AutoML リファレンス

Version 2023.1
2024-01-02

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

AutoML リファレンス
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

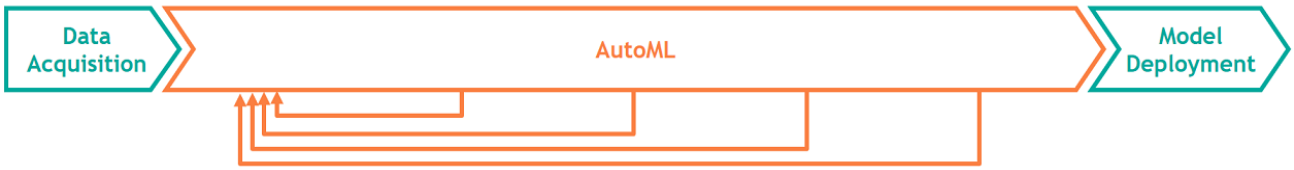
AutoML の概要

注釈　このドキュメントでは AutoML の詳細を説明します。IntegratedML でプロバイダとして AutoML を使用する方法は、IntegratedML ユーザ・ガイド を参照してください。

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® データ・プラットフォームに収容されています。機械学習プロセスのいくつかの主要な構成要素を自動化し、ユーザのデータを使用して正確な予測モデルを迅速に作成できるように設計されています。

図 1-1: 機械学習プロセス

図 1-2: 機械学習プロセスの自動化



AutoML でモデルをトレーニングした後は、IntegratedML に用意されている SQL 構文を使用して、モデルを簡単に導入できます。

2

AutoML の主な機能

ここでは、予測モデルをすばやく生成するため、AutoML で使用されるいくつかの機械学習機能について説明します。

2.1 自然言語処理

AutoML は自然言語処理 (NLP) を利用してテキスト特徴量を数値特徴量に変換し、予測モデルを生成します。また、TIFDF (Term frequency-inverse document frequency : 単語の出現頻度と逆文書頻度) を使用して、テキスト列とリスト列のキーワードを評価します。

2.2 Multi-Hot エンコーディング

ほとんどのデータはスパースですが、機械学習アルゴリズムが理解できるのはデンス・データのみです。大半のデータ・モデリング・ワークフローでは、スパース・データをデンス・データに変換する困難で面倒な作業を手動で行わなければならず、データ・サイエンティストの負担になっています。

このような手動ステップを必要とする多くのワークフローとは異なり、AutoML はこの変換をシームレスに実行します。リストと一対多のリレーションシップが適切に “Multi-Hot エンコーディング” され、複数値を表す列になります。

例えば、各 person の持病のリストを含むテーブルを考えてみます。

Person	条件
Person A	[‘diabetes’, ‘osteoporosis’, ‘asthma’]
Person B	[‘osteoporosis’, ‘hypertension’]
Person C	[‘asthma’, ‘hypertension’]
Person D	[‘hypertension’, ‘asthma’]

多くの機械学習関数では、これらのリストは別個のエンティティとして one-hot エンコーディングで処理され、次のように変換されます。

Person	[‘diabetes’, ‘osteoporosis’, ‘asthma’]	[‘osteoporosis’, ‘hypertension’]	[‘asthma’, ‘hypertension’]	[‘hypertension’, ‘asthma’]
Person A	[1]	0	0	0
Person B	0	[1]	0	0
Person C	0	0	[1]	0
Person D	0	0	0	[1]

AutoML ではむしろ、bag-of-words を使用して、各リストの値ごとに別個の列を作成します。

Person	‘diabetes’	‘osteoporosis’	‘asthma’	‘hypertension’
Person A	[1]	[1]	[1]	0
Person B	0	[1]	0	[1]
Person C	0	0	[1]	[1]
Person D	0	0	[1]	[1]

他の関数では、各 person はそれぞれ別個の持病リストを持つものとして処理されるのに対し、AutoML の手法では、モデルがこれらの各 person の持病セット間のパターンを適切に見つけることができます。

AutoML では、順序は重要ではないと見なします。Person C と Person D は、順序が異なるだけの、同じ持病セットを共有しています。他の関数はこれら 2 つのリストを異なるものとして処理しますが、AutoML は同じものであると識別します。

3

特徴量エンジニアリング

AutoML は、2 つの重要な特徴量エンジニアリングの手順を実行します。

- 列の型分類
- データ変換

これらの手順により、データと使用されている機械学習アルゴリズムとの適合性を確保して、パフォーマンスを著しく向上させることができます。

3.1 列の型分類

AutoML はまずデータセット内の列を調べて、特定の Python データ型として分類します。DDL データ型から Python データ型への変換の詳細は、"DDL から Python への型変換" を参照してください。

以下に、列の型とその分類の方法を示します。

数値列

数値列は、int8、int64、float32 など、数値 pandas データ型を持つ列です。以下を除き、この条件を満たす、すべての列が含まれます。

- 無視される列
- timedelta データ型の列
- 一意の値が 1 つのみの列

数値データのように見える列が、数値列として不適切に分類されることがあります。例えば、さまざまな項目の ID 番号が含まれる列がある場合、ID 番号 1000 は ID 番号 2000 の“半分”ではありません。VARCHAR 値で数値データを再キャストすることで、これらの列をカテゴリ列として適切に処理することができます。

カテゴリ列

カテゴリ列は、カテゴリカル値を含む列です。つまり、比較的少数の固定数の値が表示されます。以下の条件を満たします。

- category または object pandas データ型である。
- 無視される列が含まれていない。
- リスト列が含まれていない。

- 一意の値の数は、値の合計数の 10% 未満である。

テキスト列

テキスト列は、値が文章のように見える列です。AutoML は、4 つ以上の語を含む値を探します。以下の条件を満たします。

- category または object pandas データ型である。
- 無視される列が含まれていない。
- カテゴリ列が含まれていない。
- リスト列が含まれていない。
- 一意の値の数は、値の合計数の 10% 未満である。

リスト列

リスト列は、リスト値を含む列です。以下の条件を満たします。

- category または object pandas データ型である。
- 無視される列が含まれていない。
- 以下のいずれかの型であるか、これらの型を含む。
  - InterSystems IRIS データ型 %Library.String:list。
  - InterSystems IRIS データ型 %Library.String:array。
  - Python リスト。列の空でない最初の 10 個の値を調べて、各値の型が Python リストかどうかを確認して、判断します。
  - 文字列配列。列の空でない最初の 10 個の値を調べて、各値の型が string で、文字 [, ending character] で始まり、長さが 2 文字以上かどうかを確認して、判断します。

ブーリアン列

ブーリアン列は、bool pandas データ型を持つ列です。さらに、無視される列を含まないという条件を満たします。

無視される列

無視される列は、無視され、トレーニングの前に削除される列です。これには、以下のものがあります。

- ID 列
- ラベル列
- 一意の値が 1 つのみの列 (datetime pandas データ型の列を除く)

日/時列

日/時列は、datetime pandas データ型を持つ列です。さらに、無視される列を含まないという条件を満たします。

作成されるその他の日/時列については、以下のセクションを参照してください。

3.1.1 DDL から Python への型変換

以下の表に、AutoML がデータ列を分類するために使用する、DDL データ型から Python データ型へのマッピングを示します。

DDL データ型	Python データ型
BIGINT	integer
BINARY	bytes
BIT	Boolean
DATE	datetime64 (numpy)
DECIMAL	decimal
DOUBLE	float
INTEGER	integer
NUMERIC	float
REAL	float
SMALLINT	integer
TIME	datetime64 (numpy)
TIMESTAMP	datetime64 (numpy)
TINYINT	integer
VARBINARY	bytes
VARCHAR	string

DDL データ型および関連する InterSystems IRIS® データ型の詳細は、"InterSystems SQL リファレンス" の "データ型" を参照してください。

3.2 データ変換

変換関数は、データセット全体を機械学習モデルで使用される形式に変換します。これは、トレーニング・セットにはトレーニングの前に適用され、将来のデータセットには予測が行われる前に適用されます。

列の追加

追加の日/時列が作成されます。datetime 列ごとに、必要に応じて、以下の別個の列が追加されます。

- 時間
- 曜日
- 月

AutoML は期間列も作成します。追加される各列は、元の日/時列のいずれかであり、この列の各値は、その特定の日/時列の日付と他のすべての日/時列の日付との間の期間です。例えば、以下の 3 つの日/時列を持つ患者データを考えてみます。

- 生年月日

- 入院日
- 退院日

AutoML はこれらの列から、経過日時（生年月日と入院日との間の期間）と滞在期間（入院日と退院日との間の期間）という 2 つの有用な期間列を作成します。

最後に、存在するリスト列ごとに、リストのサイズの列が追加されます。つまり、古い列の対応するリストの長さが、新しい列の値になります。

欠落する値の置き換え

データセットの一部の列の値が欠落していて、不完全なことがよくあります。これを補って、パフォーマンスを向上させるために、AutoML は欠落する値/Null 値に値を入力します。

- カテゴリカル列と日付列の場合、AutoML は欠落する値を列の最頻値（最も頻度の高い値）で置き換えます。
- 数値列と期間列の場合、AutoML は欠落する値を列の平均値（平均）で置き換えます。
- リスト列とテキストの場合、AutoML は欠落する値を空の文字列で置き換えます。

数値列の変換

各数値列には、標準のスカラが適合します。これには、元の数値列と、期間列およびリスト・サイズ列が含まれます。

数値列の値もビニングされてから、カテゴリカル値として使用されます。すでに存在する数値列のほかに、これらの新しいカテゴリカル・ビン列が別個に追加されます。各数値列は 4 つのビンに分割され、それぞれがその列の値の四分位数を表します。新規のビニングされた列は、カテゴリカル列として扱われます。

テキスト列とリスト列の変換

各テキスト列とリスト列には、データをトレーニングに必要な形式に変換するためのベクトライザが適合します。これは、SciKit Learn の TFIDF ベクトライザによって行われます。詳細は、こちらのドキュメントを参照してください。

次のパラメータが使用されます。

パラメータ	値
Convert to lowercase	True
Stop Words	なし
N-Gram Range	(1,1)
Max Features	10000
Norm	L2

バイナリ列

バイナリ列は 1 と 0 で構成されるように変換され、true 値は 1 にマッピングされます。

カテゴリカル列

カテゴリカル列は、one-hot エンコーディングされてからトレーニングに使用されます。

特徴量の削除

トレーニングの前の最後の手順として、冗長性の排除、トレーニング速度の改善、およびモデルの正確性の向上のために、特徴量の削除が実行されます。これは、Scikit Learn の SelectFPR 関数を使用して行われます。

次のパラメータが使用されます。

パラメータ	値
スコアリング関数	f_classif
alpha	0.2

4

使用されるアルゴリズム

AutoML は 4 つの異なるモデルを使用して予測を行います。

回帰モデルの場合：

- XGBRegressor

分類モデルの場合：

- ニューラル・ネットワーク
- ロジスティック回帰
- ランダム・フォレスト分類子

4.1 XGBRegressor

回帰モデルには、AutoML は XGBoost の XGBRegressor クラスを使用します。

このモデルのハイパーパラメータの詳細を以下に示します。

ハイパーパラメータ	値
Max Depth	3
Learning Rate	0.1
Number of Estimators	100
Objective	Squared Error
Booster	Gbtree
Tree Method	Auto
Number of Jobs	[1]
Gamma (リーフでの分割のための最小損失削減量)	0
Min Child Weight	[1]
Max Delta Step	0
L2 Regularization Lambda	[1]

ハイパーパラメータ	値
Scale Positive Weight	[1]
Base/Initial Score	0.5

4.2 ニューラル・ネットワーク

ニューラル・ネットワーク・モデルには、AutoML は Keras をラッパとして TensorFlow を使用します。

入力層のサイズは、特徴の数に基づいています。その後、この層は 100 ニューロンで構成される 1 つの非表示層に密に接続されて、ReLU 活性化関数を実装します。この非表示層は最終出力層に密に接続されて、Softmax 活性化関数を実装します。出力層のニューロンの数は、分類のために存在するクラスの数と同じです。

このモデルのハイパーパラメータの詳細を以下に示します。

ハイパーパラメータ	値
Optimizer (name)	Adam
Beta_1	0.9
Beta_2	0.999
Epsilon	1e-07
Amsgrad	False
Loss	Sparse Categorical Crossentropy

4.3 ロジスティック回帰

ロジスティック回帰モデルには、AutoML は SciKit Learn のロジスティック回帰クラスを使用します。

このモデルのハイパーパラメータの詳細を以下に示します。

ハイパーパラメータ	値
Penalty	L2
Dual Formulation	False
Tolerance	1e-4
C (逆正則化パラメータ)	[1]
Fit Intercept	True
Intercept Scaling	[1]
Class Weight	Balanced
Solver	liblinear
Max Iterations	100
Multiclass	One-vs-Rest

ハイパーパラメータ	値
Warm Start	False
Number of Jobs	[1]

4.4 ランダム・フォレスト分類子

ランダム・フォレスト分類子モデルには、AutoML は SciKit Learn のランダム・フォレスト分類子クラスを使用します。このモデルのハイパーパラメータの詳細を以下に示します。

ハイパーパラメータ	値
Number of Estimators	100
Criterion	Gini Impurity
Max Depth	なし
Min Samples to Split	2
Min Samples to be Leaf Node	[1]
Min Fraction of Total Sum of Weights to be Leaf	0
Max Features	Square root of number of features
Max Leaf Nodes	なし
Min Impurity Decrease for Split	0
Bootstrap	True
Number of Jobs	[1]
Warn Start	False
Class Weight	Balanced

5

モデルの選択プロセス

ラベル列が float 型または complex 型の場合、AutoML は XGBRegressor を使用して回帰モデルをトレーニングします。

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。

   データセットのサイズは、列数と行数を掛けて計算されます。この計算されたサイズがターゲット・サイズよりも大きい場合、サンプリングが必要になります。ターゲット・サイズを列数で割って、使用できる行数が計算されます。この数の行がデータセット全体からランダムに選択され、モデルの選択のためにのみ使用されます。

2. 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。
   - 二項分類の問題がある場合は、ROC AUC スコアリング・メトリックが使用されます。
   - それ以外の場合は、F1 スコアリング・メトリックが使用されます。

3. その後、モンテカルロ交差検証を使用して、3 つのトレーニング/テストを 70%/30% で分割して、モデルごとにこれらのスコアリング・メトリックが計算されます。トレーニング・モードに応じて、最適なモデルは次のように決定されます。

   注釈　以下にリストされている数式で、`model_score` は手順 2 のスコアリング・メトリックを表し、`model_time` はモデルのトレーニングに費やした時間を表します。

トレーニング・モード	モデル比較の式
TIME	`(model_score)/(model_time^1.2)`
BALANCE	`(model_score)/(model_time)`
SCORE	`model_score`

例えば、次の 3 つのモデルを比較するとします。

モデル	model_score	model_time
モデル A	0.7	500
モデル B	0.85	600
モデル C	0.87	800

`TIME` トレーニング・モードでは、モデル A が選択されます。

`BALANCE` トレーニング・モードでは、モデル B が選択されます。

SCORE トレーニング・モードでは、モデル C が選択されます。