワークシートがインデックスで定義されているのか (Index)、または名前で定義されているのか (Name) を指定します。
worksheet_index	*integer*	読み込むべきワークシートのインデックス。最初のワークシートは 0、2 番目は 1、というようにインデックスが指します。
worksheet_name	*文字列*	読み込むべきワークシートの名前。
data_range_mode	FirstNonBlank ExplicitRange	範囲の決定方法を指定します。
blank_rows	StopReading ReturnBlankRows	data_range_mode が *FirstNonBlank* のときに、空白行の処理方法を指定します。
explicit_range_start	*文字列*	data_range_mode が *ExplicitRange* のときに、読み込む範囲の開始点を指定します。
explicit_range_end	*文字列*	

表 36. *excelimport* ノードのプロパティー *(続き)*:

excelimport ノードのプロパティー	データ型	プロパティーの説明
read_field_names	*Boolean*	指定された範囲の最初の行がフィールド (列) 名として使用されるかどうかを指定します。

evimport ノードのプロパティー

Enterprise View ノードは、IBM SPSS Collaboration and Deployment Services Repository への接続を作成し、Enterprise View のデータをストリームに読み込み、他のユーザーがリポジトリからアクセスできるシナリオにモデルをパッケージ化できます。

表 37. *evimport* ノードのプロパティー:

evimport ノードのプロパティー	データ型	プロパティーの説明
connection	*list*	構造化プロパティー - エンタープライズ・ビューの接続を作成するパラメーターのリスト。
tablename	*文字列*	アプリケーション・ビューのテーブル名。

fixedfile ノードのプロパティー

固定長ノードで、固定長フィールド・テキスト・ファイルからデータをインポートします。ここで、ファイルのフィールドは区切られていませんが、同じ位置から始まって長さは固定されています。コンピューターによって生成されたデータや旧来のシステムのデータなどは、固定長フィールドの形式で保存されていることがよくあります。

表 38. *fixedfile* ノードのプロパティー:

fixedfile ノードのプロパティー	データ型	プロパティーの説明
record_len	*number*	各レコードの文字数を指定します。
line_oriented	*boolean*	各レコードの末尾の改行文字をスキップします。
decimal_symbol	Default Comma Period	データ・ソースで使われている小数点記号。
skip_header	*number*	最初のレコードの先頭で無視する行数を指定します。列見出しを無視する場合などに役立ちます。
auto_recognize_datetime	*boolean*	入力データの日付または時刻を自動的に特定するかどうかを指定します。
lines_to_scan	*number*	
fields	*list*	構造化プロパティー。
full_filename	*文字列*	読み込みファイルのディレクトリーを含む完全な名前。

表 38. fixedfile ノードのプロパティー (続き):

fixedfile ノードのプロパティー	**データ型**	**プロパティーの説明**
strip_spaces	None Left Right Both	インポート時に文字列の前後のスペースを破棄します。
invalid_char_mode	Discard Replace	データ入力から不正な文字 (ヌル、0、または現在のエンコード中に存在していない文字) をデータ入力から削除するか (Discard)、指定された 1 文字の記号で不正な文字を置き換えます (Replace)。
invalid_char_replacement	*文字列*	
use_custom_values	*boolean*	
custom_storage	Unknown String Integer 実数 Time Date Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	このプロパティーは、カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。

表 38. *fixedfile* ノードのプロパティー *(続き)*:

fixedfile ノードのプロパティー	データ型	プロパティーの説明
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"	このプロパティーは、カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。
custom_decimal_symbol	*field*	カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。
エンコード	StreamDefault SystemDefault "UTF-8"	テキストのエンコード方法を指定します。

sasimport ノードのプロパティー

SAS インポート・ノードで、SAS データを IBM SPSS Modeler へインポートします。

表 39. *sasimport* ノードのプロパティー:

sasimport ノードのプロパティー	データ型	プロパティーの説明
format	Windows UNIX Transport SAS7 SAS8 SAS9	インポートするファイルの形式。
full_filename	*文字列*	パスも含めた、完全なファイル名。この名前を入力します。
member_name	*文字列*	指定した SAS トランスポート・ファイルからインポートするメンバーを指定します。
read_formats	*boolean*	指定された形式ファイルから、データ形式 (変数ラベルなど) を読み込みます。
full_format_filename	*文字列*	
import_names	NamesAndLabels LabelsasNames	インポート時に変数名と変数ラベルをマッピングする方法を指定します。

simgen ノードのプロパティー

シミュレーション生成ノードにより、シミュレーション対象のデータを容易に生成することができます。このとき、ユーザー指定の統計分布を使用して最初から生成するか、既存の履歴データに対してシミュレーション適合ノードを実行して得られた分布を使用して自動的に生成することができます。これは、モデルの入力に不確定性がある状況で予測モデルの結果を評価するときに便利です。

表 40. simgen ノードのプロパティー:

simgen ノードのプロパティー	データ型	プロパティーの説明
fields	構造化プロパティー	
相関	構造化プロパティー	
max_cases	*整数*	最小値は 1000、最大値は 2,147,483,647 です。
create_iteration_field	*boolean*	
iteration_field_name	*文字列*	
replicate_results	*boolean*	
random_seed	*整数*	
overwrite_when_refitting	*boolean*	
parameter_xml	*文字列*	パラメーター XML を文字列として返します。
分布	Bernoulli ベータ 2 項 カテゴリ 指数(X) 固定 ガンマ 対数正規 負の 2 項失敗回数 負の 2 項試行回数 正規 ポアソン 範囲 三角 一様 ワイブル	
bernoulli_prob	*数値*	0 ≤ bernoulli_prob ≤ 1
beta_shape1	*数値*	0 以上でなければなりません。
beta_shape2	*数値*	0 以上でなければなりません。
beta_min	*数値*	オプション。beta_max より小さくなければなりません。
beta_max	*数値*	オプション。beta_min より大きくなければなりません。
binomial_n	*整数*	0 より大きくなければなりません。
binomial_prob	*数値*	0 ≤ binomial_prob ≤ 1 の範囲でなければなりません。
binomial_min	*数値*	オプション。binomial_max より小さくなければなりません。

表 40. simgen ノードのプロパティー (続き):

simgen ノードのプロパティー	データ型	プロパティーの説明
binomial_max	数値	オプション。binomial_min より大きくなければなりません。
exponential_scale	数値	0 より大きくなければなりません。
exponential_min	数値	オプション。exponential_max より小さくなければなりません。
exponential_max	数値	オプション。exponential_min より大きくなければなりません。
fixed_value	文字列	
gamma_shape	数値	0 以上でなければなりません。
gamma_scale	数値	0 以上でなければなりません。
gamma_min	数値	オプション。gamma_max より小さくなければなりません。
gamma_max	数値	オプション。gamma_min より大きくなければなりません。
lognormal_shape1	数値	0 以上でなければなりません。
lognormal_shape2	数値	0 以上でなければなりません。
lognormal_min	数値	オプション。lognormal_max より小さくなければなりません。
lognormal_max	数値	オプション。lognormal_min より大きくなければなりません。
negative_bin_failures_threshold	数値	0 以上でなければなりません。
negative_bin_failures_prob	数値	0 ≤ negative_bin_failures_prob ≤ 1
negative_bin_failures_min	数値	オプション。negative_bin_failures_max より小さくなければなりません。
negative_bin_failures_max	数値	オプション。negative_bin_failures_min より大きくなければなりません。
negative_bin_trials_threshold	数値	0 以上でなければなりません。
negative_bin_trials_prob	数値	0 ≤ negative_bin_trials_prob ≤ 1
negative_bin_trials_min	数値	オプション。negative_bin_trials_max より小さくなければなりません。
negative_bin_trials_max	数値	オプション。negative_bin_trials_min より小さくなければなりません。
normal_mean	数値	
normal_sd	数値	0 より大きくなければなりません。
normal_min	数値	オプション。normal_max より小さくなければなりません。
normal_max	数値	オプション。normal_min より大きくなければなりません。
poisson_mean	数値	0 以上でなければなりません。

表 40. simgen ノードのプロパティー (続き):

simgen ノードのプロパティー	データ型	プロパティーの説明
poisson_min	数値	オプション。poisson_max より小さくなければなりません。
poisson_max	数値	オプション。poisson_min より大きくなければなりません。
triangular_mode	数値	triangular_min ≤ triangular_mode ≤ triangular_max
triangular_min	数値	triangular_mode より小さくなければなりません。
triangular_max	数値	triangular_mode より大きくなければなりません。
uniform_min	数値	uniform_max より小さくなければなりません。
uniform_max	数値	uniform_min より大きくなければなりません。
weibull_rate	数値	0 以上でなければなりません。
weibull_scale	数値	0 以上でなければなりません。
weibull_location	数値	0 以上でなければなりません。
weibull_min	数値	オプション。weibull_max より小さくなければなりません。
weibull_max	数値	オプション。weibull_min より大きくなければなりません。

相関は、+1 から -1 までの任意の数字です。相関は必要な数だけ指定することができます。指定されていない相関は、すべて 0 に設定されます。不明なフィールドが存在する場合、相関値は相関行列 (または表) 上で設定する必要があり、赤いテキストで表示されます。不明なフィールドが存在する場合、ノードを実行することはできません。

statisticsimport ノードのプロパティー

IBM SPSS Statistics ファイル・ノードは、同じ形式を使用する IBM SPSS Statistics で使用される *.sav* ファイル形式のデータおよび IBM SPSS Modeler に保存されたキャッシュ・ファイルを読み込みます。

このノードのプロパティーについては、 247 ページの『statisticsimport ノードのプロパティー』に記載されています。

userinput ノードのプロパティー

ユーザー入力ノードにより、合成データをまったく新規に作成したり、既存のデータを変更して作成したりすることができます。これは、モデル作成用の検定データセットを作成する場合などに役立ちます。

表 41. *userinput* ノードのプロパティー:

userinput ノードのプロパティー	データ型	プロパティーの説明
data		各フィールドのデータの長さは異なる場合がありますが、フィールドのストレージについて一貫している必要があります。存在していないフィールドに値を設定すると、そのフィールドが作成されます。また、フィールドの値として空文字列 (" ") を設定すると、指定したフィールドが削除されます。 **注:** このプロパティーに入力する値は、数値ではなくストリングでなければなりません。例えば、数値 1、2、3 および 4 は "1 2 3 4" として入力する必要があります。
names		ノードにより生成されたフィールド名のリストを設定または返す構造化スロット。
custom_storage	Unknown String Integer 実数 Time Date Timestamp	フィールドのストレージを設定するか返す、キー・スロット。
data_mode	Combined (結合) Ordered	Combined が指定された場合、レコードは、セット値と最小/最大値のそれぞれ組み合わせについて生成されます。生成されたレコード数は、それぞれのフィールドの値の数値の積に等しくなります。Ordered が指定された場合、データ行を生成するために、各レコードの各列から 1 個の値が取られます。生成されるレコード数は、フィールドに関連付けられている最大の値に等しくなります。より小さいデータ値を持つフィールドは、ヌル値で埋められます。
values		*このプロパティーは、data により廃止されるため、使用しないでください。*

variablefile ノードのプロパティー

可変長ノードで、可変長フィールド・テキスト・ファイル、つまりフィールド数は一定でも各フィールド内の文字数が異なるレコードを含むファイルから、データを読み込みます。このノードは、固定長のヘッダー・テキストやある種の注釈があるファイルにも使用できます。

表 42. *variablefile* ノードのプロパティー:

variablefile ノードのプロパティー	データ型	プロパティーの説明
skip_header	*number*	最初のレコードの先頭で無視する文字数を指定します。

表 42. *variablefile* ノードのプロパティー *(続き)*:

variablefile ノードのプロパティー	**データ型**	**プロパティーの説明**
num_fields_auto	*boolean*	各レコードのフィールドの数を自動的に決定します。レコードは、改行文字で終わる必要があります。
num_fields	*number*	各レコードのフィールドの数を手動で指定します。
delimit_space	*boolean*	ファイルのフィールドを区切る文字を指定します。
delimit_tab	*boolean*	
delimit_new_line	*boolean*	
delimit_non_printing	*boolean*	
delimit_comma	*boolean*	この場合、コンマはストリーム内でフィールドの区切り文字と桁区切り記号の両方であるため、delimit_other を *true* に設定し、other プロパティーを使用し、コンマを区切り記号として指定します。
delimit_other	*boolean*	other プロパティーを使用して、カスタム区切り記号をユーザーが指定できます。
other	*文字列*	delimit_other が *true* に設定されているときに使用される区切り記号を指定します。
decimal_symbol	Default Comma Period	データ・ソースで使われている小数点記号を指定します。
multi_blank	*boolean*	複数の隣接するブランク区切り文字を 1 つの区切り文字として扱います。
read_field_names	*boolean*	データ・ファイル中の最初の行を列のラベルとして取り扱います。
strip_spaces	None Left Right Both	インポート時に文字列の前後のスペースを破棄します。
invalid_char_mode	Discard Replace	データ入力から不正な文字 (ヌル、0、または現在のエンコード中に存在していない文字) をデータ入力から削除するか (Discard)、指定された 1 文字の記号で不正な文字を置き換えます (Replace)。
invalid_char_replacement	*文字列*	
break_case_by_newline	*boolean*	行区切り文字が改行文字であることを指定します。
lines_to_scan	*number*	指定したデータ型をスキャンする行数を指定します。
auto_recognize_datetime	*boolean*	入力データの日付または時刻を自動的に特定するかどうかを指定します。
quotes_1	Discard PairAndDiscard IncludeAsText	インポートでの単一引用符の処理方法を指定します。

表 42. *variablefile* ノードのプロパティー *(続き)*:

variablefile ノードのプロパティー	**データ型**	**プロパティーの説明**
quotes_2	Discard PairAndDiscard IncludeAsText	インポートでの二重引用符の処理方法を指定します。
full_filename	*文字列*	読み込みファイルのディレクトリーを含む完全な名前。
use_custom_values	*boolean*	
custom_storage	Unknown String Integer 実数 Time Date Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。

表 42. *variablefile* ノードのプロパティー *(続き)*:

variablefile ノードのプロパティー	データ型	プロパティーの説明
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"	カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。
custom_decimal_symbol	*field*	カスタム (ユーザー設定) ストレージが指定される場合のみ適用されます。
エンコード	StreamDefault SystemDefault "UTF-8"	テキストのエンコード方法を指定します。

xmlimport ノードのプロパティー

XML 入力ノードを使用して、XML 形式のデータをストリームにインポートできます。ディレクトリーの 1 つのファイルまたはすべてのファイルをインポートできます。オプションで、XML 構造を読み込むスキーマ ファイルを指定できます。

表 43. *xmlimport* ノードのプロパティー:

xmlimport ノードのプロパティー	データ型	プロパティーの説明
read	single ディレクトリ	単独のデータ・ファイルを読み込む (デフォルト) か、ディレクトリー内のすべての XML ファイルを読み込みます。
recurse	*boolean*	指定したディレクトリーのすべてのサブディレクトリーから XML ファイルを追加で読み込むかどうかを指定します。
full_filename	*文字列*	(必須) インポートする XML ファイルの完全パスおよびファイル名 (read = single の場合)。
directory_name	*文字列*	(必須) XML ファイルをインポートするディレクトリーの完全パスおよび名前 (read = directory の場合)。
full_schema_filename	*文字列*	XML 構造を読み込む XSD ファイルまたは DTD ファイルの完全パスおよびファイル名。このパラメーターを使用すると、構造を XML 入力ファイルから読み込みます。

表 43. *xmlimport* ノードのプロパティー *(続き)*:

xmlimport ノードのプロパティー	**データ型**	**プロパティーの説明**
records	*文字列*	レコードの境界を定義する XPath 式 (例: /author/name)。入力ファイルにこの要素が出現するごとに、新しいレコードが作成されます。
mode	read specify	すべてのデータを読み込む (デフォルト) か、読み込む項目を指定します。
fields		インポートする項目 (要素と属性) のリスト。リスト内の各アイテムは XPath 式です。

第 10 章 レコード設定ノードのプロパティー

append ノードのプロパティー

レコード追加ノードで、レコードのセットを連結します。レコード追加ノードは、構造が似ていながらデータが異なるデータ・セットを組み合わせる場合に役立ちます。

表 44. append ノードのプロパティー:

append ノードのプロパティー	データ型	プロパティーの説明
match_by	Position Name	メイン・データ・ソース中のフィールドの位置 (Position) 、または入力データ・セット中のフィールド名 (Name) を基準にして、データ・セットを追加できます。
match_case	*boolean*	フィールド名を比較するときに大文字と小文字の区別を有効にします。
include_fields_from	Main All	
create_tag_field	*boolean*	
tag_field_name	*文字列*	

レコード集計ノードのプロパティー

レコード集計ノードで、一連の入力レコードを要約集計された出力レコードに置き換えます。

表 45. レコード集計ノードのプロパティー:

aggregate ノードのプロパティー	データ型	プロパティーの説明
keys	*[フィールド フィールド ... フィールド]*	集計にキーとして使用できるフィールドが一覧表示されます。例えば、キー・フィールドが Sex と Region の場合、一意な M と F の、および地域 N と S のそれぞれの組み合わせに対して集計レコードが作成されます (4 つの一意な組み合わせ)。
contiguous	*boolean*	同じキー値を持つすべてのレコードが入力にグループ化されている場合 (例えば、入力がキー・フィールドにソートされる場合)、このオプションを選択します。このオプションを選択すると、パフォーマンスが向上します。
aggregates		集計する数値フィールド、および選択されている集計モードを表示する構造化プロパティー。

表 45. レコード集計ノードのプロパティー (続き):

aggregate ノードのプロパティー	データ型	プロパティーの説明
extension	文字列	重複集計フィールドに対応させる接頭辞または接尾辞を指定します (下の例を参照)。
add_as	Suffix Prefix	
inc_record_count	*boolean*	各集計レコードを作成するために集計された入力レコード数を指定する追加フィールドを作成します。
count_field	文字列	レコード度数フィールドの名前を指定します。

balance ノードのプロパティー

バランス・ノードで、データ・セットが指定した条件に合うように、データ・セットの不均衡を修正します。バランス式で、指定した比率によって条件が真 (true) の場合に、レコードの比率を調整します。

表 46. balance ノードのプロパティー:

balance ノードのプロパティー	データ型	プロパティーの説明
directives		指定された数値に基づいてフィールド値の割合を均衡にするための構造化プロパティー (次の例を参照してください)。
training_data_only	*boolean*	学習データのみがバランス化されるよう指定します。データ区分フィールドがストリーム中で指定されていない場合、このオプションは無視されます。

directives ノード・プロパティーが使用する形式は以下のとおりです。

「{ *数値文字列* } ¥ { *数値文字列*} ¥ ....{ *数値文字列* }].

注 : 文字列を式に埋め込む場合 (二重引用符を使用)、その先頭にエスケープ文字 " ¥ " を指定する必要があります。文字 " ¥ " は、引数を明確に記述するための行継続文字としても使われます。

derive_stb ノードのプロパティー

スペース-時間-ボックス・ノードは、緯度、経度、およびタイム・スタンプの各フィールドから、スペース-時間-ボックスを派生させます。頻度の高いスペース-時間-ボックスをハングアウトとして識別することもできます。

表 47. derive_stb ノードのプロパティー:

derive_stb ノードのプロパティー	データ型	プロパティーの説明
mode	IndividualRecords Hangouts	

表 47. *derive_stb* ノードのプロパティー *(続き)*:

derive_stb ノードのプロパティー	データ型	プロパティーの説明
latitude_field	*フィールド*	
longitude_field	*フィールド*	
timestamp_field	*フィールド*	
hangout_density	*密度 (density)*	単一密度。有効な密度値については、「densities」を参照してください。
densities	[*density*,*density*,..., *density*]	各 density は、STB_GH8_1DAY などの文字列です。**注:** どの density が有効であるかについては、制約があります。geohash の場合、GH1 から GH15 の値を使用できます。この部分では、以下の値を使用できます 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	*フィールド*	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	文字列でなければなりません。
min_events	*整数*	最小の有効な整数値は 2 です。

表 47. *derive_stb* ノードのプロパティー *(続き)*:

derive_stb ノードのプロパティー	データ型	プロパティーの説明
qualifying_pct	*整数*	1 から 100 の範囲でなければなりません。
add_extension_as	接頭辞 接尾辞	
name_extension	*文字列*	

distinct ノードのプロパティー

重複レコード・ノードで、重複レコードを削除します。その場合、最初の重複するレコードをデータ・ストリームに渡すか、または、最初のレコードを破棄して、その後の重複レコードをデータ・ストリームに渡します。

表 48. *distinct* ノードのプロパティー:

distinct ノードのプロパティー	データ型	プロパティーの説明
mode	Include Discard	データ・ストリームに最初の重複レコードを含めるか、最初の重複レコードを破棄して、代わりにすべての重複レコードをデータ・ストリームに渡すことができます。
grouping_fields	[*フィールド フィールド フィールド*]	レコードが同一であるかどうかを判断するために使われるフィールドを表示します。 **注:** このプロパティーは、IBM SPSS Modeler 16 以降では廃止されています。
composite_value	構造化スロット	
composite_values	構造化スロット	
inc_record_count	*boolean*	各集計レコードを作成するために集計された入力レコード数を指定する追加フィールドを作成します。
count_field	*文字列*	レコード度数フィールドの名前を指定します。
sort_keys	構造化スロット。	**注:** このプロパティーは、IBM SPSS Modeler 16 以降では廃止されています。
default_ascending	*boolean*	
low_distinct_key_count	*boolean*	キー・フィールドに少ないレコードまたは少ない一意の値を持つよう指定します。
keys_pre_sorted	*boolean*	同じキー値を持つすべてのレコードが入力で一緒にグループ化されるよう指定します。
disable_sql_generation	*boolean*	

merge ノードのプロパティー

レコード結合ノードは、複数の入力レコードを取得し、入力フィールドの全部または一部を含む 1 つの出力レコードを作成します。この機能は、内部顧客データと購入人口データのような、異なるソースからのデータを結合する場合に役立ちます。

表 49. merge ノードのプロパティー:

merge ノードのプロパティー	データ型	プロパティーの説明
method	Order キー Condition (条件)	データ・ファイル内の順序に応じてレコードが結合されるか (Order)、またはキー・フィールド内が同じ値のレコードを結合するのにキー・フィールドが使用されるかどうか (Keys)、または指定された条件を満たした場合にレコードが結合されるかどうか (Condition) を指定します。
condition	*文字列*	method が Condition に設定されている場合、レコードを含めるまたは破棄する条件を指定します。
key_fields	[*フィールド フィールド フィールド*]	
common_keys	*boolean*	
join	Inner FullOuter PartialOuter Anti	
outer_join_tag.n	*boolean*	このプロパティーでは、*n* は「データセットの選択」ダイアログ・ボックスに表示されるタグ名です。どのようなデータセット数であっても不完全なレコードを作成する可能性があるので、複数のタグ名を指定できます。
single_large_input	*boolean*	ほかの入力と比べて比較的大きな入力を指定し最適化を行うかどうかを指定します。
single_large_input_tag	*文字列*	「ラージ・データ・セットの選択」ダイアログ・ボックスに表示されるタグ名を指定します。指定できる入力データ・セットは 1 つだけであるため、このプロパティーの使用法は outer_join_tag プロパティーと少し異なります (ブール値ではなく文字列です)。
use_existing_sort_keys	*boolean*	入力がすでにキー・フィールドでソート済みかどうかを指定します。
existing_sort_keys	[{*文字列* Ascending} ¥ {*文字列* Descending}]	すでにソートされたフィールドとソート方向を指定します。

rfmaggregate ノードのプロパティー

リーセンシ、フリクエンシ、マネタリー (RFM) のレコード集計ノードを使用すると、顧客の過去のトランザクション・データを取得、未使用のデータを削除、残りのトランザクション・データをすべて単一行に結合することができます。これにより、最後のトランザクションの時期、トランザクション数、これらのトランザクションの合計金額が一覧表示されます。

表 50. rfmaggregate ノードのプロパティー:

rfmaggregate ノードのプロパティー	**データ型**	**プロパティーの説明**
relative_to	Fixed Today	トランザクションのリーセンシが計算される日付を指定します。
reference_date	*date*	Fixed が relative_to に設定されている場合にのみ使用できます。
contiguous	*boolean*	データ・ストリーム中で同じ ID を持つすべてのレコードが一緒に表示されるようにデータをソートしている場合、このオプションを選択すると処理を高速化することができます。
id_field	*field*	顧客およびトランザクションを識別するために使用するフィールドを指定します。
date_field	*field*	リーセンシを計算するために使用される日付フィールドを選択します。
value_field	*field*	マネタリー値を計算するために使用するフィールドを指定します。
extension	*文字列*	重複集計フィールドに対応させる接頭辞または接尾辞を指定します。
add_as	Suffix Prefix	extension を接尾辞として追加するか、または接頭辞として追加するかを指定します。
discard_low_value_records	*boolean*	discard_records_below 設定の使用を有効にします。
discard_records_below	*number*	RFM の合計を計算する場合に使用されないトランザクションの詳細の最小値を指定することができます。値の単位は、選択された value フィールドに関連します。
only_recent_transactions	*boolean*	specify_transaction_date 設定と transaction_within_last 設定のいずれかを使用可能にします。
specify_transaction_date	*boolean*	
transaction_date_after	*date*	specify_transaction_date が選択されている場合にのみ使用できます。データが分析に含まれた後のトランザクションの日付を指定します。
transaction_within_last	*number*	transaction_within_last が選択されている場合にのみ使用できます。レコードが分析に含まれる後の「リーセンシの相対値を計算」の日付からさかのぼった期間の数および種類 (日、週、月または年数) を指定します。

表 50. *rfmaggregate* ノードのプロパティー *(続き)*:

rfmaggregate ノードのプロパティー	データ型	プロパティーの説明
transaction_scale	Days 週 Months Years	transaction_within_last が選択されている場合にのみ使用できます。レコードが分析に含まれる後の「リーセンシの相対値を計算」の日付からさかのぼった期間の数および種類 (日、週、月または年数) を指定します。
save_r2	*boolean*	各顧客の 2 番目に最近のトランザクションの日付を表示します。
save_r3	*boolean*	save_r2 が選択されている場合にのみ使用できます。各顧客の 3 番目に最近のトランザクションの日付を表示します。

Rprocess ノードのプロパティー

R プロセス・ノードでは、IBM(r) SPSS(r) Modeler ストリームからデータを取得し、そのデータを独自のカスタム R スクリプトを使用して変更できます。データ変更後、データはストリームに返されます。

表 51. *Rprocess* ノードのプロパティー:

Rprocess ノードのプロパティー	データ型	プロパティーの説明
構文	*文字列*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*boolean*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*boolean*	

sample ノードのプロパティー

サンプル・ノードでは、レコードのサブセットを選択します。層化サンプル、クラスター・サンプル、非無作為 (構造化) サンプルなど、さまざまなサンプルの種類がサポートされています。サンプリングは、パフォーマンスの向上、および分析のための関連するレコードまたはトランザクションのグループの選択に役に立ちます。

表 52. *sample* ノードのプロパティー:

sample ノードのプロパティー	データ型	プロパティーの説明
method	Simple Complex	
mode	Include Discard	指定された条件を満たすレコードを含めるか (Include)、破棄 (Discard) します。

表 52. *sample* ノードのプロパティー *(続き)*:

sample ノードのプロパティー	データ型	プロパティーの説明
sample_type	First OneInN RandomPct	サンプリング方法を指定します。
first_n	*integer*	指定された分割点までのレコードを含めるか破棄します。
one_in_n	*number*	*n* 番目ごとにレコードを含めるか破棄します。
rand_pct	*number*	含めるか破棄するレコードのパーセンテージを指定します。
use_max_size	*boolean*	maximum_size 設定を使用可能にします。
maximum_size	*integer*	データ・ストリームに入れるまたはデータ・ストリームから破棄するサンプルの最大数を指定します。このオプションは冗長であり、そのため、First と Include が指定されているときは破棄されます。
set_random_seed	*boolean*	ランダム シード設定の使用を有効にします。
random_seed	*integer*	ランダム シードとして使用する値を指定します。
complex_sample_type	Random Systematic	
sample_units	比率 Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*integer*	
variable_proportions	*field*	
variable_counts	*field*	
use_min_stratum_size	*boolean*	
minimum_stratum_size	*integer*	このオプションは、Sample units=Proportions によって複雑なサンプルが作成された場合にのみ適用されます。
use_max_stratum_size	*boolean*	
maximum_stratum_size	*integer*	このオプションは、Sample units=Proportions によって複雑なサンプルが作成された場合にのみ適用されます。
clusters	*field*	
stratify_by	*[フィールド 1 ... フィールド N]*	
specify_input_weight	*boolean*	

表 52. *sample* ノードのプロパティー *(続き)*:

sample ノードのプロパティー	データ型	プロパティーの説明
input_weight	*field*	
new_output_weight	*文字列*	
sizes_proportions	[{string *string value*}{string *string value*}...]	sample_units=proportions および sample_size_proportions=Custom の場合、層化フィールドの値の考えられる組み合わせの値を指定します。
default_proportion	*number*	
sizes_counts	[{string *string value*}{string *string value*}...]	層化フィールドの値の考えられる組み合わせの値を指定します。使用方法は sizes_proportions と似ていますが、割合ではなく整数を指定します。
default_count	*number*	

select ノードのプロパティー

条件抽出ノードで、特定の条件に基づいて、データ・ストリームからレコードのサブセットを選択したり破棄したりできます。例えば、特定の営業地域に関連するレコードを選択できます。

表 53. *select* ノードのプロパティー:

select ノードのプロパティー	データ型	プロパティーの説明
mode	Include Discard	選択したレコードを含めるか、または破棄するかを指定します。
condition	*文字列*	レコードを含めるか、または破棄かの条件。

sort ノードのプロパティー

ソート・ノードで、1 つまたは複数のフィールド値に基づいて、レコードを昇順または降順にソートします。

表 54. *sort* ノードのプロパティー:

sort ノードのプロパティー	データ型	プロパティーの説明
keys	[{*文字列* Ascending} ¥ {*文字列* Descending}]	ソートの基準となるフィールドを指定します。ソートの方向が指定されていない場合、デフォルトが使用されます。
default_ascending	*boolean*	デフォルトのソート順を指定します。
use_existing_keys	*boolean*	前に使用されたフィールドのソート順を使用してソートを最適化するかどうかを指定します。

表 54. *sort ノードのプロパティー (続き)*:

sort ノードのプロパティー	データ型	プロパティーの説明
existing_keys		すでにソートされたフィールドとソート方向を指定します。keys プロパティーと同じ形式を使用します。

streamingts ノードのプロパティー

ストリーミング TS ノードは、1 つのステップで時系列モデルを作成してスコアリングします。時間間隔ノードは必要ありません。

表 55. *streamingts ノードのプロパティー*:

streamingts ノードのプロパティー	データ型	プロパティーの説明
custom_fields	*boolean*	custom_fields=false の場合は、上流のデータ型ノードの設定が使用されます。 custom_fields=true の場合は、targets と inputs を指定する必要があります。
targets	[*フィールド 1...フィールド N*]	
inputs	[*フィールド 1...フィールド N*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*boolean*	
conf_limit_pct	*real*	
use_time_intervals_node	*boolean*	use_time_intervals_node=true の場合は、上流の時間間隔ノードの設定が使用されます。これ以外の場合は、interval_offset_position、interval_offset、および interval_type を指定する必要があります。
interval_offset_position	LastObservation LastRecord	LastObservation は、「**最後の有効な観測**」を表します。LastRecord は、「**最後のレコードからカウント**」を表します。
interval_offset	*数値*	
interval_type	期間 年 四半期 月 WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*フィールド*	

表 55. *streamingts* ノードのプロパティー *(続き)*:

streamingts ノードのプロパティー	**データ型**	**プロパティーの説明**
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*boolean*	
detect_outliers	*boolean*	
expert_outlier_additive	*boolean*	
expert_outlier_level_shift	*boolean*	
expert_outlier_innovational	*boolean*	
expert_outlier_transient	*boolean*	
expert_outlier_seasonal_additive	*boolean*	
expert_outlier_local_trend	*boolean*	
expert_outlier_additive_patch	*boolean*	
exsmooth_model_type	シンプル HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	なし SquareRoot NaturalLog	
arima_p	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_d	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_q	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_sp	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_sd	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_sq	整数	時系列モデル作成ノードの場合と同じプロパティー
arima_transformation_type	なし SquareRoot NaturalLog	時系列モデル作成ノードの場合と同じプロパティー
arima_include_constant	*boolean*	時系列モデル作成ノードの場合と同じプロパティー
tf_arima_p.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_d.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_q.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。

表 55. *streamingts* ノードのプロパティー *(続き)*:

streamingts ノードのプロパティー	**データ型**	**プロパティーの説明**
tf_arima_sp.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_sd.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_sq.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_delay.*fieldname*	整数	時系列モデル作成ノードの場合と同じプロパティー。転送関数用。
tf_arima_transformation_type.*fieldname*	なし SquareRoot NaturalLog	
arima_detect_outlier_mode	なし 自動	
arima_outlier_additive	*boolean*	
arima_outlier_level_shift	*boolean*	
arima_outlier_innovational	*boolean*	
arima_outlier_transient	*boolean*	
arima_outlier_seasonal_additive	*boolean*	
arima_outlier_local_trend	*boolean*	
arima_outlier_additive_patch	*boolean*	
deployment_force_rebuild	*boolean*	
deployment_rebuild_mode	Count Percent	
deployment_rebuild_count	*数値*	
deployment_rebuild_pct	*数値*	
deployment_rebuild_field	*<フィールド>*	

第 11 章 フィールド設定ノードのプロパティー

anonymize ノードのプロパティー

匿名化ノードは、フィールド名や値の下流の表示方法を変換し、元のデータを隠します。これは、他のユーザーが顧客名やその他の詳細情報をなどの重要情報を使用してモデルを構築できるようにする場合に有用です。

表 56. 匿名化ノードのプロパティー:

anonymize ノードのプロパティー	データ型	プロパティーの説明
enable_anonymize	*boolean*	T に設定したときは、フィールドの値の匿名化をアクティブ化します (「値を匿名化」列でそのフィールドのために「**はい**」を選択することと同等です)。
use_prefix	*boolean*	T に設定したときは、ユーザー指定接頭辞が指定されている場合に、そのユーザー指定接頭辞が使用されます。ハッシュ・メソッドによって匿名化されるフィールドに適用され、そのフィールドの「値を置換」ダイアログの「**ユーザー設定**」ラジオ・ボタンを選択することと同等です。
prefix	*文字列*	「値を置換」ダイアログ・ボックスのテキスト・ボックスに接頭辞を入力することと同等です。デフォルトの接頭辞は、何も他に指定されていない場合は、デフォルト値です。
transformation	Random Fixed	Transform メソッドにより匿名化されたフィールドの変換パラメーターが無作為 (Random) か固定 (Fixed) かを決定します。
set_random_seed	*boolean*	T に設定したときは、指定されたシード値を使用します (変換も Random に設定されている場合)。
random_seed	*integer*	set_random_seed を設定したときは、これが乱数のシードになります。
scale	*number*	変換を Fixed に設定したときに、この値がスケールに使用されます。最大スケール値は通常 10 ですが、あふれを防止するために減少できます。
translate	*number*	変換を Fixed に設定したときに、この値が変換に使用されます。最大変換値は通常 1000 ですが、あふれを防止するために減少できます。

autodataprep ノードのプロパティー

自動データ準備 (ADP) ノードでは、データ分析、固定値の識別、問題のあるまたは役に立たない可能性のあるフィールドのスクリーニング、必要に応じた新しい属性の取得、詳細なスクリーニングおよびサンプリング手法を使用したパフォーマンスの向上などを行うことができます。完全に自動化された方法でノードを使用し、ノードで固定値を選択および適用できます。または必要に応じて変更の作成および承認、拒否または修正の前に変更をプレビューできます。

表 57. autodataprep ノードのプロパティー:

autodataprep ノードのプロパティー	**データ型**	**プロパティーの説明**
objective	Balanced Speed Accuracy Custom	
custom_fields	*boolean*	真 (true) の場合は、現在のノードのターゲット、入力、その他フィールドなどを指定することができます。偽 (false) の場合は、上流のデータ型ノードから現在の設定が使用されます。
ターゲット	*field*	1 つの対象フィールドを指定します。
inputs	[*field1 ... fieldN*]	モデルで使用される入力または予測変数フィールド。
use_frequency	*boolean*	
frequency_field	*field*	
use_weight	*boolean*	
weight_field	*field*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*boolean*	すべての日付/時間フィールドへのアクセスを制御します。
compute_time_until_date	*boolean*	
reference_date	Today Fixed	
fixed_date	*date*	
units_for_date_durations	Automatic (自動) Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*boolean*	
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic (自動) Fixed	
fixed_date_units	時 Minutes Seconds	
extract_year_from_date	*boolean*	
extract_month_from_date	*boolean*	
extract_day_from_date	*boolean*	
extract_hour_from_time	*boolean*	

表 57. *autodataprep* ノードのプロパティー *(続き)*:

autodataprep ノードのプロパティー	データ型	プロパティーの説明
extract_minute_from_time	*boolean*	
extract_second_from_time	*boolean*	
exclude_low_quality_inputs	*boolean*	
exclude_too_many_missing	*boolean*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*boolean*	
maximum_number_categories	*number*	
exclude_if_large_category	*boolean*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*boolean*	
adjust_type_inputs	*boolean*	
adjust_type_target	*boolean*	
reorder_nominal_inputs	*boolean*	
reorder_nominal_target	*boolean*	
replace_outliers_inputs	*boolean*	
replace_outliers_target	*boolean*	
replace_missing_continuous_inputs	*boolean*	
replace_missing_continuous_target	*boolean*	
replace_missing_nominal_inputs	*boolean*	
replace_missing_nominal_target	*boolean*	
replace_missing_ordinal_inputs	*boolean*	
replace_missing_ordinal_target	*boolean*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace 削除	
rescale_continuous_inputs	*boolean*	
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*boolean*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*boolean*	
maximize_association_with_target	*boolean*	
p_value_for_merging	*number*	

表 57. *autodataprep* ノードのプロパティー *(続き)*:

autodataprep ノードのプロパティー	データ型	プロパティーの説明
merge_ordinal_features	*boolean*	
merge_nominal_features	*boolean*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*boolean*	
p_value_for_binning	*number*	
perform_feature_selection	*boolean*	
p_value_for_selection	*number*	
perform_feature_construction	*boolean*	
transformed_target_name_extension	*文字列*	
transformed_inputs_name_extension	*文字列*	
constructed_features_root_name	*文字列*	
years_duration_ name_extension	*文字列*	
months_duration_ name_extension	*文字列*	
days_duration_ name_extension	*文字列*	
hours_duration_ name_extension	*文字列*	
minutes_duration_ name_extension	*文字列*	
seconds_duration_ name_extension	*文字列*	
year_cyclical_name_extension	*文字列*	
month_cyclical_name_extension	*文字列*	
day_cyclical_name_extension	*文字列*	
hour_cyclical_name_extension	*文字列*	
minute_cyclical_name_extension	*文字列*	
second_cyclical_name_extension	*文字列*	

binning ノードのプロパティー

データ分割ノードで、既存の 1 つまたは複数の連続型 (数値範囲) フィールドの値に基づいて、自動的に新しい名義型 (セット型) フィールドを作成します。例えば、連続型収入フィールドを、平均からの偏差による収入グループを含む、新しいカテゴリー・フィールドに変換することができます。新規フィールドのビンを作成すると、分割点に基づいてフィールド作成ノードを生成することができます。

表 58. *binning* ノードのプロパティー:

binning ノードのプロパティー	データ型	プロパティーの説明
fields	*[field1 field2 ... fieldn]*	変換保留中の連続型 (数値範囲) フィールド。複数のフィールドを同時にビンに分割できます。

表 58. binning ノードのプロパティー (続き):

binning ノードのプロパティー	**データ型**	**プロパティーの説明**
method	FixedWidth EqualCount Rank SDev Optimal	新規フィールドのビン (カテゴリー) の分割点を決める方法。
rcalculate_bins	Always IfNecessary	ノードが実行されるごとに、ビンが再計算され、適切なビンの中にデータが配置されるか、またはデータが既存のビンおよび追加された新規のビンに追加されるだけかを指定します。
fixed_width_name_extension	*文字列*	デフォルトの拡張子は *_BIN* です。
fixed_width_add_as	Suffix Prefix	拡張子をフィールド名の最後に追加するか (Suffix)、または先頭に追加するか (Prefix) を指定します。デフォルトの拡張子は *income_BIN* です。
fixed_bin_method	Width Count	
fixed_bin_count	*integer*	新規フィールドの固定幅ビン (カテゴリー) 数を決定するのに使用する整数を指定します。
fixed_bin_width	*real*	ビンの幅を算出するために使用する値 (整数または実数)。
equal_count_name_ extension	*文字列*	デフォルトの拡張子は *_TILE* です。
equal_count_add_as	Suffix Prefix	標準の分位を使用して生成されるフィールドに対して使用される拡張子が、Suffix (接頭辞) か Prefix (接尾辞) かを指定します。デフォルトの拡張子は、*_TILE* に *N* を付けたものになります。*N* は分位数です。
tile4	*boolean*	それぞれが 25 % のケースを含む、4 分位のビンを生成します。
tile5	*boolean*	5 つの 5 分位ビンを生成します。
tile10	*boolean*	10 個の十分位 (デシル) ビンを生成します。
tile20	*boolean*	20 個の二十分位ビンを生成します。
tile100	*boolean*	100 個の百分位 (パーセンタイル) ビンを生成します。
use_custom_tile	*boolean*	
custom_tile_name_extension	*文字列*	デフォルトの拡張子は *_TILEN* です。
custom_tile_add_as	Suffix Prefix	
custom_tile	*integer*	
equal_count_method	RecordCount ValueSum	RecordCount の方法は、同じ数のレコードを各ビンに割り当てます。一方、ValueSum では、各ビンの値の合計が同じになるようにレコードを割り当てます。

表 58. *binning* ノードのプロパティー *(続き)*:

binning ノードのプロパティー	**データ型**	**プロパティーの説明**
tied_values_method	Next Current Random	可否同数の値のデータに配置されるビンを指定。
rank_order	Ascending Descending	このプロパティーには、Ascending (もっとも小さい値が 1 となる) または Descending (もっとも大きい値が 1 となる) が含まれます。
rank_add_as	Suffix Prefix	このオプションは、ランク、ランクの比率、およびランクのパーセンテージに適用されます。
rank	*boolean*	
rank_name_extension	*文字列*	デフォルトの拡張子は *_RANK* です。
rank_fractional	*boolean*	新規フィールドの値が、ランクを非欠損ケースの重みの合計で除算した値になるように、ケースをランク付けします。小数点付き順位の範囲は 0 から 1 までです。
rank_fractional_name_ extension	*文字列*	デフォルトの拡張子は *_F_RANK* です。
rank_pct	*boolean*	各ランクが、有効な値を持つレコード数で除算された後、100 倍されます。パーセンテージの小数点付き順位の範囲は 1 から 100 までです。
rank_pct_name_extension	*文字列*	デフォルトの拡張子は *_P_RANK* です。
sdev_name_extension	*文字列*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*文字列*	デフォルトの拡張子は *_OPTIMAL* です。
optimal_add_as	Suffix Prefix	
optimal_supervisor_field	*field*	データ分割のために選択されたフィールドが関係する監督フィールドとして選ばれたフィールド。
optimal_merge_bins	*boolean*	ケース度数が小さいビンをより大きな隣接ビンに追加することを指定します。
optimal_small_bin_threshold	*integer*	
optimal_pre_bin	*boolean*	データ・セットの事前データ分割を実行することを示します。
optimal_max_bins	*integer*	過度に多数のビンを作成しないように、上限を指定します。
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	

表 58. *binning ノードのプロパティー (続き)*:

binning ノードのプロパティー	データ型	プロパティーの説明
optimal_last_bin	Unbounded Bounded	

derive ノードのプロパティー

フィールド作成ノードで、1 つまたは複数の既存フィールドから、データ値を変更するか、新しいフィールドを作成します。これで、 タイプ式、フラグ、名義、ステート、カウント、および条件式の各フィールドが作成されます。

表 59. *derive ノードのプロパティー*:

derive ノードのプロパティー	データ型	プロパティーの説明
new_name	*文字列*	新しいフィールド名。
mode	Single Multiple	1 つのフィールドか (Single)、または複数フィールドか (Multiple) を指定します。
fields	[*フィールド フィールド フィールド*]	複数フィールドを選択する場合にだけ、Multiple モードで使用。
name_extension	*文字列*	新しいフィールド名に使用する拡張子を指定します。
add_as	Suffix Prefix	拡張子をフィールド名の Prefix (先頭、接頭辞)、または Suffix (最後、接尾辞) として追加します。
result_type	式 Flag Set 状態 Count Conditional	作成可能な新しいフィールドの 6 つの種類。
formula_expr	*文字列*	フィールド作成ノードの新しいフィールド値を計算する式。
flag_expr	*文字列*	
flag_true	*文字列*	
flag_false	*文字列*	
set_default	*文字列*	
set_value_cond	*文字列*	特定の値に関連付けられた条件を提供するように構造化プロパティー。
state_on_val	*文字列*	オン (On) の条件を満たす場合の新規フィールドの値を指定します。
state_off_val	*文字列*	オフ (Off) の条件を満たす場合の新規フィールドの値を指定します。
state_on_expression	*文字列*	
state_off_expression	*文字列*	

表 59. *derive* ノードのプロパティー *(続き)*:

derive ノードのプロパティー	データ型	プロパティーの説明
state_initial	On Off	各レコードで新しいフィールドの初期値として On または Off を割り当てます。この値は、それぞれの条件が満たされるごとに変化します。
count_initial_val	*文字列*	
count_inc_condition	*文字列*	
count_inc_expression	*文字列*	
count_reset_condition	*文字列*	
cond_if_cond	*文字列*	
cond_then_expr	*文字列*	
cond_else_expr	*文字列*	

ensemble ノードのプロパティー

アンサンブル・ノードでは、2 つまたはそれ以上のモデル・ナゲットを組み合わせて 1 つのモデルよりもより正確な予測を取得します。

表 60. *ensemble* ノードのプロパティー:

ensemble ノードのプロパティー	データ型	プロパティーの説明
ensemble_target_field	*field*	アンサンブルで使用されるすべてのモデルの対象フィールドを指定します。
filter_individual_model_output	*boolean*	個々のモデルのスコアリング結果を抑制するかどうかを指定します。
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	アンサンブル・スコアを決定するために使用する方法を指定します。この設定は、選択された対象がフラグ型フィールドである場合にのみ適用されます。
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	アンサンブル・スコアを決定するために使用する方法を指定します。この設定は、選択された対象が名義型フィールドである場合にのみ適用されます。
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	票決方法が選択された場合、可否同数の解決方法を指定します。この設定は、選択された対象がフラグ型フィールドである場合にのみ適用されます。
set_voting_tie_selection	Random HighestConfidence	票決方法が選択された場合、可否同数の解決方法を指定します。この設定は、選択された対象が名義型フィールドである場合にのみ適用されます。

表 60. *ensemble* ノードのプロパティー *(続き)*:

ensemble ノードのプロパティー	データ型	プロパティーの説明
calculate_standard_error	*boolean*	対象フィールドが連続型の場合、標準誤差の計算がデフォルトで実施され、測定された値または推定された値と真の値との差異を計算し、それらの推定がどれほど近いかを示します。

filler ノードのプロパティー

置換ノードで、フィールド値の置換やストレージの変更を行います。@BLANK(@FIELD) のような、CLEM 条件に基づいて値を置換することができます。また、すべての空白値やヌル値を特定の値に置換することもできます。置換ノードは、データ型ノードと一緒に使用される場合が多く、欠損値の置き換えが行われます。

表 61. *filler* ノードのプロパティー:

filler ノードのプロパティー	データ型	プロパティーの説明
fields	[*フィールド フィールド フィールド*]	検査されて置換される値のデータ・セットのフィールド群。
replace_mode	Always Conditional Blank Null BlankAndNull	すべての値、空白値、またはヌル値を置換できます。または、指定した条件に基づいて、置換できます。
condition	*文字列*	
replace_with	*文字列*	

filter ノードのプロパティー

フィルター・ノードで、1 つの入力ノードから他の 1 つの入力ノードへ、フィールドをフィルタリング（破棄）し、フィールド名を変更し、また、フィールドを関連付けます。

default_include プロパティーの使用: default_include プロパティーの値を設定しても、すべてのフィールドが自動的に取り込まれたり除外されたりするわけではありません。単に、現在選択されている項目に対するデフォルトが決定されるだけです。これは、「フィルター・ノード」ダイアログ・ボックスで **「デフォルトでフィールドを含める」** をクリックすることと、機能的に同じです。

表 62. *filter* ノードのプロパティー:

filter ノードのプロパティー	データ型	プロパティーの説明
default_include	*boolean*	デフォルトの処理としてフィールドを通過させるかフィルターをかけるかの指定をするキー・プロパティー。このプロパティーを設定しても、すべてのフィールドが自動的に取り込まれたり除外されたりするわけではありません。選択したフィールドが、デフォルトでは取り込まれるか除外されるかを決めるだけです。
include	*boolean*	フィールドを取り込むか除外するかのキー・プロパティー。
new_name	*文字列*	

history ノードのプロパティー

時系列ノードにより、以前レコードのフィールドのデータを含む、新規フィールドが作成されます。時系列ノードは、多くの場合、時系列データなどの継続的なデータに使用されます。時系列ノードを使用する前に、ソート・ノードを使用して、データをソートしておくこともできます。

表 63. *history* ノードのプロパティー:

history ノードのプロパティー	データ型	プロパティーの説明
fields	[*フィールド フィールド フィールド*]	履歴の対象となるフィールド。
offset	*number*	時系列フィールド値を抽出する最新レコードが、現在のレコードのいくつ前にあるかを指定します。
span	*number*	値を抽出する元になるレコードの前にあるレコード数を指定します。
unavailable	Discard Leave Fill	時系列として使用する前のレコードがないデータ・セットの先頭の数レコードを通常は指しますが、その時系列値がないレコードの取り扱い方法を指定します。
fill_with	String 数値	時系列値が利用できないレコードを充填するのに使用する値 (Number) または文字列 (String) を指定します。

partition ノードのプロパティー

データ区分ノードで、モデル構築の学習、テスト、および検証の各ステージ用に、データを独立したサブセットに分割するデータ区分フィールドが生成されます。

表 64. *partition* ノードのプロパティー：

partition ノードのプロパティー	データ型	プロパティーの説明
new_name	*文字列*	ノードにより生成されたデータ区分フィールドの名前です。
create_validation	*boolean*	検証用のデータ区分を作成するかどうかを指定します。
training_size	*integer*	学習用区分に割り当てるレコード数のパーセンテージ (0-100)。
testing_size	*integer*	テスト用区分に割り当てるレコード数のパーセンテージ (0-100)。
validation_size	*integer*	検証用区分に割り当てるレコード数のパーセンテージ (0-100)。検証用データ区分を生成しない場合は無視されます。
training_label	*文字列*	学習用データ区分のラベル。
testing_label	*文字列*	テスト用データ区分のラベル。
validation_label	*文字列*	検証用データ区分のラベル。検証用データ区分を生成しない場合は無視されます。
value_mode	System SystemAndLabel Label	データ中の各データ区分を表すために使用される値を指定します。例えば、学習用サンプルは、システム整数 1 、ラベル Training、またはこの 2 つを組み合わせた 1_Training のように表されます。
set_random_seed	*boolean*	ユーザー指定のランダム シードを使用するかどうかを指定します。
random_seed	*integer*	ユーザー定義のランダム シードの値。この値が使用されるようにするには、set_random_seed を True に設定する必要があります。
enable_sql_generation	*boolean*	SQL プッシュバックを使用してレコードをデータ区分に割り当てるかどうかを指定します。
unique_field		レコードが無作為で繰り返し可能な方法でデータ区分に割り当てるよう、入力フィールドを指定します。この値が使用されるようにするには、enable_sql_generation を True に設定する必要があります。

reclassify ノードのプロパティー

データ分類ノードにより、あるカテゴリー値のセットが別のセットに変換されます。データ分類ノードは、カテゴリーを縮小したり、分析用にデータをグループ化し直したりする場合に役立ちます。

表 65. *reclassify* ノードのプロパティー:

reclassify ノードのプロパティー	データ型	プロパティーの説明
mode	Single Multiple	1 つのフィールドのカテゴリーを再分類する場合、Single を使用します。Multiple (複数) を使用すると、一度に複数のフィールドを同時に変換できます。
replace_field	*boolean*	
field	*文字列*	Single モードでしか使用できません。
new_name	*文字列*	Single モードでしか使用できません。
fields	*[field1 field2 ... fieldn]*	Multiple モードでしか使用できません。
name_extension	*文字列*	Multiple モードでしか使用できません。
add_as	Suffix Prefix	Multiple モードでしか使用できません。
reclassify	*文字列*	フィールド値用構造化プロパティー。
use_default	*boolean*	デフォルト値を使用します。
default	*文字列*	デフォルト値を指定します。
pick_list	*[string string ... string]*	ユーザーが、既知の新しい値をインポートしてテーブル内のドロップダウン・リストをデータで埋めることができるようにします。

reorder ノードのプロパティー

フィールド順序ノードで、下流のフィールド表示に使用する順序を定義します。この順序は、テーブル、リスト、およびフィールド・ピッカーなど、さまざまな場所のフィールドの表示に適用されます。この操作は、さまざまなデータ・セットにおいて、特定のフィールドをより参照しやすくする場合に役立ちます。

表 66. *reorder* ノードのプロパティー:

reorder ノードのプロパティー	データ型	プロパティーの説明
mode	Custom Auto	値を自動的に並び替えたり、ユーザー指定の順序を指定することができます。
sort_by	Name Type Storage	
ascending	*boolean*	
start_fields	*[field1 field2 ... fieldn]*	新規フィールドは、これらのフィールドの後に挿入されます。
end_fields	*[field1 field2 ... fieldn]*	新規フィールドは、これらのフィールドの前に挿入されます。

restructure ノードのプロパティー

再構成ノードで、名義型またはグラグ型フィールドを、これから別のフィールドの値で埋めることができるフィールドのグループへ変換します。例えば、*credit*、*cash*、および *debit* の値の *payment type* という名前のフィールドがある場合、3 つの新しいフィールド (*credit*、*cash*、*debit*) が作成されます。その各々には、実際の支払の値を含めることができます。

表 67. restructure ノードのプロパティー:

restructure ノードのプロパティー	データ型	プロパティーの説明
fields_from	[*category category category*] all	
include_field_name	*boolean*	再構成されるフィールド名に元のフィールド名を使用するかどうかを示します。
value_mode	OtherFields Flags	再構成されるフィールドの値を指定するためのモードを示します。OtherFields を指定すると、使用するフィールドを指定する必要があります (下を参照)。Flags を指定する場合、値は数値のフラグです。
value_fields	[フィールド フィールド フィールド]	value_mode が OtherFields の場合は必須です。値のフィールドとして使用するフィールドを指定します。

rfmanalysis ノードのプロパティー

リーセンシ、フリクエンシ、マネタリー (RFM) の分析ノードを使用すると、最後に購入したのがどのくらい最近か (リーセンシ)、どのくらい頻繁に購入するか (フリクエンシ)、トランザクション全体でいくら消費したか (マネタリー) を検証することによって、最も良い顧客となると考えられるのはどの顧客かを量的に決定することができます。

表 68. rfmanalysis ノードのプロパティー:

rfmanalysis ノードのプロパティー	データ型	プロパティーの説明
recency	*field*	リーセンシ フィールドを指定します。このフィールドは日付、タイムスタンプまたは単純な数値です。
frequency	*field*	フリクエンシ フィールドを指定します。
monetary	*field*	マネタリー・フィールドを指定します。
recency_bins	*integer*	生成されるリーセンシ ビンの数を指定します。
recency_weight	*number*	リーセンシ データに適用される重みを指定します。The default is 100.
frequency_bins	*integer*	生成されるフリクエンシ ビンの数を指定します。
frequency_weight	*number*	フリクエンシ データに適用される重みを指定します。デフォルト値は 10 です。

表 68. *rfmanalysis* ノードのプロパティー *(続き)*:

rfmanalysis ノードのプロパティー	データ型	プロパティーの説明
monetary_bins	*integer*	生成されるマネタリー・ビンの数を指定します。
monetary_weight	*number*	マネタリー・データに適用される重みを指定します。デフォルトは 1 です。
tied_values_method	Next Current	可否同数の値のデータに配置されるビンを指定。
recalculate_bins	Always IfNecessary	
add_outliers	*boolean*	recalculate_bins が IfNecessary に設定されている場合使用できます。設定されると、下限のビンの下にあるレコードが下限のビンに追加され、上限のビンの上にあるレコードが上限のビンに追加されます。
binned_field	リーセンシー 頻度 Monetary	
recency_thresholds	*value value*	recalculate_bins が Always に設定されている場合使用できます。リーセンシ ビンの上限および下限の閾値を指定します。あるビンの上限の閾値が次のビンの下限の閾値として使用されます。例えば、[10 30 60] は、最初のビンに 10 および 30 の上限および下限の閾値があり、2 番目のビンには 30 および 60 の閾値があると定義します。
frequency_thresholds	*value value*	recalculate_bins が Always に設定されている場合使用できます。
monetary_thresholds	*value value*	recalculate_bins が Always に設定されている場合使用できます。

settoflag ノードのプロパティー

フラグ設定ノードで、1 つ以上の名義型フィールドに定義されたカテゴリー値に基づいた、複数のフラグ型フィールドが派生します。

表 69. *settoflag* ノードのプロパティー:

settoflag ノードのプロパティー	データ型	プロパティーの説明
fields_from	[*category category category*] all	

表 69. *settoflag* ノードのプロパティー *(続き)*:

settoflag ノードのプロパティー	データ型	プロパティーの説明
true_value	*文字列*	フラグを設定するときにノードが使用する真 (true) の値を指定します。デフォルトは T です。
false_value	*文字列*	フラグを設定するときにノードが使用する偽 (false) の値を指定します。デフォルトは F です。
use_extension	*boolean*	新規フラグ型フィールドの接尾辞または接頭辞として、拡張子を使用します。
extension	*文字列*	
add_as	Suffix Prefix	拡張子が接尾辞 (Suffix) または接頭辞 (Prefix) として追加されることを指定します。
aggregate	*boolean*	キー・フィールドに基づいてレコードをグループ化します。真 (true) に設定されたレコードが 1 つでもあると、グループ内のすべてのフラグ型フィールドが有効になります。
keys	[*フィールド フィールド フィールド*]	キー・フィールド。

statisticstransform ノードのプロパティー

Statistics 変換ノードは、IBM SPSS Modeler のデータ・ソースに対する IBM SPSS Statistics シンタックス・コマンドの選択を行います。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

このノードのプロパティーについては、 247 ページの『statisticstransform ノードのプロパティー』に記載されています。

timeintervals ノードのプロパティー

時間区分ノードで、時系列データのモデル作成用に区分を指定し、必要に応じてラベルを作成します。値の間隔が均等に空けられていない場合は、レコード間に一律の間隔をとる必要に応じて、値を充填したり集計したりできます。

表 70. timeintervals ノードのプロパティー:

timeintervals ノードのプロパティー	**データ型**	**プロパティーの説明**
interval_type	None 期間 CyclicPeriods Years 四半期 Months DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Label Create	レコードに連続してラベルを付ける (Label) か、または指定された日付、タイムスタンプ、または時間フィールドに基づいて系列を構築するか (Create) を指定します。
field	*field*	データから系列を構築する場合は、各レコードの日付または時刻を示すフィールドを指定します。
period_start	*integer*	期間または循環する期間の開始期間を指定します。
cycle_start	*integer*	循環する期間の開始サイクル。
year_start	*integer*	適用可能な区分タイプの、最初の区分が入る年。
quarter_start	*integer*	適用可能な区分タイプの、最初の区分が入る四半期。
month_start	1 月 2 月 3 月 4 月 5 月 6 月 7 月 8 月 9 月 10 月 11 月 12 月	
day_start	*integer*	
hour_start	*integer*	
minute_start	*integer*	
second_start	*integer*	
periods_per_cycle	*integer*	循環する期間の、各サイクル内の期間数。

表 70. *timeintervals* ノードのプロパティー *(続き)*:

timeintervals ノードのプロパティー	**データ型**	**プロパティーの説明**
fiscal_year_begins	1 月 2 月 3 月 4 月 5 月 6 月 7 月 8 月 9 月 10 月 11 月 12 月	四半期単位の区分の場合、会計年度が始まる月を指定します。
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	定期的な区分 (週当たりの日数、日当たりの時間数、日当たりの分数、日当たりの秒数) の、週が始まる曜日を指定します。
day_begins_hour	*integer*	定期的な区分 (日当たりの時間数、日当たりの分数、日当たりの秒数) の、日が始まる時間を指定します。day_begins_minute および day_begins_second と組み合わせて使用し、厳密な時刻 (*8:05:01* など) を指定することができます。下の使用例を参照してください。
day_begins_minute	*integer*	定期的な区分 (日当たりの時間数、日当たりの分数、日当たりの秒数) の、日が始まる時間の分を指定します (例えば *8:05* の *5*)。
day_begins_second	*integer*	定期的な区分 (日当たりの時間数、日当たりの分数、日当たりの秒数) の、日が始まる時間の秒を指定します (例えば *08:05:17* の *17*)。
days_per_week	*integer*	定期的な区分 (週当たりの日数、日当たりの時間数、日当たりの分数、日当たりの秒数) の、週当たりの日数を指定します。
hours_per_day	*integer*	定期的な区分 (日当たりの時間数、日当たりの分数、日当たりの秒数) の、1 日の時間数を指定します。
interval_increment	1 2 3 4 5 6 10 15 20 30	日当たりの分数と日当たりの秒数について、各レコード用増分の分数または秒数を指定します。

表 70. timeintervals ノードのプロパティー (続き):

timeintervals ノードのプロパティー	**データ型**	**プロパティーの説明**
field_name_extension	*文字列*	
field_name_extension_as_prefix	*boolean*	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	フィールドの集計方法を指定します。

表 70. *timeintervals* ノードのプロパティー *(続き)*:

timeintervals ノードのプロパティー	データ型	プロパティーの説明
pad	ブランク MeanOfRecentPoints True False	フィールドの充填方法を指定します。
agg_mode	All Specify	必要に応じてデフォルトの関数ですべてのフィールドを集計または充填するかどうかを指定します。または、使用するフィールドと関数を指定します。
agg_range_default	Mean Sum Mode Min Max	連続型フィールドを集計するときに使用するデフォルトの関数を指定します。
agg_set_default	Mode First Last	名義型フィールドを集計するときに使用するデフォルトの関数を指定します。
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	ブランク MeanOfRecentPoints	連続型フィールドをパディングするときに使用するデフォルトの関数を指定します。
pad_set_default	ブランク MostRecentValue	
pad_flag_default	ブランク True False	
max_records_to_create	*integer*	系列を充填するときに作成する最大レコード数を指定します。
estimation_from_beginning	*boolean*	
estimation_to_end	*boolean*	
estimation_start_offset	*integer*	
estimation_num_holdouts	*integer*	
create_future_records	*boolean*	
num_future_records	*integer*	
create_future_field	*boolean*	
future_field_name	*文字列*	

transpose ノードのプロパティー

行列入替ノードで、レコードがフィールドになり、フィールドがレコードになるように、行内と列内のデータを交換します。

表 71. *transpose* ノードのプロパティー:

transpose ノードのプロパティー	データ型	プロパティーの説明
transposed_names	Prefix Read	新しいフィールド名は、指定された接頭辞 (Prefix) に基づいて自動的に作成できます。または、既存のデータ内のフィールドからフィールド名を読み込むことができます (Read)。
prefix	*文字列*	
num_new_fields	*integer*	接頭辞を使用する場合は、作成する新しいフィールドの最大数を指定します。
read_from_field	*field*	名前が読み込まれるフィールド。これはインスタンス化されたフィールドであることが必要です。そうでない場合は、ノードが実行されるときにエラーが発生します。
max_num_fields	*integer*	フィールドから名前を読み込む場合は、異常に大量のフィールドを作成しないように、フィールド数の上限を指定します。
transpose_type	数値 String Custom	デフォルトでは連続型のフィールドのみの行列が入れ替えられますが、代わりに、数値フィールドのカスタム (ユーザー設定) サブセットを選択またはすべての文字列フィールドを入れ替えることもできます。
transpose_fields	[*フィールド フィールド フィールド*]	Custom (ユーザー設定) オプションを使用するときに、行列を入れ替えるフィールドを指定します。
id_field_name	*field*	

type ノードのプロパティー

データ型ノードで、フィールドのメタデータとプロパティーを指定します。例えば、各フィールドに、測定の尺度 (連続型、名義型、順序型、またはフラグ) を指定し、欠損値とシステムヌルの処理のためのオプションを設定し、モデル作成の目的に対するフィールドの役割を設定し、フィールドと値のラベルを指定し、フィールドの値を指定します。

ある種の場合、ほかのノードが正しく機能するように、フラグ設定ノードの fields from プロパティーのように、データ型ノードを完全にインスタンス化する必要がある場合があります。フィールドをインスタンス化するには、次のように、テーブル・ノードを接続して実行するだけです。

表 72. *type* ノードのプロパティー:

type ノードのプロパティー	データ型	プロパティーの説明
direction	Input Target Both None Partition Split 頻度 RecordID	フィールドの役割のキープロパティー。 注: 値 In および Out は 廃止されています。 今後のリリースでは サポートが中断される 場合があります。

表 72. *type* ノードのプロパティー *(続き)*:

type ノードのプロパティー	データ型	プロパティーの説明
type	Range (範囲) Flag Set 型なし Discrete OrderedSet Default	フィールドの尺度 (以前はフィールドの「タイプ」と呼ばれていました)。type を Default に設定すると、values パラメーター設定がすべてクリアされ、value_mode の値が Specify である場合は Read にリセットされます。value_mode が Pass または Read に設定されている場合は、type を設定しても value_mode には影響しません。 注: 内部で使用されるデータ型は、 データ型ノードで表示されている データ型とは異なります。 以下のように対応しています。 Range -> 連続型 Set - > 名義型 OrderedSet -> 順序型 Discrete- > カテゴリー型
storage	Unknown String Integer 実数 Time Date Timestamp	フィールドのストレージ・タイプ用読み込み専用キー・プロパティー。
check	None 無効化 強制 Discard 警告 Abort	フィールド・タイプと範囲の検査用のキー・プロパティー。
values	[*value value*]	連続型フィールドの場合、最初の値が最小値で最後の値が最大値になります。名義型フィールドの場合、すべての値を指定します。フラグ型の場合、最初の値が *false* (偽) を、最後の値が *true* (真) を表します。このプロパティーを設定すると、value_mode プロパティーの値が自動的に Specify に設定されます。
value_mode	Read Pass Read+ Current Specify	値の設定方法を決定します。このプロパティーに Specify を直接には設定できないことに注意してください。特定の値を使用するには、values プロパティーを設定します。
extend_values	*boolean*	value_mode が Read に設定された場合に適用されます。新しく読み込んだ値を、フィールドの既存の値に追加する場合は、T を設定します。新しく読み込んだ値を優先して、既存の値を破棄する場合は、F を設定します。

表 72. type ノードのプロパティー (続き):

type ノードのプロパティー	データ型	プロパティーの説明
enable_missing	*boolean*	T を設定した場合、フィールドの欠損値の追跡が有効になります。
missing_values	[*value value ...*]	欠損データを示すデータ値を指定します。
range_missing	*boolean*	フィールドに欠損値 (空白) の範囲が定義されているかどうかを指定します。
missing_lower	*文字列*	range_missing が真 (true) の場合、欠損値範囲の下限値を指定します。
missing_upper	*文字列*	range_missing が真 (true) の場合、欠損値範囲の上限値を指定します。
null_missing	*boolean*	T を設定した場合、*ヌル値* (ソフトウェアでは $null$ として表示される未定義値) は欠損値と見なされます。
whitespace_missing	*boolean*	T を設定した場合、空白類 (スペース、タブ、および改行) だけを含む値が欠損値と見なされます。
description	*文字列*	フィールドの説明を指定します。
value_labels	*[{Value LabelString} {Value LabelString} ...]*	値のペアのためのラベルを指定します。
display_places	*integer*	フィールドが表示されるときの小数部の桁数を設定します (REAL ストレージのフィールドにのみ適用)。-1 を設定すると、ストリームのデフォルトが使用されます。
export_places	*integer*	フィールドをエクスポートするときの小数部の桁数を設定します (REAL ストレージのフィールドにのみ適用)。-1 を設定すると、ストリームのデフォルトが使用されます。
decimal_separator	DEFAULT PERIOD COMMA	フィールドの小数点記号を指定します (REAL ストレージのフィールドにのみ適用)。

表 72. *type* ノードのプロパティー *(続き)*:

type ノードのプロパティー	データ型	プロパティーの説明
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	フィールドの日付形式を設定します (DATE または TIMESTAMP ストレージのフィールドにのみ適用されます)。
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"	フィールドの日付形式を設定します (TIME または TIMESTAMP ストレージのフィールドにのみ適用されます)。
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	フィールドに数値の表示形式を設定します。
standard_places	*integer*	フィールドが標準形式で表示されるときの小数点以下の桁数を指定します。–1 を設定すると、ストリームのデフォルトが使用されます。既存の display_places スロットでもこの設定が変更されますが、現在は廃止されています。
scientific_places	*integer*	フィールドが科学系の形式で表示されるときの小数点以下の桁数を設定します。–1 を設定すると、ストリームのデフォルトが使用されます。

表 72. type ノードのプロパティー (続き):

type ノードのプロパティー	**データ型**	**プロパティーの説明**
currency_places	*integer*	フィールドが通貨の形式で表示されるときのフィールドの小数点以下の桁数を設定します。–1 を設定すると、ストリームのデフォルトが使用されます。
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	フィールドにグループ化シンボルを設定します。
column_width	*integer*	フィールドに列幅を設定します。-1 という値を指定すると、列幅は Auto に設定されます。
justify	AUTO CENTER LEFT RIGHT	フィールドに列調整を設定します。

第 12 章 グラフ作成ノードのプロパティー

グラフ作成ノードの共通のプロパティー

このセクションでは、グラフ作成ノードで使用できるプロパティーについて、共通なプロパティーとノード・タイプ固有のプロパティーも含めて説明します。

表 73. グラフ作成ノードの共通プロパティー:

グラフ作成ノードの共通プロパティー	データ型	プロパティーの説明
title	*文字列*	タイトルを指定します。例:"This is a title."
caption	*文字列*	解説を指定します。例:"This is a caption."
output_mode	画面 File	グラフ作成ノードからの出力が表示されるか、ファイルへ書き込まれるかを指定します。
output_format	BMP JPEG ファイル PNG HTML output (*.cou*)	出力のタイプを指定します。出力可能なタイプは、各ノードに応じて変化します。
full_filename	*文字列*	グラフ作成ノードから生成されたグラフの、出力先のパスとファイル名を指定します。
use_graph_size	*boolean*	下に説明する幅と高さのプロパティーを使用してグラフのサイズが明示して設定されるかどうかを制御します。画面に出力されるグラフにだけ影響します。棒グラフ・ノードには使用できません。
graph_width	*number*	use_graph_size が True の場合、グラフの幅をピクセル数で指定します。
graph_height	*number*	use_graph_size が True の場合、グラフの高さをピクセル数で指定します。

IBM Notes

オプション・フィールドの無効化: プロットのオーバーレイ・フィールドなどのオプション・フィールドは、プロパティー値を " " (空文字列) に設定することにより、オフにすることができます。

色の指定 : タイトル、解説、背景、およびラベルの色は、ハッシュ記号 (#) で始まる 16進文字列で指定することができます。

先頭の 2 桁は赤の成分を指定します。中間の 2 桁は緑の成分を指定します。末尾の 2 桁は青の成分を指定します。各桁は、0 から 9 または A から F の範囲の値になります。これらの値を組み合わせて、赤、緑、青 (RGB) を指定することができます。

注 : 色を RGB で指定する場合、フィールド・ピッカーを使用して正しい色コードを決定することができます。ピッカーを目的の色の上にかざせば、その色コードがツールヒントに表示されます。

collection ノードのプロパティー

集計棒グラフ・ノードで、他の数値フィールドの値に相対的な数値フィールドの値の棒グラフを表示します（集計棒グラフ・ノードでは、ヒストグラムに似たグラフが作成されます）。集計棒グラフは、値が時間の経過とともに変化する変数やフィールドを表示する場合に役立ちます。3 次元グラフを使用して、分布をカテゴリー別に表示するシンボル値軸を追加することもできます。

表 74. collection ノードのプロパティー:

collection ノードのプロパティー	データ型	プロパティーの説明
over_field	*field*	
over_label_auto	*boolean*	
over_label	*文字列*	
collect_field	*field*	
collect_label_auto	*boolean*	
collect_label	*文字列*	
three_D	*boolean*	
by_field	*field*	
by_label_auto	*boolean*	
by_label	*文字列*	
operation	Sum Mean Min Max SDev	
color_field	*文字列*	
panel_field	*文字列*	
animation_field	*文字列*	
range_mode	Automatic (自動) UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*boolean*	
graph_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
page_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。

distribution ノードのプロパティー

棒グラフ・ノードで、ローンの種類や性別など、シンボル値（カテゴリー）の出現頻度を表示します。通常、棒グラフ・ノードを使用してデータの不均衡を表示しますが、そのデータはモデルの作成前にバランス・ノードを使用して修正できます。

表 75. distribution ノードのプロパティー:

distribution ノードのプロパティー	データ型	プロパティーの説明
plot	SelectedFields Flags	
x_field	*field*	
color_field	*field*	オーバーレイ フィールド。
normalize	*boolean*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*boolean*	

evaluation ノードのプロパティー

評価ノードは、予測モデルの評価と比較に使用されます。評価グラフで、モデルが特定の結果をどの程度予測するかを表示します。それによって、予測値と予測の信頼度に基づいたレコードがソートされます。そして、レコードが等サイズ (**分位**) のグループに分割され、各分位のビジネスに関する基準の値が、高い方から降順で作図されます。プロットには、複数のモデルが異なる線で示されます。

表 76. evaluation ノードのプロパティー:

evaluation ノードのプロパティー	データ型	プロパティーの説明
chart_type	ゲイン レスポンス リフト プロフィット ROI ROC	
inc_baseline	*boolean*	
field_detection_method	Metadata Name	
use_fixed_cost	*boolean*	
cost_value	*number*	
cost_field	*文字列*	
use_fixed_revenue	*boolean*	
revenue_value	*number*	
revenue_field	*文字列*	
use_fixed_weight	*boolean*	
weight_value	*number*	
weight_field	*field*	

表 76. evaluation ノードのプロパティー (続き):

evaluation ノードのプロパティー	データ型	プロパティーの説明
n_tile	Quartiles (四分位) Quintles 十分位 二十分位 Percentiles (パーセンタイル) 1000-tiles	
cumulative	*フラグ*	
style	Line Point	
point_type	長方形 ドット 三角形 六角形 プラス 五角形 スター BowTie HorizontalDash VerticalDash IronCross 工場 家 大聖堂 OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle ファン	
export_data	*boolean*	
data_filename	*文字列*	
delimiter	*文字列*	
new_line	*boolean*	
inc_field_names	*boolean*	
inc_best_line	*boolean*	
inc_business_rule	*boolean*	
business_rule_condition	*文字列*	
plot_score_fields	*boolean*	
score_fields	*[フィールド 1 ... フィールド N]*	
target_field	*field*	
use_hit_condition	*boolean*	
hit_condition	*文字列*	
use_score_expression	*boolean*	
score_expression	*文字列*	
caption_auto	*boolean*	

graphboard ノードのプロパティー

グラフボード・ノードでは、単一のノードにさまざまな種類のグラフを提供しています。このノードを使用して、検証するデータ・フィールドを選択肢、選択したデータに使用できるグラフを選択できます。選択したフィールドに適さないグラフの種類は、ノードによって自動的に除外されます。

注：グラフ・タイプに有効でないプロパティーを設定した場合 (ヒストグラムに `y_field` を指定するなど)、そのプロパティーは無視されます。

表 77. graphboard ノードのプロパティー:

graphboard ノードのプロパティー	**データ型**	**プロパティーの説明**
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area (領域) ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram (ヒストグラム) Line LineChartMap LineOverlayMap Parallel Path 円グラフ PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap リボン 散布図 SPLOM 表面	グラフの種類を識別します。

表 77. *graphboard* ノードのプロパティー *(続き)*:

graphboard ノードのプロパティー	データ型	プロパティーの説明
x_field	*field*	*x* 軸のカスタム (ユーザー設定) ラベルを指定します。ラベルでのみ使用できます。
y_field	*field*	*y* 軸のカスタム (ユーザー設定) ラベルを指定します。ラベルでのみ使用できます。
z_field	*field*	3 次元グラフの一部で使用します。
color_field	*field*	ヒート・マップで使用します。
size_field	*field*	バブル・プロットで使用します。
categories_field	*field*	
values_field	*field*	
rows_field	*field*	
columns_field	*field*	
fields	*field*	
start_longitude_field	*field*	参照マップの矢印で使用します。
end_longitude_field	*field*	
start_latitude_field	*field*	
end_latitude_field	*field*	
data_key_field	*field*	さまざまなマップで使用します。
panelrow_field	*文字列*	
panelcol_field	*文字列*	
animation_field	*文字列*	
longitude_field	*field*	マップ上の座標で使用します。
latitude_field	*field*	
map_color_field	*field*	

histogram ノードのプロパティー

ヒストグラム・ノードでは、数値フィールドの値の出現頻度が示されます。多くの場合、ヒストグラム・ノードは、操作やモデルの構築前にデータを調べるために使用されます。棒グラフ・ノードと同様、ヒストグラム・ノードにより、データ内の不均衡がしばしば明らかになります。

表 78. *histogram* ノードのプロパティー:

histogram ノードのプロパティー	データ型	プロパティーの説明
field	*field*	
color_field	*field*	
panel_field	*field*	
animation_field	*field*	

表 78. *histogram* ノードのプロパティー *(続き)*:

histogram ノードのプロパティー	データ型	プロパティーの説明
range_mode	Automatic (自動) UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*boolean*	
separate_bands	*boolean*	
x_label_auto	*boolean*	
x_label	*文字列*	
y_label_auto	*boolean*	
y_label	*文字列*	
use_grid	*boolean*	
graph_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
page_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
normal_curve	*boolean*	正規分布のカーブを出力に表示するかどうかを指定します。

multiplot ノードのプロパティー

線グラフ・ノードでは、1 つの *X* フィールドに対して複数の *Y* フィールドを表示する作図が作成されます。*Y* フィールドは色付きの線で作図され、それぞれ「スタイル」フィールドを「**ライン**」に、「X モード」フィールドを「**ソート**」に設定した散布図ノードに相当します。線グラフは、複数の変数の変動を長期にわたって調査するときに役立ちます。

表 79. *multiplot* ノードのプロパティー:

multiplot ノードのプロパティー	データ型	プロパティーの説明
x_field	*field*	
y_fields	[*フィールド フィールド フィールド*]	
panel_field	*field*	
animation_field	*field*	
normalize	*boolean*	
use_overlay_expr	*boolean*	
overlay_expression	*文字列*	
records_limit	*number*	

表 79. *multiplot* ノードのプロパティー *(続き)*:

multiplot ノードのプロパティー	データ型	プロパティーの説明
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*boolean*	
x_label	*文字列*	
y_label_auto	*boolean*	
y_label	*文字列*	
use_grid	*boolean*	
graph_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
page_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。

plot ノードのプロパティー

散布図ノードで、数値フィールド間の関係が示されます。作図は、点 (散布図) または折れ線を使用して作成できます。

表 80. *plot* ノードのプロパティー:

plot ノードのプロパティー	データ型	プロパティーの説明
x_field	*field*	*x* 軸のカスタム (ユーザー設定) ラベルを指定します。ラベルでのみ使用できます。
y_field	*field*	*y* 軸のカスタム (ユーザー設定) ラベルを指定します。ラベルでのみ使用できます。
three_D	*boolean*	*y* 軸のカスタム (ユーザー設定) ラベルを指定します。3-D グラフのラベルでのみ使用できます。
z_field	*field*	
color_field	*field*	オーバーレイ フィールド。
size_field	*field*	
shape_field	*field*	
panel_field	*field*	各カテゴリー個別のグラフの作成に使用する名義型またはフラグ型フィールドを指定します。グラフは「パネル化」され、複数のグラフが 1 つの出力ウィンドウに表示されます。
animation_field	*field*	アニメーションを使用して順番に表示する一連のグラフを作成してデータ値のカテゴリーを描画する、名義型またはフラグ型フィールドを指定します。
transp_field	*field*	カテゴリーごとに異なるレベルの透過度を使用して、データ値のカテゴリーを表すフィールドを指定します。折れ線グラフでは使用できません。

表 80. plot ノードのプロパティー (続き):

plot ノードのプロパティー	データ型	プロパティーの説明
overlay_type	None 平滑化 Function	オーバーレイ関数が表示されるか、LOESS 平滑化が表示されるかを指定します。
overlay_expression	*文字列*	overlay_type が Function に設定されているときに使用される式を指定します。
style	Point Line	
point_type	長方形 ドット 三角形 六角形 プラス 五角形 スター BowTie HorizontalDash VerticalDash IronCross 工場 家 大聖堂 OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle ファン	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic (自動) UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Automatic (自動) UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic (自動) UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*boolean*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*boolean*	

表 80. *plot* ノードのプロパティー *(続き)*:

plot ノードのプロパティー	データ型	プロパティーの説明
x_label	*文字列*	
y_label_auto	*boolean*	
y_label	*文字列*	
z_label_auto	*boolean*	
z_label	*文字列*	
use_grid	*boolean*	
graph_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
page_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
use_overlay_expr	*boolean*	overlay_type の代わりに廃止される予定。

timeplot ノードのプロパティー

時系列ノードで、時系列データの 1 つ以上のセットを表示します。通常、最初に時間区分ノードを使用して *TimeLabel* フィールドを作成します。このフィールドは、*x* 軸にラベルを付けるために使用されます。

表 81. *timeplot* ノードのプロパティー:

timeplot ノードのプロパティー	データ型	プロパティーの説明
plot_series	系列 Models	
use_custom_x_field	*boolean*	
x_field	*field*	
y_fields	[*フィールド フィールド フィールド*]	
panel	*boolean*	
normalize	*boolean*	
line	*boolean*	
points	*boolean*	

表 81. *timeplot* ノードのプロパティー *(続き)*:

timeplot ノードのプロパティー	データ型	プロパティーの説明
point_type	長方形 ドット 三角形 六角形 プラス 五角形 スター BowTie HorizontalDash VerticalDash IronCross 工場 家 大聖堂 OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle ファン	
smoother	*boolean*	panel を True に設定した場合にのみ、平滑化をプロットに追加できます。
use_records_limit	*boolean*	
records_limit	*integer*	
symbol_size	*number*	マーカー・サイズを指定します。
panel_layout	Horizontal Vertical	

web ノードのプロパティー

Web グラフ・ノードで、複数のシンボル値 (カテゴリー) フィールドの値の関係の強さが示されます。このグラフでは、接続の強さを示すためにさまざまな幅の線が使用されます。Web グラフ・ノードを使用して、例えば、E コマース・サイトで購入されたさまざまな商品の関係を調査できます。

表 82. *web* ノードのプロパティー:

web ノードのプロパティー	データ型	プロパティーの説明
use_directed_web	*boolean*	
fields	[*フィールド フィールド フィールド*]	
to_field	*field*	
from_fields	[*フィールド フィールド フィールド*]	
true_flags_only	*boolean*	
line_values	Absolute OverallPct PctLarger PctSmaller	

表 82. *web* ノードのプロパティー *(続き)*:

web ノードのプロパティー	データ型	プロパティーの説明
strong_links_heavier	*boolean*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*number*	
links_above	*number*	
discard_links_min	*boolean*	
links_min_records	*number*	
discard_links_max	*boolean*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*boolean*	
web_display	Circular Network Directed Grid	
graph_background	*色*	標準のグラフ色は、このセクションの最初に説明されています。
symbol_size	*number*	マーカー・サイズを指定します。

第 13 章 モデル作成ノードのプロパティー

モデル作成ノードの共通プロパティー

次のプロパティーは、複数またはすべてのデータベース・モデル作成ードに共通です。個別のモデル作成ノードに関しては、必要に応じてドキュメント内に例外を記載しています。

表 83. モデル作成ノードの共通プロパティー:

プロパティー	値	プロパティーの説明
custom_fields	*boolean*	真 (true) の場合は、現在のノードのターゲット、入力、その他フィールドなどを指定することができます。偽 (false) の場合は、上流のデータ型ノードから現在の設定が使用されます。
ターゲット または targets	*field* or [*field1 ... fieldN*]	モデルのタイプによって、単一の対象フィールドまたは複数の対象フィールドを指定します。
inputs	[*field1 ... fieldN*]	モデルで使用される入力または予測変数フィールド。
partition	*field*	
use_partitioned_data	*boolean*	区分フィールドが定義される場合、このオプションは学習データ区分からのデータのみがモデル構築に使用されるようにします。
use_split_data	*boolean*	
splits	*[フィールド 1 ... フィールド N]*	分割モデル作成に使用する、フィールドを選択します。use_split_data が True に設定されている場合にのみ有効です。
use_frequency	*boolean*	各モデル・タイプで言及するとおり、重みフィールドおよび度数フィールドが特定のモデルで使用されます。
frequency_field	*field*	
use_weight	*boolean*	
weight_field	*field*	
use_model_name	*boolean*	
model_name	*文字列*	ユーザーが指定する新規モデル名。
mode	Simple (単純) Expert	

anomalydetection ノードのプロパティー

異常値検出ノードで、「正常な」データのパターンに合致しない異常ケースや外れ値を識別します。このノードで、外れ値が既知のパターンに当てはまらなかったり、何を探しているのかはっきりしなかったりする場合でも、外れ値を識別できます。

表 84. anomalydetection ノードのプロパティー:

anomalydetection ノードのプロパティー	値	プロパティーの説明
inputs	*[フィールド 1 ... フィールド N]*	異常値検出モデルは、指定の入力フィールドに基づいてレコードをスクリーニングします。ターゲット・フィールドは使用しません。重みフィールドおよび度数フィールドも使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	レコードに異常としてフラグを設定するための、分割値を決めるのに使用される方法を指定します。
index_level	*number*	異常としてフラグを設定するための最小分割値を指定します。
percent_records	*number*	学習データ内のレコードの割合 (%) に基づいてレコードにフラグを設定するための、閾値を設定します。
num_records	*number*	学習データ内のレコードの数に基づいてレコードにフラグを設定するための、閾値を設定します。
num_fields	*integer*	各異常レコードに報告するフィールド数。
impute_missing_values	*boolean*	
adjustment_coeff	*number*	距離の計算時,É連続型とカテゴリー・フィールド間に指定された関連の重みのバランスをとるために使用される値。
peer_group_num_auto	*boolean*	ピア・グループ数を自動的に計算します。
min_num_peer_groups	*integer*	peer_group_num_auto が True に設定されている場合に使用されるピア・グループの最小数を指定します。
max_num_per_groups	*integer*	ピア・グループの最大数を指定します。
num_peer_groups	*integer*	peer_group_num_auto が False に設定されている場合に使用されるピア・グループの数を指定します。
noise_level	*number*	クラスタリング中の外れ値の処理方法を決定します。0 から 0.5 までの値を指定してください。

表 84. *anomalydetection* ノードのプロパティー *(続き)*:

anomalydetection ノードのプロパティー	値	プロパティーの説明
noise_ratio	*number*	ノイズのバッファリングに使用されるコンポーネントに割り当てられる、メモリーの一•”を指定します。0 から 0.5 までの値を指定してください。

apriori ノードのプロパティー

Apriori ノードで、データからルール・セットを抽出し、情報内容が最も充実したルールを引き出します。Apriori には、5 種類のルール選択方法があり、高度なインデックス作成方法を使用して、大きなデータ・セットが効率的に処理されます。大きな問題の場合は、一般に、Apriori の方が高速に学習できます。保持できるルール数に特に制限はありません。また、最大 32 の前提条件を持つルールを処理できます。Apriori では、入力フィールドと出力フィールドのすべてがカテゴリーであることが必要ですが、この種類のデータに合わせて最適化されているので、よりよいパフォーマンスを実現します。

表 85. *apriori* ノードのプロパティー:

apriori ノードのプロパティー	値	プロパティーの説明
consequents	*field*	Apriori モデルは標準的な対象フィールドおよび入力フィールドの結果と条件を使用します。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
antecedents	[*field1 ... fieldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*boolean*	
optimize	Speed Memory	
use_transactional_data	*boolean*	
contiguous	*boolean*	
id_field	*文字列*	
content_field	*文字列*	
mode	Simple (単純) Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	

表 85. *apriori* ノードのプロパティー *(続き)*:

apriori ノードのプロパティー	値	プロパティーの説明
optimize	Speed Memory	モデル作成が速度とメモリーのどちらにより最適化されるかを指定します。

autoclassifier ノードのプロパティー

自動分類ノードは、2種類の結果 (yes/no、 churn/don't churn など) を生じる多くの異なるモデルを作成および比較し、与えられた分析への最善のアプローチを選ぶことができるようになります。多くのモデル作成アルゴリズムに対応し、希望する方法、各特定のオプション、そして結果を比較するための基準を選択することができます。このノードで、指定されたオプションに基づいてモデルのセットが生成され、指定された基準に基づいて最善の候補がランク付けされます。

表 86. *autoclassifier* ノードのプロパティー:

autoclassifier ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	フラグ型対照の場合、自動分類ノードは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。重みフィールドおよび度数フィールドも指定することができます。詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
ranking_measure	Accuracy Area_under_curve 利益 Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*integer*	モデル・ナゲットに含まれるモデルの数。1 と 100の間の整数を指定します。
calculate_variable_importance	*boolean*	
enable_accuracy_limit	*boolean*	
accuracy_limit	*integer*	0 と 100 の間の整数です。
enable_ area_under_curve _limit	*boolean*	
area_under_curve_limit	*number*	0.0 と 1.0 の間の実数。
enable_profit_limit	*boolean*	
profit_limit	*number*	1 以上の整数。
enable_lift_limit	*boolean*	
lift_limit	*number*	1.0 を超える実数。
enable_number_of_variables_limit	*boolean*	
number_of_variables_limit	*number*	1 以上の整数。

表 86. *autoclassifier* ノードのプロパティー *(続き)*:

autoclassifier ノードのプロパティー	値	プロパティーの説明
use_fixed_cost	*boolean*	
fixed_cost	*number*	0.0 を超える実数。
variable_cost	*field*	
use_fixed_revenue	*boolean*	
fixed_revenue	*number*	0.0 を超える実数。
variable_revenue	*field*	
use_fixed_weight	*boolean*	
fixed_weight	*number*	0.0 を超える実数。
variable_weight	*field*	
lift_percentile	*number*	0 と 100 の間の整数です。
enable_model_build_time_limit	*boolean*	
model_build_time_limit	*number*	個々のモデルのそれぞれを構築するためにかかる時間を制限するために分数を設定する整数。
enable_stop_after_time_limit	*boolean*	
stop_after_time_limit	*number*	自動分類の実行のための全体経過時間を制限するために時間数を設定する実数。
enable_stop_after_valid_model_produced	*boolean*	
use_costs	*boolean*	
<algorithm>	*boolean*	特定のアルゴリズムの使用の有効、無効を切り替えます。
<algorithm>.<property>	*文字列*	特定のアルゴリズムのプロパティー値を設定します。 詳しくは、トピック『アルゴリズム・プロパティーの設定』を参照してください。

アルゴリズム・プロパティーの設定

自動分類ノードのアルゴリズム名は cart、chaid、quest、c50、logreg、decisionlist、bayesnet、discriminant、svm、および knn です。

自動数値ノードのアルゴリズム名は cart、chaid、neuralnetwork、genlin、svm、regression、linear、および knn です。

自動クラスター・ノードのアルゴリズム名は、twostep、k-means、および kohonen です。

プロパティー名は、各アルゴリズムノードのために文書化されている標準です。

ピリオドなどの句読点を含むアルゴリズム・プロパティーは、一重引用符で囲む必要があります。

複数の値をプロパティーに割り当てることもできます。

注:

- true および false の値を設定するときは、小文字を使用します (False のように大文字は使用しません)。
- 自動分類ノードで特定のアルゴリズム・オプションが使用可能でない場合、または値の範囲ではなく、1 つの値だけを指定できるときは、標準の方法でノードにアクセスするときと同じ制限が、スクリプトにも適用されます。

autocluster ノードのプロパティー

自動クラスター・ノードは、同様の特性を持つレコードのグループを識別するクラスタリング・モデルを推定し、比較します。ノードは他の自動化モデル作成ノードと同じように動作し、複数の組み合わせのオプションを単一のモデル作成の実行で検証できます。モデルは、クラスター・モデルの有用性をフィルタリングおよびランク付けする基本的な指標を使用して比較し、特定のフィールドの重要度に基づいて指標を提供します。

表 87. autocluster ノードのプロパティー:

autocluster ノードのプロパティー	値	プロパティーの説明
evaluation	*field*	注 : 自動クラスター・ノードのみ。重要度の値を計算するフィールドを識別します。また、どれだけクラスターがフィールドの値を区別するか、どれだけ正確にモデルがこのフィールドを予測するかを識別するために使用することができます。
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance (重要度)	
ranking_dataset	Training Test	
summary_limit	*integer*	レポートに一覧するモデルの数。1 と 100 の間の整数を指定します。
enable_silhouette_limit	*boolean*	
silhouette_limit	*integer*	0 と 100 の間の整数です。
enable_number_less_limit	*boolean*	
number_less_limit	*number*	0.0 と 1.0 の間の実数。
enable_number_greater_limit	*boolean*	
number_greater_limit	*number*	1 以上の整数。
enable_smallest_cluster_limit	*boolean*	
smallest_cluster_units	パーセンテージ Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*integer*	1 以上の整数。
enable_largest_cluster_limit	*boolean*	
largest_cluster_units	パーセンテージ Counts	

表 87. *autocluster* ノードのプロパティー *(続き)*:

autocluster ノードのプロパティー	値	プロパティーの説明
largest_cluster_limit_percentage	*number*	
largest_cluster_limit_count	*integer*	
enable_smallest_largest_limit	*boolean*	
smallest_largest_limit	*number*	
enable_importance_limit	*boolean*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	0 と 100 の間の整数です。
importance_limit_less_than	*number*	0 と 100 の間の整数です。
<algorithm>	*boolean*	特定のアルゴリズムの使用の有効、無効を切り替えます。
<algorithm>.<property>	*文字列*	特定のアルゴリズムのプロパティー値を設定します。 詳しくは、トピック 139 ページの『アルゴリズム・プロパティーの設定』を参照してください。

autonumeric ノードのプロパティー

自動数値ノードでは、多くのさまざまな方法を使用し、連続する数値範囲の結果を求めてモデルを推定し比較します。このノードは、自動分類ノードと同じ方法で動作し、1 回のモデル作成のパスで、複数の組み合わせのオプションを使用し試すアルゴリズムを選択することができます。使用できるアルゴリズムには、ニューラル・ネットワーク、C&R Tree、CHAID、線型回帰、一般化線型回帰、サポート・ベクトル・マシン (SVM) が含まれています。モデルは、相関、相対エラー、または使用された変数の数に基づいて比較できます。

表 88. *autonumeric* ノードのプロパティー:

autonumeric ノードのプロパティー	値	プロパティーの説明
custom_fields	*boolean*	真 (True) の場合、データ型ノード設定の代わりにカスタム・フィールド設定が使用されます。
ターゲット	*field*	自動数値ノードは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。重みフィールドおよび度数フィールドも指定することができます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
inputs	*[field1 ... field2]*	
partition	*field*	
use_frequency	*boolean*	
frequency_field	*field*	
use_weight	*boolean*	
weight_field	*field*	

表 88. autonumeric ノードのプロパティー (続き):

autonumeric ノードのプロパティー	値	プロパティーの説明
use_partitioned_data	*boolean*	データ区分フィールドが定義されている場合、学習データだけがモデルの構築に使用されます。
ranking_measure	Correlation (相関) NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*integer*	モデル・ナゲットに含まれるモデルの数。1と 100の間の整数を指定します。
calculate_variable_importance	*boolean*	
enable_correlation_limit	*boolean*	
correlation_limit	*integer*	
enable_number_of_fields_limit	*boolean*	
number_of_fields_limit	*integer*	
enable_relative_error_limit	*boolean*	
relative_error_limit	*integer*	
enable_model_build_time_limit	*boolean*	
model_build_time_limit	*integer*	
enable_stop_after_time_limit	*boolean*	
stop_after_time_limit	*integer*	
stop_if_valid_model	*boolean*	
<algorithm>	*boolean*	特定のアルゴリズムの使用の有効、無効を切り替えます。
<algorithm>.<property>	*文字列*	特定のアルゴリズムのプロパティー値を設定します。 詳しくは、トピック 139 ページの『アルゴリズム・プロパティーの設定』を参照してください。

bayesnet ノードのプロパティー

ベイズ・ネットワーク・ノードを使用すると、観測された情報および記録された情報を実際の知識を組み合わせることによって確率モデルを作成し、発生の尤度を確立できます。このノードは、主に分類に使用される Tree Augmented Naïve Bayes (TAN) および Markov Blanke ネットワークに焦点を当てています。

表 89. *bayesnet* ノードのプロパティー：

bayesnet ノードのプロパティー	値	プロパティーの説明
inputs	*[フィールド 1 ... フィールド N]*	ベイズ・ネットワーク・モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。連続フィールドは自動的に分割されます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue_training_existing_model	*boolean*	
structure_type	TAN MarkovBlanket	Bayesian ネットワークを構築時に使用する構造を選択します。
use_feature_selection	*boolean*	
parameter_learning_method	Likelihood Bayes	親の値が認識されるノード間の条件付き確率テーブルを推定するために用いる方法を指定します。
mode	Expert Simple	
missing_values	*boolean*	
all_probabilities	*boolean*	
independence	Likelihood Pearson	2 つの変数のペアの観測がお互いに独立しているかどうかを評価するために用いる方法を指定します。
significance_level	*number*	独立性を判断するための分割値を指定します。
maximal_conditioning_set	*number*	独立性検定に使用する条件変数の最大数を指定します。
inputs_always_selected	*[フィールド 1 ... フィールド N]*	ベイズ・ネットワーク構築時にデータ・セットのどのフィールドを常に使用するかを指定します。 注：対象フィールドは常に選択されます。
maximum_number_inputs	*number*	ベイズ・ネットワーク構築で使用する入力フィールドの最大数を指定します。
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

buildr ノードのプロパティー

R 構築ノードを使用すると、IBM SPSS Modeler に展開されているモデル作成およびモデル・スコアリングを実行するためのカスタムの R スクリプトを入力できます。

表 90. buildr プロパティー:

buildr ノードのプロパティー	値	プロパティーの説明
build_syntax	*文字列*	モデル作成用の R スクリプト・シンタックス。
score_syntax	*文字列*	モデル・スコアリング用の R スクリプト・シンタックス。
convert_flags	StringsAndDoubles LogicalValues	フラグ型フィールドを変換するためのオプション。
convert_datetime	*boolean*	日付形式または日付/時刻形式の変数を R の日付/時刻形式に変換するためのオプション。
convert_datetime_class	POSIXct POSIXlt	日付形式または日付/時刻形式の変数のうち、どの形式の変数を変換するかを指定するためのオプション。
convert_missing	*boolean*	欠損値を R の NA 値に変換するためのオプション。
output_html	*boolean*	R モデル・ナゲットのタブにグラフを表示するためのオプション。
output_text	*boolean*	R モデル・ナゲットのタブに R コンソールのテキスト出力を書き込むためのオプション。

c50 ノードのプロパティー

C5.0 ノードは、デシジョン・ツリーとルール・セットのどちらかを構築します。このモデルは、各レベルで最大の情報の対応をもたらすフィールドに基づいてサンプルを分割します。対象フィールドは、カテゴリーでなければなりません。複数の分割を 2 つ以上のサブグループに分割できます。

表 91. c50 ノードのプロパティー:

c50 ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	C50 モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。重みフィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
output_type	DecisionTree RuleSet	
group_symbolics	*boolean*	
use_boost	*boolean*	
boost_num_trials	*number*	
use_xval	*boolean*	
xval_num_folds	*number*	

表 91. *c50 ノードのプロパティー (続き)*:

c50 ノードのプロパティー	値	プロパティーの説明
mode	Simple Expert	
favor	Accuracy Generality	精度 (Accuracy) または一般化 (Generality) を選択。
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*boolean*	
costs	*構造化*	これは構造化されたプロパティーです。
use_winnowing	*boolean*	
use_global_pruning	*boolean*	デフォルトはオン (True)。
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

carma ノードのプロパティー

CARMA モデルは、入力または対象フィールドを指定しなくても、データからルールのセットを抽出します。Apriori とは対照的に、CARMA ノードでは、前提条件サポートだけではなく、ルール・サポート (前提条件と結果の両方のサポート) を対象とした構築の設定が可能です。これは、生成されたルールをさまざまなアプリケーションで活用できることを意味します。例えば、この休暇シーズンに販売促進する項目を結果とする、商品またはサービス (前提条件) のリストを調べることができます。

表 92. *carma ノードのプロパティー*:

carma ノードのプロパティー	値	プロパティーの説明
inputs	*[フィールド 1 ... フィールド N]*	CARMA モデルは対象フィールドでなく、入力フィールドのリストを使用します。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
id_field	*field*	モデル作成の ID フィールドとして使用するフィールド。
contiguous	*boolean*	ID フィールドの ID が連続するかどうかを指定します。
use_transactional_data	*boolean*	
content_field	*field*	
min_supp	*数値 (パーセント)*	前提条件範囲(サポート) ではなく、ルール範囲に関連します。デフォルト値は 20% です。

表 92. carma ノードのプロパティー (続き):

carma ノードのプロパティー	値	プロパティーの説明
min_conf	数値 (パーセント)	デフォルト値は 20% です。
max_size	number	デフォルトは 10 です。
mode	Simple Expert	デフォルトは Simple です。
exclude_multiple	boolean	複数の結果を持つルールを除外します。デフォルトは False です。
use_pruning	boolean	デフォルトは False です。
pruning_value	number	デフォルトは 500 です。
vary_support	boolean	
estimated_transactions	integer	
rules_without_antecedents	boolean	

cart ノードのプロパティー

C&R Tree (分類と回帰ツリー) ノードは、デシジョン・ツリーを生成し、将来の観測値を予測または分類できるようにします。この方法は再帰的なデータ区分を使用して学習レコードを複数のセグメントに分割し、各ステップで不純性を最小限に抑えます。ツリーのノードが「純粋」であると考えられるのは、ノード中にあるケースの 100% が、対象フィールドのある特定のカテゴリーに分類される場合です。対象フィールドおよび入力フィールドは、数値範囲またはカテゴリー (名義型、順序型、フラグ) が使用できます。すべての分岐は 2 分割です (2 つのサブグループのみ)。

表 93. cart ノードのプロパティー:

cart ノードのプロパティー	値	プロパティーの説明
ターゲット	field	C&R Tree モデルは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。度数フィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue_training_existing_model	boolean	
objective	標準 ブースティング バギング psm	psm は非常に大きいデータ・セットに使用され、Server の接続が必要です。
model_output_type	Single InteractiveBuilder	
use_tree_directives	boolean	

表 93. cart ノードのプロパティー (続き):

cart ノードのプロパティー	値	プロパティーの説明
tree_directives	*文字列*	ツリーの成長のためのディレクティブ (式) を指定します。ディレクティブ (式) は、改行や引用符のエスケープ処理を回避するために、三重の引用符で囲むことができます。ディレクティブは、データやモデルリング・オプションの些細な変更に依存するため、他のデータセットに対しては一般化できません。
use_max_depth	Default Custom	
max_depth	*integer*	最大ツリー深さ (0 から 1000)。use_max_depth = Custom の場合にのみ使用します。
prune_tree	*boolean*	オーバーフィットしないようにツリーを剪定します。
use_std_err	*boolean*	リスクにおける最大差 (標準誤差) を使用します。
std_err_multiplier	*number*	最大差。
max_surrogates	*number*	最大代理フィールド :
use_percentage	*boolean*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*boolean*	
costs	*構造化*	構造化プロパティー。
priors	Data Equal Custom	
custom_priors	*構造化*	構造化プロパティー。
adjust_priors	*boolean*	
trails	*number*	ブーストまたはバグのコンポーネント・モデル数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	カテゴリー型対象のデフォルト結合ルール。
range_ensemble_method	Mean Median	連続型対象のデフォルト結合ルール。
large_boost	*boolean*	特に大きなデータセットのブースティングを適用します。
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	

表 93. cart ノードのプロパティー (続き):

cart ノードのプロパティー	値	プロパティーの説明
train_pct	*number*	オーバーフィット防止セット。
set_random_seed	*boolean*	結果の複製オプション。
seed	*number*	
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

chaid ノードのプロパティー

CHAID ノードはデシジョン・ツリーを生成し、カイ二乗統計値を使用して最適な分割を識別します。C&RT Tree および QUEST ノードと異なり、CHAID は、非 2 分岐ツリーを生成できます。これは、ある分岐が 3 個以上のブランチを持てることを意味します。対象フィールドおよび入力フィールドは、数値範囲 (連続型) またはカテゴリーとなります。Exhaustive CHAID は CHAID の修正版で、可能性のある分割すべてを調べることで、よりよい結果を得られますが、計算時間も長くなります。

表 94. chaid ノードのプロパティー:

chaid ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	CHAID モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。度数フィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue_training_existing_model	*boolean*	
objective	標準 ブースティング バギング psm	psm は非常に大きいデータ・セットに使用され、Server の接続が必要です。
model_output_type	Single InteractiveBuilder	
use_tree_directives	*boolean*	
tree_directives	*文字列*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*integer*	最大ツリー深さ (0 から 1000)。use_max_depth = Custom の場合にのみ使用します。
use_percentage	*boolean*	

表 94. *chaid* ノードのプロパティー *(続き)*:

chaid ノードのプロパティー	値	プロパティーの説明
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*boolean*	
costs	*構造化*	構造化プロパティー。
trails	*number*	ブーストまたはバグのコンポーネント・モデル数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	カテゴリー型対象のデフォルト結合ルール。
range_ensemble_method	Mean Median	連続型対象のデフォルト結合ルール。
large_boost	*boolean*	特に大きなデータセットのブースティングを適用します。
split_alpha	*number*	分割の有意水準：
merge_alpha	*number*	結合の有意水準。
bonferroni_adjustment	*boolean*	Bonferroni メソッドを使用して有意確率値を調整。
split_merged_categories	*boolean*	マージしたカテゴリーの再分割を許可。
chi_square	Pearson LR	カイ 2 乗統計の計算に使用される方法 (Pearson または尤度比)
epsilon	*number*	期待されるセル度数の最小変化。
max_iterations	*number*	収束のための最大反復回数。
set_random_seed	*integer*	
seed	*number*	
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*integer*	

coxreg ノードのプロパティー

Cox 回帰ノードを使用すると、打ち切りレコードの存在下でイベントまでの時間のデータの生存モデルを構築します。モデルは、対象のイベントが入力変数の指定の値で指定の時間 (*t*) に発生する確率を予測する生存関数を作成します。

表 95. *coxreg* ノードのプロパティー:

coxreg ノードのプロパティー	値	プロパティーの説明
survival_time	*field*	Cox回帰モデルは 生存時間のある 1 つのフィールドを使用します。
ターゲット	*field*	Cox 回帰モデルは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
method	Enter Stepwise BackwardsStepwise	
groups	*field*	
model_type	MainEffects Custom	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*boolean*	

表 95. *coxreg* ノードのプロパティー *(続き)*:

coxreg ノードのプロパティー	値	プロパティーの説明
ci_value	90 95 99	
correlation	*boolean*	
display_baseline	*boolean*	
survival	*boolean*	
hazard	*boolean*	
log_minus_log	*boolean*	
one_minus_survival	*boolean*	
separate_line	*field*	
value	*数値型* または *文字列*	フィールドに対して値の指定がない場合、デフォルト・オプションの「Mean」をそのフィールドで使用します。

decisionlist ノードのプロパティー

デシジョン・リスト・ノードは、母集団に関連する与えられた 2 値の結果の高いもしくは低い尤度を示すサブグループまたはセグメントを識別します。例えば、離れる可能性の少ないもしくはキャンペーンに好意的に答える可能性のある顧客を探すことができます。顧客区分を追加し、結果を比較するために他のモデルを並べて表示することによって、ビジネスに関する知識をモデルに導入することができます。デシジョン・リスト・モデルは、ルールのリストから構成され、各ルールには条件と結果が含まれます。ルールは順番に適用され、一致する最初のルールで、結果が決まります。

表 96. *decisionlist* ノードのプロパティー:

decisionlist ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	ディシジョン・リスト・モデルは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。度数フィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
model_output_type	モデル InteractiveBuilder	
search_direction	Up Down	セグメントの検索に関連します。Up は、高い確率の検索、Down は低い確率の検索と同じです。
target_value	*文字列*	指定しない場合は、フラグには真の値が想定されます。
max_rules	*integer*	残りを除外するセグメントの最大数
min_group_size	*integer*	最小セグメント・サイズ :
min_group_size_pct	*number*	最小セグメント・サイズ (パーセントとして)。

表 96. *decisionlist* ノードのプロパティー *(続き)*:

decisionlist ノードのプロパティー	値	プロパティーの説明
confidence_level	*number*	セグメント定義に追加するためにふさわしくするために、応答の尤度を向上するために入力フィールドが持つ最小しきい値。
max_segments_per_rule	*integer*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*integer*	リストの検索幅。
max_rules_per_cycle	*integer*	セグメント ルールの検索幅。
segment_growth	*number*	
include_missing	*boolean*	
final_results_only	*boolean*	
reuse_fields	*boolean*	属性 (ルールに表示される入力フィールド) の再使用を許可します。
max_alternatives	*integer*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

discriminant ノードのプロパティー

判別分析によって、ロジスティック回帰より厳密な仮説を立てることができますが、これらの仮説が一致した場合、ロジスティック回帰分析に対する様々な代替あるいは補足になります。

表 97. *discriminant* ノードのプロパティー:

discriminant ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	判別分析 モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
method	Enter Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	

表 97. *discriminant* ノードのプロパティー *(続き)*:

discriminant ノードのプロパティー	**値**	**プロパティーの説明**
covariance_matrix	WithinGroups SeparateGroups	
means	*boolean*	「詳細出力」ダイアログ・ボックスの統計オプション
univariate_anovas	*boolean*	
box_m	*boolean*	
within_group_covariance	*boolean*	
within_groups_correlation	*boolean*	
separate_groups_covariance	*boolean*	
total_covariance	*boolean*	
fishers	*boolean*	
unstandardized	*boolean*	
casewise_results	*boolean*	「詳細出力」ダイアログ・ボックスの統計オプション
limit_to_first	*number*	デフォルト値は 10 です。
summary_table	*boolean*	
leave_one_classification	*boolean*	
combined_groups	*boolean*	
separate_groups_covariance	*boolean*	**個別グループ共分散** 行列オプション
territorial_map	*boolean*	
combined_groups	*boolean*	**結合グループ**散布図オプション
separate_groups	*boolean*	**個別グループ**散布図オプション
summary_of_steps	*boolean*	
F_pairwise	*boolean*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	
F_value_entry	*number*	デフォルト値は 3.84 です。
F_value_removal	*number*	デフォルト値は 2.71 です。
probability_entry	*number*	デフォルト値は 0.05 です。
probability_removal	*number*	デフォルト値は 0.10 です。
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

factor ノードのプロパティー

因子分析ノードには、データの複雑性を整理する強力なデータ分解手法が 2 種類あります。主成分分析 (PCA) : 入力フィールドの線型結合が検出されます。成分が互いに直交する (直角に交わる) 場合に、フィールドのセット全体の分散を把握するのに役立ちます。因子分析 : 一連の観測フィールド内の相関パターンを説明する基本因子が識別されます。どちらの手法でも、元のフィールド・セットの情報を効果的に要約する少数の派生フィールドの検出が目標です。

表 98. factor ノードのプロパティー :

factor ノードのプロパティー	値	プロパティーの説明
inputs	[*field1 ... fieldN*]	主成分分析/因子モデルは対象フィールドでなく、入力フィールドのリストを使用します。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
method	PC ULS GLS ML PAF アルファ Image	
mode	Simple (単純) Expert	
max_iterations	*number*	
complete_records	*boolean*	
matrix	Correlation (相関) Covariance (共分散)	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None バリマックス DirectOblimin エカマックス Quartimax Promax	
delta	*number*	rotation で DirectOblimin を選択した場合、delta の値を指定できる。 値を指定しない場合は、delta のデフォルト値を使用。
kappa	*number*	rotation で Promax を選択した場合、kappa の値を指定できる。 値を指定しない場合は、kappa のデフォルト値を使用。

表 98. *factor* ノードのプロパティー *(続き)*:

factor ノードのプロパティー	値	プロパティーの説明
sort_values	*boolean*	
hide_values	*boolean*	
hide_below	*number*	

featureselection ノードのプロパティー

フィールド選択ノードで、（欠損値の割合などの）諸基準に基づいて入力フィールドをスクリーニングして削除にかけ、指定した目標に相対的な残りの入力フィールドの重要度をランク付けします。例えば、数百の潜在的入力フィールドを含むデータセットがあるとして、患者予後のモデリングにはどれが役に立つのでしょう?

表 99. *featureselection* ノードのプロパティー:

featureselection ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	フィールド選択モデルは指定対象に関連した予測フィールドをランク付けします。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
screen_single_category	*boolean*	True の場合、総レコード数に比べ同じカテゴリーに多くかたよったレコードを持つフィールドを選別します。
max_single_category	*number*	screen_single_category が True の場合に使用される閾値を指定します。
screen_missing_values	*boolean*	True の場合、レコードの総数のパーセントで表すレコード数になるまで、多すぎる欠損値フィールドをスクリーニング(選別)します。
max_missing_values	*number*	
screen_num_categories	*boolean*	True の場合、レコードの総数に対して多すぎるカテゴリーを減らす目的で、フィールドをスクリーニング(選別)します。
max_num_categories	*number*	
screen_std_dev	*boolean*	True の場合、指定された最小値以下の標準偏差で、フィールドをスクリーニング (選別) します。
min_std_dev	*number*	
screen_coeff_of_var	*boolean*	True の場合、指定された最小値以下の分散係数で、フィールドをスクリーニング (選別) します。
min_coeff_of_var	*number*	

表 99. *featureselection* ノードのプロパティー *(続き)*:

featureselection ノードのプロパティー	値	プロパティーの説明
criteria	Pearson Likelihood CramersV Lambda (ラムダ)	カテゴリー対象に対するカテゴリー予測値のランク付けのときに、重要な値が基準とする測定単位を指定します。
unimportant_below	*number*	重要、境界、非重要として変数をランク付けするときに使用される閾値 *p* を指定します。0.0 から 1.0 の値を指定します。
important_above	*number*	0.0 から 1.0 の値を指定します。
unimportant_label	*文字列*	非重要ランクのラベルを指定します。
marginal_label	*文字列*	
important_label	*文字列*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*boolean*	selection_mode が ImportanceLevel に設定されているときに、重要なフィールドを選択するかどうかを指定します。
select_marginal	*boolean*	selection_mode が ImportanceLevel に設定されているときに、境界フィールドを選択するかどうかを指定します。
select_unimportant	*boolean*	selection_mode が ImportanceLevel に設定されているときに、重要でないフィールドを選択するかどうかを指定します。
importance_value	*number*	selection_mode が ImportanceValue に設定されているときに、使用する分割値を指定します。0 から 100 の値を指定します。
top_n	*integer*	selection_mode が TopN に設定されているときに、使用する分割値を指定します。0 から 1000 の値を指定します。

genlin ノードのプロパティー

一般化線型モデルは、指定したリンク関数によって従属変数が因子および共変量と線型関係になるよう、一般線型モデルを拡張したものです。さらにこのモデルでは、非正規分布の従属変数を使用することができます。線型回帰、ロジスティック回帰、カウント・データに関するログ線型モデル、そして区間打切り生存モデルなど、統計モデルの機能性が数多く含まれています。

表 100. genlin ノードのプロパティー:

genlin ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	一般化線型モデルは、名義型またはフラグ型の 1 つの対象フィールドおよび 1 つ以上の入力フィールドが必要です。重みフィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
use_weight	*boolean*	
weight_field	*field*	フィールドのデータ型は連続型だけです。
target_represents_trials	*boolean*	
trials_type	Variable FixedValue	
trials_field	*field*	フィールドのデータ型はフラグ型または順序型です。
trials_number	*number*	デフォルト値は 10 です。
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*field*	フィールドのデータ型は連続型だけです。
offset_value	*number*	Must be a real number.
base_category	Last First	
include_intercept	*boolean*	
mode	Simple Expert	
distribution (分布)	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: 逆ガウス。 NEGBIN: 負の 2 項分布。
negbin_para_type	Specify 推定値	
negbin_parameter	*number*	デフォルト値は 1 で、負でない実数を含む必要があります。
tweedie_parameter	*number*	

表 100. genlin ノードのプロパティー (続き):

genlin ノードのプロパティー	値	プロパティーの説明
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: 補ログ・マイナス・ログ。 LOGC: 補対数。 NEGBIN: 負の 2 項分布。 NLOGLOG: 負ログ・マイナス・ログ。 CUMCAUCHIT: 累積コーチット。 CUMCLOGLOG: 累積補ログ マイナス・ログ。 CUMLOGIT: 累積ロジット。 CUMNLOGLOG: 累積負ログ・マイナス・ログ。 CUMPROBIT: 累積プロビット。
power	*number*	値は 0 でない実数である必要があります。
method	ハイブリッド Fisher NewtonRaphson	
max_fisher_iterations	*number*	デフォルト値は 1 です。正の整数値だけが使用できます。
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
scale_value	*number*	デフォルト値は 1 です。0 を超える必要があります。
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*number*	デフォルト値は 100 です。0 以上の整数だけを使用できます。
max_step_halving	*number*	デフォルト値は 5 です。正の整数値だけが使用できます。
check_separation	*boolean*	
start_iteration	*number*	デフォルト値は 20 です。正の整数値だけが使用できます。
estimates_change	*boolean*	
estimates_change_min	*number*	デフォルト値は 1E-006 です。正の数値だけが使用できます。
estimates_change_type	Absolute Relative	
loglikelihood_change	*boolean*	
loglikelihood_change_min	*number*	正の数値だけが使用できます。
loglikelihood_change_type	Absolute Relative	

表 100. genlin ノードのプロパティー (続き):

genlin ノードのプロパティー	値	プロパティーの説明
hessian_convergence	*boolean*	
hessian_convergence_min	*number*	正の数値だけが使用できます。
hessian_convergence_type	Absolute Relative	
case_summary	*boolean*	
contrast_matrices	*boolean*	
descriptive_statistics	*boolean*	
estimable_functions	*boolean*	
model_info	*boolean*	
iteration_history	*boolean*	
goodness_of_fit	*boolean*	
print_interval	*number*	デフォルト値は 1 です。正の整数である必要があります。
model_summary	*boolean*	
lagrange_multiplier	*boolean*	
parameter_estimates	*boolean*	
include_exponential	*boolean*	
covariance_estimates	*boolean*	
correlation_estimates	*boolean*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*number*	デフォルト値は 0.0001 です。
confidence_interval	*number*	デフォルト値は 95 です。
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

表 100. genlin ノードのプロパティー (続き):

genlin ノードのプロパティー	値	プロパティーの説明
adjusted_propensity_partition	Test Validation	

glmm ノードのプロパティー

一般化線型混合モデル (GLMN) は線型モデルを拡張したため、対象が非正規分布となる場合があり、指定されたリンク関数を介して因子および共変量に線形に関連し、観測が相関できるようになりました。一般化線型混合モデルは、単純な線型回帰から正規化されていない時系列データの複雑なマルチレベル・モデルまで、さまざまなモデルをカバーします。

表 101. glmm ノードのプロパティー:

glmm ノードのプロパティー	値	プロパティーの説明
residual_subject_spec	*構造化*	指定したカテゴリー型フィールドの組み合わせにより、データ・セット内の被験者が一意に定義されることが必要です。
repeated_measures	*構造化*	反復する観察の特定に使用されるフィールド。
residual_group_spec	[*field1 ... fieldN*]	反復効果共変量パラメーターの独立セットを定義するフィールド。
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	残差の共変量構造を指定します。
custom_target	*boolean*	上流のノードで定義された対象を使用するか (false) または target_field によって指定されたカスタム対象を使用するか (true) を定義します。
target_field	*field*	custom_target が true の場合対象として使用するフィールド。
use_trials	*boolean*	試行回数を指定する追加フィールド又は値を、対象フィールドが一連の試行が発生する様々なイベントである場合に使用するかどうかを示します。デフォルトは false です。
use_field_or_value	フィールド Value	フィールドまたは値を使用して試行回数を指定するかどうかを示します。
trials_field	*field*	試行回数の指定に使用するフィールド。
trials_value	*integer*	試行回数の指定に使用する値。指定する場合、最小値は 1 です。

表 101. glmm ノードのプロパティー (続き):

glmm ノードのプロパティー	値	プロパティーの説明
use_custom_target_reference	*boolean*	カスタム参照カテゴリーをカテゴリー型対象に使用するかどうかを示します。デフォルトは false です。
target_reference_value	*文字列*	use_custom_target_reference が true の場合使用する参照カテゴリー。
dist_link_combination	Nominal (名義) Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	対象の値の分布に関する一般モデル。Custom を選択して、target_distribution で提供されたリストから分布を指定します。
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson (ポワソン))	dist_link_combination が Custom の場合の対象の値の分布。
link_function_type	IDENTITY LOGC LOG CLOGLOG LOGIT NLOGLOG PROBIT POWER CAUCHIT	対象値を予測値に関連付けるリンク関数。target_distribution が Binomial の場合、リストされているどのリンク関数でも使用できます。 target_distribution が Multinomial の場合、 CLOGLOG、CAUCHIT、LOGIT、 NLOGLOG、または PROBIT を使用できます。 target_distribution が Binomial 以外および Multinomial 以外の場合、 IDENTITY、LOG、または POWER を使用できます。
link_function_param	*number*	使用するリンク関数パラメーター値。normal_link_function または link_function_type が POWER の場合にのみ使用できます。
use_predefined_inputs	*boolean*	固定効果フィールドを入力フィールドとして上流で定義されたフィールドとするか (true) fixed_effects_list のフィールドとするか (false) を指定します。デフォルトは false です。
fixed_effects_list	*構造化*	use_predefined_inputs が false の場合、固定効果フィールドとして使用する入力フィールドを指定します。
use_intercept	*boolean*	true (デフォルト) の場合、モデルに定数項を含みます。
random_effects_list	*構造化*	ランダム効果として指定するフィールドのリスト。

表 101. glmm ノードのプロパティー (続き):

glmm ノードのプロパティー	値	プロパティーの説明
regression_weight_field	*field*	分析の重みフィールドとして使用するフィールド。
use_offset	None offset_value offset_field	オフセットを指定する方法を示します。値 None は、オフセットが使用されないことを意味します。
offset_value	*number*	use_offset が offset_value の場合オフセットに使用する値。
offset_field	*field*	use_offset が offset_field の場合オフセット値に使用する値。
target_category_order	Ascending Descending Data	カテゴリー型対象のソート順。値 Data は、データ内のソート順を使用するよう指定します。デフォルトは Ascending です。
inputs_category_order	Ascending Descending Data	Sorting order for categorical predictors. 値 Data は、データ内のソート順を使用するよう指定します。デフォルトは Ascending です。
max_iterations	*integer*	アルゴリズムで実行される反復の最大回数です。負の数ではない整数。デフォルト値は 100 です。
confidence_level	*integer*	モデル係数の区間推定の計算に使用する確信度。 負の数ではない整数。最小値は 100、デフォルト値は 95 です。
degrees_of_freedom_method	Fixed Varied	自由度が有意性検定に計算される方法を指定します。
test_fixed_effects_coeffecients	モデル Robust	パラメーター推定共変量マトリックスを計算する方法。
use_p_converge	*boolean*	パラメーター収束のオプション。
p_converge	*数値*	空白または任意の正の値。
p_converge_type	絶対値 Relative	
use_l_converge	*boolean*	対数尤度収束のオプション。
l_converge	*数値*	空白または任意の正の値。
l_converge_type	絶対値 Relative	
use_h_converge	*boolean*	Hessian 収束のオプション。
h_converge	*数値*	空白または任意の正の値。
h_converge_type	絶対値 Relative	
max_fisher_steps	*整数*	
singularity_tolerance	*数値*	
use_model_name	*boolean*	モデルのカスタム名を使用するか (true) システムによって生成された名前を使用するか (false) を指定します。デフォルトは false です。

表 101. glmm ノードのプロパティー (続き):

glmm ノードのプロパティー	値	プロパティーの説明
model_name	*文字列*	use_model_name が true のときに、使用するモデルを指定します。
confidence	onProbability onIncrease	スコアリングの確信度を計算する基準 (最も高い予測確率、または最も高い予測確率と 2 番目に高い予測確率との差)。
score_category_probabilities	*boolean*	true の場合、カテゴリー型対象の予測確率を生成します。デフォルトは false です。
max_categories	*integer*	score_category_probabilities が true のときに、使用するカテゴリーの最大数を指定します。
score_propensity	*boolean*	true の場合、フィールドの「true」の結果の確率を示すフラグ型対象フィールドの傾向スコアを生成します。
emeans	*structure*	固定効果リストの各カテゴリー型フィールドについて、推定周辺平均を生成するかどうかを指定します。
covariance_list	*structure*	固定効果リストの各カテゴリー型フィールドについて、推定周辺平均を計算する場合に平均値を使用するかカスタム値を使用するかを指定します。
mean_scale	Original Transformed (変換)	対象の元の尺度に基づいて (デフォルト)、またはリンク関数変換に基づいて推定周辺平均を計算するかどうかを指定します。
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	複数の対比で仮定検定を実行する場合に使用する調整方法。

kmeans ノードのプロパティー

K-Means ノードで、データ・セットが異なるグループ (つまりクラスター) へ、クラスタリングされます。この方法で、固定数のクラスターを定義し、クラスターにレコードを繰り返し割り当てて、これ以上調整してもモデルが改善されなくなるまで、クラスターの中心を調整します。*K*-means では、結果を予測するのではなく、入力フィールドのセット内のパターンを明らかにするために、「非監視学習」として知られるプロセスが使用されます。

表 102. kmeans ノードのプロパティー:

kmeans ノードのプロパティー	値	プロパティーの説明
inputs	[*field1 ... fieldN*]	K-means モデルは入力フィールドのセットでクラスター分析を行いますが、対象フィールドは使用しません。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。

表 102. kmeans ノードのプロパティー (続き):

kmeans ノードのプロパティー	値	プロパティーの説明
num_clusters	*number*	
gen_distance	*boolean*	
cluster_label	String 数値	
label_prefix	*文字列*	
mode	Simple (単純) Expert	
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	
optimize	Speed Memory	モデル作成が速度とメモリーのどちらにより最適化されるかを指定します。

knn ノードのプロパティー

k が整数である場合、*k* 最近隣 (KNN) ノードは、新しいケースを、予測領域の新しいケースに最も近い *k* 個のオブジェクトのカテゴリーまたは値と関連付けます。類似したケースはお互いに近く、類似していないケースはお互いに離れています。

表 103. knn ノードのプロパティー:

knn ノードのプロパティー	値	プロパティーの説明
analysis	PredictTarget IdentifyNeighbors	
objective	バランス Speed Accuracy Custom	
normalize_ranges	*boolean*	
use_case_labels	*boolean*	次のオプションを有効化するチェック・ボックス。
case_labels_field	*field*	
identify_focal_cases	*boolean*	次のオプションを有効化するチェック・ボックス。
focal_cases_field	*field*	
automatic_k_selection	*boolean*	
fixed_k	*integer*	automatic_k_selectio が False の場合にのみ有効です。
minimum_k	*integer*	automatic_k_selectio が True の場合のみ使用できます。
maximum_k	*integer*	

表 103. knn ノードのプロパティー (続き):

knn ノードのプロパティー	値	プロパティーの説明
distance_computation	ユークリッド CityBlock	
weight_by_importance	*boolean*	
range_predictions	Mean Median	
perform_feature_selection	*boolean*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*boolean*	
number_to_select	*integer*	
minimum_change	*number*	
validation_fold_assign_by_field	*boolean*	
number_of_folds	*integer*	validation_fold_assign_by_field が False の場合にのみ有効です。
set_random_seed	*boolean*	
random_seed	*number*	
folds_field	*field*	validation_fold_assign_by_field が True の場合にのみ有効です。
all_probabilities	*boolean*	
save_distances	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

kohonen ノードのプロパティー

Kohonen ノードは、ニューラル・ネットワークの一種であり、データ・セットをクラスター化して異なるグループを形成する目的で使用できます。ネットワークの学習が完了すると、類似のレコードは出力マップで互い近くに表示され、違いの大きいレコードほど離れたところに表示されます。強度の高いユニットを識別するために生成されたモデル内で、各ユニットが獲得した観察の数値を調べることができます。これは、適切なクラスター数についてのヒントになる場合があります。

表 104. kohonen ノードのプロパティー:

kohonen ノードのプロパティー	値	プロパティーの説明
inputs	[*field1 ... fieldN*]	Kohonen モデルは対象フィールドでなく、入力フィールドのリストを使用します。度数フィールドおよび重みフィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue	*boolean*	

表 104. *kohonen* ノードのプロパティー *(続き)*:

kohonen ノードのプロパティー	値	プロパティーの説明
show_feedback	*boolean*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	モデル作成が速度とメモリーのどちらにより最適化されるかを指定します。
cluster_label	*boolean*	
mode	Simple (単純) Expert	
width	*number*	
length	*number*	
decay_style	Linear Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

linear ノードのプロパティー

線型回帰モデルは、対象と 1 つまたは複数の予測値との線型の関係に基づいて連続型対象を予測します。

表 105. *linear* ノードのプロパティー:

linear ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	1 つの対象フィールドを指定します。
inputs	[*field1 ... fieldN*]	モデルで使用される入力または入力または予測変数フィールド。
continue_training_existing_model	*boolean*	
objective	標準 バギング ブースティング psm	psm は非常に大きいデータ・セットに使用され、Server の接続が必要です。
use_auto_data_preparation	*boolean*	
confidence_level	*number*	

表 105. linear ノードのプロパティー (続き):

linear ノードのプロパティー	値	プロパティーの説明
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*boolean*	
max_effects	*number*	
use_max_steps	*boolean*	
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
use_random_seed	*boolean*	
random_seed	*number*	
use_custom_model_name	*boolean*	
custom_model_name	*文字列*	
use_custom_name	*boolean*	
custom_name	*文字列*	
tooltip	*文字列*	
keywords	*文字列*	
annotation	*文字列*	

logreg ノードのプロパティー

ロジスティック回帰は、入力フィールドの値に基づいてレコードを分類する統計手法です。線型回帰と似ていますが、数値範囲ではなくカテゴリー対象フィールドを使用します。

表 106. logreg ノードのプロパティー：

logreg ノードのプロパティー	**値**	**プロパティーの説明**
ターゲット	*field*	ロジスティック回帰モデルは 1 つの対象フィールドおよび 1 つ以上の入力フィールドを使用します。度数フィールドおよび重みフィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
logistic_procedure	Binomial Multinomial	
include_constant	*boolean*	
mode	Simple (単純) Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	
model_type	MainEffects FullFactorial Custom	モデル タイプとして FullFactorial が指定されている場合、ステップ手法が指定されたとしても、実行されません。その代わりに、強制投入法 (Enter) が使用されます。 モデル タイプに Custom が設定されてもユーザー設定フィールド (custom fields) が指定されていない場合は、主効果モデルが構築されます。
custom_terms	[*{BP Sex}{BP}{Age}*]	
multinomial_base_category	*文字列*	参照カテゴリーの決定方法を指定します。
binomial_categorical_input	*文字列*	
binomial_input_contrast	インジケーター Simple (単純) Difference Helmert Repeated Polynomial Deviation	コントラストを決定する方法を指定するカテゴリー入力用のキー・プロパティー。
binomial_input_category	First Last	参照カテゴリーを決定する方法を指定するカテゴリー入力用のキー・プロパティー。
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*boolean*	

表 106. *logreg* ノードのプロパティー *(続き)*:

logreg ノードのプロパティー	値	プロパティーの説明
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	*boolean*	
max_terms	*number*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll 包含関係 None	
max_iterations	*number*	
max_steps	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	*boolean*	
history_steps	*number*	
summary	*boolean*	
likelihood_ratio	*boolean*	
asymptotic_correlation	*boolean*	
goodness_fit	*boolean*	
parameters	*boolean*	
confidence_interval	*number*	

表 106. logreg ノードのプロパティー (続き):

logreg ノードのプロパティー	値	プロパティーの説明
asymptotic_covariance	*boolean*	
classification_table	*boolean*	
stepwise_summary	*boolean*	
info_criteria	*boolean*	
monotonicity_measures	*boolean*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*boolean*	
binomial_parameters	*boolean*	
binomial_iteration_history	*boolean*	
binomial_classification_plots	*boolean*	
binomial_ci_enable	*boolean*	
binomial_ci	*number*	
binomial_residual	outliers all	
binomial_residual_enable	*boolean*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	

neuralnet ノードのプロパティー

注意: このリリースでは、機能が拡張された新しいバージョンのニューラル・ネットワーク・モデル作成ノードを使用することができます。新しいバージョンについては、次のセクションで説明します (*neuralnetwork*)。旧バージョンでモデルを作成およびスコアリングできますが、新しいバージョンを使用するようスクリプトを更新することをお勧めします。参考として、旧バージョンの詳細を以下に示します。

表 107. neuralnet ノードのプロパティー:

neuralnet ノードのプロパティー	値	プロパティーの説明
targets	[*field1 ... fieldN*]	ニューラル・ノードには、 1 つ以上の対象フィールドと 1 つ以上の入力フィールドが必要です。度数および重みフィールドは無視されます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。

表 107. *neuralnet* ノードのプロパティー *(続き)*:

neuralnet ノードのプロパティー	値	プロパティーの説明
method	高速 Dynamic Multiple 剪定 ExhaustivePrune RBFN	
prevent_overtrain	*boolean*	
train_pct	*number*	
set_random_seed	*boolean*	
random_seed	*number*	
mode	Simple (単純) Expert	
stop_on	Default Accuracy サイクル Time	停止モード。
accuracy	*number*	停止精度。
cycles	*number*	学習サイクル。
time	*number*	学習時間 (分)。
continue	*boolean*	
show_feedback	*boolean*	
binary_encode	*boolean*	
use_last_model	*boolean*	
gen_logfile	*boolean*	
logfile_name	*文字列*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*文字列*	
m_non_pyramids	*boolean*	
m_persistence	*number*	

表 107. neuralnet ノードのプロパティー (続き):

neuralnet ノードのプロパティー	値	プロパティーの説明
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*boolean*	
r_alpha	*number*	
r_eta	*number*	
optimize	Speed Memory	モデル作成が速度とメモリーのどちらにより最適化されるかを指定します。
calculate_variable_importance	*boolean*	注 : 前回のリリースで使用した sensitivity_analysis プロパティーは、このプロパティーにより廃止されます。古いプロパティーはまだサポートされますが、calculate_variable_importance をお勧めします。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

neuralnetwork ノードのプロパティー

ニューラル・ネットワーク・ノードは、人間の脳が情報を処理する方法を単純化したモデルを使用します。ニューラル・ネットワーク・ノードは、連係する多数の単純な処理単位をシミュレートします。処理単位は、ニューロンを抽象化したものと表現できます。ニューラル・ネットワークは強力な一般関数推定法であり、学習させたり、適用するには、最低限の統計学および数学の知識しか必要ありません。

表 108. neuralnetwork ノードのプロパティー:

neuralnetwork ノードのプロパティー	値	プロパティーの説明
targets	[*field1 ... fieldN*]	対象フィールドを指定します。

表 108. *neuralnetwork* ノードのプロパティー *(続き)*:

neuralnetwork ノードのプロパティー	**値**	**プロパティーの説明**
inputs	[*field1 ... fieldN*]	モデルで使用される入力または入力または予測変数フィールド。
splits	*[field1 ... fieldN*	分割モデル作成に使用する、フィールドを選択します。
use_partition	*boolean*	区分フィールドが定義される場合、このオプションは学習データ区分からのデータのみがモデル構築に使用されるようにします。
continue	*boolean*	既存モデルの学習を継続 :
objective	標準 バギング ブースティング psm	psm は非常に大きいデータ・セットに使用され、Server の接続が必要です。
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*boolean*	
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*boolean*	
max_time	*number*	
use_max_cycles	*boolean*	
max_cycles	*number*	
use_min_accuracy	*boolean*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*boolean*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_custom_model_name	*boolean*	
custom_model_name	*文字列*	
confidence	onProbability onIncrease	
score_category_probabilities	*boolean*	

表 108. *neuralnetwork* ノードのプロパティー *(続き)*:

neuralnetwork ノードのプロパティー	値	プロパティーの説明
max_categories	*number*	
score_propensity	*boolean*	
use_custom_name	*boolean*	
custom_name	*文字列*	
tooltip	*文字列*	
keywords	*文字列*	
annotation	*文字列*	

quest ノードのプロパティー

QUEST ノードには、デシジョン・ツリーの構築用に2 分岐の方法が用意されています。これは、大規模な C&R ツリー分析が必要とする処理時間を短縮すると同時に、より多くの分割を可能にする入力値が優先される分類ツリー内の傾向を低減するように設計されています。入力フィールドは、数値範囲 (連続型) にできますが、目標変数はカテゴリーでなければなりません。すべての分割は 2 分岐です。

表 109. *quest* ノードのプロパティー:

quest ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	QUEST モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。度数フィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue_training_existing_model	*boolean*	
objective	標準 ブースティング バギング psm	psm は非常に大きいデータ・セットに使用され、Server の接続が必要です。
model_output_type	Single InteractiveBuilder	
use_tree_directives	*boolean*	
tree_directives	*文字列*	
use_max_depth	Default Custom	
max_depth	*integer*	最大ツリー深さ (0 から 1000)。use_max_depth = Custom の場合にのみ使用されます。
prune_tree	*boolean*	オーバーフィットしないようにツリーを剪定します。
use_std_err	*boolean*	リスクにおける最大差 (標準誤差) を使用します。

表 109. *quest* ノードのプロパティー *(続き)*:

quest ノードのプロパティー	値	プロパティーの説明
std_err_multiplier	*number*	最大差。
max_surrogates	*number*	最大代理フィールド :
use_percentage	*boolean*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*boolean*	
costs	*構造化*	構造化プロパティー。
priors	Data Equal Custom	
custom_priors	*構造化*	構造化プロパティー。
adjust_priors	*boolean*	
trails	*number*	ブーストまたはバグのコンポーネント・モデル数。
set_ensemble_method	Voting HighestProbability HighestMeanProbability	カテゴリー型対象のデフォルト結合ルール。
range_ensemble_method	Mean Median	連続型対象のデフォルト結合ルール。
large_boost	*boolean*	特に大きなデータセットのブースティングを適用します。
split_alpha	*number*	分割の有意水準 :
train_pct	*number*	オーバーフィット防止セット。
set_random_seed	*boolean*	結果の複製オプション。
seed	*number*	
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

regression ノードのプロパティー

線型回帰は、データを要約する一般的な統計手法であり、予測された出力値と実際の出力値の違いを最小限にする直線または面を当てはめることにより予測を行います。

注 : 回帰ノードは、今後のリリースでは線型ノードに置き換えられます。今後、線型回帰には線型モデルを使用することをお勧めします。

表 110. regression ノードのプロパティー：

regression ノードのプロパティー	**値**	**プロパティーの説明**
ターゲット	*field*	回帰モデルは単一の対象フィールドおよび 1 つ以上の入力フィールドを使用します。重みフィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
method	Enter Stepwise Backwards Forwards	
include_constant	*boolean*	
use_weight	*boolean*	
weight_field	*field*	
mode	Simple (単純) Expert	
complete_records	*boolean*	
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	引数には二重引用符を使用します。
stepping_method	useP useF	useP : F 値の確率を使用 useF: F 値を使用
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*boolean*	
confidence_interval	*boolean*	
covariance_matrix	*boolean*	
collinearity_diagnostics	*boolean*	
regression_coefficients	*boolean*	
exclude_fields	*boolean*	
durbin_watson	*boolean*	
model_fit	*boolean*	
r_squared_change	*boolean*	
p_correlations	*boolean*	

表 110. *regression* ノードのプロパティー *(続き)*:

regression ノードのプロパティー	値	プロパティーの説明
descriptives	*boolean*	
calculate_variable_importance	*boolean*	

sequence ノードのプロパティー

シーケンス・ノードで、シーケンシャルな、または時間経過が伴うデータ内のアソシエーション・ルールを検出します。予測可能な順序で起こる傾向にあるアイテム・セットのリストを、シーケンスと呼びます。例えば、顧客がひげそりとアフター・シェーブ ローションを購入した場合、その顧客は次の購入時にシェービング クリームを購入する可能性があります。シーケンス・ノードは CARMA アソシエーション・ルール・アルゴリズムに基づいており、効率的な 2 段階通過法を使用してシーケンスを検出します。

表 111. *sequence* ノードのプロパティー:

sequence ノードのプロパティー	値	プロパティーの説明
id_field	*field*	シーケンス・モデルを作成するには、ID フィールドを指定する必要があります。さらにオプションで時間フィールドと 1 つ以上の内容フィールドを指定します。重みフィールドおよび度数フィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
time_field	*field*	
use_time_field	*boolean*	
content_fields	[*field1 ... fieldn*]	
contiguous	*boolean*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple (単純) Expert	
use_max_duration	*boolean*	
max_duration	*number*	
use_gaps	*boolean*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*boolean*	
pruning_value	*number*	
set_mem_sequences	*boolean*	
mem_sequences	*integer*	

slrm ノードのプロパティー

SLRM (自己学習応答モデル) ノードを使用するとモデルを構築でき、単一または少数の新しいケースを使用して全データを使用するモデルの保持をすることなく、モデルの再見積もりを行うことができます。

表 112. slrm ノードのプロパティー:

slrm ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	対象フィールドは名義型またはフラグ型である必要があります。度数フィールドも指定できます。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
target_response	*field*	フラグ型である必要があります。
continue_training_existing_model	*boolean*	
target_field_values	*boolean*	すべて使用:ソースのすべての値を使用します。 指定:必要な値を選択します。
target_field_values_specify	*[フィールド 1 ... フィールド N]*	
include_model_assessment	*boolean*	
model_assessment_random_seed	*number*	Must be a real number.
model_assessment_sample_size	*number*	Must be a real number.
model_assessment_iterations	*number*	Number of iterations.
display_model_evaluation	*boolean*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Ascending Descending	高いスコアまたは低いスコアのどちらを持つオファーが最初に表示されるかを指定します。
model_reliability	*boolean*	
calculate_variable_importance	*boolean*	

statisticsmodel ノードのプロパティー

Statistics モデル・ノードを使用すると、PMML を作成する IBM SPSS Statistics 手続きを実行してデータを分析および使用することができます。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

このノードのプロパティーについては、 248 ページの『statisticsmodel ノードのプロパティー』に記載されています。

svm ノードのプロパティー

サポート・ベクター・マシン (SVM) ノードを使用すると、オーバーフィットすることなく、データを 2 つのグループのいずれかに分類することができます。SVM は、非常に多数の入力フィールドを含むデータ・セットなど、広範なデータ・セットを処理することができます。

表 113. svm ノードのプロパティー:

svm ノードのプロパティー	値	プロパティーの説明
all_probabilities	*boolean*	
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (default) 1.0E-4 1.0E-5 1.0E-6	最適化アルゴリズムを停止するタイミングを決定します。
regularization	*number*	C パラメーターとしても知られています。
precision	*number*	対象フィールドの尺度が Continuous の場合にのみ使用されます。
kernel	RBF (デフォルト) Polynomial シグモイド Linear	変換に使用されるカーネル関数のタイプ。
rbf_gamma	*number*	kernel が RBF の場合にのみ使用します。
gamma	*number*	kernel が Polynomial または Sigmoid の場合にのみ使用します。
bias	*number*	
degree	*number*	kernel が Polynomial の場合にのみ使用します。
calculate_variable_importance	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	
adjusted_propensity_partition	Test Validation	

timeseries ノードのプロパティー

時系列ノードは、時系列から指数平滑法、1 変量の自己回帰型統合移動平均法 (ARIMA)、および多変量 ARIMA (または転送関数) モデルを推測し、将来のパフォーマンスの予測を作成します。時系列ノードは、時間区分ノードによって常に先行される必要があります。

表 114. *timeseries* ノードのプロパティー:

timeseries ノードのプロパティー	値	プロパティーの説明
targets	*field*	時系列ノードは、オプションで 1 つ以上の入力フィールドを予測値として使用しながら、1 つ以上の対象フィールドを予測します。度数フィールドおよび重みフィールドは使用しません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
continue	*boolean*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*boolean*	
consider_seasonal	*boolean*	
detect_outliers	*boolean*	
expert_outlier_additive	*boolean*	
expert_outlier_level_shift	*boolean*	
expert_outlier_innovational	*boolean*	
expert_outlier_level_shift	*boolean*	
expert_outlier_transient	*boolean*	
expert_outlier_seasonal_additive	*boolean*	
expert_outlier_local_trend	*boolean*	
expert_outlier_additive_patch	*boolean*	
exsmooth_model_type	Simple (単純) HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	
arima_d	*integer*	
arima_q	*integer*	
arima_sp	*integer*	
arima_sd	*integer*	
arima_sq	*integer*	

表 114. *timeseries* ノードのプロパティー *(続き)*:

timeseries ノードのプロパティー	値	プロパティーの説明
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*boolean*	
tf_arima_p. *fieldname*	*integer*	転送関数用。
tf_arima_d. *fieldname*	*integer*	転送関数用。
tf_arima_q. *fieldname*	*integer*	転送関数用。
tf_arima_sp. *fieldname*	*integer*	転送関数用。
tf_arima_sd. *fieldname*	*integer*	転送関数用。
tf_arima_sq. *fieldname*	*integer*	転送関数用。
tf_arima_delay. *fieldname*	*integer*	転送関数用。
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	転送関数用。
arima_detect_outlier_mode	None Automatic (自動)	
arima_outlier_additive	*boolean*	
arima_outlier_level_shift	*boolean*	
arima_outlier_innovational	*boolean*	
arima_outlier_transient	*boolean*	
arima_outlier_seasonal_additive	*boolean*	
arima_outlier_local_trend	*boolean*	
arima_outlier_additive_patch	*boolean*	
conf_limit_pct	*real*	
max_lags	*integer*	
events	*fields*	
scoring_model_only	*boolean*	多く (1 万単位) の時系列のモデルに使用します。

twostep ノードのプロパティー

TwoStep ノードで、2 段階のクラスター化手法が使用されます。最初のステップでは、データを 1 度通過させて、未処理の入力データを管理可能な一連のサブクラスターに圧縮します。2 番目のステップでは、階層クラスター化手法を使用して、サブクラスターをより大きなクラスターに結合させていきます。TwoStep には、学習データに最適なクラスター数を自動的に推定するという利点があります。また、フィールド・タイプの混在や大規模データ・セットも効率よく処理できます。

表 115. twostep ノードのプロパティー：

twostep ノードのプロパティー	**値**	**プロパティーの説明**
inputs	[*field1 ... fieldN*]	TwoStep モデルは対象フィールドでなく、入力フィールドのリストを使用します。重みフィールドおよび度数フィールドは認識されません。 詳しくは、トピック 135 ページの『モデル作成ノードの共通プロパティー』を参照してください。
standardize (標準化)	*boolean*	
exclude_outliers	*boolean*	
percentage	*number*	
cluster_num_auto	*boolean*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String 数値	
label_prefix	*文字列*	
distance_measure	ユークリッド Loglikelihood	
clustering_criterion	AIC BIC	

第 14 章 モデル・ナゲット・ノードのプロパティー

モデル・ナゲット・ノードは、他のノードと同じ共通のプロパティーを共有しています。 詳しくは、トピック 60 ページの『共通のノード・プロパティー』を参照してください。

applyanomalydetection ノードのプロパティー

異常値検出モデル作成ノードを使用して、異常値検出モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyanomalydetection* です。モデル作成ノード自体のスクリプトの詳細は、トピック 136 ページの『anomalydetection ノードのプロパティー』を参照してください。

表 116. applyanomalydetection ノードのプロパティー:

applyanomalydetection ノードのプロパティー	**値**	**プロパティーの説明**
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	スコアリング用に、作成される出力を決めます。
num_fields	*integer*	報告するフィールド数。
discard_records	*boolean*	レコードが出力から廃棄されるかどうかを示します。
discard_anomalous_records	*boolean*	異常なレコードと異常ではない レコードのどちらを廃棄するかの標識。デフォルトは off です。この場合、異常ではない レコードが廃棄されます。on を指定すると、異常なレコードが廃棄されます。このプロパティーは、discard_records プロパティーが有効な場合のみ使用することができます。

applyapriori ノードのプロパティー

Apriori モデル作成ノードを使用して、Apriori モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyapriori* です。モデル作成ノード自体のスクリプトの詳細は、トピック 137 ページの『apriori ノードのプロパティー』を参照してください。

表 117. applyapriori ノードのプロパティー:

applyapriori ノードのプロパティー	**値**	**プロパティーの説明**
max_predictions	*数値 (整数)*	
ignore_unmatached	*boolean*	
allow_repeats	*boolean*	
check_basket	NoPredictions Predictions NoCheck	

表 117. *applyapriori* ノードのプロパティー *(続き)*:

applyapriori ノードのプロパティー	値	プロパティーの説明
criterion	Confidence Support RuleSupport Lift Deployability	

applyautoclassifier ノードのプロパティー

自動分類モデル作成ノードを使用して、自動分類モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyautoclassifier* です。モデル作成ノード自体のスクリプトの詳細は、トピック 138 ページの『autoclassifier ノードのプロパティー』を参照してください。

表 118. *applyautoclassifier* ノードのプロパティー:

applyautoclassifier ノードのプロパティー	値	プロパティーの説明
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	アンサンブル・スコアを決定するために使用する方法を指定します。この設定は、選択された対象がフラグ型フィールドである場合にのみ適用されます。
flag_voting_tie_selection	Random HighestConfidence RawPropensity	票決方法が選択された場合、可否同数の解決方法を指定します。この設定は、選択された対象がフラグ型フィールドである場合にのみ適用されます。
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	アンサンブル・スコアを決定するために使用する方法を指定します。この設定は、選択された対象がセット型フィールドである場合にのみ適用されます。
set_voting_tie_selection	Random HighestConfidence	票決方法が選択された場合、可否同数の解決方法を指定します。この設定は、選択された対象が名義型フィールドである場合にのみ適用されます。

applyautocluster ノードのプロパティー

自動クラスター・モデル作成ノードを使用して、自動クラスター・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyautocluster* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 140 ページの『autocluster ノードのプロパティー』を参照してください。

applyautonumeric ノードのプロパティー

自動数値モデル作成ノードを使用して、自動数値モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyautonumeric* です。モデル作成ノード自体のスクリプトの詳細は、トピック 141 ページの『autonumeric ノードのプロパティー』を参照してください。

表 119. applyautonumeric ノードのプロパティー:

applyautonumeric ノードのプロパティー	値	プロパティーの説明
calculate_standard_error	*boolean*	

applybayesnet ノードのプロパティー

ベイズ・ネットワーク・モデル作成ノードを使用して、ベイズ・ネットワーク・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applybayesnet* です。モデル作成ノード自体のスクリプトの詳細は、トピック 142 ページの『bayesnet ノードのプロパティー』を参照してください。

表 120. applybayesnet ノードのプロパティー:

applybayesnet ノードのプロパティー	値	プロパティーの説明
all_probabilities	*boolean*	
raw_propensity	*boolean*	
adjusted_propensity	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applyc50 ノードのプロパティー

C5.0 モデル作成ノードを使用して、C5.0 モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyc50* です。モデル作成ノード自体のスクリプトの詳細は、トピック 144 ページの『c50 ノードのプロパティー』を参照してください。

表 121. applyc50 ノードのプロパティー:

applyc50 ノードのプロパティー	値	プロパティーの説明
sql_generate	Never NoMissingValues	ルールセット実行時の SQL 生成オプションの設定に使用します。
calculate_conf	*boolean*	SQL 生成が有効になっている場合に利用できます。このプロパティーには、生成されたツリー中の確信度計算が含まれています。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applycarma ノードのプロパティー

CARMA モデル作成ノードを使用して、CARMA モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applycarma* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 145 ページの『carma ノードのプロパティー』を参照してください。

applycart ノードのプロパティー

C&R Tree モデル作成を使用して、C&R Tree モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applycart* です。モデル作成ノード自体のスクリプトの詳細は、トピック 146 ページの『cart ノードのプロパティー』を参照してください。

表 122. applycart ノードのプロパティー:

applycart ノードのプロパティー	値	プロパティーの説明
sql_generate	Never MissingValues NoMissingValues	ルールセット実行時の SQL 生成オプションの設定に使用します。
calculate_conf	*boolean*	SQL 生成が有効になっている場合に利用できます。このプロパティーには、生成されたツリー中の確信度計算が含まれています。
display_rule_id	*boolean*	フィールドが 1 つスコアリング出力に追加されますが、これは各レコードを割り当てるターミナル・ノードに ID を示すためのものです。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applychaid ノードのプロパティー

CHAID モデル作成ノードを使用して、CHAID モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applychaid* です。モデル作成ノード自体のスクリプトの詳細は、トピック 148 ページの『chaid ノードのプロパティー』を参照してください。

表 123. applychaid ノードのプロパティー:

applychaid ノードのプロパティー	値	プロパティーの説明
sql_generate	Never MissingValues	
calculate_conf	*boolean*	
display_rule_id	*boolean*	フィールドが 1 つスコアリング出力に追加されますが、これは各レコードを割り当てるターミナル・ノードに ID を示すためのものです。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applycoxreg ノードのプロパティー

Cox モデル作成ノードを使用して、Cox モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applycoxreg* です。モデル作成ノード自体のスクリプトの詳細は、トピック 149 ページの『coxreg ノードのプロパティー』を参照してください。

表 124. applycoxreg ノードのプロパティー:

applycoxreg ノードのプロパティー	値	プロパティーの説明
future_time_as	間隔 フィールド	
time_interval	*number*	
num_future_times	*integer*	
time_field	*field*	
past_survival_time	*field*	
all_probabilities	*boolean*	
cumulative_hazard	*boolean*	

applydecisionlist ノードのプロパティー

デシジョン・リスト・モデル作成ノードを使用して、デシジョン・リスト・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applydecisionlist* です。モデル作成ノード自体のスクリプトの詳細は、トピック 151 ページの『decisionlist ノードのプロパティー』を参照してください。

表 125. applydecisionlist ノードのプロパティー:

applydecisionlist ノードのプロパティー	値	プロパティーの説明
enable_sql_generation	*boolean*	真に設定したときは、デシジョン・リスト・モデルが SQL へプッシュバックされるように IBM SPSS Modeler が試行します。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applydiscriminant ノードのプロパティー

判別分析モデル作成ノードを使用して、判別分析モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applydiscriminant* です。モデル作成ノード自体のスクリプトの詳細は、トピック 152 ページの『discriminant ノードのプロパティー』を参照してください。

表 126. applydiscriminant ノードのプロパティー:

applydiscriminant ノードのプロパティー	値	プロパティーの説明
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applyfactor ノードのプロパティー

因子分析モデル作成ノードを使用して、因子分析モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyfactor* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 154 ページの『factor ノードのプロパティー』を参照してください。

applyfeatureselection ノードのプロパティー

フィールド選択モデル作成ノードを使用して、フィールド選択モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyfeatureselection* です。モデル作成ノード自体のスクリプトの詳細は、トピック 155 ページの『featureselection ノードのプロパティー』を参照してください。

表 127. applyfeatureselection ノードのプロパティー:

applyfeatureselection ノードのプロパティー	値	プロパティーの説明
selected_ranked_fields		モデル・ブラウザー内で検査されるランク付きのフィールドを指定します。
selected_screened_fields		モデル・ブラウザー内で検査されるスクリーニングされたフィールドを指定します。

applygeneralizedlinear ノードのプロパティー

一般化線型 (genlin) モデル作成ノードを使用して、一般化線型モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applygeneralizedlinear* です。モデル作成ノード自体のスクリプトの詳細は、トピック 156 ページの『genlin ノードのプロパティー』を参照してください。

表 128. applygeneralizedlinear ノードのプロパティー:

applygeneralizedlinear ノードのプロパティー	値	プロパティーの説明
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applyglmm ノードのプロパティー

GLMM モデル作成ノードを使用して、GLMM モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyglmm* です。モデル作成ノード自体のスクリプトの詳細は、トピック 160 ページの『glmm ノードのプロパティー』を参照してください。

表 129. applyglmm ノードのプロパティー:

applyglmm ノードのプロパティー	値	プロパティーの説明
confidence	onProbability onIncrease	スコアリングの確信度を計算する基準 (最も高い予測確率、または最も高い予測確率と 2 番目に高い予測確率との差)。

表 129. *applyglmm ノードのプロパティー (続き)*:

applyglmm ノードのプロパティー	値	プロパティーの説明
score_category_probabilities	*boolean*	これを True に設定すると、カテゴリー型対象の予測確率が出力されます。カテゴリーごとにフィールドが作成されます。デフォルトは False です。
max_categories	*integer*	確率を予測するカテゴリーの最大数。score_category_probabilities が True の場合にのみ使用されます。
score_propensity	*boolean*	これを True に設定すると、フラグ型対象のモデルの未調整の傾向スコア (「真」の結果の尤度) が出力されます。データ区分が有効な場合は、検定データ区分に基づいて調整済み傾向スコアも出力されます。デフォルトは False です。

applykmeans ノードのプロパティー

K-means モデル作成ノードを使用して、K-means モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applykmeans* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 163 ページの『kmeans ノードのプロパティー』を参照してください。

applyknn ノードのプロパティー

KNN モデル作成ノードを使用して、KNN モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyknn* です。モデル作成ノード自体のスクリプトの詳細は、トピック 164 ページの『knn ノードのプロパティー』を参照してください。

表 130. *applyknn ノードのプロパティー*:

applyknn ノードのプロパティー	値	プロパティーの説明
all_probabilities	*boolean*	
save_distances	*boolean*	

applykohonen ノードのプロパティー

Kohonen モデル作成ノードを使用して、Kohonen モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applykohonen* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 165 ページの『kohonen ノードのプロパティー』を参照してください。

applylinear ノードのプロパティー

線型モデル作成ノードを使用して、線型モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applylinear* です。モデル作成ノード自体のスクリプトの詳細は、トピック 166 ページの『linear ノードのプロパティー』を参照してください。

表 131. applylinear ノードのプロパティー:

applylinear ノードのプロパティー	値	プロパティーの説明
use_custom_name	*boolean*	
custom_name	*文字列*	
enable_sql_generation	*boolean*	

applylogreg ノードのプロパティー

ロジスティック回帰モデル作成ノードを使用して、ロジスティック回帰モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applylogreg* です。モデル作成ノード自体のスクリプトの詳細は、トピック 167 ページの『logreg ノードのプロパティー』を参照してください。

表 132. applylogreg ノードのプロパティー:

applylogreg ノードのプロパティー	値	プロパティーの説明
calculate_raw_propensities	*boolean*	
calculate_conf	*boolean*	
enable_sql_generation	*boolean*	

applyneuralnet ノードのプロパティー

ニューラル・ネットワーク・モデル作成ノードを使用して、ニューラル・ネットワーク・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyneuralnet* です。モデル作成ノード自体のスクリプトの詳細は、トピック 170 ページの『neuralnet ノードのプロパティー』を参照してください。

注意: 機能が拡張された新しいバージョンのニューラル・ネットワーク ナゲットがこのリリースで使用できます。新しいバージョンについては次の項で説明します (*applyneuralnetwork*)。以前のバージョンは現在も使用できますが、スクリプトを更新して新しいバージョンを使用することをお勧めします。旧バージョンの詳細を参照用に記載しておりますが、それに対するサポートは今後のリリースで廃止されます。

表 133. applyneuralnet ノードのプロパティー:

applyneuralnet ノードのプロパティー	値	プロパティーの説明
calculate_conf	*boolean*	SQL 生成が有効になっている場合に利用できます。このプロパティーには、生成されたツリー中の確信度計算が含まれています。
enable_sql_generation	*boolean*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applyneuralnetwork ノードのプロパティー

ニューラル・ネットワーク・モデル作成ノードを使用して、ニューラル・ネットワーク・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyneuralnetwork* です。モデル作成ノード自体のスクリプトの詳細は、トピック 172 ページの『neuralnetwork ノードのプロパティー』を参照してください。

表 134. applyneuralnetwork ノードのプロパティー:

applyneuralnetwork ノードのプロパティー	値	プロパティーの説明
use_custom_name	*boolean*	
custom_name	*文字列*	
confidence	onProbability onIncrease	
score_category_probabilities	*boolean*	
max_categories	*number*	
score_propensity	*boolean*	

applyquest ノードのプロパティー

QUEST モデル作成ノードを使用して、QUEST モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyquest* です。モデル作成ノード自体のスクリプトの詳細は、トピック 174 ページの『quest ノードのプロパティー』を参照してください。

表 135. applyquest ノードのプロパティー:

applyquest ノードのプロパティー	値	プロパティーの説明
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	*boolean*	
display_rule_id	*boolean*	フィールドが 1 つスコアリング出力に追加されますが、これは各レコードを割り当てるターミナル・ノードに ID を示すためのものです。
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applyregression ノードのプロパティー

線型回帰モデル作成ノードを使用して、線型回帰モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyregression* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 175 ページの『regression ノードのプロパティー』を参照してください。

applyr ノードのプロパティー

R 構築ノードを使用して R モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyr* です。モデル作成ノード自体のスクリプトの詳細は、トピック 143 ページの『buildr ノードのプロパティー』を参照してください。

表 136. applyr ノードのプロパティー

applyr ノードのプロパティー	値	プロパティーの説明
score_syntax	*文字列*	モデル・スコアリング用の R スクリプト・シンタックス。
convert_flags	StringsAndDoubles LogicalValues	フラグ型フィールドを変換するためのオプション。
convert_datetime	*boolean*	日付形式または日付/時刻形式の変数を R の日付/時刻形式に変換するためのオプション。
convert_datetime_class	POSIXct POSIXlt	日付形式または日付/時刻形式の変数のうち、どの形式の変数を変換するかを指定するためのオプション。
convert_missing	*boolean*	欠損値を R の NA 値に変換するためのオプション。

applyselflearning ノードのプロパティー

自己学習応答モデル (SLRM) モデル作成ノードを使用して、SLRM モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applyselflearning* です。モデル作成ノード自体のスクリプトの詳細は、トピック 178 ページの『slrm ノードのプロパティー』を参照してください。

表 137. applyselflearning ノードのプロパティー:

applyselflearning ノードのプロパティー	値	プロパティーの説明
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	ascending descending	高いスコアまたは低いスコアのどちらを持つオファーが最初に表示されるかを指定します。
model_reliability	*boolean*	「設定」タブでモデルの信頼性を考慮します。

applysequence ノードのプロパティー

シーケンス・モデル作成ノードを使用して、シーケンス・モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applysequence* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 177 ページの『sequence ノードのプロパティー』を参照してください。

applysvm ノードのプロパティー

SVM モデル作成ノードを使用して、SVM モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applysvm* です。モデル作成ノード自体のスクリプトの詳細は、トピック 179 ページの『svm ノードのプロパティー』を参照してください。

表 138. applysvm ノードのプロパティー:

applysvm ノードのプロパティー	値	プロパティーの説明
all_probabilities	*boolean*	
calculate_raw_propensities	*boolean*	
calculate_adjusted_propensities	*boolean*	

applytimeseries ノードのプロパティー

時系列モデル作成ノードを使用して、時系列モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applytimeseries* です。モデル作成ノード自体のスクリプトの詳細は、トピック 179 ページの『timeseries ノードのプロパティー』を参照してください。

表 139. applytimeseries ノードのプロパティー:

applytimeseries ノードのプロパティー	値	プロパティーの説明
calculate_conf	*boolean*	
calculate_residuals	*boolean*	

applytwostep ノードのプロパティー

TwoStep モデル作成ノードを使用して、TwoStep モデル・ナゲットを生成することができます。このモデル・ナゲットのスクリプト名は、*applytwostep* です。このモデル・ナゲットの他のプロパティーはありません。モデル作成ノード自体のスクリプトの詳細は、トピック 181 ページの『twostep ノードのプロパティー』を参照してください。

第 15 章 データベース・モデル作成ノードのプロパティー

IBM SPSS Modeler は、Microsoft SQL Server Analysis Services、Oracle Data Mining、IBM DB2 InfoSphere Warehouse、IBM Netezza Analytics を含む、データベース・ベンダーから入手可能なデータ・マイニングとモデル作成ツールとの統合をサポートしています。IBM SPSS Modeler ネイティブ・データベース・アルゴリズムを使用して、アプリケーション内からのモデルの構築およびスコアリングがすべて可能です。データベース・モデルは、このセクションで説明するプロパティーを使用してスクリプトで作成および処理することも可能です。

Microsoft モデル作成ノードのプロパティー

Microsoft モデル作成ノードのプロパティー

共通のプロパティー

次のプロパティーは、Microsoft データベース・モデル作成ノードに共通です。

表 140. Microsoft ノードの共通プロパティー:

Microsoft ノードの共通プロパティー	値	プロパティーの説明
analysis_database_name	*文字列*	Analysis Services データベースの名前。
analysis_server_name	*文字列*	Analysis Services ホストの名前。
use_transactional_data	*boolean*	入力データがテーブル形式またはトランザクション形式かを指定します。
inputs	*[フィールド フィールド フィールド]*	テーブル形式の入力フィールド。
ターゲット	*field*	予測フィールド (MS クラスタリング・ノードまたはシーケンス・クラスタリング・ノードには該当しない)。
unique_field	*field*	キー・フィールド。
msas_parameters	*構造化*	アルゴリズム・パラメーター。 詳しくは、トピック 197 ページの『アルゴリズム・パラメーター』を参照してください。
with_drillthrough	*boolean*	「ドリルスルーあり」オプション。

MS ディシジョン・ツリー

mstree タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS クラスタリング

mscluster タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS アソシエーション・ルール

次のプロパティーは、msassoc タイプのノードで使用できます。

表 141. *msassoc* ノードのプロパティー:

msassoc ノードのプロパティー	値	プロパティーの説明
id_field	*field*	データの各トランザクションを特定します。
trans_inputs	*[フィールド フィールド フィールド]*	トランザクションデータの入力フィールド。
transactional_target	*field*	予測データ (トランザクション・データ)。

MS Naive Bayes

msbayes タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS Linear Regression

msregression タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS Neural Network

msneuralnetwork タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS Logistic Regression

mslogistic タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS Time Series

mstimeseries タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Microsoft プロパティーを参照してください。

MS Sequence Clustering

次のプロパティーは、mssequencecluster タイプのノードで使用できます。

表 142. *mssequencecluster* ノードのプロパティー:

mssequencecluster ノードのプロパティー	値	プロパティーの説明
id_field	*field*	データの各トランザクションを特定します。
input_fields	*[フィールド フィールド フィールド]*	トランザクションデータの入力フィールド。
sequence_field	*field*	シーケンス ID。
target_field	*field*	予測フィールド (テーブル形式データ)。

アルゴリズム・パラメーター

各 Microsoft データベース・モデル タイプには、msas_parameters プロパティーを使用して設定できる、特定のパラメーターがあります。

これらのパラメーターは SQL から取得されます。各ノードに関連するパラメーターを見るには

1. キャンバスにデータベース入力ノードを配置します。
2. データベース入力ノードを開きます。
3. 「**データ・ソース**」 ドロップダウン・リストから有効なソースを選択します。
4. 「**テーブル名**」 リストから有効なテーブルを選択します。
5. 「**OK**」 をクリックして、データベース入力ノードを閉じます。
6. プロパティーを一覧表示したい Microsoft データベース・モデル作成ノードを追加します。
7. データベース・モデル作成ノードを開きます。
8. 「**エキスパート**」 タブを選択します。

このノードの使用できる msas_parameters プロパティーが表示されます。

Microsoft モデル・ナゲットのプロパティー

Microsoft データベース・モデル作成ノードを使用して作成されるモデル・ナゲットのプロパティーを、次に示します。

MS ディシジョン・ツリー

表 143. MS デシジョン・ツリーのプロパティー:

applymstree ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前
datasource	*文字列*	SQL Server の ODBC データ・ソース 名 (DSN) の名前
sql_generate	*boolean*	SQL 生成を有効にします。

MS Linear Regression

表 144. MS 線型回帰のプロパティー:

applymsregression ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前

MS Neural Network

表 145. MS ニューラル・ネットワークのプロパティー:

applymsneuralnetwork ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前

MS Logistic Regression

表 146. MS ロジスティック回帰のプロパティー:

applymslogistic ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前

MS Time Series

表 147. MS タイム・シリーズのプロパティー:

applymstimeseries ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前
start_from	new_prediction historical_prediction	将来の予測を行うか過去の予測を行うかを指定します。
new_step	*number*	将来の予測の開始時間を定義します。
historical_step	*number*	過去の予測の開始時間を定義します。
end_step	*number*	予測の終了時間を定義します。

MS Sequence Clustering

表 148. MS シーケンス・クラスタリングのプロパティー:

applymssequencecluster ノードのプロパティー	**値**	**説明**
analysis_database_name	*文字列*	このノードは、ストリームの中で直接スコアされます。 このプロパティーは Analysis Services データベース名の識別に使用します。
analysis_server_name	*文字列*	Analysis サーバー・ホストの名前

Oracle モデル作成ノードのプロパティー

Oracle モデル作成ノードのプロパティー

次のプロパティーは、各 Oracle データベース・モデリング・ノードに共通です。

表 149. Oracle ノードの共通プロパティー:

Oracle ノードの共通プロパティー	**値**	**プロパティーの説明**
ターゲット	*field*	
inputs	*フィールドのリスト*	
partition	*field*	モデル構築の学習、テスト、および検証の各ステージ用に、データを独立したサブセット (サンプル) に分割するフィールド。
datasource		
username		
password		
epassword		
use_model_name	*boolean*	
model_name	*文字列*	ユーザーが指定する新規モデル名。
use_partitioned_data	*boolean*	区分フィールドが定義される場合、このオプションは学習データ区分からのデータのみがモデル構築に使用されるようにします。
unique_field	*field*	
auto_data_prep	*boolean*	Oracle 自動データ準備機能を有効化または無効化します (11g データベースのみ)。
costs	*構造化*	構造化プロパティー。
mode	Simple (単純) Expert	Simple に設定されている場合、個々のノード・プロパティーに記述されているように、特定のプロパティーは無視されます。
use_prediction_probability	*boolean*	
prediction_probability	*文字列*	
use_prediction_set	*boolean*	

Oracle Naive Bayes

次のプロパティーは、oranb タイプのノードで使用できます。

表 150. oranb ノードのプロパティー:

oranb ノードのプロパティー	**値**	**プロパティーの説明**
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	*構造化*	構造化プロパティー。

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle Adaptive Bayes

次のプロパティーは、oraabn タイプのノードで使用できます。

表 151. oraabn ノードのプロパティー:

oraabn ノードのプロパティー	**値**	**プロパティーの説明**
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*boolean*	*
execution_time_limit	*integer*	値は 1 以上でなければなりません。*
max_naive_bayes_predictors	*integer*	値は 1 以上でなければなりません。*
max_predictors	*integer*	値は 1 以上でなければなりません。*
priors	Data Equal Custom	
custom_priors	*構造化*	構造化プロパティー。

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle Support Vector Machines

次のプロパティーは、orasvm タイプのノードで使用できます。

表 152. orasvm ノードのプロパティー:

orasvm ノードのプロパティー	**値**	**プロパティーの説明**
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*integer*	Gaussian カーネル専用。値は 1 以上でなければなりません。*
convergence_tolerance	*number*	値は 1 以上でなければなりません。*
use_standard_deviation	*boolean*	Gaussian カーネル専用。*
standard_deviation	*number*	値は 1 以上でなければなりません。*
use_epsilon	*boolean*	回帰モデルのみです。*
epsilon	*number*	値は 1 以上でなければなりません。*
use_complexity_factor	*boolean*	*
complexity_factor	*number*	*
use_outlier_rate	*boolean*	単一バリアントのみです。*
outlier_rate	*number*	単一バリアントのみです。 0.0–1.0.*

表 152. *orasvm* ノードのプロパティー *(続き)*:

orasvm ノードのプロパティー	**値**	**プロパティーの説明**
weights	Data Equal Custom	
custom_weights	*構造化*	構造化プロパティー。

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle 一般化線型モデル

次のプロパティーは、oraglm タイプのノードで使用できます。

表 153. *oraglm* ノードのプロパティー:

oraglm ノードのプロパティー	**値**	**プロパティーの説明**
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*boolean*	*
row_weights_field	*field*	*
save_row_diagnostics	*boolean*	*
row_diagnostics_table	*文字列*	*
coefficient_confidence	*number*	*
use_reference_category	*boolean*	*
reference_category	*文字列*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*boolean*	*

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle Decision Tree

次のプロパティーは、oradecisiontree タイプのノードで使用できます。

表 154. *oradecisiontree* ノードのプロパティー:

oradecisiontree ノードのプロパティー	**値**	**プロパティーの説明**
use_costs	*boolean*	
impurity_metric	Entropy (エントロピー) Gini	
term_max_depth	*integer*	2–20.*

表 154. *oradecisiontree* ノードのプロパティー *(続き)*:

oradecisiontree ノードのプロパティー	**値**	**プロパティーの説明**
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*
term_minrec_node	*integer*	値は 1 以上でなければなりません。*
term_minrec_split	*integer*	値は 1 以上でなければなりません。*
display_rule_ids	*boolean*	*

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle O-Cluster

次のプロパティーは、oraocluster タイプのノードで使用できます。

表 155. *oraocluster* ノードのプロパティー:

oraocluster ノードのプロパティー	**値**	**プロパティーの説明**
max_num_clusters	*integer*	値は 1 以上でなければなりません。*
max_buffer	*integer*	値は 1 以上でなければなりません。*
sensitivity	*number*	0.0–1.0.*

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle KMeans

次のプロパティーは、orakmeans タイプのノードで使用できます。

表 156. *orakmeans* ノードのプロパティー:

orakmeans ノードのプロパティー	**値**	**プロパティーの説明**
num_clusters	*integer*	値は 1 以上でなければなりません。*
normalization_method	zscore minmax none	
distance_function	ユークリッド コサイン	
iterations	*integer*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	デフォルトは Variance です。*
num_bins	*integer*	値は 1 以上でなければなりません。*
block_growth	*integer*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle NMF

次のプロパティーは、oranmf タイプのノードで使用できます。

表 157. oranmf ノードのプロパティー:

oranmf ノードのプロパティー	**値**	**プロパティーの説明**
normalization_method	minmax none	
use_num_features	*boolean*	*
num_features	*integer*	0–1. デフォルト値はアルゴリズムによってデータから推定されます。
random_seed	*number*	*
num_iterations	*integer*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*boolean*	*

* mode が Simple に設定されている場合、プロパティーは無視されます。

Oracle Apriori

次のプロパティーは、oraapriori タイプのノードで使用できます。

表 158. oraapriori ノードのプロパティー:

oraapriori ノードのプロパティー	**値**	**プロパティーの説明**
content_field	*field*	
id_field	*field*	
max_rule_length	*integer*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*boolean*	

Oracle 最小記述長 (MDL)

oramdl タイプのノードには、特定のプロパティーが定義されていません。このセクションの冒頭にある共通 Oracle プロパティーを参照してください。

Oracle Attribute Importance (AI)

次のプロパティーは、oraai タイプのノードで使用できます。

表 159. oraai ノードのプロパティー:

oraai ノードのプロパティー	**値**	**プロパティーの説明**
custom_fields	*boolean*	真 (true) の場合は、現在のノードのターゲット、入力、その他フィールドなどを指定することができます。偽 (false) の場合は、上流のデータ型ノードから現在の設定が使用されます。

表 159. oraai ノードのプロパティー (続き):

oraai ノードのプロパティー	値	プロパティーの説明
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*boolean*	selection_mode が ImportanceLevel に設定されているときに、重要なフィールドを選択するかどうかを指定します。
important_label	*文字列*	「重要」ランクのラベルを指定します。
select_marginal	*boolean*	selection_mode が ImportanceLevel に設定されているときに、境界フィールドを選択するかどうかを指定します。
marginal_label	*文字列*	「境界」ランクのラベルを指定します。
important_above	*number*	0.0–1.0.
select_unimportant	*boolean*	selection_mode が ImportanceLevel に設定されているときに、重要でないフィールドを選択するかどうかを指定します。
unimportant_label	*文字列*	「非重要」ランクのラベルを指定します。
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	selection_mode が ImportanceValue に設定されているときに、使用する分割値を指定します。0 から 100 の値を指定します。
top_n	*number*	selection_mode が TopN に設定されているときに、使用する分割値を指定します。0 から 1000 の値を指定します。

Oracle モデル・ナゲットのプロパティー

Oracle ノードを使用して作成されるモデル・ナゲットのプロパティーを、次に示します。

Oracle Naive Bayes

applyoranb タイプのノードには、特定のプロパティーが定義されていません。

Oracle Adaptive Bayes

applyoraabn タイプのノードには、特定のプロパティーが定義されていません。

Oracle Support Vector Machines

applyorasvm タイプのノードには、特定のプロパティーが定義されていません。

Oracle Decision Tree

次のプロパティーは、applyoradecisiontree タイプのノードで使用できます。

表 160. applyoradecisiontree ノードのプロパティー:

applyoradecisiontree ノードのプロパティー	値	プロパティーの説明
use_costs	*boolean*	

表 160. applyoradecisiontree ノードのプロパティー (続き):

applyoradecisiontree ノードのプロパティー	値	プロパティーの説明
display_rule_ids	*boolean*	

Oracle O-Cluster

applyoraocluster タイプのノードには、特定のプロパティーが定義されていません。

Oracle KMeans

applyorakmeans タイプのノードには、特定のプロパティーが定義されていません。

Oracle NMF

次のプロパティーは、applyoranmf タイプのノードで使用できます。

表 161. applyoranmf ノードのプロパティー:

applyoranmf ノードのプロパティー	値	プロパティーの説明
display_all_features	*boolean*	

Oracle Apriori

このモデル・ナゲットはスクリプトに適用できません。

Oracle MDL

このモデル・ナゲットはスクリプトに適用できません。

IBM DB2 モデル作成ノードのプロパティー

IBM DB2 モデル作成ノードのプロパティー

次のプロパティーは、各 IBM InfoSphere Warehouse (ISW) データベース・モデリング・ノードに共通です。

表 162. ISW ノードの共通プロパティー:

ISW ノードの共通プロパティー	値	プロパティーの説明
inputs	*フィールドのリスト*	
datasource		
username		
password		
epassword		
enable_power_options	*boolean*	
power_options_max_memory	*integer*	値は 33 以上でなければなりません。
power_options_cmdline	*文字列*	
mining_data_custom_sql	*文字列*	
logical_data_custom_sql	*文字列*	

表 162. ISW ノードの共通プロパティー (続き):

ISW ノードの共通プロパティー	値	プロパティーの説明
mining_settings_custom_sql		

ISW ディシジョン・ツリー

次のプロパティーは、db2imtree タイプのノードで使用できます。

表 163. db2imtree ノードのプロパティー:

db2imtree ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	
perform_test_run	*boolean*	
use_max_tree_depth	*boolean*	
max_tree_depth	*integer*	値は 1 以上です。
use_maximum_purity	*boolean*	
maximum_purity	*number*	0 と 100 の間の数値です。
use_minimum_internal_cases	*boolean*	
minimum_internal_cases	*integer*	値は 2 以上です。
use_costs	*boolean*	
costs	*構造化*	構造化プロパティー。

ISW アソシエーション

次のプロパティーは、db2imassoc タイプのノードで使用できます。

表 164. db2imassoc ノードのプロパティー:

db2imassoc ノードのプロパティー	値	プロパティーの説明
use_transactional_data	*boolean*	
id_field	*field*	
content_field	*field*	
data_table_layout	basic limited_length	
max_rule_size	*integer*	値は 3 以上でなければなりません。
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*boolean*	
item_constraints_type	Include Exclude	
use_taxonomy	*boolean*	
taxonomy_table_name	*文字列*	DB2 テーブルの名前は、分類の詳細に格納されます。
taxonomy_child_column_name	*文字列*	分類テーブルの子カラムの名前。子カラムには、項目名またはカテゴリー名が含まれます。
taxonomy_parent_column_name	*文字列*	分類テーブルの親カラムの名前。親カラムには、カテゴリー名が含まれます。

表 164. *db2imassoc ノードのプロパティー (続き)*:

db2imassoc ノードのプロパティー	値	プロパティーの説明
load_taxonomy_to_table	*boolean*	IBM SPSS Modeler に保存されている分類情報をモデルの構築時に、分類テーブルにアップロードするかどうかをコントロールします。すでに分類テーブルが存在する場合、そのテーブルは削除されます。分類情報は、モデル構築ノードと共に保存され、「**カテゴリーの編集**」 ボタンと 「**分類法の編集**」 ボタンを使用して編集できます。

ISW シーケンス

次のプロパティーは、db2imsequence タイプのノードで使用できます。

表 165. *db2imsequence ノードのプロパティー*:

db2imsequence ノードのプロパティー	値	プロパティーの説明
id_field	*field*	
group_field	*field*	
content_field	*field*	
max_rule_size	*integer*	値は 3 以上でなければなりません。
min_rule_support	*number*	0–100%
min_rule_confidence	*number*	0–100%
use_item_constraints	*boolean*	
item_constraints_type	Include Exclude	
use_taxonomy	*boolean*	
taxonomy_table_name	*文字列*	DB2 テーブルの名前は、分類の詳細に格納されます。
taxonomy_child_column_name	*文字列*	分類テーブルの子カラムの名前。子カラムには、項目名またはカテゴリー名が含まれます。
taxonomy_parent_column_name	*文字列*	分類テーブルの親カラムの名前。親カラムには、カテゴリー名が含まれます。
load_taxonomy_to_table	*boolean*	IBM SPSS Modeler に保存されている分類情報をモデルの構築時に、分類テーブルにアップロードするかどうかをコントロールします。すでに分類テーブルが存在する場合、そのテーブルは削除されます。分類情報は、モデル構築ノードと共に保存され、「**カテゴリーの編集**」 ボタンと 「**分類法の編集**」 ボタンを使用して編集できます。

ISW 回帰

次のプロパティーは、db2imreg タイプのノードで使用できます。

表 166. *db2imreg ノードのプロパティー*:

db2imreg ノードのプロパティー	値	プロパティーの説明
ターゲット	*field*	

表 166. db2imreg ノードのプロパティー (続き):

db2imreg ノードのプロパティー	**値**	**プロパティーの説明**
regression_method	transform linear polynomial rbf	regression_method が rbf に設定されている場合にのみ適用されるプロパティーについては、次の表を参照してください。
perform_test_run	*field*	
limit_rsquared_value	*boolean*	
max_rsquared_value	*number*	値の範囲は 0.0 から 1.0 です。
use_execution_time_limit	*boolean*	
execution_time_limit_mins	*integer*	値は 1 以上です。
use_max_degree_polynomial	*boolean*	
max_degree_polynomial	*integer*	
use_intercept	*boolean*	
use_auto_feature_selection_method	*boolean*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*boolean*	
min_significance_level	*number*	
use_min_significance_level	*boolean*	

次のプロパティーは、regression_method が rbf に設定されている場合にのみ適用されます。

表 167. regression_method が rbf に設定される場合の db2imreg ノードのプロパティー:

db2imreg ノードのプロパティー	値	プロパティーの説明
use_output_sample_size	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
output_sample_size	*integer*	デフォルトは 2 です。 最小値は 1 です。
use_input_sample_size	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
input_sample_size	*integer*	デフォルトは 2 です。 最小値は 1 です。
use_max_num_centers	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
max_num_centers	*integer*	デフォルトは 20 です。 最小値は 1 です。
use_min_region_size	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
min_region_size	*integer*	デフォルトは 15 です。 最小値は 1 です。
use_max_data_passes	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
max_data_passes	*integer*	デフォルトは 5 です。 最小値は 2 です。
use_min_data_passes	*boolean*	true の場合、値はデフォルトに自動的に設定されます。
min_data_passes	*integer*	デフォルトは 5 です。 最小値は 2 です。

ISW クラスタリング

次のプロパティーは、db2imcluster タイプのノードで使用できます。

表 168. db2imcluster ノードのプロパティー:

db2imcluster ノードのプロパティー	**値**	**プロパティーの説明**
cluster_method	demographic kohonen birch	
kohonen_num_rows	*integer*	
kohonen_num_columns	*integer*	
kohonen_passes	*integer*	
use_num_passes_limit	*boolean*	
use_num_clusters_limit	*boolean*	
max_num_clusters	*integer*	値は 2 以上です。
birch_dist_measure	log_likelihood euclidean	デフォルトは log_likelihood です。
birch_num_cfleaves	*integer*	デフォルトは 1000 です。
birch_num_refine_passes	*integer*	デフォルトは 3、最小値は 1 です。
use_execution_time_limit	*boolean*	
execution_time_limit_mins	*integer*	値は 1 以上です。
min_data_percentage	*number*	0–100%
use_similarity_threshold	*boolean*	
similarity_threshold	*number*	値の範囲は 0.0 から 1.0 です。

ISW Naive Bayes

次のプロパティーは、db2imnbs タイプのノードで使用できます。

表 169. db2imnb ノードのプロパティー:

db2imnb ノードのプロパティー	**値**	**プロパティーの説明**
perform_test_run	*boolean*	
probability_threshold	*number*	デフォルトは 0.001 です。 最小値は 0、最大値は 1.000 です。
use_costs	*boolean*	
costs	*構造化*	構造化プロパティー。

ISW ロジスティック回帰

次のプロパティーは、db2imlog タイプのノードで使用できます。

表 170. db2imlog ノードのプロパティー:

db2imlog ノードのプロパティー	**値**	**プロパティーの説明**
perform_test_run	*boolean*	
use_costs	*boolean*	

表 170. *db2imlog* ノードのプロパティー *(続き)*:

db2imlog ノードのプロパティー	値	プロパティーの説明
costs	*構造化*	構造化プロパティー。

ISW 時系列

注 : 入力フィールド・パラメーターはこのノードには使用されません。入力フィールド・パラメーターがスクリプトにない場合、ノードに入力フィールドではなく、受信フィールドとして*時間*および*対象*があることを示す警告が表示されます。

次のプロパティーは、db2imtimeseries タイプのノードで使用できます。

表 171. *db2imtimeseries* ノードのプロパティー:

db2imtimeseries ノードのプロパティー	値	プロパティーの説明
time	*field*	整数、時間、日付が使用できます。
targets	*フィールドのリスト*	
forecasting_algorithm	arima exponential_smoothing seasonal_trend_decomposition	
forecasting_end_time	auto integer date time	
use_records_all	*boolean*	false の場合は、use_records_start と use_records_end を設定する必要があります。
use_records_start	*整数/時間/日付*	時間フィールドの種類によって異なります
use_records_end	*整数/時間/日付*	時間フィールドの種類によって異なります
interpolation_method	none linear exponential_splines cubic_splines	

IBM DB2 モデル・ナゲットのプロパティー

IBM DB2 ISW ノードを使用して作成されるモデル・ナゲットのプロパティーを、次に示します。

ISW ディシジョン・ツリー

applydb2imtree タイプのノードには、特定のプロパティーが定義されていません。

ISW アソシエーション

このモデル・ナゲットはスクリプトに適用できません。

ISW シーケンス

このモデル・ナゲットはスクリプトに適用できません。

ISW 回帰

applydb2imreg タイプのノードには、特定のプロパティーが定義されていません。

ISW クラスタリング

applydb2imcluster タイプのノードには、特定のプロパティーが定義されていません。

ISW Naive Bayes

applydb2imnb タイプのノードには、特定のプロパティーが定義されていません。

ISW ロジスティック回帰

applydb2imlog タイプのノードには、特定のプロパティーが定義されていません。

ISW 時系列

このモデル・ナゲットはスクリプトに適用できません。

IBM Netezza Analytics モデル作成ノードのプロパティー

Netezza モデル作成ノードのプロパティー

次のプロパティーは、各 IBM Netezza データベース・モデリング・ノードに共通です。

表 172. Netezza ノードの共通プロパティー:

Netezza ノードの共通プロパティー	値	プロパティーの説明
custom_fields	*boolean*	真 (true) の場合は、現在のノードのターゲット、入力、その他フィールドなどを指定することができます。偽 (false) の場合は、上流のデータ型ノードから現在の設定が使用されます。
inputs	*[フィールド 1 ... フィールド N]*	モデルで使用される入力または予測変数フィールド。
ターゲット	*field*	対象フィールド (連続型またはカテゴリー型)。
record_id	*field*	一意のレコード ID として使用されるフィールド。
use_upstream_connection	*boolean*	true (デフォルト) の場合、 上流のノードで指定された接続の詳細。move_data_to_connection が指定されている場合は使用されません。
move_data_connection	*boolean*	true の場合、データは connection に指定されたデータベースに移動します。use_upstream_connection が指定されている場合は使用されません。

表 172. *Netezza ノードの共通プロパティー (続き)*:

Netezza ノードの共通プロパティー	値	プロパティーの説明
connection	*構造化*	モデルが保存される Netezza データベースの接続文字列。以下の形式の構造化プロパティー。 ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' {true|false}] ここで、 <dsn> は データ・ソース名です。 <username> と <psw> は、データベースのユーザー名とパスワードです。 <catname> は、カタログ名です。 <conn_attribs> は、接続の属性です。 true | false は、パスワードが必要かどうかを示します。
table_name	*文字列*	モデルが保存されるデータベース・テーブルの名前。
use_model_name	*boolean*	true の場合、model_name によって指定された名前をモデルの名前として使用します。そうでない場合、モデル名はシステムによって作成されます。
model_name	*文字列*	ユーザーが指定する新規モデル名。
include_input_fields	*boolean*	true の場合、すべての入力フィールドを下流に渡します。そうでない場合、record_id とモデルによって生成されたフィールドのみが渡されます。

Netezza デシジョン・ツリー

次のプロパティーは、netezzadectree タイプのノードで使用できます。

表 173. *netezzadectree ノードのプロパティー*:

netezzadectree ノードのプロパティー	値	プロパティーの説明
impurity_measure	Entropy (エントロピー) Gini	ツリーの分割に最も良い場所を評価するのに使用される、不純度の測定。
max_tree_depth	*integer*	ツリーが成長可能な最大レベル数。デフォルトは 62 です (可能な最大値)。
min_improvement_splits	*number*	分割が発生する不純度の改善の最小値。デフォルトは 0.01 です。
min_instances_split	*integer*	分割が発生する前に残る分割されていないレコードの最小数。デフォルトは 2 です (可能な最小値)。
weights	*構造化*	クラスの相対的重み。 構造化プロパティー。 デフォルトの重みはすべてのクラスで 1 です。
pruning_measure	Acc wAcc	デフォルトは Acc (精度) です。wAcc (重み付き精度) は、剪定を適用する際にクラスの重みを考慮します。

表 173. *netezzadectree* ノードのプロパティー *(続き)*:

netezzadectree ノードのプロパティー	**値**	**プロパティーの説明**
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	デフォルトでは、allTrainingData を使用してモデルの精度を推定します。partitionTrainingData を使用して、使用する学習データの割合を、useOtherTable を使用して指定したデータベース・テーブルの学習データ・セットを使用します。
perc_training_data	*number*	prune_tree_options が partitionTrainingData に設定されている場合、学習に使用するデータの割合を指定します。
prune_seed	*integer*	prune_tree_options が partitionTrainingData に設定されている場合、分析結果の複製に使用するランダム シード。デフォルトは 1 です。
pruning_table	*文字列*	モデルの精度を推定するために個別の剪定データ・セットのテーブル名。
compute_probabilities	*boolean*	true の場合、予測フィールドのほか、確信度 (確率) フィールドを生成します。

Netezza K-Means

次のプロパティーは、netezzakmeans タイプのノードで使用できます。

表 174. *netezzakmeans* ノードのプロパティー:

netezzakmeans ノードのプロパティー	**値**	**プロパティーの説明**
distance_measure	ユークリッド マンハッタン キャンベラ maximum	データ・ポイント間の教理を測定する方法。
num_clusters	*integer*	作成するクラスター数。デフォルトは 3。
max_iterations	*integer*	モデルの学習を停止する前のアルゴリズムの反復数。デフォルトは 5。
rand_seed	*integer*	分析結果の反復に使用するランダム シード。デフォルトは 12345。

Netezza ベイズ・ネットワーク

次のプロパティーは、netezzabayes タイプのノードで使用できます。

表 175. netezzabayes ノードのプロパティー:

netezzabayes ノードのプロパティー	**値**	**プロパティーの説明**
base_index	*integer*	内部管理の最初の入力フィールドに割り当てられる数値の識別子。デフォルトは 777。
sample_size	*integer*	属性の値が非常に大きい場合に最小するサンプルのサイズ。デフォルトは 10,000。
display_additional_information	*boolean*	true の場合、メッセージのダイアログ・ボックスに追加の進捗状況の情報を表示します。
type_of_prediction	best neighbors nn-neighbors	使用する予測アルゴリズムの種類: 最適 (相関度が最も高い近隣)、近隣 (近隣の重み付き予測)、NN 近隣 (null 以外の近隣)。

Netezza Naive Bayes

次のプロパティーは、netezzanaivebayes タイプのノードで使用できます。

表 176. netezzanaivebayes ノードのプロパティー:

netezzanaivebayes ノードのプロパティー	**値**	**プロパティーの説明**
compute_probabilities	*boolean*	true の場合、予測フィールドのほか、確信度 (確率) フィールドを生成します。
use_m_estimation	*boolean*	true の場合、推定時に 0 の確立を回避する m 推定方法を使用します。

Netezza KNN

次のプロパティーは、netezzaknn タイプのノードで使用できます。

表 177. netezzaknn ノードのプロパティー:

netezzaknn ノードのプロパティー	**値**	**プロパティーの説明**
weights	*構造化*	重みを各クラスに割り当てる構造化プロパティー。
distance_measure	ユークリッド マンハッタン キャンベラ Maximum	データ・ポイント間の教理を測定する方法。
num_nearest_neighbors	*integer*	特定のケースの最近隣数。デフォルトは 3。
standardize_measurements	*boolean*	true の場合、距離の値を計算する前に連続型入力フィールドの測定を標準化します。
use_coresets	*boolean*	true の場合、大規模なデータ・セットに対して計算を高速化するコアセット・サンプリングを使用しています

Netezza 分裂クラスタリング

次のプロパティーは、netezzadivcluster タイプのノードで使用できます。

表 178. netezzadivcluster ノードのプロパティー:

netezzadivcluster ノードのプロパティー	**値**	**プロパティーの説明**
distance_measure	ユークリッド マンハッタン キャンベラ Maximum	データ・ポイント間の教理を測定する方法。
max_iterations	*integer*	モデルの学習が停止する前に、実行するアルゴリズム反復の最大回数。デフォルトは 5 です。
max_tree_depth	*integer*	データ・セットを分割することができるレベルの最大数。デフォルトは 3 です。
rand_seed	*integer*	分析を複製するために使用されるランダムシード。デフォルトは 12345。
min_instances_split	*integer*	分割可能な最小レコード数。デフォルトは 5。
level	*integer*	レコードをスコアリングする階層レベル。デフォルトは -1。

Netezza PCA

次のプロパティーは、netezzapca タイプのノードで使用できます。

表 179. netezzapca ノードのプロパティー:

netezzapca ノードのプロパティー	**値**	**プロパティーの説明**
center_data	*boolean*	true（デフォルト）の場合、このオプションをチェックした場合、分析前にデータのセンタリングを（または「平均値減算」）を実行します。
perform_data_scaling	*boolean*	true の場合、分析前にデータのスケーリングを行います。そうすることで、別の変数が異なる単位で測定されるとき、分析が恣意的でないようにします。
force_eigensolve	*boolean*	true の場合、を計算する精度が低くなってもより高速な方法を使用します。
pc_number	*integer*	データ・セットを減少する主要成分の数。デフォルトは 1。

Netezza 回帰ツリー

次のプロパティーは、netezzaregtree タイプのノードで使用できます。

表 180. netezzaregtree ノードのプロパティー:

netezzaregtree ノードのプロパティー	**値**	**プロパティーの説明**
max_tree_depth	*integer*	ルート・ノードの前にツリーが成長できるレベルの最大数。デフォルトは 10 です。
split_evaluation_measure	Variance	ツリーを分割するのに最適な場所を評価するために使用される、クラスの不純度の測定。デフォルト (現在唯一のオプション) は Variance。

表 180. *netezzaregtree* ノードのプロパティー *(続き)*:

netezzaregtree ノードのプロパティー	**値**	**プロパティーの説明**
min_improvement_splits	*number*	ツリー内に新しい分割が作成される前に純度を減少させる最小数。
min_instances_split	*integer*	分割可能な最小レコード数。
pruning_measure	mse r2 pearson spearman	剪定に使用する方法
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	デフォルトでは、allTrainingData を使用してモデルの精度を推定します。partitionTrainingData を使用して、使用する学習データの割合を、useOtherTable を使用して指定したデータベース・テーブルの学習データ・セットを使用します。
perc_training_data	*number*	prune_tree_options が PercTrainingData に設定されている場合、学習に使用するデータの割合を指定します。
prune_seed	*integer*	prune_tree_options が PercTrainingData に設定されている場合、分析結果の複製に使用するランダム シード。デフォルトは 1 です。
pruning_table	*文字列*	モデルの精度を推定するために個別の剪定データ・セットのテーブル名。
compute_probabilities	*boolean*	true の場合、割り当てられたクラスの分散が出力に含まれるべきかどうかを指定ます。

Netezza 線型回帰

次のプロパティーは、netezzalineregression タイプのノードで使用できます。

表 181. *netezzalineregression* ノードのプロパティー:

netezzalineregression ノードのプロパティー	**値**	**プロパティーの説明**
use_svd	*boolean*	true の場合、元のマトリックスの代わりに特異値分解マトリックスを使用して速度と数値の精度を向上させます。
include_intercept	*boolean*	true (デフォルト) の場合、ソリューションの全体の精度が向上します。
calculate_model_diagnostics	*boolean*	true の場合、モデルの診断を計算します。

Netezza 時系列

次のプロパティーは、netezzatimeseries タイプのノードで使用できます。

表 182. *netezzatimeseries* ノードのプロパティー:

netezzatimeseries ノードのプロパティー	**値**	**プロパティーの説明**
time_points	*field*	時系列の日付または時刻の値を含む入力フィールド。
time_series_ids	*field*	時系列 ID を含むフィールド。入力に複数の時系列が含まれる場合に使用します。
model_table	*field*	Netezza 時系列モデルが保存されるデータベース・テーブルの名前。
description_table	*field*	時系列名および説明を含む入力テーブルの名前。
seasonal_adjustment_table	*field*	指数平滑化または季節的傾向分解アルゴリズムによって計算された季節性調整値を保存する出力テーブル名。
algorithm_name	SpectralAnalysis または spectral ExponentialSmoothing または esmoothing ARIMA SeasonalTrendDecomposition または std	時系列モデリングに使用するアルゴリズム
trend_name	N A DA M DM	指数平滑化の傾向タイプ。 N - none A - 付加 DA - 付加減衰 M - 倍数 DM - 倍数減衰
seasonality_type	N A M	指数平滑化の季節性タイプ。 N - none A - 付加 M - 倍数
interpolation_method	linear cubicspline exponentialspline	使用する補間方法。
timerange_setting	SD SP	使用する時刻範囲の設定。 SD - システムが決定 (全範囲の時系列データを使用) SP - earliest_time と latest_time を使用してユーザーが指定
earliest_time	*Date*	開始時刻と終了時刻 (timerange_setting が SP の場合)。 形式: <yyyy>-<mm>-<dd>
latest_time		

表 182. *netezzatimeseries* ノードのプロパティー *(続き)*:

netezzatimeseries ノードのプロパティー	**値**	**プロパティーの説明**
arima_setting	SD SP	ARIMA アルゴリズムの設定 (algorithm_name が ARIMA に設定されている場合のみ使用されます)。 SD - system-determined SP - user-specified arima_setting = SP の場合は、次のパラメーターを使用して季節性の値と非季節性の値を設定してください。
p_symbol	less eq lesseq	ARIMA - パラメーター p、d、q、sp、sd、および sq の演算子。 less - より小さい eq - 等しい lesseq - 以下
d_symbol		
q_symbol		
sp_symbol		
sd_symbol		
sq_symbol		
p	*integer*	ARIMA - 自己相関の非季節性の度合い。
q	*integer*	ARIMA - 自己相関の非季節性導出値。
d	*integer*	ARIMA - モデル内の移動平均の非季節性数値。
sp	*integer*	ARIMA - 自己相関の季節性の度合い。
sq	*integer*	ARIMA - 自己相関の季節性導出値。
sd	*integer*	ARIMA - モデル内の移動平均の季節性数値。
advanced_setting	SD SP	詳細設定の処理方法を決定します。 SD - system-determined SP - period、units_period、forecast_setting を使用してユーザーが指定。
period	*integer*	units_period と組み合わせて指定した季節性サイクルの長さ。スペクトル解析には適用できません。

表 182. *netezzatimeseries* ノードのプロパティー *(続き)*:

netezzatimeseries ノードのプロパティー	**値**	**プロパティーの説明**
units_period	ms s min h d wk q y	period の表現単位。 ms - ミリ秒 s - 秒 min - 分 h - 時間 d - 日 wk - 週 q - 四半期 y - 年 例えば、1 週間ごとの時系列の場合は period に 1 を指定し、units_period に wk を指定します。
forecast_setting	forecasthorizon forecasttimes	予測の実行方法を指定します。
forecast_horizon	*文字列*	forecast_setting = forecasthorizon の場合に、予測の終点を指定します。 形式: <yyyy>-<mm>-<dd>
forecast_times	[{'*date*'}, {'*date*'},..., {'*date*'}]	forecast_setting = forecasttimes の場合に、予測を実行するために使用する時間を指定します。 形式: <yyyy>-<mm>-<dd>
include_history	*boolean*	過去の値を出力に含めるかどうかを示します。
include_interpolated_values	*boolean*	補間されたの値を出力に含めるかどうかを示します。include_history が false の場合は適用されません。

Netezza 一般化線型

次のプロパティーは、netezzaglm タイプのノードで使用できます。

表 183. *netezzaglm* ノードのプロパティー:

netezzaglm ノードのプロパティー	**値**	**プロパティーの説明**
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	分布のタイプ。デフォルトは bernoulli です。
dist_params	*number*	使用する分布パラメーター値。distribution が Negativebinomial の場合にのみ適用されます。

表 183. netezzaglm ノードのプロパティー (続き):

netezzaglm ノードのプロパティー	**値**	**プロパティーの説明**
trials	*integer*	distribution が Binomial の場合にのみ適用されます。ターゲット応答が一連の試行が発生するさまざまなイベントの場合、target フィールドにはイベント数、trials フィールドには試行回数が含まれます。
model_table	*field*	Netezza 一般化線型モデルが保存されるデータベース・テーブルの名前。
maxit	*integer*	アルゴリズムが実行できる反復の最大回数。デフォルトは 20 です。
eps	*number*	アルゴリズムが適合度モデルの検索を停止する最大誤差の値 (科学的表記)。デフォルトは -3、つまり 1E-3 または 0.001 です。
tol	*number*	誤差が 0 として扱われる値 (科学的表記)。デフォルトは -7、つまり 1E-7 (または 0.0000001) を下回る誤差の値が有意でないとカウントされます。
link_func	識別 inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	使用するリンク関数。デフォルトは logit です。
link_params	*number*	使用するリンク関数パラメーター値。link_function が power または oddspower の場合にのみ適用されます。
interaction	[{[*colnames1*],[*levels1*]},{[*colnames2*],[*levels2*]},...,{[*colnamesN*],[*levelsN*]},]	フィールド間の交互作用を指定します。*colnames* は、入力フィールドのリストです。また、各フィールドの *level* は常に 0 です。
intercept	*boolean*	true の場合、モデルに定数項を含みます。

Netezza モデル・ナゲットのプロパティー

次のプロパティーは、Netezza データベース・モデリング ナゲットに共通です。

表 184. Netezza モデル・ナゲットの共通プロパティー:

Netezza モデル・ナゲットの共通プロパティー	値	プロパティーの説明
connection	*文字列*	モデルが保存される Netezza データベースの接続文字列。
table_name	*文字列*	モデルが保存されるデータベース・テーブルの名前。

他のモデルナゲットのプロパティーは、対応するモデリングのノードの場合と同じです。

モデル・ナゲットのスクリプト名は以下の通りです。

表 185. Netezza モデル・ナゲットのスクリプト名:

モデル・ナゲット	スクリプト名
デシジョン・ツリー	applynetezzadectree
K-Means	applynetezzakmeans
ベイズ・ネット	applynetezzabayes
Naive Bayes	applynetezzanaivebayes
KNN	applynetezzaknn
分裂クラスタリング	applynetezzadivcluster
PCA	applynetezzapca
回帰ツリー	applynetezzaregtree
線形回帰	applynetezzalineregression
時系列	applynetezzatimeseries
一般化線型	applynetezzaglm

第 16 章 出力ノードのプロパティー

出力ノードのプロパティーは、ほかの種類のノードのプロパティーと少し異なっています。出力ノードのプロパティーは、特定のノード・オプションを参照するというよりは、参照を出力オブジェクトに格納します。このことはテーブルから値を取得して、それをストリーム・パラメーターとして設定するような場合などに役立ちます。

このセクションで、出力ノードで使用できるスクリプト用のプロパティーを説明します。

analysis ノードのプロパティー

精度分析ノードで、予測モデルの能力を評価して正確な予測を生成します。分析ノードでは、1 つ以上のモデル・ナゲットについて、予測値と実際値をさまざまな方法で比較します。また、分析ノードでは予測モデル同士を比較できます。

表 186. analysis ノードのプロパティー:

analysis ノードのプロパティー	**データ型**	**プロパティーの説明**
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。
by_fields	[*フィールド フィールド フィールド*]	
full_filename	*文字列*	ディスク、データ、または HTML の出力を選択した場合の、出力ファイルの名前。
coincidence	*boolean*	
performance	*boolean*	
evaluation_binary	*boolean*	
confidence	*boolean*	
threshold (しきい値)	*number*	
improve_accuracy	*number*	
inc_user_measure	*boolean*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	

表 186. *analysis ノードのプロパティー (続き)*:

analysis ノードのプロパティー	データ型	プロパティーの説明
user_compute	[Mean Sum Min Max SDev]	

dataaudit ノードのプロパティー

データ検査ノードでは、欠損値、外れ値、および極値に関する情報の他、各フィールドの要約統計量、ヒストグラムや棒グラフを含む、データを広範に検査するための手段を提供しています。結果は把握しやすい行列形式で表示され、ソートしたり、フルサイズのグラフやデータ準備ノードを生成することができます。

表 187. *dataaudit ノードのプロパティー*:

dataaudit ノードのプロパティー	データ型	プロパティーの説明
custom_fields	*boolean*	
fields	*[field1 ... fieldN]*	
overlay	*field*	
display_graphs	*boolean*	出力行列中のグラフ表示をオンまたはオフにするために使用されます。
basic_stats	*boolean*	
advanced_stats	*boolean*	
median_stats	*boolean*	
calculate	Count Breakdown	欠損値の計算に使用します。計算方法のいずれか、または両方を選択するか、またはどちらも選択しません。
outlier_detection_method	std iqr	外れ値および極値の検出方法を指定します。
outlier_detection_std_outlier	*number*	outlier_detection_method が std の場合、外れ値の定義に使用する数値を指定します。
outlier_detection_std_extreme	*number*	outlier_detection_method が std の場合、外れ値の定義に使用する数値を指定します。
outlier_detection_iqr_outlier	*number*	outlier_detection_method が iqr の場合、外れ値の定義に使用する数値を指定します。
outlier_detection_iqr_extreme	*number*	outlier_detection_method が iqr の場合、外れ値の定義に使用する数値を指定します。
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。

表 187. *dataaudit* ノードのプロパティー *(続き)*:

dataaudit ノードのプロパティー	データ型	プロパティーの説明
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。
paginate_output	*boolean*	output_format が HTML の場合、出力がページに分割されるようにします。
lines_per_page	*number*	paginate_output と共に使用する場合は、出力ページあたりの行数を指定します。
full_filename	*文字列*	

matrix ノードのプロパティー

クロス集計ノードで、フィールド間の関係を示すテーブルを作成します。一般的にこのノードは、2 つのシンボル値フィールドの関係を示す場合によく使用されますが、フラグ型フィールド間または数値型フィールド間の関係を示すこともできます。

表 188. *matrix* ノードのプロパティー:

matrix ノードのプロパティー	データ型	プロパティーの説明
fields	Selected Flags Numerics	
row	*field*	
column	*field*	
include_missing_values	*boolean*	ユーザーによる欠損値 (空白) とシステムによる欠損値 (ヌル) が、行と列の出力に含まれるかどうかを指定します。
cell_contents	CrossTabs Function	
function_field	*文字列*	
function	Sum Mean Min Max SDev	
sort_mode	ソートなし Ascending Descending	
highlight_top	*number*	ゼロでない場合に真 (true)。

表 188. *matrix* ノードのプロパティー *(続き)*:

matrix ノードのプロパティー	データ型	プロパティーの説明
highlight_bottom	*number*	ゼロでない場合に真 (true) 。
display	[Counts Expected Residuals (残差) RowPct ColumnPct TotalPct]	
include_totals	*boolean*	
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。Formatted と Delimited の両方が、テーブル内で行と列を入れ替える修飾子 transposed を伴うことができます。
paginate_output	*boolean*	output_format が HTML の場合、出力がページに分割されるようにします。
lines_per_page	*number*	paginate_output と共に使用する場合は、出力ページあたりの行数を指定します。
full_filename	*文字列*	

means ノードのプロパティー

平均比較ノードでは、独立したグループ間で、または関連するフィールドのペア間で著しい違いがあるかどうかを調べるために、平均を比較します。例えば、販売促進活動の前後で平均収益を比較したり、販売促進活動を受けなかった顧客と受けた顧客からの収益を比較することができます。

表 189. *means* ノードのプロパティー:

means ノードのプロパティー	データ型	プロパティーの説明
means_mode	BetweenGroups BetweenFields	データに実行する平均統計処理の種類を指定します。
test_fields	[フィールド 1 ... フィールド N]	means_mode が BetweenGroups に設定されているときのテスト・フィールドを指定します。
grouping_field	*field*	グループにまとめるフィールドを指定します。

表 189. means ノードのプロパティー (続き):

means ノードのプロパティー	データ型	プロパティーの説明
paired_fields	[{field1 field2} {field3 field4} ...]	means_mode が BetweenFields に設定されているときに使用するフィールドのペアを指定します。
label_correlations	*boolean*	相関ラベルが出力に表示されるかどうかを指定します。この設定が適用されるのは、means_mode を BetweenFields に設定した場合のみです。
correlation_mode	Probability Absolute	確率 (Probability) または絶対値 (Absolute) のどちらかで相関にラベルを付けることを指定します。
weak_label	*文字列*	
medium_label	*文字列*	
strong_label	*文字列*	
weak_below_probability	*number*	correlation_mode が Probability に設定されているときに、弱い相関の分割値を指定します。この値は、例えば 0.90 のように、0 と 1 の間にする必要があります。
strong_above_probability	*number*	強い相関の分割値。
weak_below_absolute	*number*	correlation_mode が Absolute に設定されているときに、弱い相関の分割値を指定します。この値は、例えば 0.90 のように、0 と 1 の間にする必要があります。
strong_above_absolute	*number*	強い相関の分割値。
unimportant_label	*文字列*	
marginal_label	*文字列*	
important_label	*文字列*	
unimportant_below	*number*	低いフィールド重要度の分割値。この値は、例えば 0.90 のように、0 と 1 の間にする必要があります。
important_above	*number*	
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	使用する名前。
output_mode	画面 File	出力ノードから生成された出力の出力先を指定します。
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定します。
full_filename	*文字列*	

表 189. means ノードのプロパティー (続き):

means ノードのプロパティー	データ型	プロパティーの説明
output_view	Simple (単純) Advanced	出力に単純な (Simple) ビューが表示されるか、または詳細な (Advanced) ビューが表示されるかを指定します。

report ノードのプロパティー

レポート・ノードで、固定テキスト、およびデータやデータから導かれた他の式を含む、フォーマット済みレポートを作成します。レポートの書式は、固定テキストとデータの出力構成を定義するテキスト テンプレートを使用して指定します。テンプレート内の HTML タグを使用し、また「出力」タブでオプションを設定することで、カスタムのテキスト書式設定を提供できます。テンプレート内の CLEM 式を使用して、データ値やその他の条件出力を含めることができます。

表 190. report ノードのプロパティー:

report ノードのプロパティー	データ型	プロパティーの説明
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	出力のタイプを指定するために使用されます。
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
text	*文字列*	
full_filename	*文字列*	
highlights	*boolean*	
title	*文字列*	
lines_per_page	*number*	

Routput ノードのプロパティー

R 出力ノードでは、独自のカスタム R スクリプトを使用して、データおよびモデル・スコアリングの結果を分析できます。分析はテキストまたはグラフィックで出力できます。出力はマネージャー領域の「**出力**」タブに追加されます。あるいは、出力をファイルにリダイレクトできます。

表 191. Routput ノードのプロパティー:

Routput ノードのプロパティー	データ型	プロパティーの説明
構文	*文字列*	

表 191. *Routput* ノードのプロパティー *(続き)*:

Routput ノードのプロパティー	データ型	プロパティーの説明
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*boolean*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*boolean*	
output_name	Auto Custom	
custom_name	*文字列*	
output_to	画面 File	
output_type	Graph Text	
full_filename	*文字列*	
graph_file_type	HTML COU	
text_file_type	HTML TXT COU	

setglobals ノードのプロパティー

グローバル・ノードで、データを走査し、CLEM 式で使用できる要約値を算出します。例えば、グローバル・ノードを使用して、*「年齢」* という名前のフィールドの統計量を算出し、次に CLEM 式に @GLOBAL_MEAN(年齢) 関数を挿入して*年齢* の全体的な平均を算出することができます。

表 192. *setglobals* ノードのプロパティー:

setglobals ノードのプロパティー	データ型	プロパティーの説明
globals	[Sum Mean Min Max SDev]	構造化プロパティー
clear_first	*boolean*	
show_preview	*boolean*	

simeval ノードのプロパティー

シミュレーション評価ノードは、指定された予測される対象フィールドを評価し、対象フィールドの分布と相関情報を提供します。

表 193. *simeval* ノードのプロパティー:

simeval ノードのプロパティー	データ型	プロパティーの説明
対象	*フィールド*	
iteration	*フィールド*	
presorted_by_iteration	*boolean*	
max_iterations	*数値*	
tornado_fields	*[フィールド 1...フィールド N]*	
plot_pdf	*boolean*	
plot_cdf	*boolean*	
show_ref_mean	*boolean*	
show_ref_median	*boolean*	
show_ref_sigma	*boolean*	
num_ref_sigma	*数値*	
show_ref_pct	*boolean*	
ref_pct_bottom	*数値*	
ref_pct_top	*数値*	
show_ref_custom	*boolean*	
ref_custom_values	*[数値 1...数値 N]*	
category_values	カテゴリー Probabilities Both	
category_groups	カテゴリ 反復回数	
create_pct_table	*boolean*	
pct_table	4 分位 区間(E) Custom	
pct_intervals_num	*数値*	
pct_custom_values	*[数値 1...数値 N]*	

simfit ノードのプロパティー

シミュレーション・フィッティング・ノードは、各フィールドのデータの統計的な分布を調べ、最も適合する分布を各フィールドに割り当ててシミュレーション生成ノードを生成 (または更新) します。この後、シミュレーション生成ノードを使用して、シミュレートするデータを生成することができます。

表 194. *simfit* ノードのプロパティー:

simfit ノードのプロパティー	データ型	プロパティーの説明
build	Node XMLExport Both	
use_source_node_name	*boolean*	

表 194. *simfit* ノードのプロパティー *(続き)*:

simfit ノードのプロパティー	データ型	プロパティーの説明
source_node_name	*文字列*	生成または更新される入力ノードのカスタム名。
use_cases	すべて LimitFirstN	
use_case_limit	*整数*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*整数*	
parameter_xml_filename	*文字列*	
generate_parameter_import	*boolean*	

statistics ノードのプロパティー

記述統計ノードでは、数値型フィールドに関する基本的な集計情報が提供されます。このノードで、個々のフィールドの要約統計量とフィールド間の相関が計算されます。

表 195. *statistics* ノードのプロパティー:

statistics ノードのプロパティー	データ型	プロパティーの説明
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。
full_filename	*文字列*	
examine	[*フィールド フィールド フィールド*]	
correlate	[*フィールド フィールド フィールド*]	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Probability Absolute	確率 (Probability) または絶対値 (Absolute) のどちらかで相関にラベルを付けることを指定します。
label_correlations	*boolean*	
weak_label	*文字列*	
medium_label	*文字列*	

表 195. *statistics* ノードのプロパティー *(続き)*:

statistics ノードのプロパティー	データ型	プロパティーの説明
strong_label	*文字列*	
weak_below_probability	*number*	correlation_mode が Probability に設定されているときに、弱い相関の分割値を指定します。この値は、例えば 0.90 のように、0 と 1 の間にする必要があります。
strong_above_probability	*number*	強い相関の分割値。
weak_below_absolute	*number*	correlation_mode が Absolute に設定されているときに、弱い相関の分割値を指定します。この値は、例えば 0.90 のように、0 と 1 の間にする必要があります。
strong_above_absolute	*number*	強い相関の分割値。

statisticsoutput ノードのプロパティー

Statistics 出力ノードを使用すると、IBM SPSS Statistics 手続きを呼び出し、IBM SPSS Modeler データを分析することができます。さまざまな IBM SPSS Statistics 分析手続きにアクセスできます。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

このノードのプロパティーについては、 248 ページの『statisticsoutput ノードのプロパティー』に記載されています。

table ノードのプロパティー

テーブル・ノードで、データがテーブル形式で表示されます。このデータは、ファイルにも書き込めます。この機能は、データの値を調査したり、データを読みやすい形式でエクスポートする必要がある場合に役立ちます。

表 196. *table* ノードのプロパティー:

table ノードのプロパティー	データ型	プロパティーの説明
full_filename	*文字列*	ディスク、データ、または HTML の出力を選択した場合の、出力ファイルの名前。
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。

表 196. table ノードのプロパティー (続き):

table ノードのプロパティー	**データ型**	**プロパティーの説明**
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。
transpose_data	*boolean*	エクスポート前にデータの行列を入れ替えて、行がフィールドを、列がレコードを表すようにします。
paginate_output	*boolean*	output_format が HTML の場合、出力がページに分割されるようにします。
lines_per_page	*number*	paginate_output と共に使用する場合は、出力ページあたりの行数を指定します。
highlight_expr	*文字列*	
output	*文字列*	ノードで直前に構築されたテーブルへの参照を保持する、読み取り専用プロパティー。
value_labels	*[{Value LabelString} {Value LabelString} ...]*	値のペアのためのラベルを指定します。
display_places	*integer*	フィールドが表示されるときの小数部の桁数を設定します (REAL ストレージのフィールドにのみ適用)。-1 を設定すると、ストリームのデフォルトが使用されます。
export_places	*integer*	フィールドをエクスポートするときの小数部の桁数を設定します (REAL ストレージのフィールドにのみ適用)。-1 を設定すると、ストリームのデフォルトが使用されます。
decimal_separator	DEFAULT PERIOD COMMA	フィールドの小数点記号を指定します (REAL ストレージのフィールドにのみ適用)。

表 196. table ノードのプロパティー (続き):

table ノードのプロパティー	**データ型**	**プロパティーの説明**
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	フィールドの日付形式を設定します (DATE または TIMESTAMP ストレージのフィールドにのみ適用されます)。
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"	フィールドの日付形式を設定します (TIME または TIMESTAMP ストレージのフィールドにのみ適用されます)。
column_width	*integer*	フィールドに列幅を設定します。-1 という値を指定すると、列幅は Auto に設定されます。
justify	AUTO CENTER LEFT RIGHT	フィールドに列調整を設定します。

transform ノードのプロパティー

変換ノードによって、選択フィールドに適用する前に変換の結果を選択し、視覚的に確認することができます。

表 197. transform ノードのプロパティー:

transform ノードのプロパティー	データ型	プロパティーの説明
fields	[*field1... fieldn*]	変換で使用するフィールド。
formula	All Select	すべての変換を計算するか、選択した変換を計算するかを指定します。
formula_inverse	*boolean*	逆変換を使用するかどうかを指定します。
formula_inverse_offset	*number*	式で使用するデータ・オフセットを指定します。ユーザーが指定しない限り、デフォルトで 0 に設定されます。
formula_log_n	*boolean*	$\log_n$ 変換を使用するかどうかを指定します。
formula_log_n_offset	*number*	
formula_log_10	*boolean*	$\log_{10}$ 変換を使用するかどうかを指定します。
formula_log_10_offset	*number*	
formula_exponential	*boolean*	指数変換 (e^x) を使用するかどうかを指定します。
formula_square_root	*boolean*	平方根変換を使用するかどうかを指定します。
use_output_name	*boolean*	ユーザー設定の出力名が使用されるかどうかを指定します。
output_name	*文字列*	use_output_name が真 (true) のときに、使用する名前を指定します。
output_mode	画面 File	出力ノードから生成される出力の、出力先を指定します。
output_format	HTML (*.html*) Output (*.cou*)	出力のタイプを指定するために使用されます。
paginate_output	*boolean*	output_format が HTML の場合、出力がページに分割されるようにします。
lines_per_page	*number*	paginate_output と共に使用する場合は、出力ページあたりの行数を指定します。
full_filename	*文字列*	ファイル出力に使用するファイル名を指定します。

第 17 章 エクスポート・ノードのプロパティー

エクスポート・ノードの共通プロパティー

次のプロパティーは、すべてのエクスポート・ノードに共通しています。

表 198. エクスポート・ノードの共通プロパティー:

プロパティー	値	プロパティーの説明
publish_path	*文字列*	公開されたイメージおよびパラメーター・ファイルに使用するルート名を指定します。
publish_metadata	*boolean*	イメージの入力および出力、それらのデータ・モデルを説明するメタデータ・ファイルを作成するかどうかを指定します。
publish_use_parameters	*boolean*	ストリーム・パラメーターが *.par ファイルに含まれるかどうかを指定します。
publish_parameters	*文字列のリスト*	使用するパラメーターを指定します。
execute_mode	export_data publish	ストリームを公開せずにノードを実行するかどうか、ノードの実行時にストリームを自動的に公開するかどうかを指定します。

asexport ノードのプロパティー

Analytic Server エクスポートにより、Hadoop 分散ファイル・システム (HDFS) でストリームを実行することができます。

表 199. asexport ノードのプロパティー:

asexport ノードのプロパティー	**データ型**	**プロパティーの説明**
data_source	*文字列*	データ・ソースの名前。
export_mode	*文字列*	エクスポートしたデータを既存のデータ・ソースに追加する (**append**) か、既存のデータ・ソースを上書きする (**overwrite**) かを指定します。
host	*文字列*	Analytic Server ホストの名前。
ポート	*整数*	Analytic Server が listen するポート。
tenant	*文字列*	マルチテナント環境における所属先テナントの名前。シングル・テナント環境ではデフォルトで **ibm** になります。
set_credentials	*boolean*	Analytic Server でのユーザー認証が SPSS Modeler Server と同じである場合は、これを **false** に設定してください。それ以外の場合は **true** に設定してください。

表 199. *asexport* ノードのプロパティー *(続き)*:

asexport ノードのプロパティー	データ型	プロパティーの説明
user_name	*文字列*	Analytic Server にログインするためのユーザー名。set_credentials が true の場合にのみ必要です。
パスワード	*文字列*	Analytic Server にログインするためのパスワード。set_credentials が true の場合にのみ必要です。

cognosexport ノードのプロパティー

IBM Cognos BI エクスポート・ノードは、Cognos BI データベースで読み取ることができる形式でデータをエクスポートできます。

注 : このノードの場合 Cognos 接続と ODBC 接続を定義する必要があります。

Cognos 接続

Cognos 接続のプロパティーは次のとおりです。

表 200. *cognosexport* ノードのプロパティー:

cognosexport ノードのプロパティー	データ型	プロパティーの説明
cognos_connection	*{"field","field", ... ,"field"}*	Cognos サーバーの接続の詳細を含むリストのプロパティー。形式は次のとおりです。 {"*Cognos_server_URL*", *login_mode*, "*namespace*", "*username*", "*password*"} ここで、 *Cognos_server_URL* は、エクスポートする Cognos サーバーの URL です。 *login_mode* は匿名ログインが使用されるかどうかを示し、true または false となります。true の場合、次のフィールドは "" に設定されます。 *namespace* はサーバーへのログインに使用するセキュリティー認証プロバイダを指定します。 *username* および *password* は Cognos サーバーにログインする際に使用するユーザー名とパスワードです。
cognos_package_name	*文字列*	データをエクスポートしている Cognos データ・ソース (通常はデータベース) のパスおよび名前。次に例を示します。 /Public Folders/MyPackage
cognos_datasource	*文字列*	
cognos_export_mode	Publish ExportFile	
cognos_filename	*文字列*	

ODBC 接続

ODBC 接続のプロパティーは次のセクションの databaseexport に示されているものと同じです。ただし、datasource プロパティーは有効ではありません。

databaseexport ノードのプロパティー

データベース・エクスポート・ノードで、データを ODBC 対応のリレーショナル・データ・ソースに書き込みます。ODBC データ・ソースに書き込むには、データ・ソースが存在し、そのデータ・ソースに対する書き込み権限を取得している必要があります。

表 201. databaseexport ノードのプロパティー:

databaseexport ノードのプロパティー	**データ型**	**プロパティーの説明**
datasource	*文字列*	
username	*文字列*	
password	*文字列*	
epassword	*文字列*	このスロットは、実行時に読み込み専用になります。暗号化パスワードを生成するには、「ツール」メニューの「パスワード暗号化ツール」を使用してください。 詳しくは、トピック 49 ページの『暗号化パスワードの生成』を参照してください。
table_name	*文字列*	
write_mode	Create Append Merge	
map	*文字列*	ストリーム・フィールド名をデータベース列名にマッピングします (write_mode が Merge の場合にのみ有効)。 結合の場合、すべてのフィールドをマッピングしてエクスポートする必要があります。 データベース内に存在しないフィールド名が、新しい列として追加されます。
key_fields	*[フィールド フィールド ... フィールド]*	キーに使用されるストリーム・フィールドを指定します。map プロパティーは、データベースでストリーム・フィールド内で対応する内容を表示します。
join	Database Add	

表 201. databaseexport ノードのプロパティー (続き):

databaseexport ノードのプロパティー	**データ型**	**プロパティーの説明**
drop_existing_table	*boolean*	
delete_existing_rows	*boolean*	
default_string_size	*integer*	
type		スキーマ タイプの設定に用いられる構造化プロパティー。
generate_import	*boolean*	
use_custom_create_table_command	*boolean*	*custom_create_table* スロットを使用して、標準の CREATE TABLE SQL コマンドを変更します。
custom_create_table_command	*文字列*	標準の CREATE TABLE SQL コマンドの代わりに使用する文字列コマンドを指定します。
use_batch	*boolean*	次のプロパティーは、データベースのバルク・ロード用の詳細オプションです。Use_batch に真 (true) の値を指定すると、行単位のデータベースへのコミットが無効になります。
batch_size	*number*	メモリーにコミットする前にデータベースに送信するレコード数を指定します。
bulk_loading	Off ODBC External	バルク・ロードの種類を指定します。ODBC および External 用の付加オプションを次に示します。
not_logged	*boolean*	
odbc_binding	Row Column	ODBC 経由のバルク・ロードにおける、行方向または列方向のバインドを指定します。
loader_delimit_mode	Tab Space Other	外部プログラム経由のバルク・ロードの場合に、区切り文字の種類を指定します。Other は、loader_other_delimiter プロパティーと組み合わせて選択し、コンマ (,) のような区切り文字を指定します。
loader_other_delimiter	*文字列*	
specify_data_file	*boolean*	真 (true) を設定すると、以下の data_file プロパティーが有効になります。このプロパティーには、データベースにバルク・ロードする際の書き込み先のファイル名とパスを指定することができます。
data_file	*文字列*	

表 201. *databaseexport* ノードのプロパティー *(続き)*:

databaseexport ノードのプロパティー	データ型	プロパティーの説明
specify_loader_program	*boolean*	真 (true) を設定すると、以下の loader_program プロパティーが有効になります。このプロパティーには、外部ローダー・スクリプトまたはプログラムの名前と場所を指定することができます。
loader_program	*文字列*	
gen_logfile	*boolean*	真 (true) を設定すると、以下の logfile_name が有効になります。このプロパティーには、エラー・ログを生成するための、サーバー上のファイル名を指定することができます。
logfile_name	*文字列*	
check_table_size	*boolean*	真 (true) を設定すると、IBM SPSS Modeler からエクスポートされる行数に対応してデータベースのテーブル・サイズを確実に増加させるために、テーブル検査が実施されます。
loader_options	*文字列*	ローダー・プログラムに対して、-comment および -specialdir のような、他の引数を指定します。
export_db_primarykey	*boolean*	指定されたフィールドがプライマリキーかどうかを指定します。
use_custom_create_index_command	*boolean*	true の場合、すべてのインデックスに対してカスタム SQL (ユーザー指定のSQL) を有効にします。
custom_create_index_command	*文字列*	カスタム SQL (ユーザー指定のSQL) が有効にされている場合、インデックスの作成に使用される SQL コマンドを指定します。(この値は、下に示す特定のインデックスに対して上書きできます。)
indexes.INDEXNAME.fields		必要な場合は指定されたインデックスを作成し、そのインデックスに含まれるフィールド名を一覧表示します。
indexes.INDEXNAME.use_custom_create_index_command	*boolean*	特定のインデックスに対してカスタム SQL (ユーザー指定のSQL) を有効または無効にするのに使用されます。
indexes.INDEXNAME.custom_create_ommand		指定されたインデックスに使用されるカスタム SQL (ユーザー指定のSQL) を使用します。
indexes.INDEXNAME.remove	*boolean*	true の場合、指定されたインデックスをインデックスのセットから削除します。

表 201. *databaseexport ノードのプロパティー (続き)*:

databaseexport ノードのプロパティー	データ型	プロパティーの説明
table_space	*文字列*	作成されるテーブル・スペースを指定します。
use_partition	*boolean*	分布ハッシュ・フィールドが使用されるよう指定します。
partition_field	*文字列*	分布ハッシュ・フィールドの内容を消去します。

注：一部のデータベースでは、圧縮してエクスポートするためのデータベース・テーブルの作成を指定することができます (SQL の CREATE TABLE MYTABLE (...) COMPRESS YES; と同等)。この機能をサポートするには、以下のように use_compression プロパティーと compression_mode プロパティーを指定します。

表 202. *圧縮機能を使用する場合の databaseexport ノードのプロパティー*:

databaseexport ノードのプロパティー	データ型	プロパティーの説明
use_compression	*boolean*	true に設定した場合は、圧縮によるエクスポート用のテーブルを作成します。
compression_mode	Row Page	SQL Server データベースの圧縮レベルを設定します。
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	Oracle データベースの圧縮レベルを設定します。OLTP、Query_High、Query_Low、Archive_High、Archive_Low の各値を使用するには、Oracle 11gR2 以上が必要です。

datacollectionexport ノードのプロパティー

IBM SPSS Data Collection エクスポート・ノードは、IBM SPSS Data Collection の市場調査ソフトウェアで使用する形式でデータを出力します。このノードを使用するには、IBM SPSS Data Collection Data Library がインストールされている必要があります。

表 203. *datacollectionexport ノードのプロパティー*:

datacollectionexport ノードのプロパティー	データ型	プロパティーの説明
metadata_file	*文字列*	出力するメタデータ・ファイルの名前。
merge_metadata	上書き MergeCurrent	
enable_system_variables	*boolean*	エクスポートされた *.mdd* ファイルに IBM SPSS Data Collection システム変数を含むかどうかを指定します。

表 203. *datacollectionexport* ノードのプロパティー *(続き)*:

datacollectionexport ノードのプロパティー	データ型	プロパティーの説明
casedata_file	*文字列*	ケース・データがエクスポートされる *.sav* ファイルの名前。
generate_import	*boolean*	

excelexport ノードのプロパティー

Excel エクスポート・ノードは、Microsoft Excel 形式（*.xls*）でデータを出力します。オプションで、ノードが実行されるときに自動的に Excel が起動し、エクスポートするファイルを開けるように選択できます。

表 204. *excelexport* ノードのプロパティー:

excelexport ノードのプロパティー	データ型	プロパティーの説明
full_filename	*文字列*	
excel_file_type	Excel2003 Excel2007	
export_mode	Create Append	
inc_field_names	*boolean*	フィールド名がワークシートの最初の行に表示されるかどうかを指定します。
start_cell	*文字列*	エクスポートの開始セルを指定します。
worksheet_name	*文字列*	書き込むワークシートの名前。
launch_application	*boolean*	Excel が結果のファイルで呼び出されるかどうかを指定します。Excel を起動するパスは、「ヘルパー・アプリケーション」ダイアログ・ボックス（「ツール」メニューから「ヘルパー・アプリケーション」）内で指定する必要があります。
generate_import	*boolean*	出力されたデータ・ファイルを読み込む Excel 入力ノードが生成されるかどうかを指定します。

outputfile ノードのプロパティー

ファイル・ノードでは、データが区切り文字で区切られたテキスト・ファイルへ出力されます。このことは、他の分析ソフトウェアや表計算ソフトウェアに読み込める形式でデータをエクスポートする場合に、役立ちます。

表 205. *outputfile* ノードのプロパティー:

outputfile ノードのプロパティー	データ型	プロパティーの説明
full_filename	*文字列*	出力ファイルの名前。
write_mode	上書き Append	
inc_field_names	*boolean*	
use_newline_after_records	*boolean*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*char*	
quote_mode	None Single Double Other	
other_quote	*boolean*	
generate_import	*boolean*	
エンコード	StreamDefault SystemDefault "UTF-8"	

sasexport ノードのプロパティー

SAS エクスポート・ノードで、SAS または SAS 互換ソフトウェア・パッケージで読み込むデータを、SAS 形式で出力できます。3 つの SAS ファイル形式が利用可能です。SAS for Windows/OS2、SAS for UNIX、または SAS バージョン 7/8

表 206. *sasexport* ノードのプロパティー:

sasexport ノードのプロパティー	データ型	プロパティーの説明
format	Windows UNIX SAS7 SAS8	バリアント・プロパティー・ラベル・フィールド。
full_filename	*文字列*	
export_names	NamesAndLabels NamesAsLabels	エクスポート時にフィールド名を IBM SPSS Modeler から IBM SPSS Statistics または SAS変数名に関連付けます。
generate_import	*boolean*	

statisticsexport ノードのプロパティー

Statistics エクスポート・ノードでは、IBM SPSS Statistics *.sav* 形式でデータを出力します。*.sav* ファイルは、IBM SPSS Statistics Base およびその他の製品で読み込むことができます。この形式は、IBM SPSS Modeler のキャッシュ・ファイルでも使用されます。

このノードのプロパティーについては、 249 ページの『statisticsexport ノードのプロパティー』に記載されています。

xmlexport ノードのプロパティー

XML エクスポート・ノードでは、XML 形式のファイルにデータを出力します。オプションで、エクスポートしたデータをストリームに読み込む XML 入力ノードを作成できます。

表 207. xmlexport ノードのプロパティー:

xmlexport ノードのプロパティー	データ型	プロパティーの説明
full_filename	*文字列*	(必須) XML エクスポート・ファイルの完全パスおよびファイル名。
use_xml_schema	*boolean*	XML スキーマ (XSD ファイルまたは DTD ファイル) を使用して、エクスポートされたデータの構造を制御するかどうかを指定します。
full_schema_filename	*文字列*	使用する XSD ファイルまたは DTD ファイルの完全パスおよびファイル名。use_xml_schema が true に設定されている場合にのみ必須です。
generate_import	*boolean*	エクスポートされたデータ・ファイルをストリームに読み込む XML 入力ノードを、自動的に生成します。
records	*文字列*	レコードの境界を示す XPath 式。
map	*文字列*	XML 構造にフィールド名をマッピングします。

第 18 章 IBM SPSS Statistics ノードのプロパティー

statisticsimport ノードのプロパティー

Statistics ファイル・ノードは、同じ形式を使用する IBM SPSS Statistics で使用される *.sav* ファイル形式のデータおよび IBM SPSS Modeler に保存されたキャッシュ・ファイルを読み込みます。

表 208. statisticsimport ノードのプロパティー:

statisticsimport ノードのプロパティー	**データ型**	**プロパティーの説明**
full_filename	*文字列*	パスを含む、完全なファイル名。
import_names	NamesAndLabels LabelsAsNames	変数名と変数ラベルを処理する方法。
import_data	DataAndLabels LabelsAsData	値とラベルを処理する方法。
use_field_format_for_storage	*Boolean*	インポート時に IBM SPSS Statistics フィールド形式情報を使用するかどうかを指定します。

statisticstransform ノードのプロパティー

Statistics 変換ノードは、IBM SPSS Modeler のデータ・ソースに対する IBM SPSS Statistics シンタックス・コマンドの選択を行います。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

表 209. statisticstransform ノードのプロパティー:

statisticstransform ノードのプロパティー	**データ型**	**プロパティーの説明**
syntax	*文字列*	
check_before_saving	*boolean*	項目を保存する前に、入力されたシンタックスを検証します。シンタックスが無効な場合は、エラー・メッセージを表示します。
default_include	*boolean*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。
include	*boolean*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。
new_name	*文字列*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。

statisticsmodel ノードのプロパティー

Statistics モデル・ノードを使用すると、PMML を作成する IBM SPSS Statistics 手続きを実行してデータを分析および使用することができます。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

statisticsmodel ノードのプロパティー	データ型	プロパティーの説明
syntax	*文字列*	
default_include	*boolean*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。
include	*boolean*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。
new_name	*文字列*	詳しくは、トピック 105 ページの『filter ノードのプロパティー』を参照してください。

statisticsoutput ノードのプロパティー

Statistics 出力ノードを使用すると、IBM SPSS Statistics 手続きを呼び出し、IBM SPSS Modeler データを分析することができます。さまざまな IBM SPSS Statistics 分析手続きにアクセスできます。このノードは、ライセンスが与えられた IBM SPSS Statistics のコピーが必要です。

表 210. statisticsoutput ノードのプロパティー:

statisticsoutput ノードのプロパティー	データ型	プロパティーの説明
mode	Dialog 構文	「IBM SPSS Statistics ダイアログ」オプションまたはシンタックス・エディターを選択します。
syntax	*文字列*	
use_output_name	*boolean*	
output_name	*文字列*	
output_mode	画面 File	
full_filename	*文字列*	
file_type	HTML SPV SPW	

statisticsexport ノードのプロパティー

Statistics エクスポート・ノードでは、IBM SPSS Statistics *.sav* 形式でデータを出力します。*.sav* ファイルは、IBM SPSS Statistics Base およびその他の製品で読み込むことができます。この形式は、IBM SPSS Modeler のキャッシュ・ファイルでも使用されます。

表 211. statisticsexport ノードのプロパティー:

statisticsexport ノードのプロパティー	**データ型**	**プロパティーの説明**
full_filename	*文字列*	
launch_application	*boolean*	
export_names	NamesAndLabels NamesAsLabels	エクスポート時にフィールド名を IBM SPSS Modeler から IBM SPSS Statistics または SAS変数名に関連付けます。
generate_import	*boolean*	

第 19 章 スーパーノードのプロパティー

スーパーノード固有のプロパティーを次の表に示します。共通のノード・プロパティーもスーパーノードに適用されることに注意してください。

表 212. ターミナル・スーパーノードのプロパティー:

プロパティー名	プロパティーの種類/値のリスト	プロパティーの説明
execute_method	Script Normal	
script	*文字列*	
script_language	Python レガシー	スーパーノード・スクリプトのスクリプト言語を設定します。

スーパーノードのパラメーター

スクリプトを使用すると、ストリーム・パラメーターの変更に使用する関数と同じ関数を使用してスーパーノードのパラメーターを作成または設定することができます。詳しくは、トピック 42 ページの『ストリーム、セッション、およびスーパーノード・パラメーター』を参照してください。

カプセル化ノードのプロパティー設定

スーパーノード内のノードにプロパティーを設定するには、そのスーパーノードが所有するダイアグラムにアクセスして、各種の find メソッド (findByName() や findByID() など) を使用してノードを探す必要があります。例えば、単一のデータ型ノードを含むスーパーノード・スクリプトでは以下のようになります。

```
supernode = modeler.script.supernode()
diagram = supernode.getCompositeProcessorDiagram()
# Find the type node within the supernode internal diagram
typenode = diagram.findByName("type", None)
typenode.setKeyedProperty("direction", "Drug", "Input")
typenode.setKeyedProperty("direction", "Age", "Target")
```

スーパーノード・スクリプトの制約。スーパーノードは、他のストリームを操作したり、現在のストリームを変更することはできません。

付録 A. ノード名のリファレンス

ここでは、IBM SPSS Modeler のノードのスクリプト名のリファレンスを提供します。

モデル・ナゲット名

モデル・ナゲット (生成されたモデル) は、ノード・オブジェクトと出力オブジェクトと同様に、その種類で参照できます。 次の表に、モデル・オブジェクトの参照名を一覧表示します。

これらの名前は、IBM SPSS Modeler ウィンドウの右上隅にある「モデル」パレット内のモデル・ナゲットを参照するために、特に使用されます。 スコアリングの目的でストリームに追加されたモデル・ノードを参照するには、`apply...` の接頭辞が付いた別の名前セットが使用されます。 詳しくは、トピック 183 ページの『第 14 章 モデル・ナゲット・ノードのプロパティー』を参照してください。

注: 通常の状況では、名前および 種類の両方でモデルを参照することが、混乱を避けるために推奨されます。

表 213. モデル・ナゲット名 (「モデル作成」パレット) :

モデル名	モデル
anomalydetection	異常値
Apriori	Apriori
autoclassifier	自動分類
autocluster	自動クラスター
autonumeric	自動数値
bayesnet	ベイズ・ネットワーク
c50	C5.0
carma	Carma
cart	C&R Tree
chaid	CHAID
coxreg	Cox 回帰
decisionlist	ディシジョン・リスト
discriminant	判別
因子	因子分析
featureselection	フィールド選択
genlin	一般化線型回帰
glmm	GLMM
kmeans	K-Means
knn	*k* 最近隣法
kohonen	Kohonen
線型	線型
logreg	ロジスティック回帰
neuralnetwork	ニューラル・ネットワーク

表 213. モデル・ナゲット名 (「モデル作成」パレット) (続き):

モデル名	モデル
quest	QUEST
regression	線型回帰
sequence	シーケンス
slrm	自己学習応答モデル
statisticsmodel	IBM SPSS Statistics モデル
svm	Support Vector Machine
timeseries	時系列
TwoStep	TwoStep

表 214. モデル・ナゲット名 (「データベース・モデリング」パレット):

モデル名	モデル
db2imcluster	IBM ISW クラスタリング
db2imlog	IBM ISW ロジスティック回帰
db2imnb	IBM ISW Naive Bayes
db2imreg	IBM ISW 回帰
db2imtree	IBM ISW デシジョン・ツリー
msassoc	MS アソシエーション・ルール
msbayes	MS Naive Bayes
mscluster	MS クラスタリング
mslogistic	MS Logistic Regression
msneuralnetwork	MS Neural Network
msregression	MS Linear Regression
mssequencecluster	MS Sequence Clustering
mstimeseries	MS Time Series
mstree	MS ディシジョン・ツリー
netezzabayes	Netezza ベイズ・ネットワーク
netezzadectree	Netezza デシジョン・ツリー
netezzadivcluster	Netezza 分裂クラスタリング
netezzaglm	Netezza 一般化線型
netezzakmeans	Netezza K-Means
netezzaknn	Netezza KNN
netezzalineregression	Netezza 線型回帰
netezzanaivebayes	Netezza Naive Bayes
netezzapca	Netezza PCA
netezzaregtree	Netezza 回帰ツリー
netezzatimeseries	Netezza 時系列
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Oracle ディシジョン・ツリー
oraglm	Oracle GLM

表 214. モデル・ナゲット名 (「データベース・モデリング」パレット) (続き):

モデル名	モデル
orakmeans	Oracle *k*-Means
oranb	Oracle Naive Bayes
oranmf	Oracle NMF
oraocluster	Oracle O-Cluster
orasvm	Oracle SVM

重複するモデル名の回避

生成されたモデルを操作するのにスクリプトを使用する場合、重複するモデル名を使用していると、スクリプトがあいまいになることに注意する必要があります。 これを避けるために、スクリプト作成時に、生成されたモデルには一意の名前を使用することをお勧めします。

重複するモデル名に関するオプションを設定するには

1. メニューから次の項目を選択します。

   「ツール」 > 「ユーザー・オプション」
2. **「通知」**タブをクリックします。
3. 生成されたモデルに対して重複する名前を禁止するには、**「前のモデルを置換」**を選択します。

あいまいなモデルの参照がある場合、スクリプト実行の動作は SPSS Modeler と IBM SPSS Collaboration and Deployment Services との間で異なります。SPSS Modeler クライアントには自動的に同じ名前を持つモデルを置き換えるオプション「以前のモデルを置き換える」があります (例えば、スクリプトをループで反復して随時異なる名前を作成)。しかし、このオプションは、同じスクリプトが IBM SPSS Collaboration and Deployment Services で実行される場合は使用できません。ループの終了前に、モデルに対するあいまいな参照を回避するために各反復で生成されるモデルの名前を変更するか、現在のモデルをクリアすることにより (`clear generated palette` 文の追加など)、この状況を回避することができます。

出力形式名

次の表に、すべての出力オブジェクトの形式と、それを作成するノードを一覧表示します。 各タイプの出力オブジェクトで使用できるエクスポート形式の完全なリストについては、出力タイプを作成するノードのプロパティーの説明 (121 ページの『グラフ作成ノードの共通のプロパティー』と 223 ページの『第 16 章 出力ノードのプロパティー』) を参照してください。

表 215. 出力オブジェクトの種類と、そのオブジェクトを作成するノード:

出力オブジェクトの種類	ノード
analysisoutput	分析
collectionoutput	集計
dataauditoutput	データ検査
distributionoutput	分布
evaluationoutput	評価
histogramoutput	ヒストグラム
matrixoutput	クロス集計
meansoutput	平均

表 215. 出力オブジェクトの種類と、そのオブジェクトを作成するノード (続き):

出力オブジェクトの種類	ノード
multiplotoutput	マルチ散布図
plotoutput	作図
qualityoutput	品質
reportdocumentoutput	このオブジェクトの種類はノードからのものではなく、プロジェクト・レポートに作成された出力です。
reportoutput	レポート
statisticsprocedureoutput	StatisticsOutput
statisticsoutput	記述統計
tableoutput	表
timeplotoutput	時系列グラフ
weboutput	Web

付録 B. 従来のスクリプトから Python スクリプトへの移行

従来のスクリプトの移行の概要

ここでは、IBM SPSS Modeler での Python スクリプトと従来のスクリプトの違いを要約し、従来のスクリプトを Python スクリプトに移行する方法について説明します。また、SPSS Modeler の標準的な従来のコマンドと、同等の Python コマンドのリストも示します。

一般的な差異

従来のスクリプトの設計の大部分は、OS コマンド・スクリプトが基になっています。従来のスクリプトは、行指向であり、一部のブロック構造 (`if...then...else...endif` や、`for...endfor` など) があるとしても、インデントには一般に意味がありません。

Python スクリプトでは、インデントには意味があり、同一の論理ブロックに属する複数の行は、同じレベルにインデントされている必要があります。

注: Python コードをコピーして貼り付ける場合は、注意が必要です。タブを使用してインデントされている行は、エディター上で、スペースを使用してインデントされている行と同じように見える場合があります。しかし、これらの行が同じインデントであるとは見なされないため、Python スクリプトはエラーを生成します。

スクリプト・コンテキスト

スクリプト・コンテキストは、スクリプトを実行する環境 (例えば、スクリプトを実行するストリームやスーパーノード) を定義します。従来のスクリプトでは、コンテキストは暗黙的です。つまり、例えば、ストリーム・スクリプト内のノード参照は、そのスクリプトを実行するストリーム内にあると想定されます。

Python スクリプトでは、スクリプト・コンテキストは、`modeler.script` モジュールによって明示的に提供されます。例えば、Python ストリーム・スクリプトは、以下のコードを使用して、スクリプトを実行するストリームにアクセスできます。

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

ストリームに関連した関数は、返されたオブジェクトによって呼び出すことができます。

コマンドと関数

従来のスクリプトは、コマンド指向です。つまり、スクリプトの各行は、実行する必要があるコマンドが先頭にあり、パラメーターが後に続きます。例えば、以下のとおりです。

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python は、通常、関数を定義するオブジェクト (モジュール、クラス、またはオブジェクト) によって起動される関数を使用します。例えば、以下のとおりです。

```
stream = modeler.script.stream()
typenode = stream.findByName("type", "Type)
filternode = stream.findByName("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

リテラルとコメント

IBM SPSS Modeler でよく使用される一部のリテラル・コマンドおよびコメント・コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 216. リテラルとコメントの従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
整数。例: `4`	同じ
浮動小数点数。例: `0.003`	同じ
単一引用符で囲まれた文字列。例: `'Hello'`	同じ **注:** 非 ASCII 文字が含まれている文字列リテラルには、接頭辞 `u` を付けて、Unicode として表します。
二重引用符で囲まれた文字列。例: `"Hello again"`	同じ **注:** 非 ASCII 文字が含まれている文字列リテラルには、接頭辞 `u` を付けて、Unicode として表します。

長い文字列。例:
`"""This is a string`
`that spans multiple`
`lines"""`
	同じ

リスト。例: `[1 2 3]`	`[1, 2, 3]`
変数の参照。例: `set x = 3`	`x = 3`

行の継続 (¥)。例:
`set x = [1 2 ¥`
`  3 4]`
	`x = [ 1, 2,¥`
	`  3, 4]`

ブロックのコメント。例:
`/* This is a long comment`
`over a line. */`
	`/* This is a long comment`
	`over a line. */`

行のコメント。例: `set x = 3  # make x 3`	`x = 3  # make x 3`
`undef`	`None`
`true`	`True`
偽	偽

演算子

IBM SPSS Modeler でよく使用される一部の演算子コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 217. 演算子の従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
NUM1 + NUM2 LIST + ITEM LIST1 + LIST2	NUM1 + NUM2 LIST.append(ITEM) LIST1.extend(LIST2)
NUM1 – NUM2 LIST - ITEM	NUM1 – NUM2 LIST.remove(ITEM)
NUM1 * NUM2	NUM1 * NUM2
NUM1 / NUM2	NUM1 / NUM2
= ==	==
/= /==	!=
X ** Y	X ** Y
X < Y X <= Y X > Y X >= Y	X < Y X <= Y X > Y X >= Y
X div Y X rem Y X mod Y	X // Y X % Y X % Y
および または not(EXPR)	および または not EXPR

条件とループ

IBM SPSS Modeler でよく使用される一部の条件コマンドおよびループ・コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 218. 条件とループの従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	
	or
	
	VAR = INT1
	while VAR <= INT2:
	...
	VAR += 1

for VAR in LIST ... endfor	for VAR in LIST: ...

表 218. 条件とループの従来のスクリプトから Python スクリプトへのマッピング (続き):

従来のスクリプト	Python スクリプト

for VAR in_fields_to NODE
  ...
endfor
	for VAR in NODE.getInputDataModel():
	  ...

for VAR in_fields_at NODE
  ...
endfor
	for VAR in NODE.getOutputDataModel():
	  ...

if...then
  ...
elseif...then
  ...
else
  ...
endif
	if ...:
	  ...
	elif ...:
	  ...
	else:
	  ...

with TYPE OBJECT ... endwith	同等機能なし
var VAR1	変数宣言は不要

変数(V)

従来のスクリプトでは、変数は参照される前に宣言します。例えば、以下のとおりです。

```
var mynode
set mynode = create typenode at 96 96
```

Python スクリプトでは、変数は初回の参照時に作成されます。例えば、以下のとおりです。

```
mynode = stream.createAt("type", "Type", 96, 96)
```

従来のスクリプトでは、変数の参照は ^ 演算子を使用して明示的に削除する必要があります。例えば、以下のとおりです。

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

ほとんどのスクリプト言語と同様、Python スクリプトでは、これは不要です。例えば、以下のとおりです。

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

ノード、出力、およびモデルの各タイプ

従来のスクリプトのさまざまなオブジェクト・タイプ (ノード、出力、およびモデル) では、通常、タイプがオブジェクトのタイプに追加された形になっています。例えば、フィールド作成 (Derive) ノードのタイプは、derivenode です。

```
set feature_name_node = create derivenode at 96 96
```

Python の IBM SPSS Modeler API には、node 接尾辞が含まれないため、フィールド作成ノード (Derive) のタイプは、derive です。例えば、以下のとおりです。

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

従来のスクリプトと Python スクリプトでのタイプ名の唯一の違いは、タイプ接尾辞がないことです。

プロパティー名

プロパティー名は、従来のスクリプトと Python スクリプトで同じです。例えば、可変長ファイル・ノードでは、ファイルの場所を定義するプロパティーは、両方のスクリプト環境で full_filename です。

ノードの参照

多くの従来のスクリプトは、暗黙の検索を使用して、変更するノードを見つけてアクセスします。例えば、以下のコマンドは、ラベル「Type」を使用して、現行ストリームの中でデータ型ノードを検索し、「Age」フィールドの方向 (またはモデル作成の役割) を Input に、「Drug」フィールドを Target (予測される値) に設定します。

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

Python スクリプトでは、プロパティー値を設定するための関数を呼び出す前に、ノード・オブジェクトを明示的に位置指定する必要があります。例えば、以下のとおりです。

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

注: この場合、"Target" を文字列引用符で囲む必要があります。

Python スクリプトは、ModelingRole 列挙を modeler.api パッケージで使用することもできます。

Python スクリプトのバージョンは、より冗長な場合がありますが、ノードの検索は通常 1 回のみ行われるため、ランタイム・パフォーマンスが良くなります。従来のスクリプトの例では、ノードの検索は、コマンドごとに行われます。

ID によるノードの検索もサポートされています (ノード ID は、ノード・ダイアログの「注釈」タブで確認できます)。例えば、従来のスクリプトでは、以下のようにします。

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

以下のスクリプトは、Python スクリプトを使用した場合の同じ例です。

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

プロパティーの取得と設定

従来のスクリプトは、set コマンドを使用して、値を割り当てます。 set コマンドの後ろに、プロパティー定義を続けることができます。以下のスクリプトは、プロパティーを設定するための 2 つの有効な形式を示しています。

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

Python スクリプトでは、関数 setPropertyValue() と setKeyedPropertyValue() を使用して、同じ結果が得られます。例えば、以下のとおりです。

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

従来のスクリプトでは、get コマンドを使用して、プロパティー値にアクセスできます。例えば、以下のとおりです。

```
var n v
set n = get node :filternode
set v = ^n.name
```

Python スクリプトでは、関数 getPropertyValue() を使用して、同じ結果が得られます。例えば、以下のとおりです。

```
n = stream.findByName("filter", None)
v = n.getPropertyValue("name")
```

ストリームの編集

従来のスクリプトでは、create コマンドを使用して、新しいノードを作成します。例えば、以下のとおりです。

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

Python スクリプトでは、ノードを作成するためのさまざまなメソッドがストリームに用意されています。例えば、以下のとおりです。

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

従来のスクリプトでは、connect コマンドを使用して、ノード間のリンクを作成します。例えば、以下のとおりです。

```
connect ^agg to ^select
```

Python スクリプトでは、link メソッドを使用して、ノード間のリンクを作成します。例えば、以下のとおりです。

```
stream.link(agg, select)
```

従来のスクリプトでは、disconnect コマンドを使用して、ノード間のリンクを削除します。例えば、以下のとおりです。

```
disconnect ^agg from ^select
```

Python スクリプトでは、unlink メソッドを使用して、ノード間のリンクを削除します。例えば、以下のとおりです。

```
stream.unlink(agg, select)
```

従来のスクリプトでは、position コマンドを使用して、ストリーム・キャンバスにノードを配置したり、他のノード間にノードを配置したりします。例えば、以下のとおりです。

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

Python スクリプトでは、2 つの異なるメソッド setXYPosition と setPositionBetween を使用して、同じ結果が得られます。以下に例を示します。

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

ノード操作

IBM SPSS Modeler でよく使用される一部のノード操作コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 219. ノード操作の従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
create *nodespec* at x y	*stream*.create(*type*, *name*) *stream*.createAt(*type*, *name*, x, y) *stream*.createBetween(*type*, *name*, preNode, postNode) *stream*.createModelApplier(*model*, *name*)
connect *fromNode* to *toNode*	*stream*.link(*fromNode*, *toNode*)
delete *node*	*stream*.delete(*node*)
disable *node*	*stream*.setEnabled(*node*, False)
enable *node*	*stream*.setEnabled(*node*, True)
disconnect *fromNode* from *toNode*	*stream*.unlink(*fromNode*, *toNode*) *stream*.disconnect(*node*)
duplicate *node*	*node*.duplicate()
execute *node*	*stream*.runSelected(*nodes*, *results*) *stream*.runAll(*results*)
flush *node*	*node*.flushCache()
position *node* at x y	*node*.setXYPosition(x, y)
position *node* between *node1* and *node2*	*node*.setPositionBetween(*node1*, *node2*)
rename *node* as *name*	*node*.setLabel(*name*)

ループ

従来のスクリプトでは、サポートされている主なループ・オプションが 2 つあります。

- *カウント型* ループ。インデックス変数が、2 つの整数の境界の間で変化します。
- *シーケンス型* ループ。一連の値をループして、現在の値をループ変数にバインドします。

以下のスクリプトは、従来のスクリプトでのカウント型ループの例です。

```
for i from 1 to 10
 println ^i
endfor
```

以下のスクリプトは、従来のスクリプトでのシーケンス型ループの例です。

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

以下のような他のタイプのループも使用可能です。

- モデル・パレットのモデル、または出力パレットの出力を反復する。
- ノードに入るフィールドまたはノードから出るフィールドを反復する。

Python スクリプトでも、さまざまなタイプのループをサポートしています。以下のスクリプトは、Python スクリプトでのカウント型ループの例です。

```
i = 1
while i <= 10:
 print i
 i += 1
```

以下のスクリプトは、Python スクリプトでのシーケンス型ループの例です。

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

シーケンス型ループは非常に柔軟であり、IBM SPSS Modeler API メソッドと組み合わせることにより、従来のスクリプトの大部分のユース・ケースをサポートできます。以下の例は、Python スクリプトでシーケンス型ループを使用して、ノードから出るフィールドを反復する方法を示しています。

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

ストリームの実行

ストリームの実行中に、生成されたモデルまたは出力オブジェクトが、いずれかのオブジェクト・マネージャーに追加されます。従来のスクリプトでは、スクリプトは、作成されたオブジェクトをオブジェクト・マネージャーから位置指定するか、生成された最新の出力に、その出力を生成したノードからアクセスする必要があります。

Python でのストリームの実行は、実行により生成されたモデルまたは出力オブジェクトが、実行関数に渡されるリストに返されるという点で異なります。このため、ストリームの実行結果に、より簡単にアクセスできます。

従来のスクリプトは、以下の 3 つのストリーム実行コマンドをサポートしています。

- `execute_all` は、ストリーム内のすべての実行可能ターミナル・ノードを実行します。
- `execute_script` は、スクリプト実行の設定に関係なく、ストリーム・スクリプトを実行します。
- `execute` *`node`* は、指定したノードを実行します。

Python スクリプトは、以下のような同様の関数をサポートしています。

- *`stream`*`.runAll(`*`results-list`*`)` は、ストリーム内のすべての実行可能ターミナル・ノードを実行します。
- *`stream`*`.runScript(`*`results-list`*`)` は、スクリプト実行の設定に関係なく、ストリーム・スクリプトを実行します。
- *`stream`*`.runSelected(`*`node-array`*`, `*`results-list`*`)` は、指定したノードのセットを、指定した順に実行します。
- *`node`*`.run(`*`results-list`*`)` は、指定したノードを実行します。

従来のスクリプトでは、オプションの整数コードを指定した `exit` コマンドを使用して、ストリームの実行を終了できます。例えば、以下のとおりです。

```
exit 1
```

Python スクリプトでは、以下のスクリプトを使用して、同じ結果が得られます。

```
modeler.script.exit(1)
```

ファイル・システムおよびリポジトリーによるオブジェクトへのアクセス

従来のスクリプトでは、open コマンドを使用して、既存のストリーム、モデル、または出力オブジェクトを開くことができます。例えば、以下のとおりです。

```
var s
set s = open stream "c:/my streams/modeling.str"
```

Python スクリプトには、セッションからアクセス可能で、同じような作業を実行できる TaskRunner クラスがあります。例えば、以下のとおりです。

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

従来のスクリプトを使用してオブジェクトを保存するには、save コマンドを使用します。例えば、以下のとおりです。

```
save stream s as "c:/my streams/new_modeling.str"
```

同等の Python スクリプトのアプローチでは、TaskRunner クラスを使用します。例えば、以下のとおりです。

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

IBM SPSS Collaboration and Deployment Services Repository ベースの操作は、retrieve および store コマンドを使用することよって、従来のスクリプトでサポートされています。例えば、以下のとおりです。

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

Python スクリプトでは、セッションに関連付けられているリポジトリー・オブジェクトによって、同等の機能にアクセスできます。

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

注: リポジトリーにアクセスするには、有効なリポジトリー接続を使用してセッションが構成されている必要があります。

ストリーム操作

IBM SPSS Modeler でよく使用される一部のストリーム操作コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 220. ストリーム操作の従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
create stream *DEFAULT_FILENAME*	*taskrunner*.createStream(*name*, *autoConnect*, *autoManage*)
close stream	*stream*.close()
clear stream	*stream*.clear()
get stream *stream*	同等機能なし
load stream *path*	同等機能なし
open stream *path*	*taskrunner*.openStreamFromFile(*path*, *autoManage*)

表 220. ストリーム操作の従来のスクリプトから Python スクリプトへのマッピング (続き):

従来のスクリプト	Python スクリプト
save *stream* as *path*	*taskrunner*.saveStreamToFile(*stream*, *path*)
retreive stream *path*	*repository*.retreiveStream(*path*, *version*, *label*, *autoManage*)
store *stream* as *path*	*repository*.storeStream(*stream*, *path*, *label*)

モデルの操作

IBM SPSS Modeler でよく使用される一部のモデル操作コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 221. モデル操作の従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
open model *path*	*taskrunner*.openModelFromFile(*path*, *autoManage*)
save *model* as *path*	*taskrunner*.saveModelToFile(*model*, *path*)
retrieve model *path*	*repository*.retrieveModel(*path*, *version*, *label*, *autoManage*)
store *model* as *path*	*repository*.storeModel(*model*, *path*, *label*)

ドキュメント出力操作

IBM SPSS Modeler でよく使用される一部のドキュメント出力操作コマンドには、Python スクリプトの同等コマンドがあります。これは、SPSS Modeler の既存の従来のスクリプトを、IBM SPSS Modeler 16 で使用できるように、Python スクリプトに変換するのに役立ちます。

表 222. ドキュメント出力操作の従来のスクリプトから Python スクリプトへのマッピング:

従来のスクリプト	Python スクリプト
open output *path*	*taskrunner*.openDocumentFromFile(*path*, *autoManage*)
save *output* as *path*	*taskrunner*.saveDocumentToFile(*output*, *path*)
retrieve output *path*	*repository*.retrieveDocument(*path*, *version*, *label*, *autoManage*)
store *output* as *path*	*repository*.storeDocument(*output*, *path*, *label*)

従来のスクリプトと Python スクリプトのその他の違い

レガシー・スクリプトは、IBM SPSS Modeler プロジェクトの操作をサポートしています。 Python スクリプトは、現在、これをサポートしていません。

従来のスクリプトは、ステート型 オブジェクト (ストリームおよびモデルの組み合わせ) をいくらかサポートしています。ステート型オブジェクトは、IBM SPSS Modeler 8.0 以降、廃止されました。 Python スクリプトは、ステート型オブジェクトをサポートしていません。

Python スクリプトは、従来のスクリプトでは使用できない、以下の追加の機能を提供しています。

- クラス定義と関数定義

- エラー処理
- より高度な入出力サポート
- 外部のサード・パーティー・モジュール

特記事項

本書は IBM が世界各国で提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Centrino、Intel Centrino ロゴ、Celeron、Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

A

B

C

D

E

F

G

H

O

P

Q

R

S

T

U

V

W

X

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21