# Driverless AI Experiment: ponasowa

Generated by: h2oai

Generated on: Wed Apr 15 04:27:35 2020

[Driverless AI Experiment: ponasowa 1](#_Toc20130252)

[Experiment Overview 1](#_Toc20130253)

[Data Overview 3](#_Toc20130254)

[Methodology 5](#_Toc20130255)

[Data Sampling 10](#_Toc20130256)

[Validation Strategy 10](#_Toc20130257)

[Model Tuning 11](#_Toc20130258)

[Feature Evolution 13](#_Toc20130259)

[Feature Transformations 13](#_Toc20130260)

[Final Model 15](#_Toc20130261)

[Alternative Models 21](#_Toc20130262)

[Deployment 22](#_Toc20130263)

[Partial Dependence Plots 23](#_Toc20130264)

[Appendix 25](#_Toc20130266)

## Experiment Overview

Driverless AI built 1 XGBoostGBMModel to predict *Default* given 24 original features from the input dataset *Card\_train.* This classification experiment completed in 2 minutes and 39 seconds (0:02:39), using 20 of the 24 original features, and 2 of the 26 engineered features.

### Performance

|  |  |
| --- | --- |
| **Dataset** | **AUC** |
| Internal Validation | 0.778 |
| Test Data | Test Data not Provided |

### Driverless Settings

|  |  |  |  |
| --- | --- | --- | --- |
| Dial Settings | Description | Setting Value | Range of Possible Values |
| Accuracy | Controls accuracy needs of the model | 3 | 1-10 |
| Time | Controls duration of the experiment | 3 | 1-10 |
| Interpretability | Controls complexity of the model | 8 | 1-10 |

### System Specifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Address** | **System** | **System Memory** | **CPUs** | **GPUs** |
| http://127.0.0.1:12345 | Docker/Linux | 60 GB | 8 | 1 |

### Versions

|  |  |
| --- | --- |
| **Driverless AI version** | 1.8.5.1 |
| **h2o4gpu version** | 0.3.2 |
| **h2o\_mli version** | 0.1.106 |
| **mojo2\_runtime version** | 2.2.0 |
| **procsy version** | 0.6.0 |
| **pydatatable version** | 0.11.0a269 |
| **vis\_data\_server version** | 2.0.4 |

## Data Overview

This section provides information on the datasets used for the experiment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **data** | **file path** | **file size** | **number of rows** | **number of columns** |
| training | ./tmp/eaa648cc-7ed0-11ea-96d6-0242ac110002/Card\_train.1586924634.452401.bin | 2.3 MiB | 23,999 | 25 |
| validation | Not provided | None | None | None |
| testing | Not provided | None | None | None |

### Training Data

The training data consists of both numeric and categorical columns.

The summary of the columns is shown below:

#### Numeric Columns

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **name** | **data\_type** | **min** | **mean** | **max** | **std** | **unique** | **freq of mode** |
| ID | int | 2.000 | 15,037.974 | 30,000.000 | 8,652.141 | 23,999 | 1 |
| CreditLimit | int | 10,000.000 | 167,997.403 | 800,000.000 | 130,585.996 | 80 | 2,724 |
| Education | int | 0.000 | 1.855 | 6.000 | 0.788 | 7 | 11,255 |
| Age | int | 21.000 | 35.507 | 79.000 | 9.203 | 54 | 1,277 |
| Status1 | int | -2.000 | -0.151 | 1.000 | 0.875 | 4 | 11,761 |
| Status2 | int | -2.000 | -0.160 | 2.000 | 1.129 | 5 | 12,551 |
| Status3 | int | -2.000 | -0.181 | 3.000 | 1.158 | 6 | 12,571 |
| Status4 | int | -2.000 | -0.234 | 4.000 | 1.133 | 7 | 13,121 |
| Status5 | int | -2.000 | -0.274 | 5.000 | 1.114 | 7 | 13,472 |
| Status6 | int | -2.000 | -0.296 | 6.000 | 1.142 | 8 | 12,988 |
| BillAmt1 | int | -154,973.000 | 51,102.599 | 746,814.000 | 73,602.907 | 18,635 | 1,609 |
| BillAmt2 | int | -69,777.000 | 48,950.813 | 605,943.000 | 70,873.894 | 18,370 | 2,025 |
| BillAmt3 | int | -157,264.000 | 46,789.205 | 1,664,089.000 | 69,379.414 | 18,077 | 2,305 |
| BillAmt4 | int | -170,000.000 | 43,028.569 | 616,836.000 | 64,115.442 | 17,721 | 2,571 |
| BillAmt5 | int | -81,334.000 | 40,183.782 | 823,540.000 | 60,920.045 | 17,331 | 2,811 |
| BillAmt6 | int | -209,051.000 | 38,784.077 | 699,944.000 | 59,752.504 | 16,932 | 3,194 |
| PayAmt1 | int | 0.000 | 5,654.421 | 873,552.000 | 16,608.025 | 6,878 | 4,223 |
| PayAmt2 | int | 0.000 | 5,949.978 | 1,684,259.000 | 24,270.503 | 6,850 | 4,354 |
| PayAmt3 | int | 0.000 | 5,187.113 | 889,043.000 | 16,863.107 | 6,530 | 4,754 |
| PayAmt4 | int | 0.000 | 4,875.825 | 621,000.000 | 16,073.541 | 6,088 | 5,092 |
| PayAmt5 | int | 0.000 | 4,834.457 | 426,529.000 | 15,586.997 | 6,016 | 5,290 |
| PayAmt6 | int | 0.000 | 5,217.418 | 528,666.000 | 17,407.197 | 6,062 | 5,701 |

#### Boolean Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **name** | **data\_type** | **min** | **mean** | **max** | **std** | **freq of max value** |
| Default | bool | False | 0.2212 | True | 0.415 | 5,308 |

#### Categorical Columns

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **unique** | **top** | **freq of top value** |
| Sex | 2 | F | 9,502 |
| Marriage | 4 | S | 12,782 |

### Shifts Detected

Driverless AI can perform shift detection between the training, validation, and testing datasets. It does this by training a binomial model to predict which dataset a record belongs to. For example, it may find that it is able to separate the training and testing data with an AUC of 0.8 using only the column: C1 as the predictor. This indicates that there is some sort of drift in the distribution of C1 between the training and testing data.

For this experiment, Driverless AI was not able to check for distribution shifts because only the training dataset was supplied by the user.

## Methodology

This section describes the experiment methodology.

### Assumptions and Limitations

Driverless AI trains all models based on the training data provided (in this case: *Card\_train*). It is the assumption of Driverless AI that this dataset is representative of the data that will be seen when scoring.

Driverless AI may perform shift detection between the train data and another dataset. If a shift in distribution is detected, this may indicate that the data that will be used for scoring may have distributions not represented in the training data.

For this experiment, Driverless AI was not able to detect any shift in distribution between train data and another dataset because no validation or test data was provided.

### Experiment Pipeline

For this experiment, Driverless AI performed the following steps to find the optimal final model:



The steps in this pipeline are described in more detail below:

* **Ingest Data** 
  + - detected column types
* **Feature Preprocessing**
  + turned raw features into numeric
* **Model and Feature Tuning**

This stage combines random hyperparameter tuning with feature selection and generation. Features in each iteration are updated using variable importance from the previous iteration as a probabilistic prior to decide what new features to create. The best performing model and features are then passed to the feature evolution stage.

* + found the optimal parameters for xgboost, lightgbm, constant, linear and decision tree models by training models with different parameters
  + the best parameters are those that generate the largest **AUC** on the internal validation data
  + 7 models trained and scored to evaluate features and model parameters
* **Feature Evolution**

This stage uses a genetic algorithm to find the best set of model parameters and feature transformations to be used in the final model.

* + found the best representation of the data for the final model training by creating and evaluating **26** features over **9** iterations
  + trained and scored 6 models to further evaluate engineered features
* **Final Model**
  + created the best model from the feature engineering iterations
    - no stacked ensemble is done due to accuracy or ensemble level settings (consider increasing accuracy or the ensemble\_level)
* **Create Scoring Pipeline** 
  + created and exported the MOJO and Python scoring pipeline
    - MOJO Scoring Pipeline: h2oai\_experiment\_04c896ce-7ed1-11ea-96d6-0242ac110002/mojo\_pipeline/mojo.zip
    - Python Scoring Pipeline: h2oai\_experiment\_04c896ce-7ed1-11ea-96d6-0242ac110002/scoring\_pipeline/scorer.zip

Driverless AI trained models throughout the experiment in an effort to determine the best parameters, model dataset, and optimal final model. The stages are described below:

|  |  |  |
| --- | --- | --- |
| Driverless AI Stage | Timing (seconds) | Number of Models |
| Data Preparation | 14.96 | 0 |
| Model and Feature Tuning | 26.48 | 7 |
| Feature Evolution | 36.42 | 6 |
| Final Pipeline Training | 45.50 | 9 |

### Experiment Settings

Below are the settings selected for the experiment by h2oai. The Defined Parameters represent the high-level parameters.

**Defined Parameters**

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| is\_classification | True |
| enable\_gpus | True |
| seed | True |
| accuracy | 3 |
| time | 3 |
| interpretability | 8 |
| time\_groups\_columns | None |
| num\_prediction\_periods | None |
| num\_gap\_periods | None |
| is\_timeseries | False |

These Accuracy, Time, and Interpretability settings map to the following internal configuration of the Driverless AI experiment:

|  |  |
| --- | --- |
| **Internal Parameter** | **Value** |
| data filtered | False |
| number of feature engineering iterations | 30 |
| number of models trained per iteration | 2 |
| early stopping rounds | 5 |
| monotonicity constraint | True |
| number of model tuning model combinations | 6 |
| number of base learners in ensemble | 0 |
| time column | [OFF] |

#### Details

* **data filtered**: Driverless AI may filter the training data depending on the number of rows and the Accuracy setting.
  + for this experiment, the training data was not filtered.
* **number of feature engineering iterations**: the number of iterations performed of feature engineering.
* **number of models evaluated per iteration**: for each feature engineering iteration, Driverless AI trains multiple models. Each model is trained with a different set of predictors or features. The goal of this step is to determine which types of features lead to the largest AUC.
* **early stopping rounds**: if Driverless AI does not see any improvement after 5 iterations of feature engineering, the feature engineering step is automatically stopped.
* **monotonicity constraint**: if enabled, the models will only have monotone relationships between the predictors and target variable.
* **number of model tuning combinations**: the number of model tuning combinations evaluated to determine the optimal model settings for the lightgbm and xgboost models.
* **number of base learners in ensemble**: the number of base models used to create the final ensemble.
* **time column**: the column that provides the time column. If a time column is provided, feature engineering and model validation will respect the causality of time. If the time column is turned off, no time order is used for modeling and data may be shuffled randomly (any potential temporal causality will be ignored).

## Data Sampling

Driverless AI did not perform any down sampling of the data.

## Validation Strategy

Driverless AI automatically split the training data to determine the performance of the model parameter tuning and feature engineering steps. For the experiment, Driverless AI randomly split the data into **3/4 training** and **1/4 validation**.

## Model Tuning

The table below shows the score and training time of the xgboost, lightgbm, constant, linear and decision tree models evaluated by Driverless AI. The table shows the parameter tuning models evaluated, ordered based on a combination of largest score and lowest training time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **job order** | **booster** | **nfeatures** | **scores** | **training times** |
| 3 | gbtree | 23 | 0.7833 | 2.2397 |
| 0 | lightgbm | 23 | 0.7814 | 1.002 |
| 1 | lightgbm | 24 | 0.7814 | 1.3111 |
| 2 | decision tree | 24 | 0.7675 | 0.5954 |
| 4 | gblinear | 24 | 0.7097 | 1.4459 |
| 5 | constant | 1 | 0.5 | 0.2421 |

More detailed information on the parameters evaluated for each algorithm is shown below.

### gbtree tuning

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **tree method** | **grow policy** | **max depth** | **max leaves** | **colsample bytree** | **subsample** | **nfeatures** | **scores** | **training times** |
| gpu\_hist | depthwise | 6.0 | 0.0 | 0.8 | 0.7 | 23 | 0.7833 | 2.2397 |

### lightgbm tuning

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **tree method** | **grow policy** | **max depth** | **max leaves** | **colsample bytree** | **subsample** | **nfeatures** | **scores** | **training times** |
| gpu\_hist | depthwise | 6.0 | 0.0 | 0.8 | 0.7 | 23 | 0.7814 | 1.002 |
| gpu\_hist | depthwise | 6.0 | 0.0 | 0.8 | 0.7 | 24 | 0.7814 | 1.3111 |

### constant tuning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **job order** | **booster** | **nfeatures** | **scores** | **training times** |
| 5 | constant | 1 | 0.5 | 0.2421 |

### gblinear tuning

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **reg alpha** | **reg lambda** | **n lambda** | **min lambda fraction** | **nfeatures** | **scores** | **training times** |
| 0.0005 | 0.001 | 1 | 0.0001 | 24 | 0.7097 | 1.4459 |

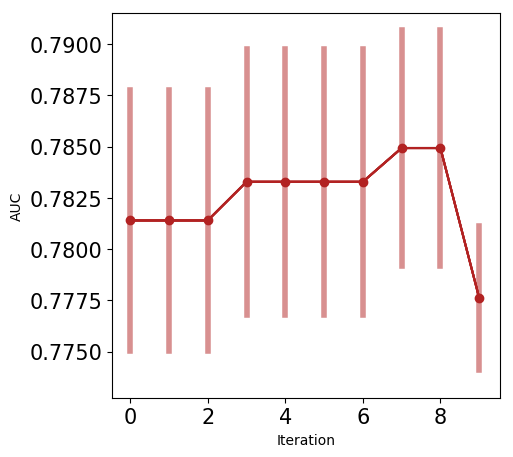
### decision tree tuning

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tree method** | **grow policy** | **max depth** | **max leaves** | **nfeatures** | **scores** | **training times** |
| gpu\_hist | lossguide | 10.0 | 128.0 | 24 | 0.7675 | 0.5954 |

## Feature Evolution

During the Model and Feature Tuning Stage, Driverless AI evaluates the effects of different types of algorithms, algorithm parameters, and features. The goal of the Model and Feature Tuning Stage is to determine the best algorithm and parameters to use during the Feature Evolution Stage. In the Feature Evolution Stage, Driverless AI trained lightgbm and xgboost models (6) where each model evaluated a different set of features. The Feature Evolution Stage uses a genetic algorithm to search the large feature engineering space.

The graph below shows the effect the Model and Feature Tuning Stage and Feature Evolution Stage had on the performance.



Based on the experiment settings and column types in the dataset, Driverless AI was able to explore the following transformers:

* **ClusterDistTransformer**: the Cluster Distance Transformer clusters selected numeric columns and uses the distance to a specific cluster as a new feature.
* **ClusterTETransformer**: the Cluster Target Encoding Transformer clusters selected numeric columns and calculates the mean of the response column for each cluster. The mean of the response is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.
* **InteractionsTransformer**: the Interactions Transformer adds, divides, multiplies, and subtracts two numeric columns in the data to create a new feature. This transformation uses a smart search to identify which feature pairs to transform. Only interactions that improve the baseline model score are kept.
* **NumToCatTETransformer**: the Numeric to Categorical Target Encoding Transformer converts numeric columns to categoricals by binning and then calculates the mean of the response column for each group. The mean of the response for the bin is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.
* **NumToCatWoETransformer**: the Numeric to Categorical Weight of Evidence Transformer converts a numeric column to categorical by binning and then calculates Weight of Evidence for each bin. The Weight of Evidence is used as a new feature. Weight of Evidence measures the “strength” of a grouping for separating good and bad risk and is calculated by taking the log of the ratio of distributions for a binary response column.
* **TruncSVDNumTransformer**: the Truncated SVD Transformer trains a Truncated SVD model on selected numeric columns and uses the components of the truncated SVD matrix as new features.
* **CVTargetEncodeTransformer**: the Cross Validation Target Encoding Transformer calculates the mean of the response column for each value in a categorical column and uses this as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.
* **FrequentTransformer**: the Frequent Transformer calculates the frequency for each value in categorical column(s) and uses this as a new feature. This count can be either the raw count or the normalized count.
* **WeightOfEvidenceTransformer**: the Weight of Evidence Transformer calculates Weight of Evidence for each value in categorical column(s). The Weight of Evidence is used as a new feature. Weight of Evidence measures the “strength” of a grouping for separating good and bad risk and is calculated by taking the log of the ratio of distributions for a binary response column.
* **CatTransformer**: the Categorical Transformer sorts a categorical column in lexicographical order and uses the order index created as a new feature. This transformer works with models that can handle categorical features.
* **OneHotEncodingTransformer**: the One-hot Encoding transformer converts a categorical column to a series of boolean features by performing one-hot encoding. The boolean features are used as new features.
* **DatesTransformer**: the Date Transformer retrieves any date or time values, including: Year, Quarter, Month, Day, Day of Year, Week, Weekday, Hour, Minute, Second.
* **IsHolidayTransformer**: the Is Holiday Transformer determines if a date column is a holiday. A boolean column indicating if the date is a holiday is added as a new feature. Creates a separate feature for holidays in the United States, United Kingdom, Germany, Mexico, and the European Central Bank. Other countries available in the python Holiday package can be added via the configuration file.
* **TextBiGRUTransformer**: the Text Bidirectional GRU Transformer trains a bi-directional GRU TensorFlow model on word embeddings created from a text feature to predict the response column. The GRU prediction is used as a new a feature. Cross Validation is used when training the GRU model to prevent overfitting.
* **TextCNNTransformer**: the Text CNN Transformer trains a CNN TensorFlow model on word embeddings created from a text feature to predict the response column. The CNN prediction is used as a new a feature. Cross Validation is used when training the CNN model to prevent overfitting.
* **TextCharCNNTransformer**: the Text Character CNN Transformer trains a CNN TensorFlow model on character embeddings created from a text feature to predict the response column. The CNN prediction is used as a new a feature. Cross Validation is used when training the CNN model to prevent overfitting.
* **TextLinModelTransformer**: the Text Linear Model Transformer trains a linear model on a TF-IDF matrix created from a text feature to predict the response column. The linear model prediction is used as a new feature. Cross Validation is used when training the linear model to prevent overfitting.
* **TextTransformer**: the Text Transformer tokenizes a text column and creates a TFIDF matrix (term frequency-inverse document frequency) or count (count of the word) matrix. This may be followed by dimensionality reduction using truncated SVD. Selected components of the TF-IDF/Count matrix are used as new features.
* **CVCatNumEncodeTransformer**: the Cross Validation Categorical to Numeric Encoding Transformer calculates an aggregation of a numeric column for each value in a categorical column (ex: calculate the mean Temperature for each City) and uses this aggregation as a new feature.
* **NumCatTETransformer**: the Numeric Categorical Target Encoding Transformer calculates the mean of the response column for several selected columns. If one of the selected columns is numeric, it is first converted to categorical by binning. The mean of the response column is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

## Feature Transformations

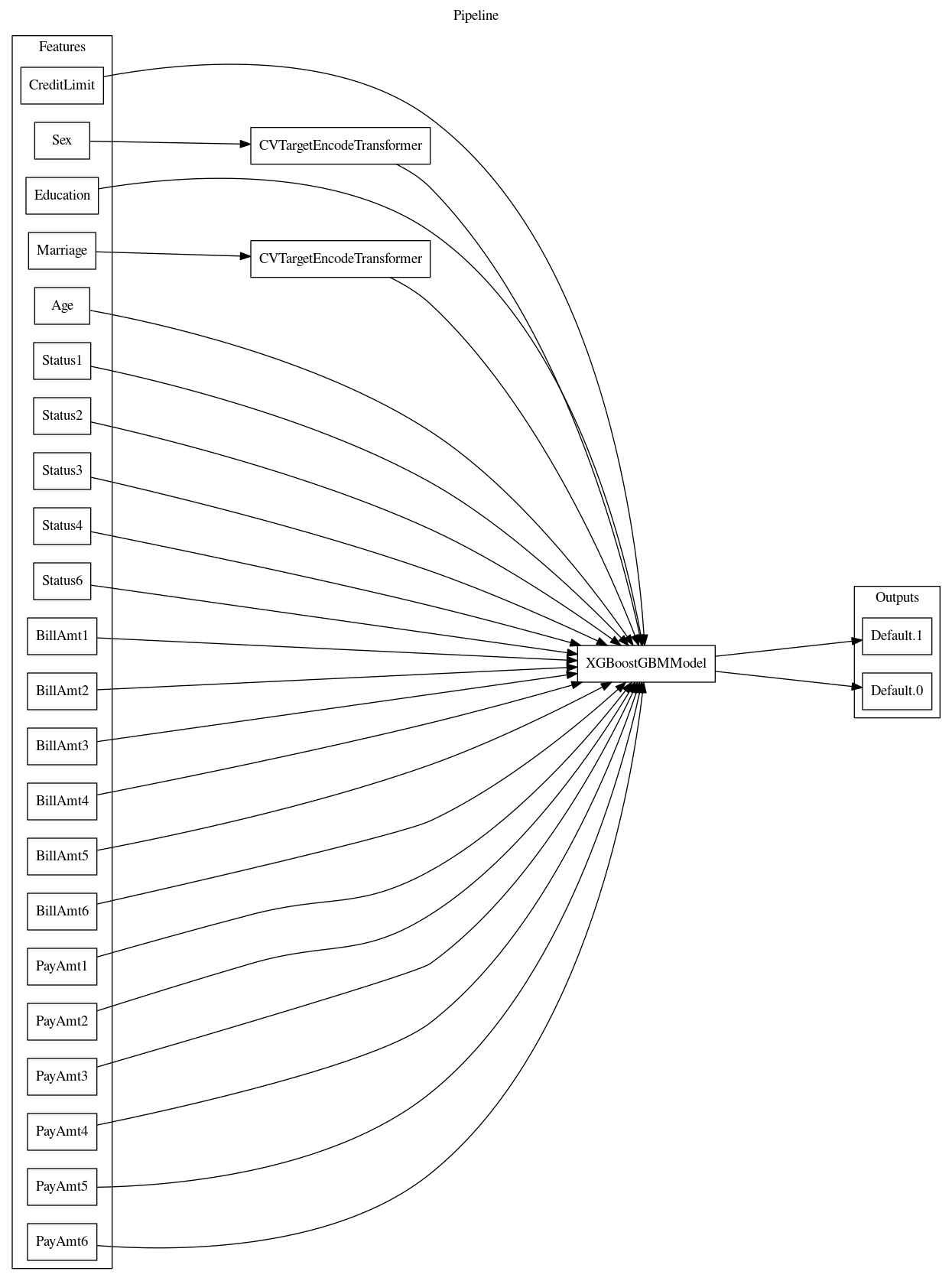
The result of the Feature Evolution Stage is a set of features to use for the final model. Some of these features were automatically created by Driverless AI. The top features used in the final model are shown below, ordered by importance. The features in the table are limited to the top 50, restricted to those with relative importance greater than or equal to 0.003. If no transformer was applied, the feature is an original column.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Feature** | **Description** | **Transformer** | **Relative Importance** |
| 1 | 15\_Status1 | Status1 (Orig) | None | 1.0 |
| 2 | 16\_Status2 | Status2 (Orig) | None | 0.4762 |
| 3 | 17\_Status3 | Status3 (Orig) | None | 0.3713 |
| 4 | 1\_BillAmt1 | BillAmt1 (Orig) | None | 0.234 |
| 5 | 7\_CreditLimit | CreditLimit (Orig) | None | 0.1949 |
| 6 | 9\_PayAmt1 | PayAmt1 (Orig) | None | 0.1806 |
| 7 | 10\_PayAmt2 | PayAmt2 (Orig) | None | 0.1431 |
| 8 | 18\_Status4 | Status4 (Orig) | None | 0.1325 |
| 9 | 20\_Status6 | Status6 (Orig) | None | 0.127 |
| 10 | 12\_PayAmt4 | PayAmt4 (Orig) | None | 0.1235 |
| 11 | 11\_PayAmt3 | PayAmt3 (Orig) | None | 0.1215 |
| 12 | 4\_BillAmt4 | BillAmt4 (Orig) | None | 0.1044 |
| 13 | 2\_BillAmt2 | BillAmt2 (Orig) | None | 0.0909 |
| 14 | 0\_Age | Age (Orig) | None | 0.083 |
| 15 | 13\_PayAmt5 | PayAmt5 (Orig) | None | 0.0766 |
| 16 | 6\_BillAmt6 | BillAmt6 (Orig) | None | 0.0697 |
| 17 | 5\_BillAmt5 | BillAmt5 (Orig) | None | 0.0668 |
| 18 | 3\_BillAmt3 | BillAmt3 (Orig) | None | 0.0653 |
| 19 | 14\_PayAmt6 | PayAmt6 (Orig) | None | 0.0576 |
| 20 | 21\_CVTE: Marriage.0 | Out-of-fold mean of the response grouped by: ['Marriage'] using 5 folds [internal parameters:(10, 3, 10)] | Cross Validation Target Encoding | 0.0518 |
| 21 | 22\_CVTE: Sex.0 | Out-of-fold mean of the response grouped by: ['Sex'] using 5 folds [internal parameters:(10, 3, 10)] | Cross Validation Target Encoding | 0.044 |
| 22 | 8\_Education | Education (Orig) | None | 0.0339 |

## Final Model

**Pipeline**

Final XGBoostGBMModel pipeline with ensemble\_level=0 transforming 22 original features -> 22 features in each of 1 models each of 4 fold hyperparameters averaged and re-fit as single model.:



**Details**

* The fitted features of the final model are the best features found during the feature engineering iterations.
* The target transformer indicates the type of transformation applied to the target column.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model Index** | **Type** | **Model Weight** | **Num Folds** | **Fitted features** | **Target Transformer** |
| 0 | XGBoostGBMModel | 1 | 4 | 22 | LabelEncoder |

* Model Index: 0 has a weight of 1 in the final ensemble

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Type** | **learning rate** | **model class name** | **grow policy** | **max leaves** | **subsample** | **index** | **tree method** | **max depth** | **colsample bytree** |
| XGBoostGBMModel | 0.05 | XGBoostGBMModel | depthwise | 0 | 0.6 | 0 | gpu\_hist | 5 | 0.5 |

For a complete list of the parameters of the final model, see the Appendix.

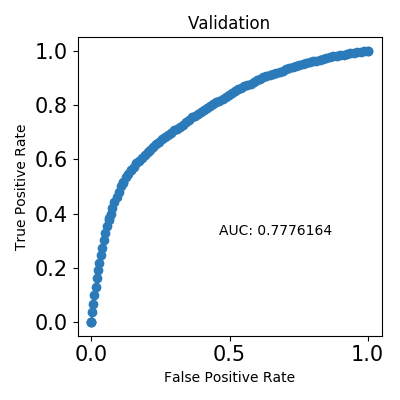
**Performance of Final Model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scorer** | **Optimized** | **Better score is** | **Final ensemble scores on validation (internal or external holdout(s)) data** | **Final ensemble standard deviation on validation (internal or external holdout(s)) data** |
| ACCURACY |  | higher | 0.8138672 | 0.002481037 |
| AUC | \* | higher | 0.7776164 | 0.003504786 |
| AUCPR |  | higher | 0.533133 | 0.008701996 |
| F05 |  | higher | 0.5628527 | 0.007161527 |
| F1 |  | higher | 0.5438757 | 0.006514066 |
| F2 |  | higher | 0.6386541 | 0.004620936 |
| GINI |  | higher | 0.5552328 | 0.007009572 |
| LOGLOSS |  | lower | 0.4337286 | 0.003290567 |
| MACROAUC |  | higher | 0.7776164 | 0.003504786 |
| MCC |  | higher | 0.4153862 | 0.008276535 |

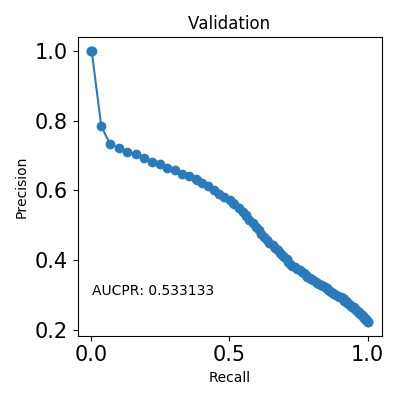
**Validation Confusion Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Predicted: 0** | **Predicted: 1** | **error** |
| Actual: 0 | 16,034 | 2,657 | 14% |
| Actual: 1 | 2,333 | 2,975 | 44% |

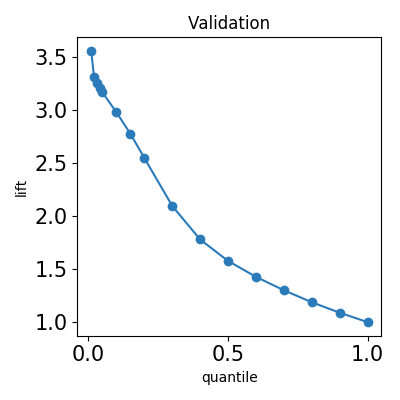
*Receiver Operating Characteristic Curve*



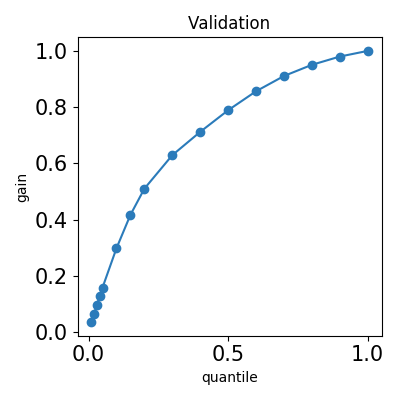
*Precision Recall Curve*



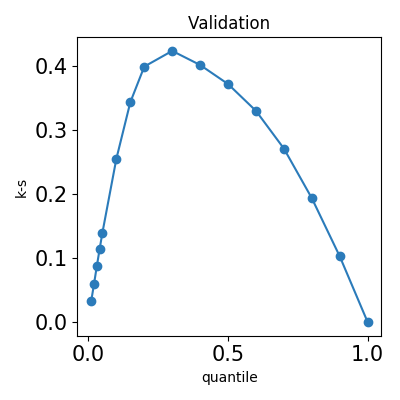
*Cumulative Lift*



*Cumulative Gains*



*Kolmogorov–Smirnov*



## Alternative Models

During the experiment, Driverless AI trained 13 alternative models. The following algorithms were evaluated during the Driverless AI experiment:

|  |  |  |  |
| --- | --- | --- | --- |
| **algorithm** | **package** | **version** | **documentation** |
| gbtree | xgboost | 0.90 | XGBoost: eXtreme Gradient Boosting library. Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTORS.md |
| lightgbm | lightgbm | 2.2.4 | LightGBM, Light Gradient Boosting Machine. Contributors: https://github.com/microsoft/LightGBM/graphs/contributors. |
| constant | custom package | 1.8.5.1 | reference model that predicts a constant aimed at minimizing the given scorer |
| gblinear | xgboost | 0.90 | XGBoost: eXtreme Gradient Boosting library. Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTORS.md |
| decision tree | lightgbm | 2.2.4 | LightGBM, Light Gradient Boosting Machine. Contributors: https://github.com/microsoft/LightGBM/graphs/contributors. |

Driverless AI is able to evaluate the algorithms: XGBoost GBM, XGBoost Dart, XGBoost GLM, LightGBM, RuleFit, Tensorflow, and FTRL models. The table below explains why certain algorithms were not selected for the final model, if any.

|  |  |
| --- | --- |
| **algorithm** | **selection** |
| rulefit | algorithm not evaluated due to experiment configuration |
| tensorflow | algorithm not evaluated due to experiment configuration |
| ftrl | algorithm not evaluated due to experiment configuration |
| dart | algorithm not evaluated due to experiment configuration |
| gblinear | not selected due to low performance during model tuning stage |
| decision tree | not selected due to low performance during model tuning stage |
| lightgbm | not selected due to low performance during feature evolution stage |
| gbtree | selected for final model |

## Deployment

For this experiment, both Python and MOJO Scoring Pipelines are available for productionizing the final model pipeline for a given row of data or table of data.

### Python Scoring Pipeline

This package contains an exported model and Python 3.6 source code examples for productionizing models built using H2O Driverless AI. The Python Scoring Pipeline is located here:

* **h2oai\_experiment\_04c896ce-7ed1-11ea-96d6-0242ac110002/scoring\_pipeline/scorer.zip**

The files in this package allow you to transform and score on new data in a couple of different ways:

* From Python 3.6, you can import a scoring module, then use the module to transform and score on new data.
* From other languages and platforms, you can use the TCP/HTTP scoring service bundled with this package to call into the scoring pipeline module through remote procedure calls (RPC).

### MOJO Scoring Pipeline

Note: The MOJO Scoring Pipeline is currently in a beta state. Updates and improvements will continue to be made in subsequent Driverless AI releases. The MOJO Scoring Pipeline is located here:

* **h2oai\_experiment\_04c896ce-7ed1-11ea-96d6-0242ac110002/mojo\_pipeline/mojo.zip**

For completed experiments, Driverless AI converts models to MOJOs (Model Objects, Optimized). A MOJO is a scoring engine that can be deployed in any Java environment for scoring in real time.

## Partial Dependence Plots

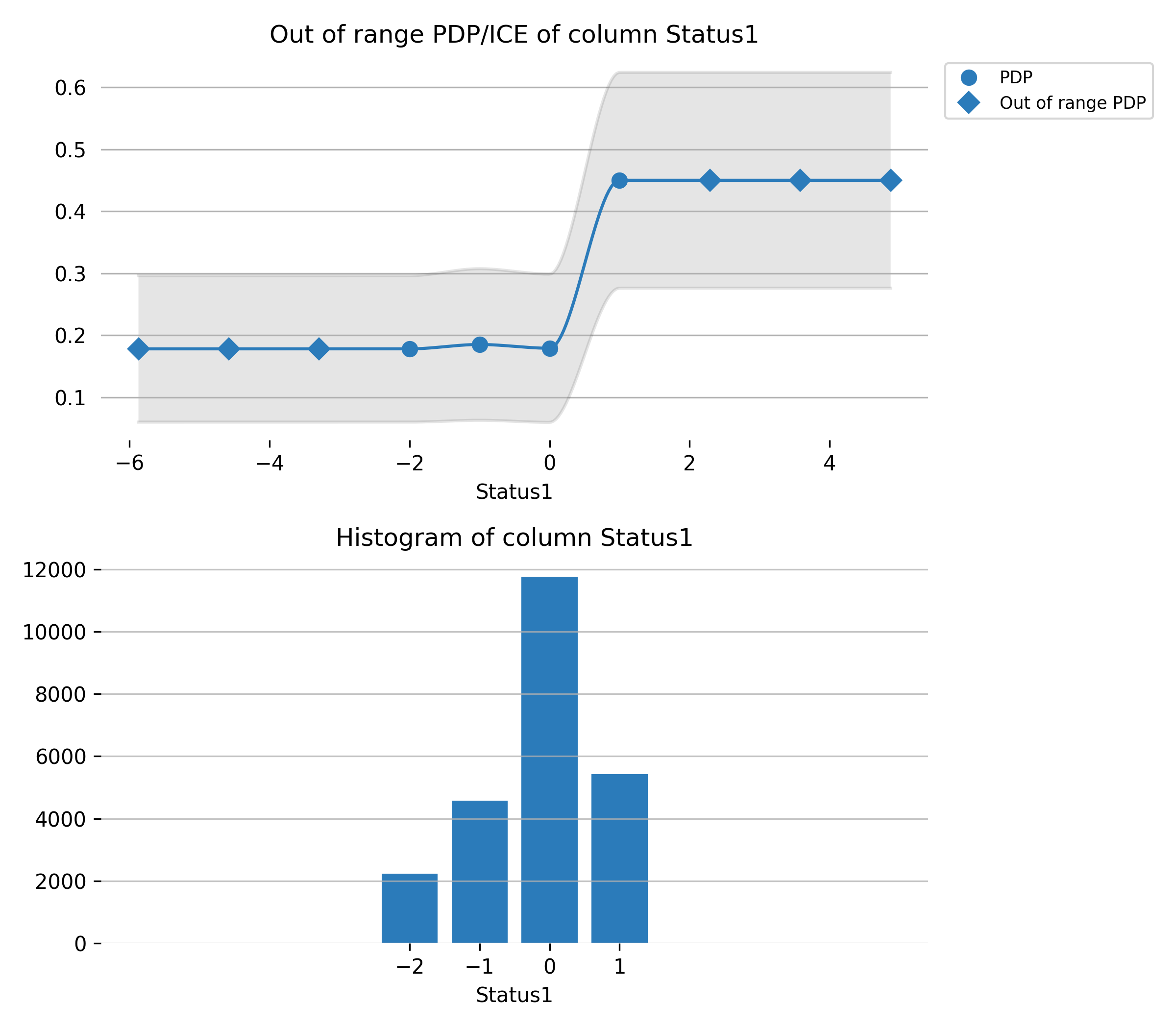
Partial dependence plots show the partial dependence as a function of specific values for a feature subset. The plots show how machine-learned response functions change based on the values of an input feature of interest, while taking nonlinearity into consideration and averaging out the effects of all other input features. Partial dependence plots enable increased transparency in a model and enable the ability to validate and debug a model by comparing a feature's average predictions across its domain to known standards and reasonable expectations.

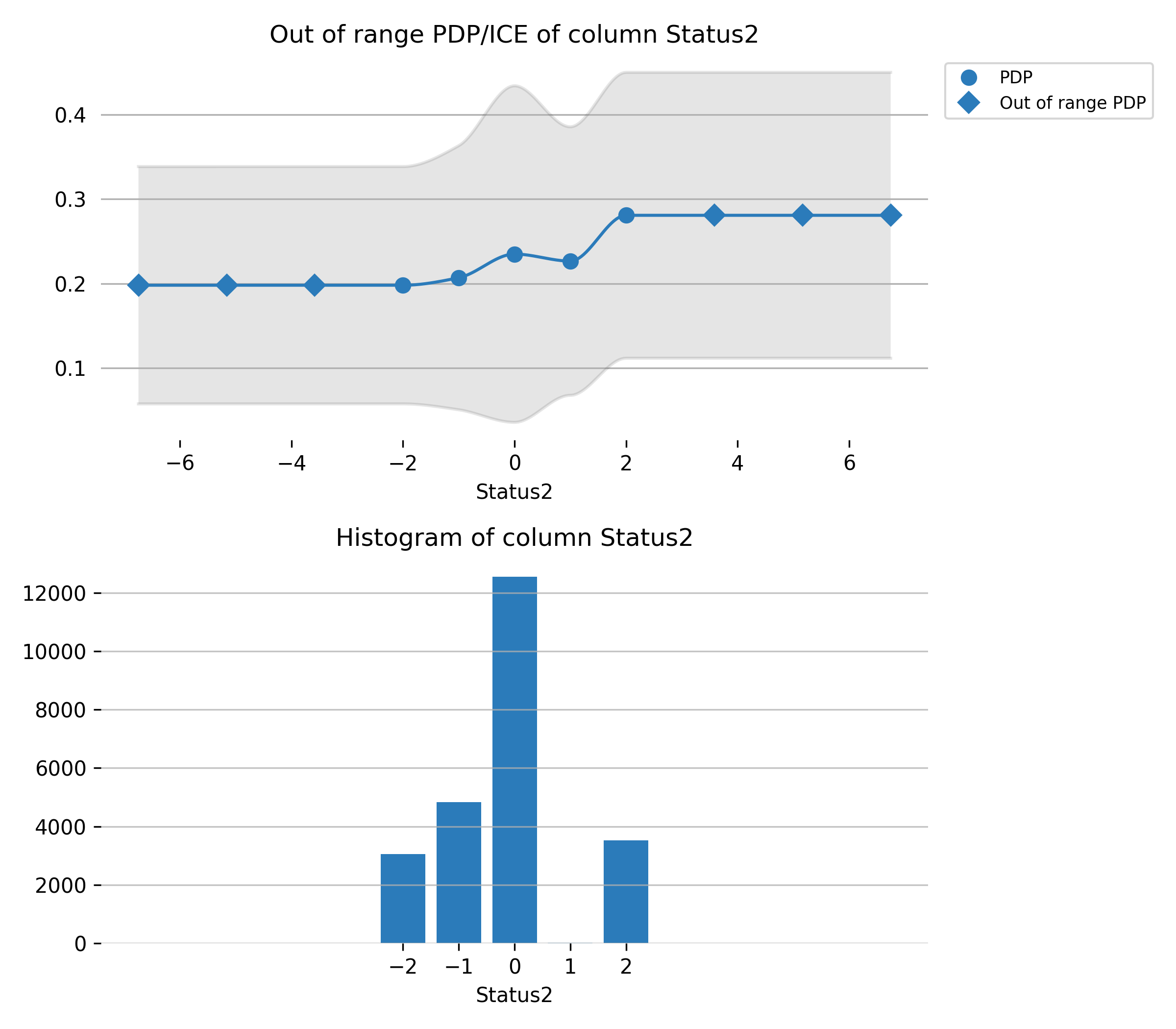
The partial dependence plots are shown for the top 4 original variables. The top 4 original variables are chosen based on their Component Based Variable Importance. Partial Dependence computation reached maximum allowed time 20 seconds.

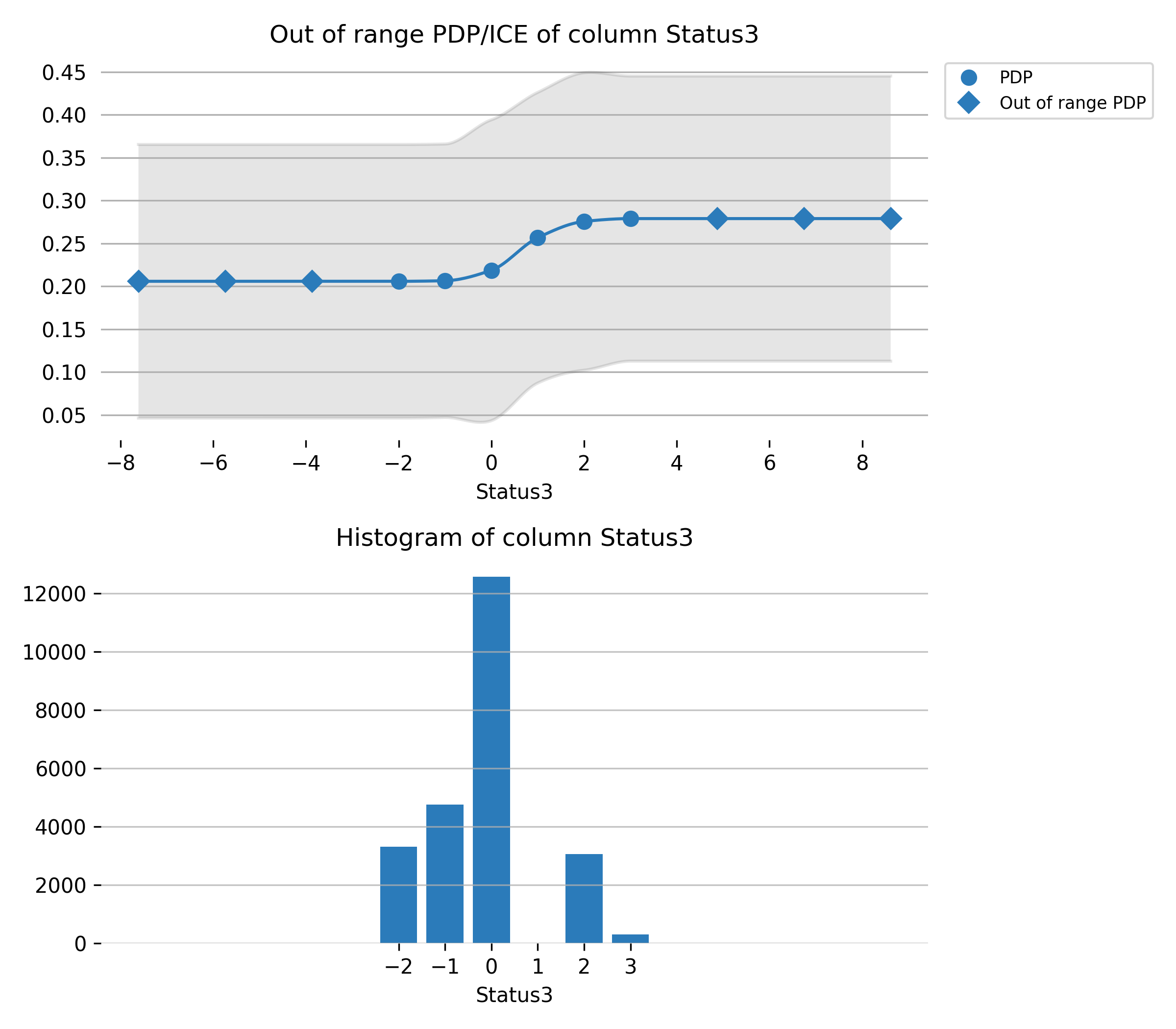
**Plot Details**

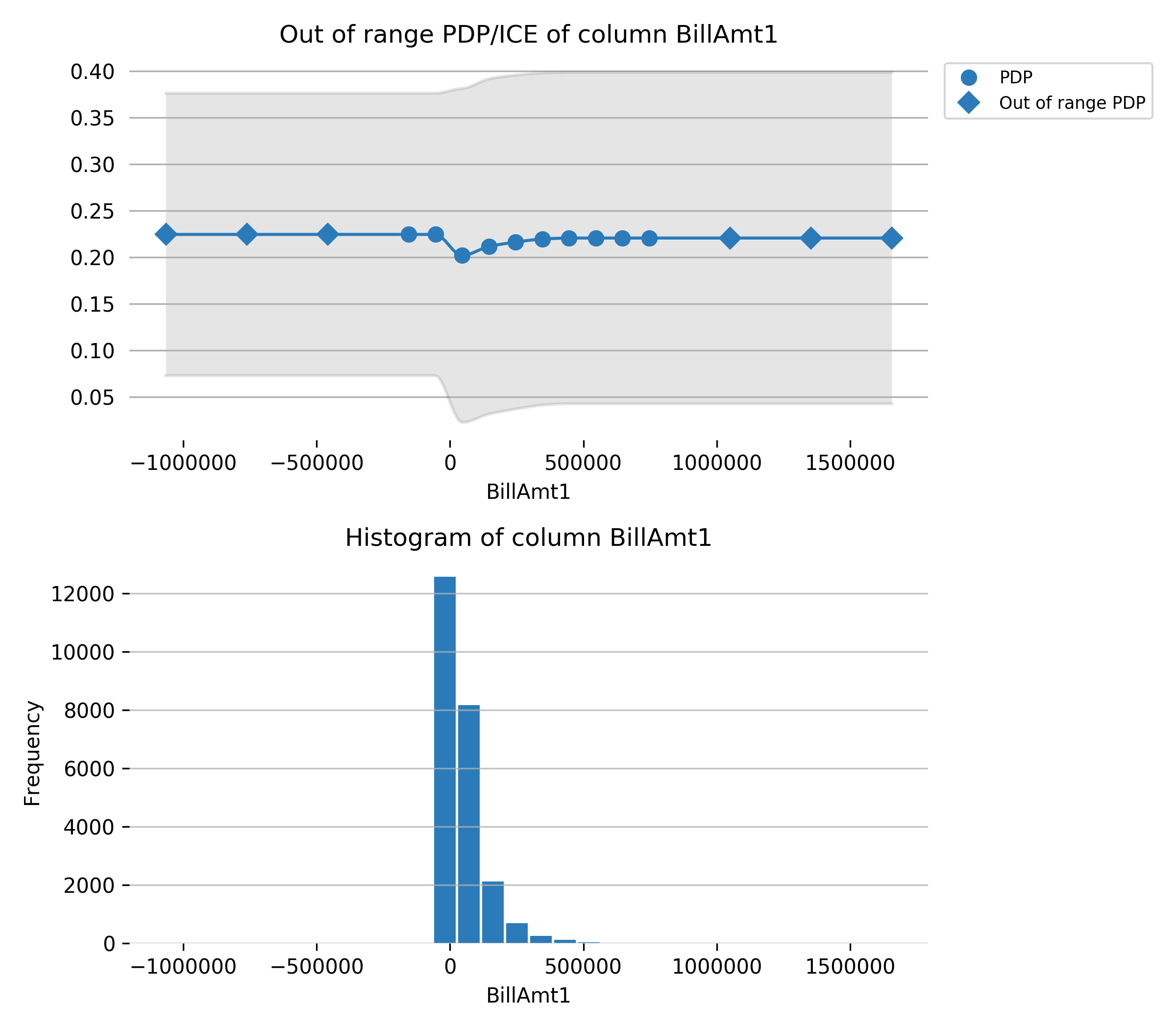
In the Driverless AI PDP, the y-axis represents the mean response, and a shaded region (for numeric features) or shaded bar (for categorical features) represents 1 standard deviation. Out-of-range PDP (diamond markers) represent values outside feature intervals seen in the data, unseen categorical values, or missing values.

For continuous features, numeric values up to 3 standard deviations lower than the minimum training value and higher than the maximum training value are feed into the model. For categorical features, an unseen categorical value is feed into the model denoted by UNSEEN (if the categorical value "UNSEEN" already exists in the training data, the out-of-range is done on a value called "UNSEEN\_[x]," where x is some integer).

Feature **Status1**

Feature **Status2**

Feature **Status3**

Feature **BillAmt1**

## Appendix

### Final Model Details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model Index** | **Type** | **Model Weight** | **Num Folds** | **Fitted features** | **Target Transformer** |
| 0 | XGBoostGBMModel | 1 | 4 | 22 | LabelEncoder |

**Model Index: Final Model - Single Model**

|  |  |
| --- | --- |
| **parameter** | **value** |
| accuracy | 3 |
| booster | gbtree |
| colsample\_bytree | 0.5 |
| debug\_verbose | 0 |
| disable\_gpus | False |
| dummy | False |
| early\_stopping\_rounds |  |
| early\_stopping\_threshold | 0 |
| encoder |  |
| ensemble\_level | 1 |
| eval\_metric | aucpr |
| gamma | 0.01 |
| gpu\_id | 0 |
| grow\_policy | depthwise |
| interpretability | 8 |
| labels | [0, 1] |
| learning\_rate | 0.05 |
| lossguide | False |
| max\_bin | 128 |
| max\_delta\_step | 0.0 |
| max\_depth | 5 |
| max\_leaves | 0 |
| min\_child\_weight | 1 |
| model\_class\_name | XGBoostGBMModel |
| model\_id | Final Model - Single Model |
| model\_origin | SEQUENCE |
| monotone\_constraints | (0,0,0,0,0,0,1,1,1,1,0,1,0,0,0,0,0,0,0,-1,0,0) |
| n\_estimators | 116 |
| n\_gpus | 1 |
| n\_jobs | 4 |
| num\_class | 1 |
| num\_classes | 2 |
| objective | binary:logistic |
| pred\_gap |  |
| pred\_periods |  |
| random\_state | 1234 |
| reg\_alpha | 0.0 |
| reg\_lambda | 10.0 |
| scale\_pos\_weight | 1.0 |
| score\_f\_name | AUC |
| seed | 1234 |
| silent | True |
| subsample | 0.6 |
| target |  |
| tgc |  |
| time\_column |  |
| time\_tolerance | 3 |
| train\_shape | [23999, 25] |
| tree\_method | gpu\_hist |
| tsp |  |
| valid\_shape |  |
| nfolds | 1 |

**Config Overrides**

The Config Overrides represent the fine-control parameters.

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| vis\_server\_port | 12346 |
| procsy\_port | 12347 |
| h2o\_port | 12348 |
| master\_redis\_password | t1Oi8w3coZczF50428ABc6XCCsBvlU3O |
| data\_directory | ./tmp |
| authentication\_method | local |
| local\_htpasswd\_file | /config/htpasswd |
| recipe\_load\_raise\_on\_any\_error | True |
| prob\_lagsinteraction | 0.2 |
| prob\_lagsaggregates | 0.2 |
| prob\_default\_lags | 0.2 |
| prob\_lag\_non\_targets | 0.1 |
| included\_scorers | ['ACCURACY', 'AUC', 'AUCPR', 'F05', 'F1', 'F2', 'GINI', 'LOGLOSS', 'MACROAUC', 'MAE', 'MAPE', 'MCC', 'MER', 'MSE', 'R2', 'RMSE', 'RMSLE', 'RMSPE', 'SMAPE'] |
| included\_models | ['CONSTANT', 'DECISIONTREE', 'FTRL', 'GLM', 'IMBALANCEDLIGHTGBM', 'IMBALANCEDXGBOOSTGBM', 'LIGHTGBM', 'RULEFIT', 'TENSORFLOW', 'XGBOOSTDART', 'XGBOOSTGBM'] |
| included\_transformers | ['CVCatNumEncodeTransformer', 'CVTargetEncodeTransformer', 'CatOriginalTransformer', 'CatTransformer', 'ClusterDistTransformer', 'ClusterTETransformer', 'DateOriginalTransformer', 'DateTimeOriginalTransformer', 'DatesTransformer', 'EwmaLagsTransformer', 'FrequentTransformer', 'InteractionsTransformer', 'IsHolidayTransformer', 'IsolationForestAnomalyNumCatAllColsTransformer', 'IsolationForestAnomalyNumCatTransformer', 'IsolationForestAnomalyNumericTransformer', 'LagsAggregatesTransformer', 'LagsInteractionTransformer', 'LagsTransformer', 'LexiLabelEncoderTransformer', 'NumCatTETransformer', 'NumToCatTETransformer', 'NumToCatWoEMonotonicTransformer', 'NumToCatWoETransformer', 'OneHotEncodingTransformer', 'OriginalTransformer', 'TextBiGRUTransformer', 'TextCNNTransformer', 'TextCharCNNTransformer', 'TextLinModelTransformer', 'TextTransformer', 'TruncSVDNumTransformer', 'WeightOfEvidenceTransformer'] |
| feature\_brain\_level | 0 |
| n\_estimators\_list\_no\_early\_stopping | 50,100,200,300 |
| override\_lag\_sizes |  |
| experiment\_id | 04c896ce-7ed1-11ea-96d6-0242ac110002 |
| experiment\_tmp\_dir | ./tmp/h2oai\_experiment\_04c896ce-7ed1-11ea-96d6-0242ac110002 |
| reproducible | True |