IntegratedML の使用

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

IntegratedML の使用
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

図一覧

1

IntegratedML の概要

IntegratedML は、自動機械学習機能を SQL から直接使用して予測モデルを作成および使用できる InterSystems IRIS® の機能です。

1.1 目的

成功している組織は、大量のデータを効果的に利用するアプリケーションを開発する必要性を認識しています。そして、機械学習を使用して大規模なデータセットから予測モデルをトレーニングし、そのデータに基づいて重要な意思決定を行いたいと考えています。このことから、機械学習モデルを構築するための専門知識を持たない組織はきわめて不利な状況に置かれています。インターシステムズが IntegratedML を作成した理由はここにあります。

IntegratedML では、開発者やデータ・アナリストは、特徴量エンジニアリングや機械学習アルゴリズムに関する専門知識がなくても、SQL 環境内に機械学習モデルを導入することができます。IntegratedML を使用すると、開発者は SQL クエリを使用して機械学習モデルを作成、トレーニング、検証、および実行できます。

IntegratedML により、機械学習を使用する際の参入障壁が著しく下がり、未加工データから実装済みモデルへの迅速な移行が可能になります。IntegratedML は、データ・サイエンティストに取って代わるのではなく、データ・サイエンティストを補完することを目的としています。

1.2 機械学習の概要

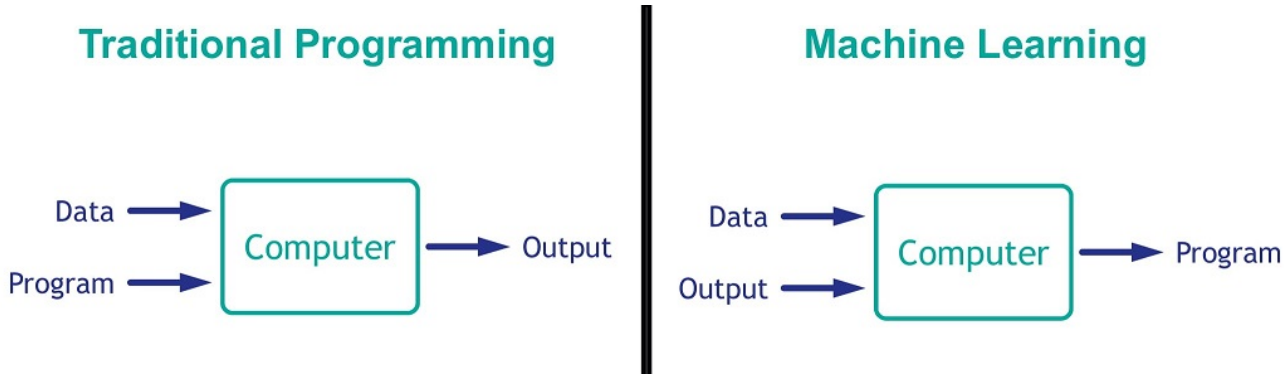
IntegratedML を理解するには、まず、よく使用されるいくつかの用語を理解している必要があります。

- 機械学習
- モデル
- トレーニング
- 特徴とラベル
- モデルの検証

機械学習とは

機械学習とは、データからパターンを識別および抽出して、予測モデルを構築して使用するコンピュータ・アルゴリズムの研究分野です。

図 1-1: 従来のプログラミングと機械学習



従来のプログラミングでは、入力データに対して実行すると目的の出力が生成されるプログラムを手動で開発していました。機械学習では、コンピュータがサンプル・データと既知の（または予想される）出力を取得してプログラム（この場合、予測モデル）を開発し、その後さらなるデータに対してそのプログラムを実行できます。

モデルのトレーニング

トレーニング・プロセスとは、機械学習アルゴリズムが予測モデルを開発する方法です。アルゴリズムはサンプル・データ、すなわちトレーニング・データを使用して、入力を目的の出力にマッピングするパターンを識別します。これらの入力（特徴）と出力（ラベル）は、データ・セットの列です。トレーニングされた機械学習モデルでは、特徴と生成されたラベルとの間に、アルゴリズムで導出された関係があります。

モデルの検証

モデルをトレーニングした後、導入する前にモデルを検証して、トレーニングに使用したデータ以外でもそのモデルが有用であることを確認できます。モデルの検証とは、モデルの出力と実際のデータの結果を比較して、モデルの予測パフォーマンスを評価するプロセスです。モデルのトレーニングにはトレーニング・データを使用しましたが、検証にはテスト・データを使用します。最も単純なケースでは、テスト・データセットは、トレーニング・データとは別に確保しておいた元のデータセットのデータです。

モデルの使用

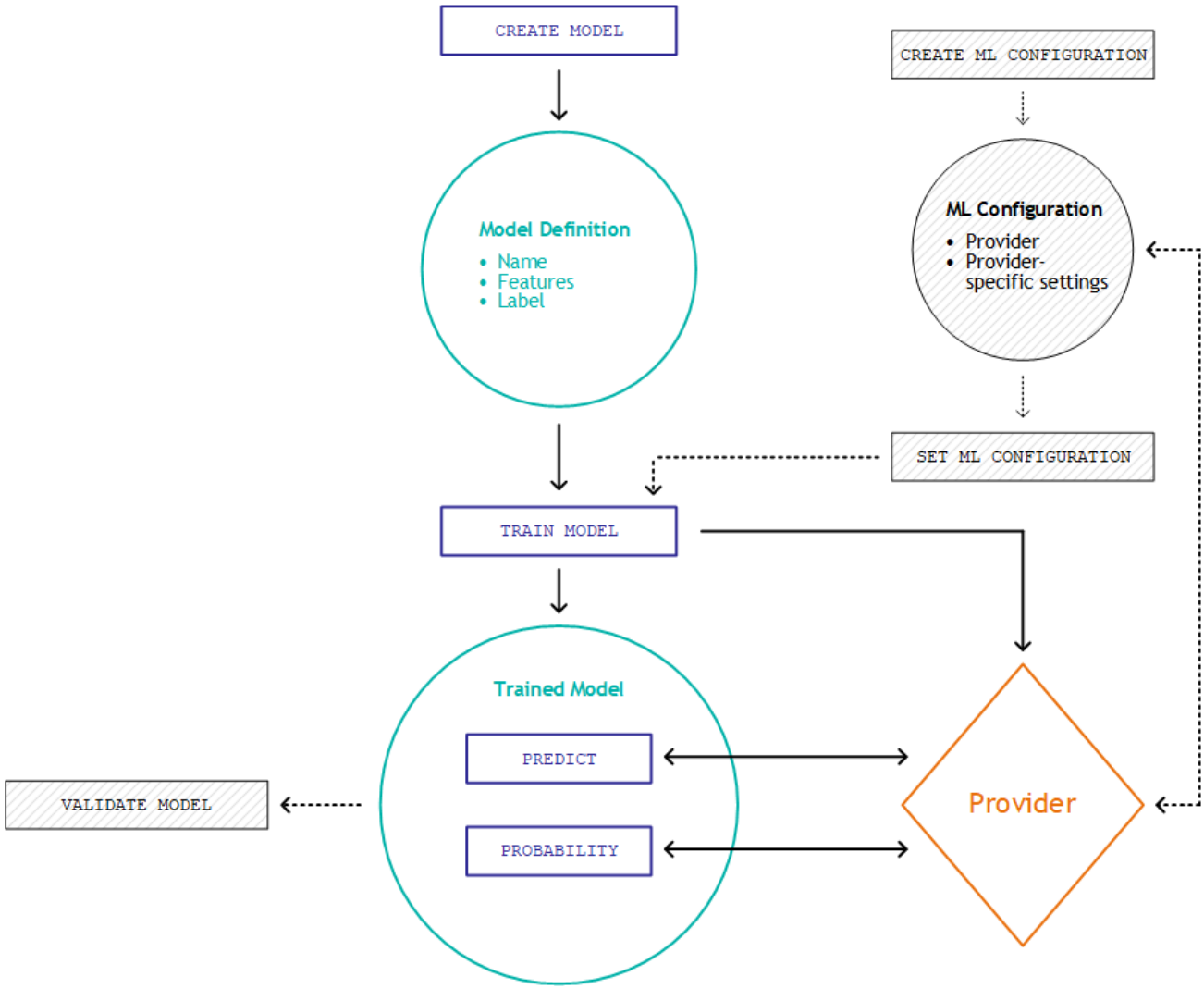
トレーニングされた機械学習モデルを使用し、新しいデータに基づいて予測を行います。このデータは、トレーニング・データおよびテスト・データと同じ特徴を収めていますが、ラベル列はありません。ラベルはモデルの出力であるからです。

2

IntegratedML の基本

IntegratedML は、自動機械学習機能を SQL から直接使用して予測モデルを作成および使用できる InterSystems IRIS® の機能です。

図 2-1: IntegratedML のワークフロー



1. IntegratedML を使用するには、まず入力フィールド (特徴)、予測されたフィールド (ラベル)、およびこれらのフィールドのデータ型に関するメタデータを含むモデル定義を指定します。モデル定義にはデータの構造のみが格納され、データ自体は格納されません。
   - オプション ー トレーニングを実行するプロバイダを指定する ML 構成を選択できます。トレーニングの前にこの構成をカスタマイズすることも、何もアクションを行わずにシステムの既定の構成を使用することもできます。
2. アクティブな ML 構成で指定されたプロバイダを使用し、データを使ってモデルをトレーニングします。プロバイダは構造化プロセスを使用して、さまざまな機械学習モデル・タイプ (線形回帰、ランダム・フォレストなど) のパフォーマンスをデータセットと比較して、適切なモデルを返します。このプロセスはプロバイダによって異なります。
   - オプション ー モデルのトレーニング後、テスト・データを使用してモデルを検証し、モデルの予測パフォーマンスを評価できます。
3. これで、トレーニングされたモデルを SQL 関数で呼び出して、データに基づいて予測をすることができます。

定義

IntegratedML 固有の用語については、以下の定義を参照してください。

モデル

モデルは、IntegratedML で使用されるプライマリ・オブジェクトです。モデル・エンティティには、次の 2 つのタイプがあります。

- モデル定義 ー IntegratedML では、モデルはテーブルやインデックスと同じようにデータベース・スキーマの一部です。CREATE MODEL 文は、新しいモデル定義をスキーマに導入します。このモデル定義では、トレーニングに使用される ML 構成と共に特徴、ラベル、およびデータ型を指定します。
- トレーニングされたモデル ー TRAIN MODEL コマンドはモデル定義を使用し、構成で指定されたプロバイダを使ってモデルをトレーニングします。このトレーニングされたモデルを使用して、データに基づいて予測を行います。

プロバイダ

複数の組織が ML-as-a-Service を提供しており、顧客のデータセットに基づいて機械学習モデルを開発するためのツールや計算能力を提供しています。これらの自動化ソリューションは一般にスタンドアロン・アプリケーションとして提供され、顧客のデータセットに直接接続するフレームワークはありません。このため、機械学習フレームワークに応じて変化する条件に従って、データを他のワークフローにエクスポートする作業が必要になります。

IntegratedML は、自動機械学習機能を InterSystems IRIS® データ・プラットフォーム内に直接組み込み、InterSystems IRIS のデータと自動化されたワークフローとの間の接続を容易にすることで、これらの問題に対処します。プロバイダは、IntegratedML の共通インタフェース内でアクセスできる強力な機械学習フレームワークです。利用できるプロバイダは、以下のとおりです。

- AutoML ー インターシステムズが開発した機械学習エンジンで、InterSystems IRIS に収容されています
- H2O ー オープンソースの自動機械学習プラットフォーム
- DataRobot ー 高度なエンタープライズ自動機械学習プラットフォーム

ML 構成

ML 構成は、IntegratedML がモデルのトレーニングに使用する設定の集まりです。構成では主として、トレーニングを実行する機械学習プロバイダを指定します。プロバイダによっては、URL や API トークンなど接続に必要な情報も構成で指定します。既定の ML 構成がインストール時に直ちに有効になるため、最も単純なケースでは調整は不要です。オプションで、個々のニーズに合わせて追加の構成を作成および変更することもできます。

2.1 モデル定義の作成

モデルをトレーニングする前に、CREATE MODEL 文を使用してモデル定義を指定する必要があります。モデル定義は、IntegratedML がモデルのトレーニングに使用するテンプレートで、入力フィールド（特徴）、予測されたフィールド（ラベル）、およびこれらのフィールドのデータ型に関するメタデータが含まれます。モデル定義にはデータの構造のみが格納され、データ自体は格納されません。

2.1.1 構文 – CREATE MODEL

CREATE MODEL 文の構文は以下のとおりです。

```
CREATE MODEL model-name PREDICTING (label-column) [ WITH feature-column-clause ] FROM model-source [
USING json-object-string ]
```

または

```
CREATE MODEL model-name PREDICTING (label-column) WITH feature-column-clause [ FROM model-source ] [
USING json-object-string ]
```

または

```
CREATE MODEL model-name PREDICTING (label-column) WITH feature-column-clause FROM model-source [ USING
 json-object-string ]
```

2.1.2 例 – CREATE MODEL

以下の例では、CREATE MODEL 文におけるさまざまな節の使用法を示します。

FROM による特徴列の選択

以下のコマンドは、モデル定義 `HousePriceModel` を作成します。ラベル列（予測される列）は、`Price` です。`HouseData` テーブルの列は、FROM 節を使用することにより、暗黙的にモデル定義の特徴列として提供されます。

```
CREATE MODEL HousePriceModel PREDICTING (Price) FROM HouseData
```

重要　CREATE MODEL 文で FROM を使用しない場合、TRAIN MODEL 文で FROM が必要になります。

WITH による特徴列の選択

以下のコマンドは、上記と同じモデル定義 `HousePriceModel` を作成しますが、WITH 節を使用して特徴列とそのデータ型を明示的に指定します。

```
CREATE MODEL HousePriceModel PREDICTING (Price) WITH (TotSqft numeric, num_beds integer, num_baths
 numeric)
```

USING によるトレーニング・パラメータの選択

以下のコマンドは、オプションの USING 節を使用して、トレーニングに使うプロバイダのパラメータを指定します。USING 節の詳細は、"トレーニング・パラメータの追加 (USING 節)" を参照してください。

```
CREATE MODEL HousePriceModel PREDICTING (Price) FROM HouseData USING {"seed": 3}
```

2.1.3 モデルのためのデータの準備

モデル定義を作成する前に、以下の項目を考慮してデータセットを準備する必要があります。

- データを単一のビューまたはテーブルにまとめます。
- 特徴を評価します。
  - 列のいくつかの行に欠落値または Null 値があると、トレーニングするモデルに悪影響を及ぼす可能性があるため、該当の列を削除することが必要になります。CASE 式を使用して、列の NULL を任意の値に置き換えることも検討できます。
  - 文字数の多いデータを使用すると、モデルのトレーニングがかなり遅くなります。

2.1.4 詳細情報

INFORMATION_SCHEMA.ML_MODELS ビューで、モデル定義を確認することができます。

モデル定義に関して実行できるその他の操作の詳細は、"モデルのメンテナンス" を参照してください。

CREATE MODEL 文の詳細は、"InterSystems SQL リファレンス" を参照してください。

2.2 モデルのトレーニング

モデル定義を作成した後、TRAIN MODEL 文を使用して予測モデルをトレーニングできます。IntegratedML は、ML 構成で指定されたプロバイダを使用してこのモデルをトレーニングします。プロバイダは構造化プロセスを使用して、さまざまな機械学習モデル・タイプ (線形回帰、ランダム・フォレストなど) のパフォーマンスをデータと比較して、適切なモデルを返します。

2.2.1 構文 – TRAIN MODEL

TRAIN MODEL 文の構文は以下のとおりです。

```
TRAIN MODEL model-name [ AS preferred-model-name ] [ NOT DEFAULT ] [ FOR label-column ] [ WITH
feature-column-clause ] [ FROM model-source ] [ USING json-object-string ]
```

2.2.2 例 – TRAIN MODEL

以下の例では、TRAIN MODEL 文におけるさまざまな節の使用法を示します。

最も単純な構文

以下のコマンドは、HousePriceModel モデル定義を使用してモデルをトレーニングします。

```
TRAIN MODEL HousePriceModel
```

重要　CREATE MODEL 文で FROM を使用しなかった場合、TRAIN MODEL 文で FROM が必要になります。

FROM によるトレーニング・ データの選択

以下のコマンドは、`HousePriceModel` モデル定義および `HouseData` をトレーニング・データとして使用してモデルをトレーニングします。

```
TRAIN MODEL HousePriceModel FROM HouseData
```

AS によるトレーニング実行の命名

以下のコマンドは、`HousePriceModel` モデル定義を使用してモデルをトレーニングします。このトレーニングされたモデルは、`HousePriceModelTrained` という名前で保存されます。

```
TRAIN MODEL HousePriceModel AS HousePriceModelTrained FROM HouseData
```

WITH による特徴列のマッチング

以下のコマンドは、`HousePriceModel` モデル定義でモデルをトレーニングし、FOR 節と WITH 節を使用して、トレーニング・セットとモデル定義との間で、ラベル列と特徴列をそれぞれ明示的に合わせます。

```
TRAIN MODEL HousePriceModel FOR house_price WITH (TotSqft = house_area, num_beds = beds, num_baths
 = bathrooms) FROM OtherHouseData
```

USING によるトレーニング・パラメータの選択

以下のコマンドは、オプションの USING 節を使用して、トレーニングに使うプロバイダのパラメータを指定します。USING 節の詳細は、"トレーニング・パラメータの追加 (USING 節)" を参照してください。

```
TRAIN MODEL HousePriceModel USING {"seed": 3}
```

2.2.3 トレーニング・パラメータの追加 (USING 節)

USING 節を使用すると、プロバイダがモデルをトレーニングする方法に影響を与えるパラメータの値を指定できます。機械学習の専門家は、この特徴を使用して、ニーズに合わせてトレーニング実行を細かく調整できます。

例えば、以下のようにすることができます。

```
TRAIN MODEL my-model USING {"seed": 3}
```

USING 節を使用して、プロバイダ固有のトレーニング・パラメータを渡すことができます。この節には、パラメータ名とパラメータ値のキーと値のペアを記述した JSON 文字列を使用できます。これらの値のペアでは大文字と小文字が区別されます。

CREATE MODEL 文と TRAIN MODEL 文だけでなく ML 構成でも USING 節を渡すことができます。以下のように解決されます。

- TRAIN MODEL コマンドの USING 節で指定したパラメータは、CREATE MODEL コマンドまたは既定の ML 構成で指定した同じパラメータの値を上書きします。
- CREATE MODEL コマンドの USING 節で指定したパラメータは、TRAIN MODEL コマンドに暗黙的に使用され、既定の ML 構成で指定した同じパラメータの値を上書きします。
- CREATE MODEL コマンドまたは TRAIN MODEL コマンドで USING を指定しない場合、モデルでは既定の ML 構成で指定された USING 節が使用されます。

すべてのパラメータ名は文字列として渡す必要があり、値はパラメータに固有の型で渡す必要があります。リストは、コンマで区切った文字列で入力する必要があります。

各プロバイダで使用できるパラメータの詳細は、以下を参照してください。

- AutoML
- H2O
- DataRobot

2.2.4 詳細情報

トレーニングされたモデルとトレーニング実行の結果は、それぞれ INFORMATION_SCHEMA.ML_TRAINED_MODELS ビューおよび INFORMATION_SCHEMA.ML_TRAINING_RUNS ビューで確認できます。トレーニングされたモデルは、トレーニングに使用されたモデル定義に関連付けられています。

トレーニングされたモデルに関して実行できるその他の操作の詳細は、"モデルのメンテナンス" を参照してください。

TRAIN MODEL 文の詳細は、"InterSystems SQL リファレンス" を参照してください。

2.3 モデルの検証

トレーニング中、プロバイダはトレーニングされたモデルを出力するプロセス全体を通して検証を実行します。IntegratedML が提供する VALIDATE MODEL 文を使用して、モデルに対して独自の検証を実行できます。VALIDATE MODEL は、提供されたテスト・セットに基づいて、回帰モデルと分類モデルの両方の簡易なメトリックを返します。

2.3.1 構文 – VALIDATE MODEL

VALIDATE MODEL 文の構文は以下のとおりです。

```
VALIDATE MODEL trained-model-name [ AS validation-run-name ] [ USE preferred-trained-model-name ] [
WITH feature-column-clause ] FROM testing-data-set
```

2.3.2 例 – VALIDATE MODEL

以下の例では、VALIDATE MODEL 文におけるさまざまな節の使用法を示します。

最も単純な構文

以下のコマンドは、HouseTesting をテスト・データ・セットとして使用して、トレーニングされた HousePriceModel を検証します。

```
VALIDATE MODEL HousePriceModel From HouseTesting
```

AS による検証実行の命名

以下のコマンドは、HouseTesting をテスト・データ・セットとして使用して、トレーニングされた HousePriceModel を検証し、検証実行を HousePriceValidation として保存します。

```
VALIDATE MODEL HousePriceModel AS HousePriceValidation From HouseTesting
```

WITH による特徴列のマッチング

以下のコマンドは、トレーニングされた HousePriceModel を検証し、WITH 節を使用してテスト・データ・セット HouseTesting の特徴列を明示的にマッチングします。

```
VALIDATE MODEL HousePriceModel WITH (TotSqft = area, num_beds = beds, num_baths = baths) From
HouseTesting
```

2.3.3 詳細情報

検証実行とその結果については、それぞれ INFORMATION_SCHEMA.ML_VALIDATION_RUNS ビューと INFORMATION_SCHEMA.ML_VALIDATION_METRICS ビューで確認できます。

VALIDATE MODEL 文および検証メトリックの詳細は、"InterSystems SQL リファレンス" を参照してください。

2.4 予測

トレーニングされたモデルにはそれぞれ、専門の関数 PREDICT があり、この関数がプロバイダを呼び出して、該当する行セットの各行の結果を予測します。分類モデルにはさらに PROBABILITY 関数があり、この関数がプロバイダを呼び出して、指定された値がモデルに対して正しい結果である確率を返します。

これらはスカラ関数であり、SQL クエリの任意の場所で、他のフィールドや関数と組み合わせて使用できます。

2.4.1 PREDICT

PREDICT 関数を使用し、指定されたモデル (したがって、プロバイダ) を該当する行セットの各行に適用して、ラベル列の予測値 (回帰モデルの場合) または最も可能性の高い値 (分類モデルの場合) を返すことができます。各行には入力列 (特徴列) があり、モデルはこの入力列から出力 (ラベル) を返します。行セットは、必要な特徴列とラベル列を含む任意の行のセットとすることができます。

構文

PREDICT 関数の構文は以下のとおりです。

```
PREDICT(model-name [ USE trained-model-name ] [ WITH feature-column-clause ] )
```

例

以下の文では、モデル HousePriceModel の専門の関数 PREDICT をさまざまな形式で使用しています。

```
SELECT *, PREDICT(HousePriceModel) FROM NewHouseData
```

```
SELECT * FROM NewHouseData WHERE PREDICT(HousePriceModel) > 500000
```

WITH 節を使用して、すべてのデータセットに基づくのではなく、トレーニング・セットの列に対応する指定の値に基づいて予測できます。各引数は、CREATE MODEL 文で指定した順序どおりに記述する必要があります。欠落している引数は空のコンマで表現できます。例えば、以下の文では 2 セットの列データに基づいて予測が実行されます。

```
SELECT PREDICT(HousePriceModel WITH ({4200, 5, 4},{3800, , 3}))
```

また、WITH を使用して、モデルの作成元のデータセットとは異なるデータセットの列とモデルの列とのマッピングを指定することもできます。以下の文は、モデルから別のデータセットの列に特徴列をマッピングします。

```
SELECT PREDICT(HousePriceModel WITH (TotSqft = house_area, num_beds = beds, num_baths = bathrooms) FROM
 OtherHouseData
```

詳細情報

PREDICT 関数の詳細は、"InterSystems SQL リファレンス" を参照してください。

2.4.2 PROBABILITY

分類モデルの場合、PROBABILITY 関数を使用して、指定したラベル値が予測されたラベル値である確率を行ごとに返すことができます。これにより、その値の予測の相対的な強度を評価できます。

構文

PROBABILITY 関数の構文は以下のとおりです。

```
PROBABILITY(model-name [ USE trained-model-name ] FOR label-value [ WITH feature-column-clause ] )
```

例

以下の文では、モデル `Iris_mdoel` 専用の関数 PROBABILITY をさまざまな形式で使用しています。

```
SELECT *, PROBABILITY(Iris_Model FOR 'iris-setosa') FROM Iris_Flower_Set
```

```
SELECT * FROM Iris_Flower_Set WHERE PROBABILITY(Iris_Model FOR 'iris-setosa') < 0.3
```

以下の文では、モデル `EmailFilter` の専門の関数 PROBABILITY を使用しています。これは、ブーリアン値 0 または 1 を唯一の出力とする二項分類モデルであるため、暗黙的な FOR 値 1 を使用して、FOR 節を省略できます。

```
SELECT * EmailData WHERE PROBABILITY(EmailFilter) > 0.7
```

詳細情報

PROBABILITY 関数の詳細は、"InterSystems SQL リファレンス" を参照してください。

2.5 ウォークスルー

このウォークスルーでは、IntegratedML がアプリケーション通じて実世界のシナリオに提供するシンプルかつ強力な構文を示します。ユーザは少数の SQL クエリを使用し、自身のデータを使って検証済みの予測モデルを開発します。

医療システムの管理者は、患者の再入院率が増加していることを憂慮しています。患者システムを評価する際、医者が全体的にもっと慎重になるべきかもしれませんが、何を探せばよいかを示す基準が定義されていません。新しい分析ソリューションに多大な投資を行う前に、管理者からデータ・アナリストに対し、再入院した患者のプロファイルで傾向を見つけるモデルを迅速に開発する課題が課せられます。データは InterSystems IRIS® データベース・プラットフォームに格納されているため、アナリストは IntegratedML を使用すれば、手動によるフォーマット設定やプラットフォーム外部からのデータの移動を必要とする他のソリューションよりも、ずっと速く開発できることを知っています。

データの準備

IntegratedML を使用する前に、アナリストはデータを準備し、データがクリーンですぐにトレーニングに使用できることを確認します。使いやすいように、必要なデータを複数のテーブルから単一のビューに入れます。この例では、ビューに `Hospital.PatientDataView` という名前が付けられています。

構成のカスタマイズ

アナリストは、既定の構成で IntegratedML を使用することを選択します。モデルのトレーニングにさまざまなプロバイダを使用できることは分かっていますが、スピードと使いやすさから、追加の構文が不要な既定の構成を使用することにしました。

モデルの作成

データが揃い、単一のビューにまとめられたので、アナリストは自動機械学習機能でトレーニングするモデル定義を作成します。`PatientReadmission` というこの定義は、`IsReadmitted` を予測するラベル列として指定します。

```
CREATE MODEL PatientReadmission PREDICTING (IsReadmitted) FROM Hospital.PatientDataView
```

モデルのトレーニング

次に、アナリストはモデルをトレーニングします。

```
TRAIN MODEL PatientReadmission
```

トレーニングのためにカスタマイズされたパラメータを指定する必要はありません。

モデルの検証

アナリストは、準備したテスト・データセット（`Hospital.PatientDataViewTesting`）を使用してモデルを検証し、パフォーマンスに関するメトリックを取得して、それらのメトリックを確認します。

```
VALIDATE MODEL PatientReadmission FROM Hospital.PatientDataViewTesting
SELECT * FROM INFORMATION_SCHEMA.ML_VALIDATION_METRICS
```

モデルを使用した予測

モデルをトレーニングして検証したので、次にアナリストはモデルを適用して、同じスキーマを持つさまざまなデータセットに基づいて予測を行います。この 1 週間に入院した患者の情報を含むデータセット `Hospital.NewPatientDataView` にモデルを適用し、再入院の可能性のある患者がいるかどうかを確認します。

```
SELECT ID FROM Hospital.NewPatientDataView WHERE PREDICT(PatientReadmission) = 1
```

まとめ

まとめると、アナリストは以下の SQL クエリを入力して、未加工データを有効な予測モデルに変換しました。

```
CREATE MODEL PatientReadmission PREDICTING (IsReadmitted) FROM Hospital.PatientDataView
TRAIN MODEL PatientReadmission
VALIDATE MODEL PatientReadmission FROM Hospital.PatientDataViewTesting
SELECT * FROM INFORMATION_SCHEMA.ML_VALIDATION_METRICS
SELECT ID FROM Hospital.NewPatientDataView WHERE PREDICT(PatientReadmission) = 1
```

3
プロバイダ

プロバイダは、IntegratedML の共通インタフェース内でアクセスできる強力な機械学習フレームワークです。トレーニングに使用するプロバイダを選択するには、目的のプロバイダを指定する ML 構成を選択します。

USING 節を使用して、これらのプロバイダに固有の追加のパラメータを渡すことができます。詳細は、"トレーニング・パラメータの追加 (USING 節)" を参照してください。

3.1 AutoML

AutoML はインターシステムズが開発した自動機械学習システムで、InterSystems IRIS® に収容されています。IntegratedML、AutoML は、モデルを迅速にトレーニングして正確な結果を生成します。また、AutoML は基本的な自然言語処理 (NLP) を備えているため、プロバイダは構造化されていないテキストを含む特徴列を機械学習モデルにすばやく組み込むことができます。

%AutoML は IntegratedML のシステムの既定の ML 構成で、AutoML をプロバイダとして指しています。

3.1.1 トレーニング・パラメータ – AutoML

USING 節を使用して、トレーニング・パラメータを渡すことができます。例えば、以下のようにすることができます。

```
TRAIN MODEL my-model USING {"seed": 3}
```

AutoML では、以下のパラメータをトレーニング・クエリに渡すことができます。

トレーニング・パラメータ	説明
seed	乱数ジェネレータを初期化するためのシード。複数のトレーニング実行での再現性を実現するために、シードとして任意の整数を手動で設定できます。既定では、seed は“None”に設定されます。
verbosity	トレーニング実行出力の詳細度を指定します。この出力は ML_TRAINING_RUNS ビューで確認できます。verbosity には以下のいずれかのオプションを指定できます。 · 0 – 最小または出力なし。 · 1 – 中程度の出力。 · 2 – すべての出力。verbosity の既定の設定です。
TrainMode	分類モデルでのモデル選択のメトリックを指定します。TrainMode には以下のいずれかのオプションを指定できます。 · “TIME” – モデルの選択で、より短いトレーニング時間を優先します。 · “BALANCE” – モデルの選択で、モデルのスコアとトレーニング時間との比率が等しいかどうかによって各モデルを比較します。 · “SCORE” – モデルの選択で、トレーニングの実行時間を考慮しません。TrainMode の既定の設定です。 これらの各種モードの詳細は、″AutoML リファレンス″ を参照してください。
MaxTime	トレーニングの実行を開始するために割り当てる時間値 (分)。これによって必ずしもトレーニング時間が制限されるわけではありません。例えば、MaxTime を 3,000 分に設定していて、あるモデルのトレーニング後に 2 分残っていれば別のモデルをトレーニングできます。既定では、MaxTime は 14400 分に設定されています。 注釈 このパラメータは、TrainMode を“TIME”に設定している場合にのみ適用できます。
MinimumDesiredScore	選択したトレーニング・モードに関係なく、分類モデルの選択で許可する最小スコア。0 ～ 1 の範囲で任意の値を設定できます。既定では、MinimumDesiredScore は 0 に設定されています。 注釈 このパラメータは、TrainMode を“TIME”に設定している場合にのみ適用できます。 トレーニングしたロジスティック回帰モデルまたはランダム・フォレスト分類子モデルのスコアが MinimumDesiredScore を超える場合、AutoML ではニューラル・ネットワーク・モデルがトレーニングされません。分類モデルに使用する各種モデルの詳細は、″AutoML リファレンス″ を参照してください。

3.1.2 特徴量エンジニアリング

AutoML では、特徴量エンジニアリングを使用して既存の特徴を変更し、新しい特徴を作成し、不要な特徴を削除します。これらのステップによって、トレーニングの速度とパフォーマンスが向上します。

- 列タイプの分類により、モデルで特徴を正しく使用できます
- 特徴量削減により、冗長性が排除され、正確性が向上します
- カテゴリカル特徴の One-hot エンコーディング
- 不完全なデータセットの欠落値/Null 値の入力
- 必要に応じて、時間/日/月/年に関連する新しい列を作成し、時間に関連するデータから洞察を生み出します。

3.1.3 モデルの選択

回帰モデルが適切であると判断された場合、AutoML は回帰モデルを開発するための単一プロセスを使用します。

分類モデルの場合、AutoML は以下の選択プロセスを使用して、最も正確なモデルを決定します。

1. データセットが大きすぎる場合、AutoML はデータをダウン・サンプル (圧縮) して、モデル選択プロセスを高速化します。モデルの選択後は、引き続き完全なデータセットがトレーニングに使用されます。
2. 適切なスコアリング・メトリックを使用するために、AutoML はデータセットに二項分類の問題があるかどうか、または複数のクラスがあるかどうかを確認します。
3. AutoML はモンテカルロ交差検証を使用して、データセット全体のトレーニングのための最適なスコアリング・メトリックを持つモデルを選択します。

注釈　　このモデル選択プロセスの詳細は、"AutoML リファレンス" を参照してください。

3.1.4 既知の問題

AutoML に必要なライブラリがないため、RHEL7 インストールでは AutoML プロバイダを実行できないことがあります。TRAIN MODEL 文の実行中に発生する一般的なエラーには、`xgboost.core.XGBoostError` などのテキストが含まれることがあります。これらのエラー・メッセージに表示されている必須パッケージをインストールすることで、この問題を修正できる可能性があります。

AutoML は Python を使用して実装されていますが、AutoML Python パッケージと Embedded Python パッケージとの分離が不適切になることがあります。その結果、AutoML が正しく動作するために必要なパッケージが見つからないことがあります。この問題を回避するには、InterSystems IRIS インスタンスで Python の sys.path に <path to instance>/lib/automl を追加します。そのためには、%SYS.Python.Shell() で Python シェルを開き、以下のコマンドを入力します。

```
import sys
sys.path.append("<path to instance>\\lib\\automl")
```

3.1.5 詳細情報

AutoML の仕組みの詳細は、"AutoML リファレンス" を参照してください。

3.2 H2O

%H2O を ML 構成として設定することで、H2O をプロバイダとして指定できます。

PROVIDER が H2O を指す新しい ML 構成を作成することもできます。

3.2.1 トレーニング・パラメータ – H2O

USING 節を使用して、トレーニング・パラメータを渡すことができます。例えば、以下のようにすることができます。

```
TRAIN MODEL my-model USING {"seed": 3}
```

予想される入力およびこれらのパラメータの処理方法については、H2O のドキュメントを参照してください。不明なパラメータを指定すると、トレーニング中にエラーが発生します。

H2O プロバイダを使用してモデルをトレーニングする場合、`max_models` パラメータは既定で 5 に設定されます。

3.2.2 モデルの選択

String 型、Integer 型、または Binary 型として分類されるラベル列は、分類モデルになります。他の型はすべて、回帰モデルになります。Integer 型の列を H2O によって回帰モデルとしてトレーニングさせる場合は、キーと値のペア `"model_type":"regression"` を USING 節に追加する必要があります。例えば以下のようにします。

```
TRAIN MODEL h2o-model USING {"model_type": "regression"}
```

3.2.3 トレーニング・ログ出力

H2O を使用してモデルをトレーニングした後、INFORMATION_SCHEMA.ML_TRAINING_RUNS ビューの LOG 列をクエリできます。

3.2.4 既知の問題

- H2O プロバイダを使用してトレーニングする際、以下のエラー・メッセージが表示されることがあります。

  ```
  LogMessage: %ML Provider '%ML.H2O.Provider' is not available on this instance
    > ERROR #5002: ObjectScript error: <READ>%GetResponse+4^%Net.Remote.Object.1
  ```

  その場合、以下を実行してこの問題を解決することができます。

  1. 管理ポータルにログインします。
  2. **[システム管理]→[構成]→[接続性]→[External Language Servers]** に移動します。
  3. **%IntegratedML Server** という名前のサーバを選択します。
  4. **[JVM 引数]** フィールドに以下を追加します。

     ```
     -Djava.net.preferIPv6Addresses=true -Djava.net.preferIPv4Addresses=false
     ```

- USING 節を使用して H2O プロバイダにシード・パラメータを設定しても、再現可能なトレーニング実行は保証されません。これは、H2O の既定のトレーニング設定には、5 に設定された `max_models` パラメータが含まれ、早期停止モードがトリガされるためです。H2O のドキュメントにも記載されているように、H2O の勾配ブースティング・モデル・アルゴリズムの再現性は複雑なトピックです。

3.2.5 詳細情報

H2O の詳細は、H2O のドキュメントを参照してください。

3.3 DataRobot

重要　　DataRobot の AutoML 機能を使用するには、DataRobot とビジネス関係を結んでいる必要があります。

DataRobot のクライアントは IntegratedML を使用し、InterSystems IRIS® に保存されているデータを利用してモデルをトレーニングできます。

DataRobot 構成を既定の ML 構成として選択することで、DataRobot をプロバイダとして指定できます。

```
SET ML CONFIGURATION datarobot_configuration
```

`datarobot_configuration` は ML 構成の名前で、`PROVIDER` は `DataRobot` を指しています。

3.3.1 トレーニング・パラメータ – DataRobot

USING 節を使用して、トレーニング・パラメータを渡すことができます。例えば以下のようにします。

```
TRAIN MODEL my-model USING {"seed": 3}
```

IntegratedML は DataRobot API を使用して HTTP 要求を作成し、モデリングを開始します。予想される入力およびこれらのパラメータの処理方法については、DataRobot のドキュメントを参照してください。不明なパラメータを指定すると、トレーニング中にエラーが発生します。

DataRobot プロバイダを使用してモデルをトレーニングする場合、`quickrun` パラメータは既定で `true` に設定されます。

3.4 PMML

IntegratedML は PMML コンシューマとしての PMML をサポートしており、SQL を使用して PMML モデルを容易にインポートおよび実行できます。

3.4.1 IntegratedML での PMML モデルの動作の仕組み

他のプロバイダと同様に、CREATE MODEL 文を使用して、特徴やラベルを含むモデル定義を指定します。このモデル定義には、PMML モデルに含まれるものと同じ特徴とラベルが含まれている必要があります。

TRAIN MODEL 文の動作は異なります。TRAIN MODEL 文はデータを“トレーニング”する代わりに、PMML モデルをインポートします。PMML モデルによって、特徴やラベルの情報を含め、トレーニングされたモデルのプロパティが提示されるため、トレーニングは不要です。このモデルは、USING 節によって識別されます。

重要　　モデル定義で指定した特徴列およびラベル列は、PMML モデルの特徴列およびラベル列と一致する必要があります。

CREATE MODEL 文と TRAIN MODEL 文のどちらにも FROM 節が必要ですが、指定されたデータは一切使用されません。

“トレーニングされた”PMML モデルを使用した予測は、IntegratedML のその他のトレーニングされたモデルと同様に機能します。PREDICT 関数は、PMML 定義と一致する特徴列を含むどのデータでも使用することができます。

3.4.2 PMML モデルのインポート方法

PMML モデルを使用するには、事前に %PMML を ML 構成として設定するか、PROVIDER が PMML を指す別の ML 構成を選択する必要があります。

PMML モデルは、USING 節を使用して指定できます。以下のいずれかのパラメータを選択できます。

クラス名による

"class_name" パラメータを使用して、PMML モデルのクラス名を指定できます。例を以下に示します。

```
USING {"class_name" : "IntegratedML.pmml.PMMLModel"}
```

ディレクトリ・パスによる

"file_name" パラメータを使用して、PMML モデルへのディレクトリ・パスを指定できます。例を以下に示します。

```
USING {"file_name" : "C:\temp\mydir\pmml_model.xml"}
```

3.4.3 例

以下の例では、USING 節を渡して PMML モデルを指定する複数のメソッドを示しています。

ML 構成での PMML モデルの指定

以下の一連の文では、ファイル名によって住宅価格の PMML モデルを指定する PMML 構成を作成し、その後 TRAIN MODEL 文でモデルをインポートします。

```
CREATE ML CONFIGURATION pmml_configuration PROVIDER PMML USING {"file_name" :
"C:\PMML\pmml_house_model.xml"}
SET ML CONFIGURATION pmml_configuration
CREATE MODEL HousePriceModel PREDICTING (Price) WITH (TotSqft numeric, num_beds integer, num_baths
numeric)
TRAIN MODEL HousePriceModel FROM HouseData
SELECT * FROM NewHouseData WHERE PREDICT(HousePriceModel) > 500000
```

TRAIN MODEL 文での PMML モデルの指定

以下の一連の文では、指定された %PMML 構成を使用し、TRAIN MODEL 文でクラス名によって PMML モデルを指定します。

```
SET ML CONFIGURATION %PMML
CREATE MODEL HousePriceModel PREDICTING (Price) WITH (TotSqft numeric, num_beds integer, num_baths
numeric)
TRAIN MODEL HousePriceModel FROM HouseData USING {"class_name" : "IntegratedML.pmml.PMMLHouseModel"}
SELECT * FROM NewHouseData WHERE PREDICT(HousePriceModel) > 500000
```

3.4.4 その他のパラメータ

PMML ファイルに複数のモデルが含まれる場合、IntegratedML では、既定でファイル内の最初のモデルが使用されます。ファイル内の別のモデルを指す場合は、USING 節で model_name パラメータを使用します。

```
TRAIN MODEL my_pmml_model FROM data USING {"class_name" : my_pmml_file, "model_name" : "model_2_name"}
```

4

ML 構成

ML 構成は、IntegratedML がモデルのトレーニングに使用する設定の集まりです。構成では主として、トレーニングを実行する機械学習プロバイダを指定します。プロバイダによっては、URL や API トークンなど接続に必要な情報も構成で指定します。

インストール時に %AutoML がシステムの既定の ML 構成として設定されるため、ML 構成に調整を加えずに IntegratedML を使用できます。

4.1 ML 構成の作成

インストール時のシステムの既定の ML 構成を使用できますが、モデルのトレーニングのために新しい ML 構成を作成することもできます。

4.1.1 システム管理ポータルを使用した ML 構成の作成

ML 構成を作成するには、次の手順を実行します。

1. 管理ポータルにログインします。
2. **[システム管理]→[構成]→[Machine Learning Configurations]** に移動します。
3. **[新規構成の作成]** を選択して、フィールドに以下の値を入力します。
   - **[名前]** ― ML 構成の名前。
   - **[プロバイダ]** ― ML 構成を接続する機械学習プロバイダ。

     [DataRobot] を選択した場合、以下の追加フィールドに値を入力する必要があります。

     - [URL] ― DataRobot エンドポイントの URL。
     - [API Token] ― DataRobot アカウントの API トークン。
   - **[説明]** ― オプション。ML 構成の説明テキスト。
   - [Using Clause] ― オプション。ML 構成の既定の USING 節。詳細は、"トレーニング・パラメータの追加 (USING 節)" を参照してください。
   - **[所有者]** ― この ML 構成の所有者。
4. **[保存]** を選択して、この新しい ML 構成を保存します。

この新しい ML 構成をシステムの既定として設定するには、"システムの既定の ML 構成の設定" を参照してください。

4.1.2 SQL を使用した ML 構成の作成

CREATE ML CONFIGURATION コマンドを使用して新しい構成を作成できます。

構文

CREATE ML CONFIGURATION 文の構文は以下のとおりです。

```
CREATE ML CONFIGURATION ml-configuration-name PROVIDER provider-name [ %DESCRIPTION description-string
 ] [ USING json-object-string ] provider-connection-settings
```

例

以下の例では、CREATE ML CONFIGURATION 文におけるさまざまな節の使用法を示します。

最も単純な構文

以下のコマンドは、H2O プロバイダを使用する ML 構成 `H2OConfig` を作成します。H2O に接続する際、プロバイダ接続設定は不要です。

```
CREATE ML CONFIGURATION H2OConfig PROVIDER H2O
```

USING によるトレーニング・パラメータの選択

以下のコマンドは、H2O プロバイダを使用し、既定の USING 節を指定する ML 構成 `H2OConfig` を作成します。

```
CREATE ML CONFIGURATION H2OConfig PROVIDER H2O USING {"nfolds": 4}
```

詳細情報

この新しい ML 構成をシステムの既定として設定するには、"システムの既定の ML 構成の設定" を参照してください。

CREATE ML CONFIGURATION コマンドの詳細は、"InterSystems SQL リファレンス" を参照してください。

4.2 ML 構成の設定

IntegratedML では、すぐに使用できる構成として、以下が用意されています。

- `%AutoML`
- `%H2O`
- `%PMML`

インストール時には、`%AutoML` がシステムの既定の ML 構成として設定されます。構成に調整を加えずに IntegratedML を使用できます。TRAIN MODEL 文に使用する別の ML 構成を指定したい場合は、以下のいずれかの方法で指定できます。

- SQL ー 指定されたプロセスの ML 構成を設定できます
- システム管理ポータル ー システムの既定の ML 構成を調整できます

INFORMATION_SCHEMA.ML_TRAINING_RUNS ビューにクエリを実行して、トレーニング実行にどの ML 構成が使用されたかを確認できます。

4.2.1 SQL を使用した、指定されたプロセスの ML 構成の設定

SET ML CONFIGURATION 文を使用して、指定プロセスの ML 構成を指定できます。

構文

SET ML CONFIGURATION 文の構文は以下のとおりです。

```
SET ML CONFIGURATION ml-configuration-name
```

詳細情報

SET ML CONFIGURATION 文の詳細は、"InterSystems SQL リファレンス" を参照してください。

4.2.2 システム管理ポータルを使用したシステムの既定の ML 構成の設定

システム管理ポータルの [Machine Learning Configurations] ページで、システムの既定の ML 構成を設定できます。

システムの既定の ML 構成を設定するには、次の手順を実行します。

1. 管理ポータルにログインします。
2. **[システム管理]→[構成]→[Machine Learning Configurations]** に移動します。
3. [System Default ML Configuration] の横で、任意の ML 構成を選択します。

注釈　この方法でシステムの既定の ML 構成を設定しても、新しいプロセスを開始するまで有効にはなりません。

4.3 ML 構成の管理

ML 構成を管理するために、以下の操作を実行できます。

- ML 構成の変更
- ML 構成の削除

INFORMATION_SCHEMA.ML_TRAINING_RUNS ビューにクエリを実行して、トレーニング実行にどの ML 構成が使用されたかを確認できます。

4.3.1 ML 構成の変更

既存の ML 構成のプロパティを変更できます。

4.3.1.1 システム管理ポータルを使用した ML 構成の変更

ML 構成を変更するには、次の手順を実行します。

1. 管理ポータルにログインします。
2. **[システム管理]→[構成]→[Machine Learning Configurations]** に移動します。
3. リストされた ML 構成の名前を選択して、任意の値を調整します。
4. **[保存]** を選択して、この変更した ML 構成を保存します。

4.3.1.2 SQL を使用した ML 構成の変更

ALTER ML CONFIGURATION 文を使用して構成を変更できます。

構文

ALTER ML CONFIGURATION 文の構文は以下のとおりです。

```
ALTER ML CONFIGURATION ml-configuration-name alter-options
```

alter-options は以下の 1 つ以上になります。

- `PROVIDER provider-name`
- `%DESCRIPTION description-string`
- `USING json-object-string`
- *provider-connection-settings*

詳細情報

ALTER ML CONFIGURATION コマンドの詳細は、"InterSystems SQL リファレンス" を参照してください。

4.3.2 ML 構成の削除

ML 構成を削除できます。

4.3.2.1 システム管理ポータルを使用した ML 構成の削除

ML 構成を削除するには、次の手順を実行します。

1. 管理ポータルにログインします。
2. **[システム管理]→[構成]→[Machine Learning Configurations]** に移動します。
3. 削除する ML 構成の行を見つけて、**[削除]** を選択します。

4.3.2.2 SQL を使用した ML 構成の削除

DROP ML CONFIGURATION 文を使用して構成を削除できます。

構文

DROP ML CONFIGURATION 文の構文は以下のとおりです。

```
DROP ML CONFIGURATION ml-configuration-name
```

詳細情報

DROP ML CONFIGURATION コマンドの詳細は、"InterSystems SQL リファレンス" を参照してください。

5
モデルのメンテナンス

モデルのメンテナンスは、モデルの表示、変更、および削除で構成されます。

5.1 モデルの表示

IntegratedML がトレーニングまたは検証を実行する際、このプロセスは“トレーニング実行”または“検証実行”と呼ばれます。

IntegratedML では、モデル、トレーニングされたモデル、トレーニング実行、および検証実行に関する情報を問い合わせるために、INFORMATION_SCHEMA クラス内に以下のビューが用意されています。

- ML_MODELS
- ML_TRAINED_MODELS
- ML_TRAINING_RUNS
- ML_VALIDATION_RUNS
- ML_VALIDATION_METRICS

5.1.1 ML_MODELS

このビューは、モデル定義ごとに 1 つの行を返します。

INFORMATION_SCHEMA.ML_MODELS は、以下の列で構成されます。

列名	説明
CREATE_TIME_STAMP	モデル定義が作成された時刻 (UTC)
DEFAULT_SETTINGS	モデル定義のプロバイダが使用する既定の設定
DEFAULT_TRAINED_MODEL_NAME	トレーニング済みの場合、既定のトレーニング済みモデルの名前
DEFAULT_TRAINING_QUERY	CREATE MODEL 文の FROM 節 (使用された場合)
DESCRIPTION	モデル定義の説明
MODEL_NAME	モデル定義の名前
PREDICTING_COLUMN_NAME	ラベル列の名前
PREDICTING_COLUMN_TYPE	ラベル列の型
WITH_COLUMNS	特徴列の名前

詳細情報

モデル定義の詳細は、"モデル定義の作成" を参照してください。

5.1.2 ML_TRAINED_MODELS

このビューは、トレーニングされたモデルごとに 1 つの行を返します。

INFORMATION_SCHEMA.ML_TRAINED_MODELS は、以下の列で構成されます。

列名	説明
MODEL_INFO	モデル情報
MODEL_NAME	モデル定義の名前
MODEL_TYPE	モデルのタイプ (分類または回帰)
PROVIDER	トレーニングに使用されたプロバイダ
TRAINED_MODEL_NAME	トレーニングされたモデルの名前
TRAINED_TIMESTAMP	トレーニングされたモデルが作成された時刻 (UTC)

詳細情報

トレーニングされたモデルの詳細は、"モデルのトレーニング" を参照してください。

プロバイダの詳細は、"プロバイダ" を参照してください。

5.1.3 ML_TRAINING_RUNS

このビューは、トレーニング実行ごとに 1 つの行を返します。

INFORMATION_SCHEMA.ML_TRAINING_RUNS は、以下の列で構成されます。

列名	説明
COMPLETED_TIMESTAMP	トレーニング実行が完了した時刻 (UTC)
LOG	プロバイダからのトレーニング・ログ出力
ML_CONFIGURATION_NAME	トレーニングに使用された ML 構成の名前

列名	説明
MODEL_NAME	モデル定義の名前
PROVIDER	トレーニングに使用されたプロバイダの名前
RUN_STATUS	トレーニング実行のステータス
SETTINGS	トレーニング実行のために USING 節によって渡された設定
START_TIMESTAMP	トレーニング実行が開始された時刻 (UTC)
STATUS_CODE	トレーニング・エラー (発生した場合)
TRAINING_DURATION	トレーニングの期間 (秒単位)
TRAINING_RUN_NAME	トレーニング実行の名前
TRAINING_RUN_QUERY	トレーニング用に特徴列とラベル列からデータを取り出すために使用されるクエリ

詳細情報

トレーニング実行の詳細は、"モデルのトレーニング" を参照してください。

5.1.4 ML_VALIDATION_RUNS

このビューは、検証実行ごとに 1 つの行を返します。

INFORMATION_SCHEMA.ML_VALIDATION_RUNS は、以下の列で構成されます。

列名	説明
COMPLETED_TIMESTAMP	検証実行が完了した時刻 (UTC)
LOG	検証ログ出力
MODEL_NAME	モデル定義の名前
RUN_STATUS	検証ステータス
SETTINGS	検証実行の設定
START_TIMESTAMP	検証実行が開始された時刻 (UTC)
STATUS_CODE	検証エラー (発生した場合)
TRAINED_MODEL_NAME	検証対象のトレーニングされたモデルの名前
VALIDATION_DURATION	検証期間 (秒単位)
VALIDATION_RUN_NAME	検証実行の名前
VALIDATION_RUN_QUERY	FROM によって指定されたデータセットの完全なクエリ

詳細情報

検証実行の詳細は、"モデルの検証" を参照してください。

5.1.5 ML_VALIDATION_METRICS

このビューは、検証実行の検証メトリックごとに 1 つの行を返します。

INFORMATION_SCHEMA.ML_VALIDATION_METRICS は、以下の列で構成されます。

列名	説明
METRIC_NAME	検証メトリック名
METRIC_VALUE	検証メトリック値
MODEL_NAME	モデル名
TARGET_VALUE	検証メトリックのターゲット値
TRAINED_MODEL_NAME	この実行のトレーニングされたモデルの名前
VALIDATION_RUN_NAME	検証実行の名前

詳細情報

METRIC_NAME と METRIC_VALUE を生成する検証メトリックの詳細は、"InterSystems SQL リファレンス" を参照してください。

5.2 モデルの変更

ALTER MODEL 文を使用してモデルを変更できます。

構文

ALTER MODEL 文の構文は以下のとおりです。

```
ALTER MODEL model-name alter-action
```

alter-action は、以下のいずれかにできます。

- `PURGE ALL`
- `PURGE integer DAYS`
- `DEFAULT preferred-model-name`

例

この例では、PURGE 節を使用して、モデル `WillLoanDefault` に関連するすべてのトレーニング実行データと検証実行データを削除します。

```
ALTER MODEL WillLoanDefault PURGE ALL
```

この例では、PURGE 節を使用して、モデル `WillLoanDefault` に関連する、7 日以上経過したすべてのトレーニング実行データと検証実行データを削除します。

```
ALTER MODEL WillLoanDefault PURGE 7 DAYS
```

詳細情報

"モデルの表示" にリストされたビューをクエリして、alter 文が成功したことを確認できます。

ALTER MODEL コマンドの詳細は、リファレンスを参照してください。

5.3 モデルの削除

DROP MODEL 文を使用してモデルを削除できます。

構文

DROP MODEL 文の構文は以下のとおりです。

```
DROP MODEL model-name
```

DROP MODEL は、関連するモデルのすべてのトレーニング実行と検証実行を削除します。

詳細情報

INFORMATION_SCHEMA.ML_MODELS ビューをクエリして、モデルが削除されたことを確認できます。

DROP MODEL コマンドの詳細は、"InterSystems SQL リファレンス" を参照してください。