**Driverless AI Experiment: *bobutiko***

Generated by: h2oai

Generated on: Sat Oct 5 02:16:17 2019

[Experiment Overview 1](#_Toc532810621)

[Data Overview 3](#_Toc532810622)

[Methodology 4](#_Toc532810623)

[Validation Strategy 9](#_Toc532810624)

[Model Tuning 11](#_Toc532810625)

[Feature Evolution 12](#_Toc532810626)

[Feature Transformations 12](#_Toc532810627)

[Final Model 13](#_Toc532810628)

[Alternative Models 14](#_Toc532810629)

[Deployment 15](#_Toc532810630)

[Appendix 16](#_Toc532810631)

## Experiment Overview

Driverless AI built 2 XGBoostModels, 1 GLMModel to predict *redemption\_status* given 117 original features from the input dataset *tr\_2.csv*. This classification experiment completed in 2 hours and 42 minutes (2:42:49), using 28 of the 117 original features, and 277 of the 5,387 engineered features.

### Performance

|  |  |
| --- | --- |
| **Dataset** | **AUC** |
| Provided Validation Data | 0.929 |
| Test Data | Test Data did not have Target Column |

### Driverless Settings

|  |  |  |  |
| --- | --- | --- | --- |
| Dial Settings | Description | Setting Value | Range of Possible Values |
| Accuracy | Controls accuracy needs of the model | 8 | 1-10 |
| Time | Controls duration of the experiment | 4 | 1-10 |
| Interpretability | Controls complexity of the model | 3 | 1-10 |

### System Specifications

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **System Memory** | **CPUs** | **GPUs** |
| Linux | 15 | 4 | 1 |

### Versions

|  |
| --- |
| Driverless AI Version |
| 1.6.3 |

## Data Overview

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

|  |  |  |  |
| --- | --- | --- | --- |
| **data** | **file path** | **number of rows** | **number of columns** |
| training | ./tmp/bosibata/tr\_2.csv.1570215786.44508.bin | 22,983 | 128 |
| validation | ./tmp/luhurifo/val\_2.csv.1570215787.5344608.bin | 25,720 | 128 |
| testing | ./tmp/gunowifa/test\_2.csv.1570215791.2250588.bin | 50,226 | 127 |

### Training Data

The training data consists of only numeric columns

The summary of the columns is shown below:

#### Numeric Columns

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **name** | **data\_type** | **min** | **mean** | **max** | **std** | **unique** | **freq of mode** |
| customer\_id | int | 1.000 | 788.684 | 1,580.000 | 458.641 | 799 | 132 |
| campaign\_type | real | 0.000 | 0.455 | 1.000 | 0.498 | 2 | 10,460 |
| start\_date\_day | int | 2.000 | 13.047 | 28.000 | 5.422 | 10 | 8,025 |
| start\_date\_month | int | 1.000 | 8.148 | 12.000 | 3.566 | 7 | 6,467 |
| start\_date\_year | int | 2,012.000 | 2,012.193 | 2,013.000 | 0.395 | 2 | 4,447 |
| start\_date\_dayofweek | int | 0.000 | 2.619 | 6.000 | 2.811 | 4 | 12,166 |
| start\_date\_week | int | 2.000 | 33.372 | 51.000 | 16.065 | 9 | 6,467 |
| end\_date\_day | int | 1.000 | 15.536 | 30.000 | 9.476 | 9 | 6,467 |
| end\_date\_month | int | 1.000 | 5.419 | 11.000 | 4.233 | 6 | 7,378 |
| end\_date\_year | int | 2,012.000 | 2,012.532 | 2,013.000 | 0.499 | 2 | 12,233 |
| end\_date\_dayofweek | int | 4.000 | 4.032 | 5.000 | 0.176 | 2 | 732 |
| end\_date\_week | int | 1.000 | 21.754 | 48.000 | 19.534 | 10 | 6,467 |
| coupon\_size | int | 1.000 | 21.865 | 148.000 | 27.898 | 79 | 1,621 |
| brand\_nunique | int | 1.000 | 1.144 | 6.000 | 0.450 | 5 | 20,290 |
| brand\_mode | int | 5.000 | 1,193.935 | 5,395.000 | 1,384.757 | 207 | 3,177 |
| brand\_type\_nunique | int | 1.000 | 1.012 | 2.000 | 0.110 | 2 | 281 |
| brand\_type\_mode | real | 0.000 | 0.138 | 1.000 | 0.345 | 2 | 3,177 |
| category\_nunique | int | 1.000 | 1.040 | 2.000 | 0.195 | 2 | 909 |
| category\_mode | real | 2.000 | 7.508 | 15.000 | 2.461 | 9 | 14,977 |
| age\_range | real | 0.000 | 2.424 | 5.000 | 1.199 | 6 | 4,014 |
| marital\_status | real | 0.000 | 0.301 | 1.000 | 0.459 | 2 | 2,168 |
| rented | real | 0.000 | 0.053 | 1.000 | 0.224 | 2 | 660 |
| family\_size | real | 0.000 | 1.192 | 4.000 | 1.222 | 5 | 4,768 |
| no\_of\_children | real | 0.000 | 0.849 | 2.000 | 0.841 | 3 | 1,715 |
| income\_bracket | real | 1.000 | 4.843 | 12.000 | 2.328 | 12 | 3,388 |
| trans\_size | int | 52.000 | 791.163 | 3,584.000 | 517.235 | 613 | 178 |
| item\_id\_nunique | int | 26.000 | 480.581 | 2,069.000 | 278.706 | 521 | 190 |
| item\_id\_mode | int | 159.000 | 29,984.850 | 65,468.000 | 16,189.542 | 486 | 3,797 |
| date\_isweekend\_mean | real | 0.000 | 0.270 | 0.854 | 0.115 | 785 | 132 |
| date\_month\_mean | real | 1.779 | 6.406 | 9.323 | 0.876 | 795 | 132 |
| date\_month\_mode | int | 1.000 | 5.404 | 12.000 | 3.866 | 12 | 5,482 |
| date\_week\_mean | real | 6.161 | 25.933 | 38.613 | 3.805 | 798 | 132 |
| date\_week\_mode | int | 1.000 | 21.005 | 52.000 | 16.087 | 52 | 1,775 |
| date\_dayofweek\_mean | real | 0.636 | 3.047 | 5.194 | 0.484 | 796 | 132 |
| date\_dayofweek\_mode | int | 0.000 | 3.140 | 6.000 | 1.783 | 7 | 5,038 |
| quantity\_mean | real | 1.038 | 128.223 | 3,615.358 | 256.381 | 798 | 132 |
| quantity\_sum | int | 54.000 | 105,114.737 | 2,067,985.000 | 191,690.339 | 765 | 135 |
| quantity\_max | int | 2.000 | 9,441.881 | 85,055.000 | 9,597.239 | 510 | 1,554 |
| quantity\_median | int | 1.000 | 1.009 | 4.000 | 0.143 | 4 | 22,875 |
| quantity\_std | real | 0.194 | 847.125 | 7,338.098 | 997.806 | 799 | 132 |
| other\_discount\_mean | real | -200.515 | -18.560 | -4.897 | 9.982 | 799 | 132 |
| other\_discount\_sum | real | -81,194.240 | -14,239.775 | -551.700 | 10,193.811 | 799 | 132 |
| other\_discount\_min | real | -3,097.870 | -400.160 | -39.540 | 321.558 | 379 | 1,074 |
| other\_discount\_max | real | 0.000 | 0.000 | 0.000 | 0.000 | 1 | 22,983 |
| other\_discount\_median | real | -106.150 | -4.223 | 0.000 | 6.711 | 83 | 11,182 |
| other\_discount\_std | real | 10.963 | 35.120 | 289.576 | 19.396 | 799 | 132 |
| coupon\_discount\_mean | real | -15.002 | -0.607 | 0.000 | 1.189 | 611 | 4,682 |
| coupon\_discount\_sum | real | -7,793.170 | -494.099 | 0.000 | 981.146 | 420 | 4,682 |

#### Boolean Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **name** | **data\_type** | **min** | **mean** | **max** | **std** | **freq of max value** |
| start\_date\_isweekend | bool | False | 0.4644 | True | 0.4987 | 10,674 |
| end\_date\_isweekend | bool | False | 0.0318 | True | 0.1756 | 732 |
| quantity\_min | bool | True | 1.0 | True | 0.0 | 22,983 |

### 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 checked the train, validation, and test data for any shift in distribution and found the following significant differences:

* Significant difference detected between training and validation data distribution for feature \*\*end\_date\_month\*\* (AUC: 1). Dropping this feature.

None

* Significant difference detected between training and validation data distribution for feature \*\*start\_date\_week\*\* (AUC: 1). Dropping this feature.

None

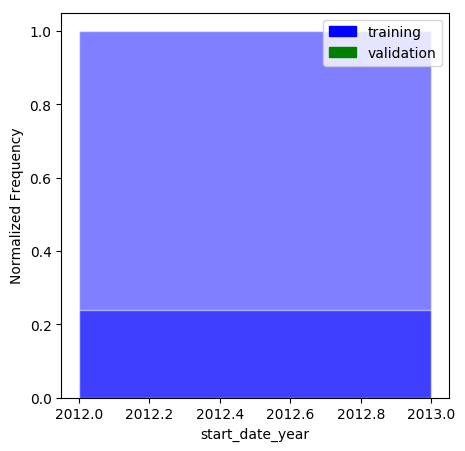
* Significant difference detected between training and validation data distribution for feature \*\*end\_date\_week\*\* (AUC: 1). Dropping this feature.

None

* Significant difference detected between training and validation data distribution for feature \*\*start\_date\_month\*\* (AUC: 0.96428). Dropping this feature.

None

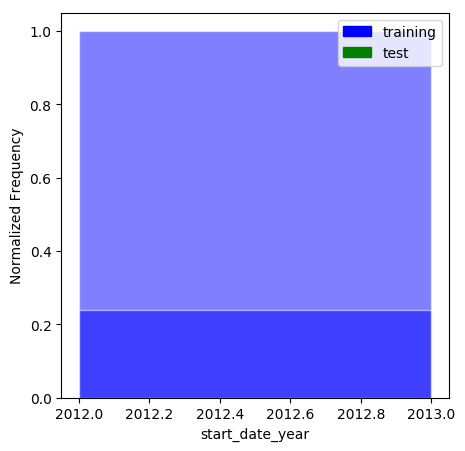
* Significant difference detected between training and validation data distribution for feature \*\*start\_date\_year\*\* (AUC: 0.9015)



* Significant difference detected between training and test data distribution for feature \*\*end\_date\_week\*\* (AUC: 0.99043). Dropping this feature.

None

* Significant difference detected between training and test data distribution for feature \*\*start\_date\_year\*\* (AUC: 0.9015). Dropping this feature.



* Significant difference detected between training and test data distribution for feature \*\*end\_date\_month\*\* (AUC: 0.88659). Dropping this feature.

None

* Significant difference detected between training and test data distribution for feature \*\*start\_date\_week\*\* (AUC: 0.83475)

None

* Significant difference detected between validation and test data distribution for feature \*\*start\_date\_week\*\* (AUC: 1). Dropping this feature.

None

* Significant difference detected between validation and test data distribution for feature \*\*end\_date\_month\*\* (AUC: 1). Dropping this feature.

None

* Significant difference detected between validation and test data distribution for feature \*\*start\_date\_month\*\* (AUC: 1)

None

## Methodology

This section describes the experiment methodology.

### Assumptions and Limitations

Driverless AI trains all models based on the training data provided (in this case: *tr\_2.csv.1570215786.44508.bin*). 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, validation, and test data. 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 performed shift detection and found significant differences described below:

|  |  |  |
| --- | --- | --- |
| **shift\_col** | **shift\_data\_first** | **shift\_data\_second** |
| end\_date\_month | training | validation |
| start\_date\_week | training | validation |
| end\_date\_week | training | validation |
| start\_date\_month | training | validation |
| start\_date\_year | training | validation |
| end\_date\_week | training | test |
| start\_date\_year | training | test |
| end\_date\_month | training | test |
| start\_date\_week | training | test |
| start\_date\_week | validation | test |
| end\_date\_month | validation | test |
| start\_date\_month | validation | test |

### 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:

1. **Ingest Data** 
   * detected column types
2. **Feature Preprocessing**
   * turned raw features into numeric
3. **Model and Feature Tuning** 
   * found the optimal parameters for linear, light gbm and xgboost models by training models with different parameters
   * the best parameters are those that generate the greatest **AUC** on the internal validation data
   * 50 models trained and scored to evaluate features and model parameters
4. **Feature Evolution** 
   * found the best representation of the data for the final model training by creating and evaluating **5,387** features over **90** iterations
   * 318 models trained and scored to further evaluate engineered features
5. **Final Model** 
   * the final model is the best model from the feature engineering iterations
   * no stacked ensemble is done because a validation dataset was provided by the user
6. **Create Scoring Pipeline** 
   * created and exported the Python scoring pipeline (no MOJO Scoring Pipeline automatically created)
   * Python Scoring Pipeline: h2oai\_experiment\_bobutiko/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 | 10.95 | 0 |
| Model and Feature Tuning | 861.24 | 50 |
| Feature Evolution | 7,727.91 | 318 |
| Final Pipeline Training | 1,148.01 | 3 |

### Experiment Settings

Below are the settings selected for the experiment by h2oai:

**Defined Parameters**



**Config Overrides**

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| recipe | auto |
| enable\_xgboost | auto |
| enable\_lightgbm | auto |
| enable\_rf | auto |
| enable\_glm | auto |
| enable\_tensorflow | off |
| enable\_rulefit | off |
| enable\_ftrl | off |
| parameter\_tuning\_num\_models | -1 |
| fixed\_ensemble\_level | -1 |
| check\_distribution\_shift | True |
| drop\_features\_distribution\_shift\_threshold\_auc | 0.6 |
| target\_transformer | auto |
| enable\_target\_encoding | True |
| time\_series\_recipe | True |
| override\_lag\_sizes |  |
| prob\_lag\_non\_targets | 0.1 |
| make\_python\_scoring\_pipeline | True |
| make\_mojo\_scoring\_pipeline | False |
| rulefit\_max\_num\_rules | -1 |
| feature\_brain\_level | 2 |
| quantile\_imbalanced\_sampling | False |
| holiday\_features | True |
| seed | 1234 |
| force\_64bit\_precision | False |
| min\_num\_rows | 100 |
| max\_orig\_cols\_selected | 10000 |
| nfeatures\_max | -1 |
| feature\_evolution\_data\_size | 100000000 |
| feature\_engineering\_effort | 7 |
| max\_feature\_interaction\_depth | 8 |
| max\_relative\_cardinality | 0.95 |
| string\_col\_as\_text\_threshold | 0.3 |
| tensorflow\_max\_epochs | 10 |
| enable\_tensorflow\_textcnn | False |
| enable\_tensorflow\_textbigru | False |
| enable\_tensorflow\_charcnn | False |
| tensorflow\_max\_epochs\_nlp | 2 |
| min\_dai\_iterations | 0 |
| max\_nestimators | 3000 |
| max\_nestimators\_feature\_evolution\_factor | 0.2 |
| max\_learning\_rate | 0.5 |
| max\_cores | -1 |
| num\_gpus\_per\_model | 1 |
| num\_gpus\_per\_experiment | -1 |
| gpu\_id\_start | 0 |
| compute\_correlation | False |
| high\_correlation\_value\_to\_report | 0.95 |
| dump\_modelparams\_every\_scored\_indiv | True |
| dump\_varimp\_every\_scored\_indiv | True |
| detailed\_traces | False |
| config\_overrides |  |

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 | 40 |
| number of models trained per iteration | 8 |
| early stopping rounds | 5 |
| monotonicity constraint | False |
| number of model tuning model combinations | 49 |
| 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 greatest 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 light gbm, xgboost and linear models.
* **number of base learners in ensemble**: the number of base models used to create the final ensemble.
* **time column**: the column that provides 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).

## Validation Strategy

Driverless AI used the validation data provided (val\_2.csv) to determine the performance of the model parameter tuning and feature engineering steps.

## Model Tuning

The table below shows the score and training time of the linear, light gbm and xgboost models evaluated by Driverless AI. The table shows the top 10 parameter tuning models evaluated, ordered based on a combination of greatest score and lowest training time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **job order** | **booster** | **nfeatures** | **scores** | **training times** |
| 48 | gbtree | 271 | 0.9206 | 10.7964 |
| 8 | gblinear | 32 | 0.9174 | 25.7306 |
| 43 | gbtree | 330 | 0.904 | 7.1622 |
| 46 | lightgbm | 71 | 0.9033 | 17.6859 |
| 19 | lightgbm | 154 | 0.9004 | 5.0258 |
| 25 | gbtree | 184 | 0.8985 | 6.2469 |
| 30 | lightgbm | 348 | 0.898 | 13.7498 |
| 33 | lightgbm | 399 | 0.8979 | 17.3708 |
| 36 | gbtree | 397 | 0.8966 | 7.4814 |
| 27 | gbtree | 168 | 0.8952 | 2.6661 |

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

### gblinear tuning

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **reg alpha** | **reg lambda** | **n lambda** | **min lambda fraction** | **nfeatures** | **scores** | **training times** |
| 0.025 | 0.05 | 30 | 1e-06 | 32 | 0.9174 | 25.7306 |

### lightgbm tuning

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **tree method** | **grow policy** | **max depth** | **max leaves** | **colsample bytree** | **subsample** | **nfeatures** | **scores** | **training times** |
| gpu\_hist | lossguide | 0.0 | 256.0 | 0.3 | 0.8 | 71 | 0.9033 | 17.6859 |
| gpu\_hist | lossguide | 0.0 | 512.0 | 0.5 | 0.9 | 154 | 0.9004 | 5.0258 |
| gpu\_hist | lossguide | 0.0 | 1024.0 | 0.2 | 0.5 | 348 | 0.898 | 13.7498 |
| gpu\_hist | lossguide | 0.0 | 32.0 | 0.3 | 1 | 399 | 0.8979 | 17.3708 |
| gpu\_hist | lossguide | 0.0 | 512.0 | 0.5 | 0.9 | 353 | 0.8929 | 16.7188 |
| gpu\_hist | lossguide | 0.0 | 128.0 | 0.45 | 0.7 | 263 | 0.9131 | 11.6726 |
| gpu\_hist | lossguide | 0.0 | 512.0 | 0.5 | 0.6 | 366 | 0.8893 | 18.6903 |
| gpu\_hist | lossguide | 0.0 | 32.0 | 0.45 | 0.5 | 372 | 0.8802 | 15.2278 |
| gpu\_hist | lossguide | 0.0 | 64.0 | 0.6 | 0.5 | 7 | 0.8775 | 2.5156 |
| gpu\_hist | lossguide | 0.0 | 1024.0 | 0.3 | 0.8 | 20 | 0.8734 | 24.663 |

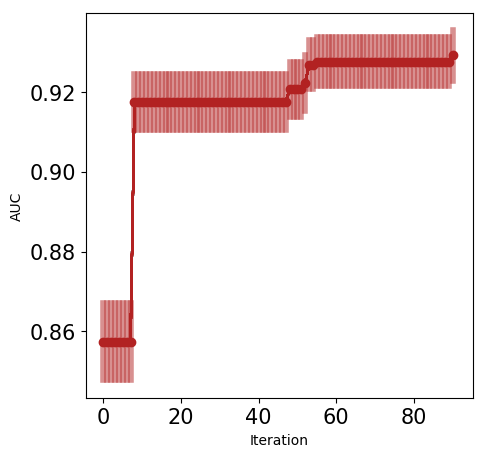
### gbtree tuning

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **tree method** | **grow policy** | **max depth** | **max leaves** | **colsample bytree** | **subsample** | **nfeatures** | **scores** | **training times** |
| gpu\_hist | depthwise | 8.0 | 0.0 | 0.7 | 0.5 | 271 | 0.9206 | 10.7964 |
| gpu\_hist | depthwise | 10.0 | 0.0 | 0.55 | 0.5 | 330 | 0.904 | 7.1622 |
| gpu\_hist | lossguide | 0.0 | 512.0 | 0.4 | 1 | 184 | 0.8985 | 6.2469 |
| gpu\_hist | lossguide | 0.0 | 256.0 | 0.3 | 0.5 | 397 | 0.8966 | 7.4814 |
| gpu\_hist | lossguide | 0.0 | 512.0 | 0.6 | 0.4 | 168 | 0.8952 | 2.6661 |
| gpu\_hist | depthwise | 8.0 | 0.0 | 0.7 | 0.5 | 69 | 0.8945 | 2.3641 |
| gpu\_hist | depthwise | 6.0 | 0.0 | 0.5 | 0.9 | 175 | 0.8912 | 3.4718 |
| gpu\_hist | depthwise | 7.0 | 0.0 | 0.35 | 0.4 | 567 | 0.8907 | 12.2364 |
| gpu\_hist | lossguide | 0.0 | 8.0 | 0.8 | 0.8 | 181 | 0.8872 | 8.3902 |
| gpu\_hist | depthwise | 8.0 | 0.0 | 0.45 | 0.9 | 444 | 0.8857 | 8.9159 |

## 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 light gbm, xgboost and linear models (318) 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 belows 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:

* **CVTargetEncodeDT**: 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**: calculates the frequency for each value in categorical column(s) and uses this as a new feature.
* **WeightOfEvidenceTransformer**: 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.
* **OHETransformer**: converts a categorical column to a series of boolean features by performing one-hot encoding. The boolean features are used as new features.
* **BulkInteractionsTransformer**: add, divide, multiply, and subtract two numeric columns in the data to create a new feature.
* **ClusterDistTransformer**: clusters selected numeric columns and uses the distance to a specific cluster as a new feature.
* **ClusterTETransformer**: 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.
* **NumToCatTETransformer**: converts a 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**: 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**: trains a Truncated SVD model on selected numeric columns and uses the components of the truncated SVD matrix as new features.
* **CVCatNumEncodeDT**: 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**: 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** |
| 516\_NumToCatTE: coupon\_discount\_sum: tfidf\_brand\_value\_top\_1: tfidf\_brand\_value\_top\_10: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_1', 'tfidf\_brand\_value\_top\_10', 'tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(20, 3, 100)] (numeric columns are bucketed into 100 equally populated bins) [internal parameters:(20, 3, 100)] | Numeric to Categorical Target Encoding | 1.0 |
| 181\_NumToCatTE: coupon\_discount\_sum: tfidf\_brand\_value\_top\_2: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_2', 'tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(20, 10, 100)] (numeric columns are bucketed into 100 equally populated bins) [internal parameters:(20, 10, 100)] | Numeric to Categorical Target Encoding | 0.8147 |
| 574\_ClusterTE: ClusterID63: coupon\_discount\_sum: cust\_coupon\_selling\_price\_sum: cust\_coupon\_trans\_size: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['ClusterID63:coupon\_discount\_sum:cust\_coupon\_selling\_price\_sum:cust\_coupon\_trans\_size:tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(20, 10, 100)] (Clustered into 63 clusters) [internal parameters:(63, False, 20, 10, 100)] | Cluster Target Encoding | 0.7223 |
| 237\_InteractionSub: coupon\_discount\_mean: tfidf\_dense\_category\_6\_0 | [coupon\_discount\_mean] - [tfidf\_dense\_category\_6\_0] | Interaction | 0.6844 |
| 195\_NumToCatTE: cust\_coupon\_coupon\_discount\_sum: tfidf\_brand\_value\_top\_4: tfidf\_item\_id\_value\_top\_2.0 | Out-of-fold mean of the response grouped by: ['cust\_coupon\_coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_4', 'tfidf\_item\_id\_value\_top\_2'] using 5 folds [internal parameters:(10, 5, 10)] (numeric columns are bucketed into 25 equally populated bins) [internal parameters:(10, 5, 10)] | Numeric to Categorical Target Encoding | 0.516 |
| 586\_ClusterTE: ClusterID100: coupon\_discount\_mean: tfidf\_brand\_value\_top\_1: tfidf\_brand\_value\_top\_2: tfidf\_brand\_value\_top\_5: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['ClusterID100:coupon\_discount\_mean:tfidf\_brand\_value\_top\_1:tfidf\_brand\_value\_top\_2:tfidf\_brand\_value\_top\_5:tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(100, 10, 100)] (Clustered into 100 clusters) [internal parameters:(100, False, 100, 10, 100)] | Cluster Target Encoding | 0.5022 |
| 254\_ClusterTE: ClusterID45: coupon\_discount\_sum: tfidf\_brand\_value\_top\_5: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['ClusterID45:coupon\_discount\_sum:tfidf\_brand\_value\_top\_5:tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(20, 1, 10)] (Clustered into 45 clusters) [internal parameters:(45, False, 20, 1, 10)] | Cluster Target Encoding | 0.4966 |
| 650\_InteractionSub: coupon\_discount\_mean: tfidf\_brand\_value\_top\_4 | [coupon\_discount\_mean] - [tfidf\_brand\_value\_top\_4] | Interaction | 0.4841 |
| 808\_NumToCatWoE: coupon\_discount\_sum: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_selling\_price\_sum: tfidf\_brand\_value\_top\_1: tfidf\_brand\_value\_top\_10: tfidf\_brand\_value\_top\_3: tfidf\_item\_id\_value\_top\_10.0 | Weight of Evidence for columns ['coupon\_discount\_sum', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_selling\_price\_sum', 'tfidf\_brand\_value\_top\_1', 'tfidf\_brand\_value\_top\_10', 'tfidf\_brand\_value\_top\_3', 'tfidf\_item\_id\_value\_top\_10'] column #0 (numeric columns are bucketed into 10 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.397 |
| 638\_NumCatTE: campaign\_type: coupon\_discount\_mean: coupon\_discount\_sum: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: tfidf\_dense\_category\_15\_0: tfidf\_item\_id\_value\_top\_2.0 | Out-of-fold mean of the response grouped by: ['campaign\_type', 'coupon\_discount\_mean', 'coupon\_discount\_sum', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'tfidf\_dense\_category\_15\_0', 'tfidf\_item\_id\_value\_top\_2'] using 5 folds [internal parameters:(10, 10, 20)] (numeric columns are bucketed into 250 equally populated bins) [internal parameters:(10, 10, 20)] | Cross Validation Target Encoding | 0.3905 |
| 68\_cust\_coupon\_other\_discount\_sum | cust\_coupon\_other\_discount\_sum (original) | None | 0.3779 |
| 912\_CVCatNumEnc: campaign\_type: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_other\_discount\_std.mean | Out-of-fold mean of 'cust\_coupon\_other\_discount\_std' grouped by: ['campaign\_type', 'cust\_coupon\_coupon\_discount\_min'] using 5 folds [internal parameters:('mean', 100)] | Cross Validation Categorical to Numeric Encoding | 0.3623 |
| 584\_InteractionDiv: coupon\_discount\_sum: tfidf\_item\_id\_value\_top\_2 | [coupon\_discount\_sum] / [tfidf\_item\_id\_value\_top\_2] | Interaction | 0.3592 |
| 764\_ClusterDist3: age\_range: selling\_price\_std.1 | Distances to cluster center after segmenting columns ['age\_range', 'selling\_price\_std'] into 3 clusters. Distance to cluster #1 [internal parameters:False] | Cluster Distance | 0.3582 |
| 666\_CVTE: campaign\_type: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: cust\_coupon\_quantity\_median.0 | Out-of-fold mean of the response grouped by: ['campaign\_type', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'cust\_coupon\_quantity\_median'] using 5 folds [internal parameters:(20, 10, 100)] | Cross Validation Target Encoding | 0.3056 |
| 268\_NumCatTE: category\_mode: date\_month\_mode: rented: tfidf\_brand\_value\_top\_10: tfidf\_item\_id\_value\_top\_2.0 | Out-of-fold mean of the response grouped by: ['category\_mode', 'date\_month\_mode', 'rented', 'tfidf\_brand\_value\_top\_10', 'tfidf\_item\_id\_value\_top\_2'] using 5 folds [internal parameters:(10, 3, None)] (numeric columns are bucketed into 25 equally populated bins) [internal parameters:(10, 3, None)] | Cross Validation Target Encoding | 0.2867 |
| 527\_NumToCatTE: coupon\_discount\_sum: tfidf\_brand\_value\_top\_2: tfidf\_brand\_value\_top\_5: tfidf\_dense\_category\_15\_0: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_2', 'tfidf\_brand\_value\_top\_5', 'tfidf\_dense\_category\_15\_0', 'tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(10, 10, 10)] (numeric columns are bucketed into 25 equally populated bins) [internal parameters:(10, 10, 10)] | Numeric to Categorical Target Encoding | 0.2699 |
| 845\_ClusterTE: ClusterID81: coupon\_discount\_mean: tfidf\_dense\_category\_15\_0.0 | Out-of-fold mean of the response grouped by: ['ClusterID81:coupon\_discount\_mean:tfidf\_dense\_category\_15\_0'] using 5 folds [internal parameters:(20, 1, 100)] (Clustered into 81 clusters) [internal parameters:(81, True, 20, 1, 100)] | Cluster Target Encoding | 0.2621 |
| 590\_NumCatTE: cust\_coupon\_date\_month\_mode: cust\_coupon\_quantity\_max: cust\_coupon\_selling\_price\_mean: income\_bracket: start\_date\_day: tfidf\_item\_id\_value\_top\_1.0 | Out-of-fold mean of the response grouped by: ['cust\_coupon\_date\_month\_mode', 'cust\_coupon\_quantity\_max', 'cust\_coupon\_selling\_price\_mean', 'income\_bracket', 'start\_date\_day', 'tfidf\_item\_id\_value\_top\_1'] using 5 folds [internal parameters:(20, 5, 20)] (numeric columns are bucketed into 10 equally populated bins) [internal parameters:(20, 5, 20)] | Cross Validation Target Encoding | 0.255 |
| 841\_ClusterTE: ClusterID38: cust\_coupon\_other\_discount\_min: tfidf\_brand\_value\_top\_4.0 | Out-of-fold mean of the response grouped by: ['ClusterID38:cust\_coupon\_other\_discount\_min:tfidf\_brand\_value\_top\_4'] using 5 folds [internal parameters:(10, 5, 20)] (Clustered into 38 clusters) [internal parameters:(38, True, 10, 5, 20)] | Cluster Target Encoding | 0.2505 |
| 335\_NumToCatWoE: coupon\_discount\_mean: tfidf\_brand\_value\_top\_3.0 | Weight of Evidence for columns ['coupon\_discount\_mean', 'tfidf\_brand\_value\_top\_3'] column #0 (numeric columns are bucketed into 100 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.2492 |
| 708\_ClusterDist9: cust\_coupon\_selling\_price\_sum: end\_date\_year: selling\_price\_mean.0 | Distances to cluster center after segmenting columns ['cust\_coupon\_selling\_price\_sum', 'end\_date\_year', 'selling\_price\_mean'] into 9 clusters. Distance to cluster #0 [internal parameters:False] | Cluster Distance | 0.2393 |
| 783\_TruncSVD: coupon\_discount\_mean: date\_week\_mode: selling\_price\_mean: tfidf\_brand\_value\_top\_10: tfidf\_dense\_category\_7\_0.1 | Component #2 of truncated SVD of ['coupon\_discount\_mean', 'date\_week\_mode', 'selling\_price\_mean', 'tfidf\_brand\_value\_top\_10', 'tfidf\_dense\_category\_7\_0'] into 2 components | Truncated SVD | 0.2378 |
| 155\_ClusterTE: ClusterID98: coupon\_discount\_sum: tfidf\_brand\_value\_top\_1: tfidf\_brand\_value\_top\_2.0 | Out-of-fold mean of the response grouped by: ['ClusterID98:coupon\_discount\_sum:tfidf\_brand\_value\_top\_1:tfidf\_brand\_value\_top\_2'] using 5 folds [internal parameters:(20, 1, 10)] (Clustered into 98 clusters) [internal parameters:(98, False, 20, 1, 10)] | Cluster Target Encoding | 0.2257 |
| 912\_CVCatNumEnc: campaign\_type: cust\_coupon\_coupon\_discount\_min: tfidf\_brand\_value\_top\_1.mean | Out-of-fold mean of 'tfidf\_brand\_value\_top\_1' grouped by: ['campaign\_type', 'cust\_coupon\_coupon\_discount\_min'] using 5 folds [internal parameters:('mean', 100)] | Cross Validation Categorical to Numeric Encoding | 0.2201 |
| 799\_NumToCatWoE: coupon\_discount\_mean: tfidf\_brand\_value\_top\_6.0 | Weight of Evidence for columns ['coupon\_discount\_mean', 'tfidf\_brand\_value\_top\_6'] column #0 (numeric columns are bucketed into 250 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.2184 |
| 723\_NumCatTE: campaign\_type: coupon\_discount\_sum: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: cust\_coupon\_quantity\_median: cust\_coupon\_selling\_price\_sum.0 | Out-of-fold mean of the response grouped by: ['campaign\_type', 'coupon\_discount\_sum', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'cust\_coupon\_quantity\_median', 'cust\_coupon\_selling\_price\_sum'] using 5 folds [internal parameters:(10, 5, None)] (numeric columns are bucketed into 250 equally populated bins) [internal parameters:(10, 5, None)] | Cross Validation Target Encoding | 0.2132 |
| 192\_NumToCatWoE: coupon\_discount\_sum.0 | Weight of Evidence for columns ['coupon\_discount\_sum'] column #0 (numeric columns are bucketed into 250 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.2074 |
| 322\_ClusterTE: ClusterID91: cust\_coupon\_other\_discount\_min.0 | Out-of-fold mean of the response grouped by: ['ClusterID91:cust\_coupon\_other\_discount\_min'] using 5 folds [internal parameters:(20, 10, 10)] (Clustered into 91 clusters) [internal parameters:(91, True, 20, 10, 10)] | Cluster Target Encoding | 0.2063 |
| 218\_CVTE: campaign\_type: cust\_coupon\_quantity\_median: cust\_coupon\_quantity\_min: quantity\_median.0 | Out-of-fold mean of the response grouped by: ['campaign\_type', 'cust\_coupon\_quantity\_median', 'cust\_coupon\_quantity\_min', 'quantity\_median'] using 5 folds [internal parameters:(10, 1, 10)] | Cross Validation Target Encoding | 0.2057 |
| 153\_NumToCatTE: coupon\_discount\_mean: tfidf\_brand\_value\_top\_10.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_mean', 'tfidf\_brand\_value\_top\_10'] using 5 folds [internal parameters:(100, 10, 20)] (numeric columns are bucketed into 100 equally populated bins) [internal parameters:(100, 10, 20)] | Numeric to Categorical Target Encoding | 0.2008 |
| 354\_CVCatNumEnc: campaign\_type: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: cust\_coupon\_quantity\_median: tfidf\_dense\_category\_6\_0.sd | Out-of-fold sd of 'tfidf\_dense\_category\_6\_0' grouped by: ['campaign\_type', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'cust\_coupon\_quantity\_median'] using 5 folds [internal parameters:('sd', 100)] | Cross Validation Categorical to Numeric Encoding | 0.1941 |
| 414\_CVCatNumEnc: campaign\_type: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: cust\_coupon\_quantity\_median: coupon\_discount\_sum.mean | Out-of-fold mean of 'coupon\_discount\_sum' grouped by: ['campaign\_type', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'cust\_coupon\_quantity\_median'] using 5 folds [internal parameters:('mean', 3)] | Cross Validation Categorical to Numeric Encoding | 0.1851 |
| 708\_ClusterDist9: cust\_coupon\_selling\_price\_sum: end\_date\_year: selling\_price\_mean.7 | Distances to cluster center after segmenting columns ['cust\_coupon\_selling\_price\_sum', 'end\_date\_year', 'selling\_price\_mean'] into 9 clusters. Distance to cluster #7 [internal parameters:False] | Cluster Distance | 0.1829 |
| 255\_InteractionMul: coupon\_discount\_sum: tfidf\_dense\_category\_6\_0 | [coupon\_discount\_sum] \* [tfidf\_dense\_category\_6\_0] | Interaction | 0.1781 |
| 708\_ClusterDist9: cust\_coupon\_selling\_price\_sum: end\_date\_year: selling\_price\_mean.6 | Distances to cluster center after segmenting columns ['cust\_coupon\_selling\_price\_sum', 'end\_date\_year', 'selling\_price\_mean'] into 9 clusters. Distance to cluster #6 [internal parameters:False] | Cluster Distance | 0.1773 |
| 385\_ClusterDist8: coupon\_discount\_mean: cust\_coupon\_selling\_price\_sum: tfidf\_brand\_value\_top\_1: tfidf\_brand\_value\_top\_8: tfidf\_dense\_category\_6\_0.3 | Distances to cluster center after segmenting columns ['coupon\_discount\_mean', 'cust\_coupon\_selling\_price\_sum', 'tfidf\_brand\_value\_top\_1', 'tfidf\_brand\_value\_top\_8', 'tfidf\_dense\_category\_6\_0'] into 8 clusters. Distance to cluster #3 [internal parameters:False] | Cluster Distance | 0.1672 |
| 896\_NumToCatWoE: coupon\_discount\_mean: coupon\_discount\_sum: tfidf\_dense\_category\_6\_0.0 | Weight of Evidence for columns ['coupon\_discount\_mean', 'coupon\_discount\_sum', 'tfidf\_dense\_category\_6\_0'] column #0 (numeric columns are bucketed into 250 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.162 |
| 573\_NumToCatWoE: coupon\_discount\_sum: tfidf\_brand\_value\_top\_5.0 | Weight of Evidence for columns ['coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_5'] column #0 (numeric columns are bucketed into 250 equally populated bins) | Numeric to Categorical Weight of Evidence | 0.158 |
| 872\_NumCatTE: campaign\_type: coupon\_discount\_mean: cust\_coupon\_coupon\_discount\_min: cust\_coupon\_quantity\_max: tfidf\_brand\_value\_top\_2: tfidf\_brand\_value\_top\_4.0 | Out-of-fold mean of the response grouped by: ['campaign\_type', 'coupon\_discount\_mean', 'cust\_coupon\_coupon\_discount\_min', 'cust\_coupon\_quantity\_max', 'tfidf\_brand\_value\_top\_2', 'tfidf\_brand\_value\_top\_4'] using 5 folds [internal parameters:(100, 5, 100)] (numeric columns are bucketed into 10 equally populated bins) [internal parameters:(100, 5, 100)] | Cross Validation Target Encoding | 0.1575 |
| 503\_NumToCatTE: coupon\_discount\_sum: tfidf\_brand\_value\_top\_2: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_sum', 'tfidf\_brand\_value\_top\_2', 'tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(10, 5, 20)] (numeric columns are bucketed into 25 equally populated bins) [internal parameters:(10, 5, 20)] | Numeric to Categorical Target Encoding | 0.154 |
| 73\_cust\_coupon\_quantity\_std | cust\_coupon\_quantity\_std (original) | None | 0.1508 |
| 520\_NumToCatTE: coupon\_discount\_sum: cust\_coupon\_other\_discount\_min: tfidf\_brand\_value\_top\_2: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_sum', 'cust\_coupon\_other\_discount\_min', 'tfidf\_brand\_value\_top\_2', 'tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(20, 3, 20)] (numeric columns are bucketed into 25 equally populated bins) [internal parameters:(20, 3, 20)] | Numeric to Categorical Target Encoding | 0.1482 |
| 17\_Freq: date\_week\_mode | Encoding of categorical levels of feature(s) ['date\_week\_mode'] to value between 0 and 1 based on their relative frequency | Frequency Encoding | 0.1388 |
| 164\_NumToCatTE: coupon\_discount\_mean: cust\_coupon\_trans\_size.0 | Out-of-fold mean of the response grouped by: ['coupon\_discount\_mean', 'cust\_coupon\_trans\_size'] using 5 folds [internal parameters:(20, 3, 100)] (numeric columns are bucketed into 100 equally populated bins) [internal parameters:(20, 3, 100)] | Numeric to Categorical Target Encoding | 0.1358 |
| 287\_CVTE: brand\_type\_nunique: campaign\_type: end\_date\_year: marital\_status.0 | Out-of-fold mean of the response grouped by: ['brand\_type\_nunique', 'campaign\_type', 'end\_date\_year', 'marital\_status'] using 5 folds [internal parameters:(100, 10, 100)] | Cross Validation Target Encoding | 0.1328 |
| 763\_ClusterTE: ClusterID57: coupon\_discount\_mean: tfidf\_brand\_value\_top\_1: tfidf\_dense\_category\_6\_0.0 | Out-of-fold mean of the response grouped by: ['ClusterID57:coupon\_discount\_mean:tfidf\_brand\_value\_top\_1:tfidf\_dense\_category\_6\_0'] using 5 folds [internal parameters:(100, 5, 100)] (Clustered into 57 clusters) [internal parameters:(57, True, 100, 5, 100)] | Cluster Target Encoding | 0.1287 |
| 197\_ClusterTE: ClusterID41: quantity\_std: selling\_price\_median: tfidf\_item\_id\_value\_top\_9.0 | Out-of-fold mean of the response grouped by: ['ClusterID41:quantity\_std:selling\_price\_median:tfidf\_item\_id\_value\_top\_9'] using 5 folds [internal parameters:(100, 1, None)] (Clustered into 41 clusters) [internal parameters:(41, False, 100, 1, None)] | Cluster Target Encoding | 0.1287 |
| 814\_ClusterDist5: coupon\_discount\_min: cust\_coupon\_date\_isweekend\_mean: income\_bracket: tfidf\_dense\_category\_9\_0.1 | Distances to cluster center after segmenting columns ['coupon\_discount\_min', 'cust\_coupon\_date\_isweekend\_mean', 'income\_bracket', 'tfidf\_dense\_category\_9\_0'] into 5 clusters. Distance to cluster #1 [internal parameters:False] | Cluster Distance | 0.1284 |
| 708\_ClusterDist9: cust\_coupon\_selling\_price\_sum: end\_date\_year: selling\_price\_mean.3 | Distances to cluster center after segmenting columns ['cust\_coupon\_selling\_price\_sum', 'end\_date\_year', 'selling\_price\_mean'] into 9 clusters. Distance to cluster #3 [internal parameters:False] | Cluster Distance | 0.1264 |

## Final Model

**Pipeline**

Final StackedEnsemble pipeline with ensemble\_level=3 transforming 76 original features -> 305 features in each of 3 models each fit on external validation set then linearly blended

**Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Index** | **Type** | **Model Weight** | **Fitted features** | **Target Transformer** |
| 0 | XGBoostModel | 0.1667 | 160 | str |
| 1 | GLMModel | 0.3333 | 42 | str |
| 2 | XGBoostModel | 0.5 | 191 | str |

* Model Index: 0 has a weight of 0.1666666667 in the final ensemble

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Type** | **index** | **max depth** | **subsample** | **learning rate** | **grow policy** | **colsample bytree** | **tree method** | **max leaves** | **Split Type** |
| XGBoostModel | 0 | 8 | 0.7 | 0.03 | depthwise | 0.7 | gpu\_hist | 0 | External |

* Model Index: 1 has a weight of 0.3333333333 in the final ensemble

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Type** | **index** | **max depth** | **subsample** | **learning rate** | **grow policy** | **colsample bytree** | **tree method** | **max leaves** | **Split Type** |
| GLMModel | 1 |  |  | 0.5 |  |  |  |  | External |

* Model Index: 2 has a weight of 0.5 in the final ensemble

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Type** | **index** | **max depth** | **subsample** | **learning rate** | **grow policy** | **colsample bytree** | **tree method** | **max leaves** | **Split Type** |
| XGBoostModel | 2 | 8 | 0.5 | 0.03 | depthwise | 0.7 | gpu\_hist | 0 | External |

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

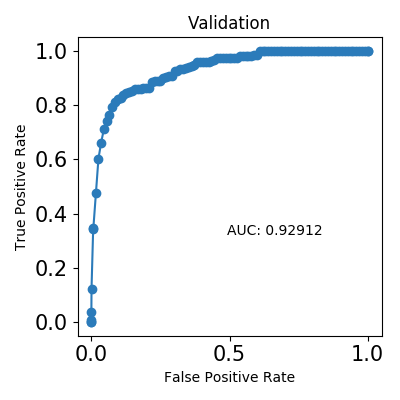
**Performance of Final Model**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 | 0.99187 | 0.00036835 |
| AUC | \* | higher | 0.92912 | 0.0062948 |
| AUCPR |  | higher | 0.24712 | 0.021744 |
| F05 |  | higher | 0.35104 | 0.026661 |
| F1 |  | higher | 0.32397 | 0.021743 |
| F2 |  | higher | 0.40275 | 0.016583 |
| GINI |  | higher | 0.85824 | 0.01259 |
| LOGLOSS |  | lower | 0.034714 | 0.0012163 |
| MCC |  | higher | 0.31832 | 0.018226 |

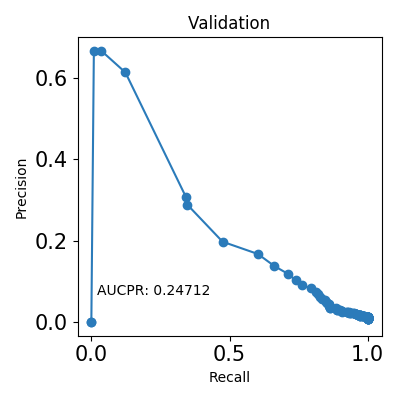
**Validation Confusion Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Predicted: 0** | **Predicted: 1** | **error** |
| Actual: 0 | 25,332 | 169 | 1% |
| Actual: 1 | 144 | 75 | 66% |

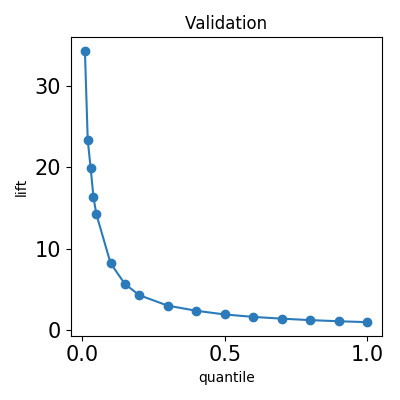
*Receiving Operator Curve*



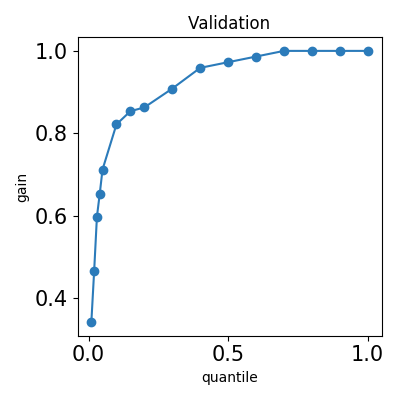
*Precision Recall Curve*



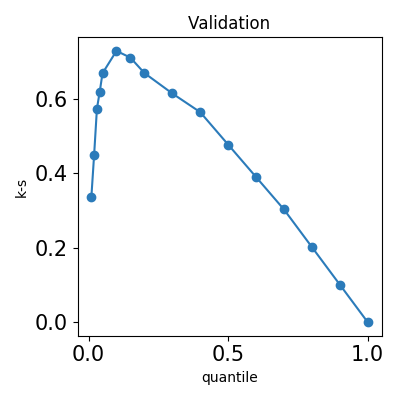
*Cumulative Lift*



*Cumulative Gains*



*Kolmogorov–Smirnov*



## Alternative Models

During the experiment, Driverless AI trained 368 alternative models. The following algorithms were evaluated during the Driverless AI experiment:

|  |  |  |  |
| --- | --- | --- | --- |
| **algorithm** | **package** | **version** | **documentation** |
| gblinear | xgboost | 0.81 | 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. |
| gbtree | xgboost | 0.81 | XGBoost: eXtreme Gradient Boosting library. Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTORS.md |

Driverless AI is able to evaluate the algorithms: XGBoost GBM, 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 |
| lightgbm | not selected due to low performance during feature evolution stage |
| gbtree | selected for final model |
| gblinear | selected for final model |

## Deployment

For this experiment, the Python Scoring Pipeline is available for productionizing the final model pipeline for a given row of data or table of data. The MOJO Scoring Pipeline can be built by clicking the **BUILD MOJO SCORING PIPELINE** button if available.

### 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\_bobutiko/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, and 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).

## Appendix

### Final Model Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model Index** | **Type** | **Model Weight** | **Fitted features** | **Target Transformer** |
| 0 | XGBoostModel | 0.1667 | 160 | str |
| 1 | GLMModel | 0.3333 | 42 | str |
| 2 | XGBoostModel | 0.5 | 191 | str |

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

|  |  |
| --- | --- |
| **fold\_id** | **0.0** |
| model\_origin | RANDOM |
| n\_gpus | 1 |
| gamma | 0.0 |
| n\_estimators | 2400 |
| min\_lambda\_fraction |  |
| min\_child\_weight | 1 |
| score\_f\_name | AUC |
| gpu\_id | 0 |
| tolerance |  |
| boosting\_type | gbdt |
| random\_state | 1234 |
| debug\_verbose | 0 |
| n\_lambda |  |
| silent | True |
| model\_id | Final Model - Single Model |
| min\_data\_in\_bin | 1 |
| monotonicity\_constraints | False |
| max\_delta\_step | 10 |
| reg\_alpha | 5.0 |
| scale\_pos\_weight | 1.0 |
| booster | gbtree |
| colsample\_bytree | 0.7 |
| min\_child\_samples | 1 |
| objective | binary:logistic |
| tree\_method | gpu\_hist |
| max\_depth | 8 |
| max\_leaves | 0 |
| early\_stopping\_threshold | 0.0 |
| coordinate\_selection |  |
| max\_bin | 128 |
| n\_jobs | 1 |
| early\_stopping\_rounds | 150 |
| learning\_rate | 0.03 |
| reg\_lambda | 0.0 |
| subsample | 0.7 |
| grow\_policy | depthwise |
| updater |  |
| eval\_metric | auc |
| num\_classes | 2 |