AutoML Reference

Version 2023.3
2024-05-16

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

AutoML Reference
InterSystems IRIS Data Platform Version 2023.3 2024-05-16

For Support questions about any InterSystems products, contact:

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

Table of Contents

List of Figures

1

Introduction to AutoML

Note: This guide covers the low-level details on AutoML. For information about using AutoML as your provider in IntegratedML, see the IntegratedML User Guide.

AutoML is an automated machine learning system developed by InterSystems, housed within InterSystems IRIS® data platform. It is designed to quickly build accurate predictive models using your data, automating several key components of the machine learning process:

Figure 1–1: The Machine Learning Process

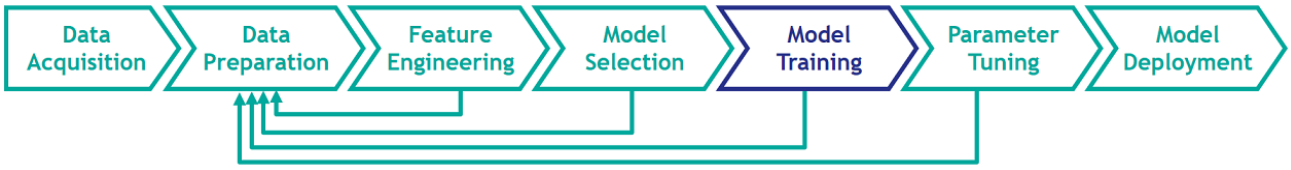

Figure 1–2: Automating the Machine Learning Process

After you train your model with AutoML, you can easily deploy your model by using the SQL syntax provided by IntegratedML.

2

Highlighted Features of AutoML

This section describes some of the machine learning features used by AutoML to smartly produce predictive models.

2.1 Natural Language Processing

AutoML leverages natural language processing (NLP) to turn your text features into numeric features for your predictive models. AutoML uses Term frequency-inverse document frequency (TFIDF) to evaluate key words in text and list columns.

2.2 Multi-Hot Encoding

While most of our data is *sparse*, machine learning algorithms can only understand *dense* data. In most data modeling workflows, data scientists are burdened with performing difficult, and cumbersome, manual transformations to convert their sparse data into dense data.

Unlike many workflows that require this manual step, AutoML performs this conversion seamlessly. Lists and one-to-many relationships are smartly "multi-hot encoded" to account for columns that are representing more than a single value.

For instance, assume a table that contains a list of medical conditions for each person:

Person	Conditions
Person A	['diabetes', 'osteoporosis', 'asthma']
Person B	['osteoporosis', 'hypertension']
Person C	['asthma', 'hypertension']
Person D	['hypertension', 'asthma']

Many machine learning functions treat these lists as separate entities, with one-hot encoding resulting in the following conversion:

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

Instead, AutoML uses bag-of-words to create a separate column for each *value* in each list:

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

While other functions would have treated each person as having a separate list of medical conditions, AutoML’s method allows a model to properly find patterns between each of these persons’ set of medical conditions.

AutoML assumes that order does not matter. Person C and Person D share the same set of medical conditions, but just ordered differently. While other functions treat those two lists differently, AutoML identifies that they are the same.

3

Feature Engineering

AutoML performs two key steps of feature engineering:

- Column Type Classification
- Data Transformation

These steps help make the data compatible with the utilized machine learning algorithms, and can greatly improve performance.

3.1 Column Type Classification

AutoML first examines the columns in the dataset and classifies them as a particular Python data type. For information about the conversion from DDL to Python data types, see DDL to Python Type Conversion.

The column types, along with how their classifications are made, are listed below:

Numeric Columns

Numeric columns are those that have the *numeric* pandas datatype, such as *int8*, *int64*, *float32*, etc. All columns meeting this condition are included, except:

- Ignored Columns.
- Columns of the *timedelta* datatype.
- Columns with only one unique value.

Some columns with seemingly numeric data may be inappropriately classified as numeric columns. For example, if had a column with ID numbers for different items, an ID number of 1000 is not "half of" an ID number of 2000. You can properly treat these columns as category columns by recasting the numeric data with VARCHAR values.

Category Columns

Category columns are those that contain categorical values, meaning there are a relatively small, fixed number of values that appear. They satisfy the following criteria:

- Must be of *category* or *object* pandas datatype.
- Must not include Ignored Columns.

- Must not include List Columns.
- The number of unique values is less than 10% the total number of values.

Text Columns

Text columns are columns where the values look like sentences. AutoML looks for values that contain 4 or more words. They satisfy the following criteria:

- Must be of the *category* or *object* pandas datatype.
- Must not include Ignored Columns.
- Must not include Category Columns.
- Must not include List Columns.
- The number of unique values is less than 10% the total number of values.

List Columns

List columns are those that contain list values. They satisfy the following criteria:

- Must be of *category* or *object* pandas datatype.
- Must not include Ignored Columns.
- Must be, or contain, one of the following types:
  - InterSystems IRIS data type %Library.String:list
  - InterSystems IRIS data type %Library.String:array
  - Python list. This is determined by checking the first 10 non-empty values of the column to see if the type of each value is a Python list.
  - String array. This is determined by checking the first 10 non-empty values of the column to see if the type of each value is a string, with starting character [, ending character], and of length at least 2.

Boolean Columns

Boolean columns are those that have the *bool* pandas datatype. They additionally satisfy the condition that they do not include Ignored Columns.

Ignored Columns

Ignored columns are those that are to be disregarded and removed before training. These include:

- The ID column.
- The label column.
- Columns with only one unique value (except for columns of *datetime* pandas datatype).

Date/Time Columns

Date/Time columns are those that have the *datetime* pandas datatype. They additionally satisfy the condition that they do not include Ignored Columns.

See below for discussion of additional date/time columns created.

3.1.1 DDL to Python Type Conversion

The following table maps DDL data types to the Python data types that AutoML uses to classify data columns.

DDL Data Type	Python Data Type
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

For information about DDL data types, and their associated InterSystems IRIS® data platform data types, see "Data Types" in the *InterSystems SQL Reference*.

3.2 Data Transformation

The Transform Function transforms the entire dataset into the form to be used by the machine learning models. It is applied on the training set before training, and on any future datasets before predictions are made.

Adding Additional Columns

Additional Date/Time columns are created. For every *datetime* column, the following separate columns are added whenever applicable:

- Hour of day.
- Day of week.
- Month of year.

AutoML also creates duration columns. Each column added represents one of the original date/time columns, and each value in this column is the duration between the dates of that particular date/time column and all other date/time columns. For example, consider patient data that has three date/time columns:

- Date of birth.

- Time of admission.
- Time of exit.

AutoML creates two useful duration columns from these columns: age (duration between date of birth and time of admission) and length of stay (duration between time of admission and exit).

Finally, for each list column present, another column is added simply with the size of the lists. That is, each value in the new column is the length of the corresponding list in the old column.

Replacing Missing Values

Datasets can often be incomplete, with missing values in some of their columns. To help compensate for this and improve performance, AutoML fills in missing/NULL values:

- For categorical and date columns, AutoML replaces missing values with the mode (most popular value) of the column.
- For numeric and duration columns, AutoML replaces missing values with the mean (average) of the column.
- For list and text columns, AutoML replaces missing values with an empty string.

Transforming Numeric Columns

For each numeric column, a standard scalar is fit. These include the original numeric columns, along with the duration and list size columns as well.

Numerical column values are also binned and then used as categorical values. These new categorical bin columns are added on separately in addition to the already present numerical columns. Each numerical column is separated into four bins, each representing a quartile of the values in that column. The new binned columns are treated as categorical columns.

Transforming Text and List Columns

For each text and list column, a vectorizer is fit to transform the data to the appropriate form needed for training. This is done with SciKit Learn's TFIDF Vectorizer. Please see their documentation.

The following parameters are used:

Parameter	Value
Convert to lowercase	True
Stop Words	None
N-Gram Range	(1,1)
Max Features	10000
Norm	L2

Binary Columns

Binary columns are simply transformed to be composed of 1's and 0's, with true values mapping to 1's.

Categorical Columns

Categorical columns are one-hot encoded before being used for training.

Feature Elimination

As the last step before training, feature elimination is performed to remove redundancy, improve training speed, and improve the accuracy of models. This is done using Scikit Learn's SelectFPR function.

The following parameters are used:

Parameter	Value
Scoring function	f_classif
alpha	0.2

4

Algorithms Used

AutoML uses four different models to make predictions.

For regression models:

- XGBRegressor

For classification models:

- Neural Network
- Logistic Regression
- Random Forest Classifier

4.1 XGBRegressor

For regression models, AutoML uses XGBoost's XGBRegressor class.

The model hyperparameters are detailed below:

Hyperparameter	Value
Max Depth	3
Learning Rate	0.1
Number of Estimators	100
Objective	Squared Error
Booster	Gbtree
Tree Method	Auto
Number of Jobs	1
Gamma (min loss reduction for partition on leaf)	0
Min Child Weight	1
Max Delta Step	0
L2 Regularization Lambda	1

Hyperparameter	Value
Scale Positive Weight	1
Base/Initial Score	0.5

4.2 Neural Network

For the Neural Network model, AutoML uses TensorFlow with Keras as a wrapper.

The input layer has its size based on the number of features. This layer is then densely connected to a single hidden layer composed of 100 neurons, which implement the ReLU Activation Function. This hidden layer is densely connected to the final output layer, which implements the Softmax Activation Function. The number of neurons in the output layer is equivalent to the number of classes present for classification.

The model hyperparameters are detailed below:

Hyperparameter	Value
Optimizer (name)	Adam
Beta_1	0.9
Beta_2	0.999
Epsilon	1e-07
Amsgrad	False
Loss	Sparse Categorical Crossentropy

4.3 Logistic Regression

For the Logistic Regression Model, AutoML uses SciKit Learn's Logistic Regression class.

The model hyperparameters are detailed below:

Hyperparameter	Value
Penalty	L2
Dual Formulation	False
Tolerance	1e-4
C (Inverse Regularization Parameter)	1
Fit Intercept	True
Intercept Scaling	1
Class Weight	Balanced
Solver	liblinear
Max Iterations	100

Hyperparameter	Value
Multiclass	One-vs-Rest
Warm Start	False
Number of Jobs	1

4.4 Random Forest Classifier

For the Random Forest Classifier model, AutoML uses SciKit Learn's Random Forest Classifier class.

The model hyperparameters are detailed below:

Hyperparameter	Value
Number of Estimators	100
Criterion	Gini Impurity
Max Depth	None
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	None
Min Impurity Decrease for Split	0
Bootstrap	True
Number of Jobs	1
Warn Start	False
Class Weight	Balanced

5

Model Selection Process

If the label column is of type float or complex, AutoML trains a regression model using XGBRegressor.

For classification models, AutoML uses the following selection process to determine the most accurate model:

1. If the dataset is too large, AutoML samples down the data to speed up the model selection process. The full dataset is still used for training after model selection.

   The size of the dataset is calculated by multiplying the number of columns by the number of rows. If this calculated size is larger than the target size, sampling is needed. The number of rows that can be utilized is calculated by dividing the target size by the number of columns. This number of rows is randomly selected from the entire dataset to be used *only* for the purposes of model selection.

2. AutoML determines if the dataset presents a binary classification problem, or if multiple classes are present.
   - If it is a binary classification problem, the ROC AUC scoring metric is used.
   - Otherwise, the F1 scoring metric is used.

3. These scoring metrics are then computed for each model using Monte Carlo cross validation, with three training/testing splits of 70%/30%. Depending on the training mode, the best model is determined as follows:

   Note: For the mathematical expressions listed below, `model_score` represents the scoring metric from step 2, while `model_time` is the time spent training the model.

Training Mode	Expression for Model Comparison
TIME	`(model_score)/(model_time^1.2)`
BALANCE	`(model_score)/(model_time)`
SCORE	`model_score`

   For example, if the following three models were being compared:

Model	model_score	model_time
Model A	0.7	500
Model B	0.85	600
Model C	0.87	800

   In the `TIME` training mode, *Model A* would be selected.

In the `BALANCE` training mode, *Model B* would be selected.

In the `SCORE` training mode, *Model C* would be selected.