# Credit Card Transactions Fraud Detection

**Group 11 Members:**

Wen-Hsuan Hung

Yun Chieh Huang

Lupeng Wang

Zhanpeng Zhang

**Project Advisor:**

Stephen Coggeshall

June 8th, 2022

# Table of Contents

# Executive Summary

For many organizations, fraudulent identity detection is so important that it makes fraud analytics indispensable. In this case, it is critical to find out fraudulent identities and stop them before their applications are processed. Admittedly, it will not be easy to build a machine learning model to detect frauds, but our team still devotes ourselves to tackle this business problem for a credit card releasing organization in this project.

In this fraud analytics project, we built numbers of  trained supervised machine learning models with different algorithms and chose the best model to identify fraudulent activities. Despite the difficulty of  building a model that is able to capture frauds perfectly, we still have confidence with our optimal supervised machine learning model to maximize the reduction of financial losses for the organization.

The main goal of this report is to provide detailed and exhaustive information regarding the data itself, the procedure of data preparation, and machine learning model building. Our team has conducted five steps to reach our project goal:

1. Data Preprocessing:
   Investigated suspicious records, filled up missing data, and removed duplicate records
2. Feature Engineering:
   Created new features with existing data
3. Feature Selection:
   Selected a subset of features to increase the model efficiency
4. Modeling:
   Built numbers of machine learning models, trained, and tested the model to acquire the best one
5. Result Analysis:
   Analyzed the result from the best model and gave recommendations

Our team has combined newly created variables and existing variables for our final model to detect fraudulent activities based on FDR rate at 3% rejection rate. Eventually, we came up with the best model, the KNN Classifier ().

## Description of data

The dataset we used for this project is actual credit card purchases from a US government organization in 2006. It includes 96,753 rows and 10 columns with the data types of float, integer, object, and datetime. The columns include record number, card number, date, merchant number, merchant description, merchant state, merchant zip code, transaction type, amount and whether the transaction is fraud or not.

### Summary Statistics Tables:

- **Table for Numeric Field:**

| Field Name | % Populated | Min | Max | Mean | Stdev | % Zero |
|---|---|---|---|---|---|---|
| Amount | 100.000% | 0.010 | 3102045.53 | 427.89 | 10006.14 | 0.000 |
| Date | 100.000% | 2006-01-01 | 2006-12-31 | N/A | N/A | N/A |

*Table 1*

- **Table for Categorical Field:**

| Field Name | % Populated | # Unique Values | Most Common Value |
|---|---|---|---|
| Recnum | 100.00% | 96753 | N/A |
| Cardnum | 100.00% | 1645 | 5142148452 |
| Merchnum | 100.00% | 13092 | 930090121224 |
| Merch description | 100.00% | 13126 | GSA-FSS-ADV |
| Merch state | 98.77% | 228 | TN |
| Merch zip | 95.19% | 4568 | 38118.0 |
| Transtype | 100.00% | 4 | P |
| Fraud | 100.00% | 2 | 0 |

*Table 2*

## Distributions of Some Fields:

We only showed some fields' distributions here based on the importance. To view the full information of all fields, the data quality report (DQR) can be found in the appendix 1.

- **Number of Transaction by Day of Week:**

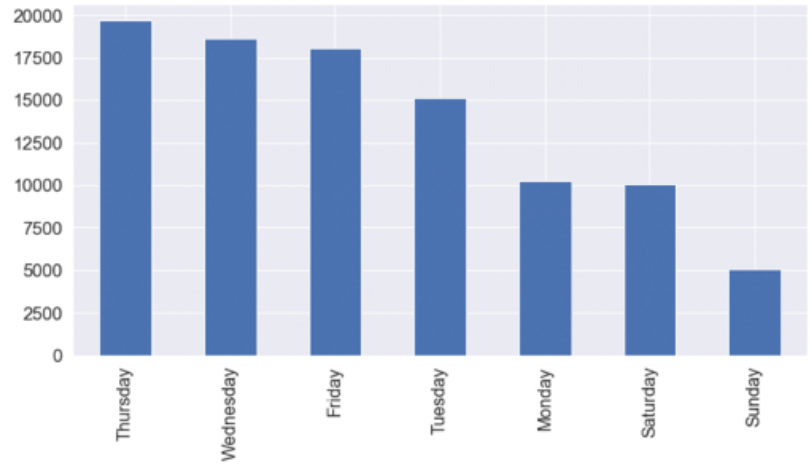



*Figure 1*

In Figure 3, Thursday got the most of the transactions, followed by Wednesday and Friday, while Sunday had the least transactions.

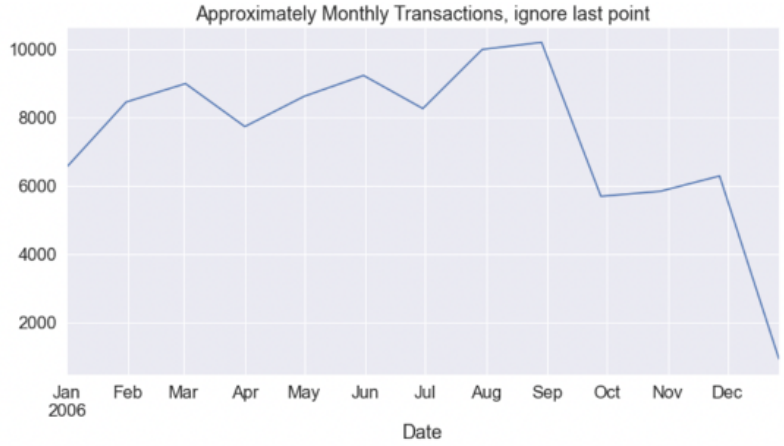



*Figure 2*

In Figure 4, September has the most transactions.

- **Fraud Label:**


*Figure 3*

The *fraud* field has two unique values of 0 and 1 indicating whether each of the records is fraud or not. 0s mean the rows of applications are benign and 1s are referring to the fraudulent applications or identities. From the histogram, we can observe the significant amount difference between those two labels. Among the 96,753 records, there are 95,694 classified as the normal application while the other 1,059 observations are categorized as fraud.

- **Number of Transactions by Merch State:**

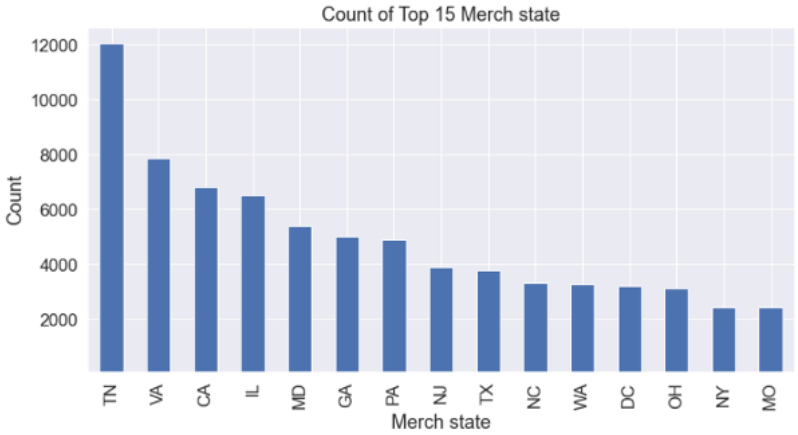



*Figure 4*

```
data['Merch state'].value_counts().tail(100)
```

```
275    1
726    1
927    1
635    1
189    1
      ..
495    1
376    1
458    1
546    1
116    1
```

*Figure 5*

In the field of *Merch State*, there are 228 unique values with around 1.23% of missing values, which can be further investigated since there are only 50 states in the U.S.A.

## Data Cleaning

After looking through all the records and fields along with their distributions and meanings, we then started to conduct the data cleaning process by removing any possible missing values, exclusions, and outliers, as well as adjusting the frivolous field values.

- **Outliers**

In the original dataset, there are 96753 rows of data, among all data, there are three types of transaction, which are “P”, “D”, “A”, “Y”. Since most of the transaction type is “P” ( 181 rows of “A”, 173 rows of “D” and 1 row of “Y”), we only keep transaction type “P” data in our further analysis. Besides, we removed the transactions which amount  are greater than 3000000. There are 96397 rows of data remaining.

- **Missing Values**

In the original dataset, there are 3198 missing values for Merchnum, 1020 missing values for Merch state, 4300 missing values for Merch zip. To start, we first replaced the null values in Merchnum column. For Merch state column, we filled in the missing values based on the value from the same merchant zip code. For merchant zip code column, missing values were filled in based on the value from the same Merch state information. If Merch state information was also missing, the values were filled in based on the rows with the same Merch description. For the rest of the missing values, the values were filled as “unknown”.

- **Create Variables**

1. **Day of Week:** we extracted day of week information from the “Date” column, and set it in new column “Dow”.

2. **Dow Risk:** we calculated the fraud risk of each day in a week. The chart shows that Sunday has the highest fraud risk, and Thursday has the lowest risk.

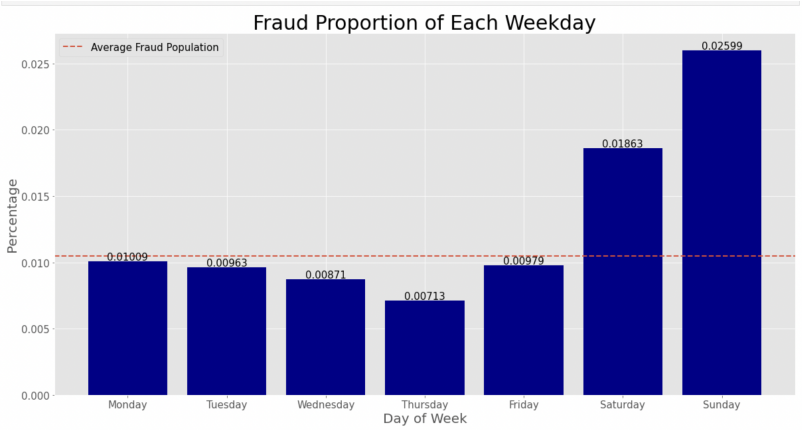



*Figure 6*

## Dataset Overview After Data Cleaning

Figure 7 shows that IN has the highest fraud percentage.

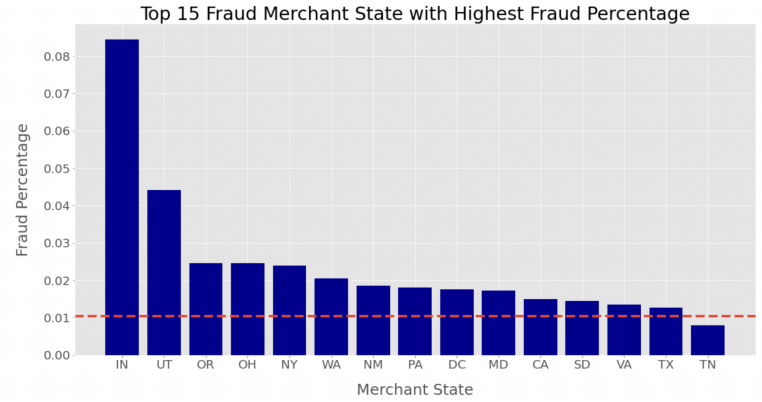



*Figure 7*

Figure 8 shows the top 15 card numbers with the highest fraud percentage. Most importantly, 5142847398, 5142117814, 5142110434, 5142228988, 5142134652, 5142189113 have a fraud percentage of 100%.

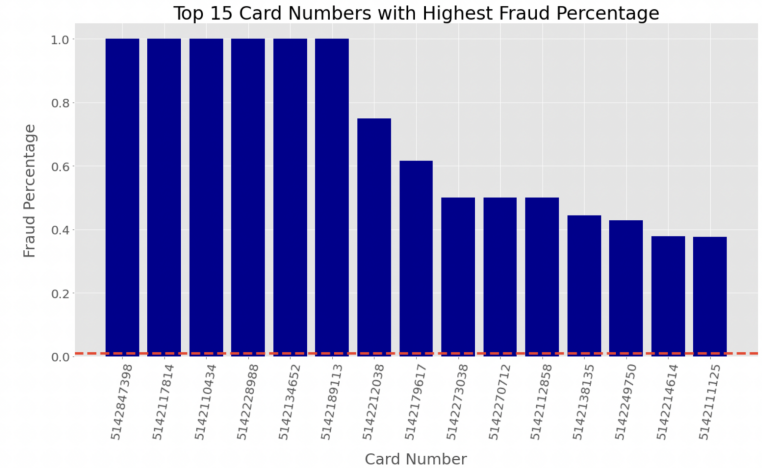



*Figure 8*

# Feature Engineering

Feature engineering is the most important step in building predictive machine learning models. A simple linear model with good variables that explain the variances in the dependent variable can outperform a complex nonlinear model with bad variables. In this step, the goal is to create as many candidate variables as possible. We don't need to worry about dimensionality or multicollinearity between variables just yet. In addition, not all variables need to be strong predictors that are univariately important. In fact, we will throw most of them away in a feature selection step.

## Variable Summary

There are 96397 columns in total. The variables include columns in the original dataset (*record number, card number, date, merchant number , merchant description, merchant state, merchant*

*zip, transaction type* (only type “P”)*, amount, fraud*. A full list of the variable names is attached in Appendix 2.

Beside combining original columns, other variables that I created can be grouped to four categories, which are amount variables, frequency variables, day-since variables and velocity change variables. Amount variables include the average, maximum, median, total, actual/average, actual/maximum, actual/median, actual/total amount for each card, merchant, card at this merchant, card in this zip code, card in this state over the past 0, 1, 3, 7, 14, 30 days.

Frequency variables represent the number of transactions with this card, merchant, card at this merchant, card in this zip code, card in this state over the past 0, 1, 3, 7, 14, 30 days. Days-since variables represent the current date minus date of the most recent transaction with the same card, merchant, card at this merchant, card in this zip code, card in this state. For records which are first seen, I set it to 365 days for days-since. Velocity change variables represent the number or amount of transactions with the same card or merchant over the past 0, 1 day divided by average daily number or amount of transactions with the same card or merchant over the past 7 days, 14 days, 30 days. We will discuss the formulas and logic we used to create each type of variable in the sections below.

| **Description of variables** | **# Variables created** | **Approximate CPU time** |
|---|---|---|
| Original fields from the dataset | 9 | |
| Date of week target encoded (average fraud percentage of that day) | 1 | 8 secs |
| New entities combining different original fields. | 26 | |
| Days since: # days since an application with that entity has been seen. Entities list is attached below. | 5 | |

| | | |
|---|---|---|
| Velocity: # records with the same entity over the last {7,14,30} days. | 299 | 4.83 secs |
| # Unique entities for that particular entity over the past {0, 1,3,7,14,30} days | 30 | |
| Amount over the past {0, 1,3,7,14,30} days | 240 | 8.11 mins |

*Table 3*

*Entities: ['Cardnum', 'Merchnum', 'state_risk', 'card_merch', 'card_zip', 'card_state', 'merch_zip', 'merch_state', 'state_des', 'state_zip', 'zip3', 'card_zip3', 'merchnum_zip', 'merchnum_zip3', 'Card_Merchdesc', 'Card_dow', 'Merchnum_desc', 'Merchnum_dow', 'Merchdesc_State', 'Merchdesc_Zip', 'Merchdesc_dow', 'Card_Merchnum_desc', 'Card_Merchnum_State', 'Card_Merchnum_Zip', 'Card_Merchdesc_State', 'Card_Merchdesc_Zip', 'Merchnum_desc_State', 'Merchnum_desc_Zip']*

- **Original Fields**

We kept all the original fields from the dataset because record is a unique identifier for each application and *fraud_label* is our dependent variable.

- **Target Encoded Day of Week**

We get the day of week from the date column in our dataset and target encoded it since it is a categorical field. Target encoding is to replace a categorical field with a numerical measure of the dependent variables for all records from that category. In this case, we replaced day of week with the average probability of fraud for each day of week.

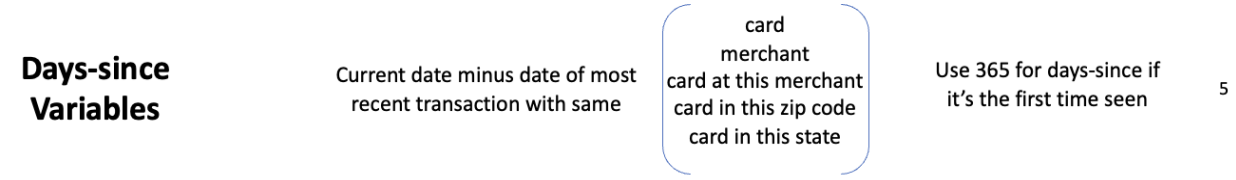
- **Entities Combining Different Original Fields**

The new columns we create includes (see full list in Appendix 2) *card_merch (Cardnum + Merchnum), card_zip (Cardnum + Merch zip*), *card_state (Cardnum + Merch state)*, *merch_zip (Merchnum + Merch zip)*, *merch_state (Merchnum + Merch state*), *state_des (Merch state + Merch description*), *state_zip (Merch state + Merch zip), zip3* (the first three digit of *Merch zip*), *card_zip3 (Cardnum + zip3), merchnum_zip (Merchnum + Merch zip), merchnum_zip3*

*(Merchnum + zip3), Card_Merchdesc (Cardnum + Merch description), Card_dow (Cardnum + Dow), Merchnum_desc (Merchnum + Merch description), Merchnum_dow (Merchnum + Dow), Merchdesc_State (Merch description + Merch state), Merchdesc_Zip (Merch description + Merch zip), Merchdesc_dow (Merch description + Dow), Card_Merchnum_desc (Cardnum + Merchnum + Merch description), Card_Merchnum_State (Cardnum + Merchnum + Merch state), Card_Merchnum_Zip (Cardnum + Merchnum + Merch zip), Card_Merchdesc_State (Cardnum + Merch description + Merch state), Card_Merchdesc_Zip (Cardnum + Merch description + Merch zip), Merchnum_desc_State (Merchnum + Merch description + Merch state), Merchnum_desc_Zip (Merchnum + Merch description + Merch zip).*

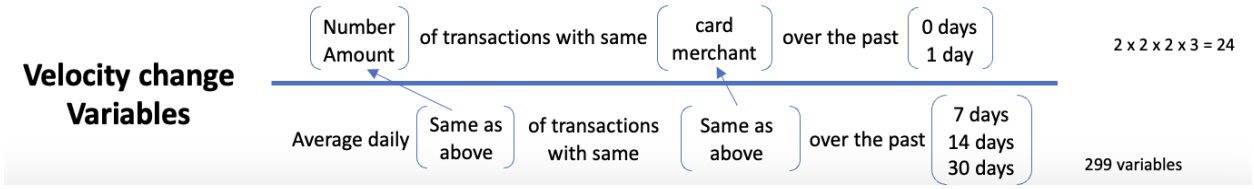
- **Days Since**

For each entity, we calculated the number of days since we last saw that entity in a previous application. A low value could be a signal of fraud.



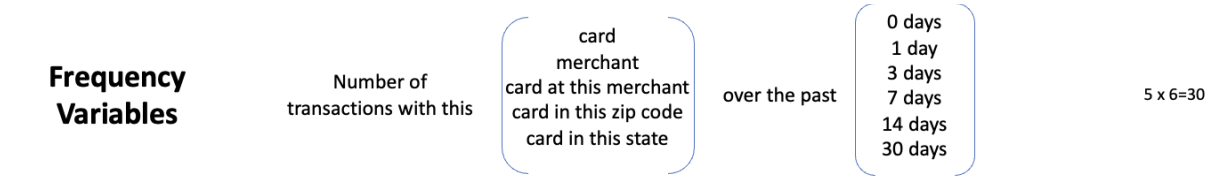

- **Velocity**

This is the number of records with the same entity over the last n days. We chose n to be a number in {7,14,30}. A high value could be a signal of fraud



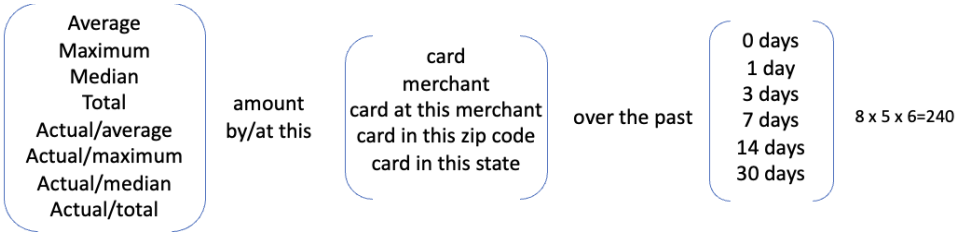

- **Unique Entity Counts (Frequency)**

This is the number of unique entities over the past {0, 1,3,7,14,30} days.




- **Amount Variables**

**Amount Variables**



## Feature Selection

After creating the candidate variables, we continued the feature engineering process through feather selection to get the number of variables down. Feature Selection is an important step in building a good model. It reduces dimensionality. High dimensionality gives the model more opportunities to overfit to noise, resulting in poor generalized performance. In addition, strange things happen in high dimensions; data points turn to outliers and sparse, and exponentially more data is needed to see true nonlinearities rather than noise. Thus, it is much more difficult to fit nonlinear models as dimensionality increases, hence linear model usually performs better with high dimensionality. Lower dimensions enable the nonlinear model to run faster to optimize model architecture and hyperparameters.

Benefits of feature selection:

- Allows the machine learning models to train faster.
- Reduces the complexity of a model and makes it easier to interpret.
- Improves the accuracy of a model when done correctly
- Reduces overfitting

In the project, feature selection identifies variables with substantial information and minimize the number of variables for the model through a two-step process - filter and wrapper. The following figure 9 illustrates in high level the feature selection process.

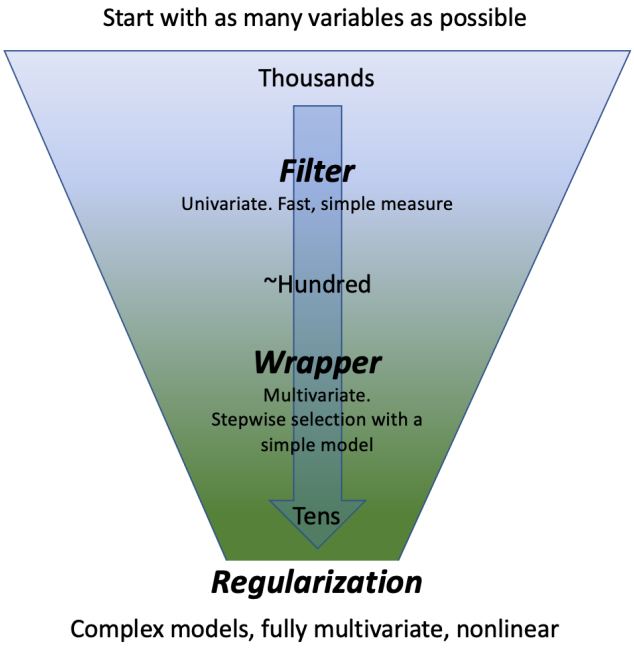



*Figure 9*

## Filter

Filter methods use a statistical methods to assess the significance of predictors outside of predictive models and retain the best variables. The filter's purpose is to determine how relevant each variable is in predicting y on its own. The kinds of data involved, both in predictors and outcomes — either numerical or categorical — must be considered while selecting filter techniques. Pearson Correlations, Mutual Information, univariate Kolmogorov-Smirnov (KS) score, and Fraud Detection Rate (FDR) are all common filter approaches for binary classification issues. For this project, two measures, KS and FDR were used to calculate the top 600 variables.

### Kolmogorov-Smirnov(KS)

KS is a statistical measure of how well two distributions are separated (goods vs. bads). For each candidate variable, plot the goods and bads separately. The more different the curves, the better the variable for separating, and thus the more important the variable is.

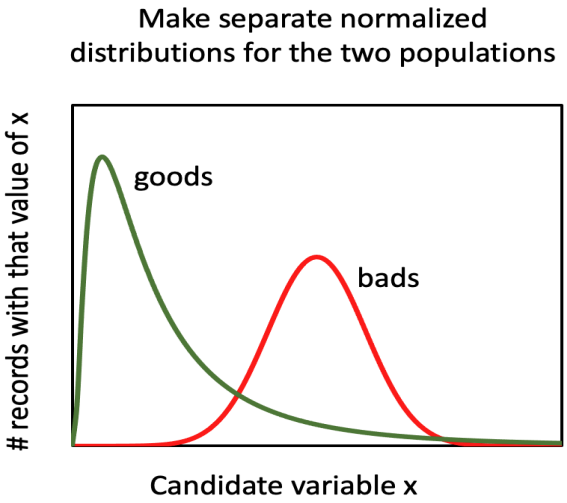
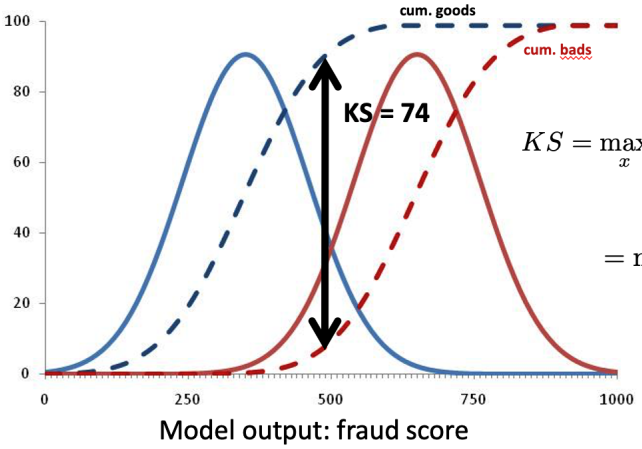



*Figure 10*

For each candidate feature, two distributions of fraud (bad) and non-fraud (goods) records are built, respectively, and then the KS score is calculated based on the formula below.

$$KS = \max_{x} \int_{x_{min}}^{x} [P_{\text{good}} - P_{\text{bad}}]\, dx$$

The KS is the maximum of the difference of the cumulative distribution. After the KS scores are calculated, variables with low KS scores would be dropped.




$$KS = \max_x \left| \int_{x_{min}}^{x} P_{\text{good}}\, dx - \int_{x_{mi}}^{x} P_{\text{bad}}\, dx \right|$$

$$= \max_x \left| \int_{x_{min}}^{x} [P_{\text{good}} - P_{\text{bad}}]\, dx \right|$$

*Figure 11*

**FDR**

The second measure we used in the filter is Fraud detection rate (FDR), a measure of goodness for fraud detection. It is more robust and meaningful than the false positive measure of goodness for highly unbalanced problems. It indicates what percentage of all the bads are caught at a particular examination cutoff location. For example, FDR 50% at 3% means the model catches 50% of all the frauds in 3% of the population.

$$FDR@3\% = \frac{Number\ of\ fraud\ caught\ at\ 3\%\ rejection\ rate}{Total\ number\ of\ frauds\ in\ the\ dataset\ (training, testing, oot)}$$

Since both metrics are typical business measures for binary classification model, we calculated the average ranking of the KS and FDR scores and selected the variables that ranked the highest (except Fraud label) by this average ranking. The number of variables selected in this step is 80, which will be further reduced in the next wrapper step.

## Wrapper

The top 80 variables will be further reduced to 20 variables in this step. There are three common types of wrapper process: forward selection, backward selection, and general stepwise selection. The wrapper method is based on a specific machine learning algorithm that tries to fit on a given dataset. It has a model "wrapped" around the process. The model can be anything. In the project, forward selection has been used to select variables. Forward selection is an iterative method in which we start with having one feature in the model. In each iteration, we keep adding the

features, which improves our model till an addition of a new variable does not improve the performance of the model will be the best combination of features to keep.
Here is the forward selection process mathematically illustrated:

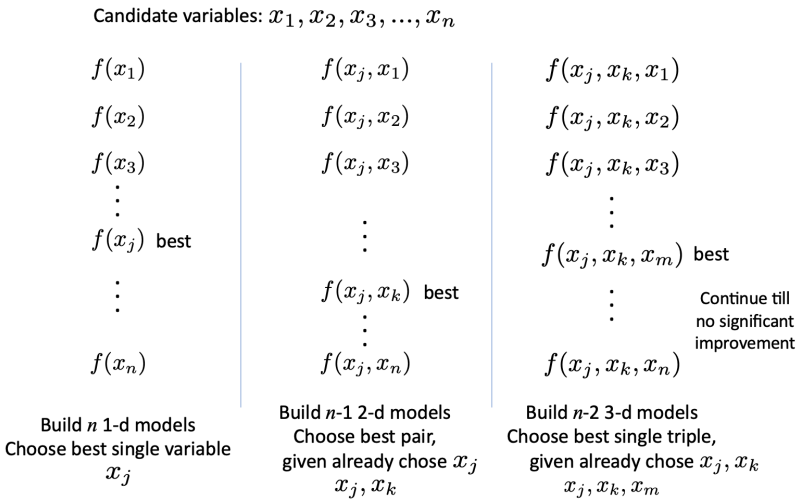



*Figure 12*

Here are results from forward selection, simple nonlinear wrapper, FDR as the measure:

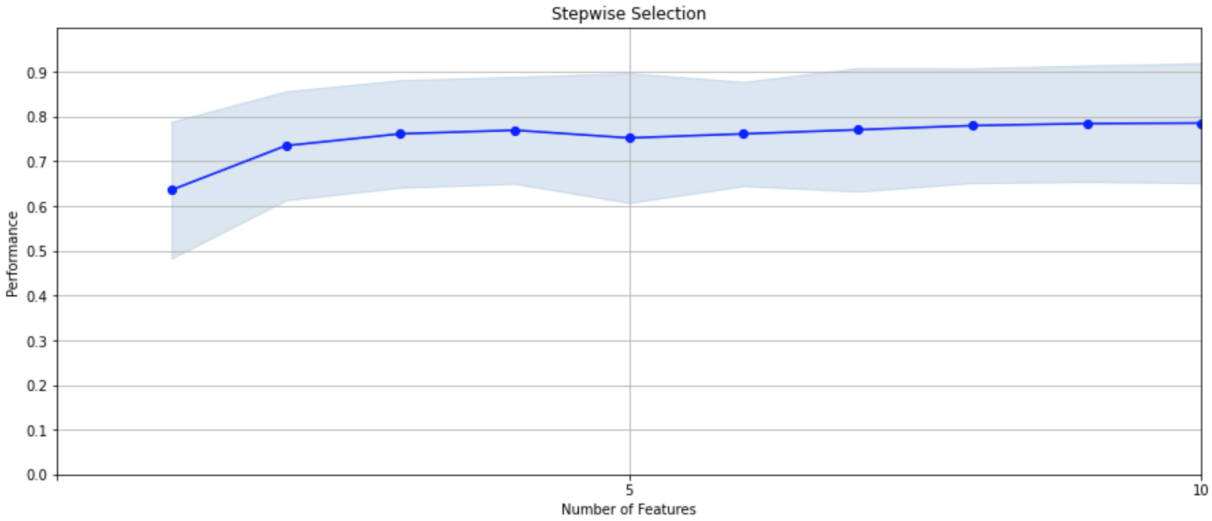



*Figure 13*

After wrapper process, a total of 20 variables were chosen to be used for modeling. The final variables are shown in the table 4 below:

| Ranking | Variable name |
|---|---|
| 1 | card_zip3_total_7 |
| 2 | Merchnum_max_7 |
| 3 | card_zip_total_14 |
| 4 | card_zip_total_60 |
| 5 | merch_zip_max_7 |
| 6 | Card_Merchnum_desc_total_60 |
| 7 | zip3_total_0 |
| 8 | card_merch_total_30 |
| 9 | card_zip_total_30 |
| 10 | card_merch_total_60 |
| 11 | Merchnum_desc_total_7 |
| 12 | Merchnum_desc_avg_7 |
| 13 | amount_cat |
| 14 | Merchnum_desc_total_14 |
| 15 | Card_Merchnum_desc_total_30 |
| 16 | Card_Merchdesc_total_60 |
| 17 | Merchnum_desc_max_7 |
| 18 | merch_zip_total_0 |
| 19 | zip3_actual/avg_60 |
| 20 | Card_Merchdesc_total_7 |

*Table 4*

# Model Algorithms

After selecting the top 30 variables, we then started to train various models along with tunning the hyperparameters to find the model that can predict the fraud label best. Models we tried in this step includes logistic regression, single decision tree, random forest, boosted tree, neural network, and naïve bayes. We will discuss different model technique as well as their performance in this section.

## Logistic Regression

Logistic regression is the generalization of the standard regression model that is used when the response variable y is binary or binomial. In this fraud project, we set goods to be 0 and bads(fraud applications) to be 1. We used this model to predict the likelihood of fraud happening in the future and we scaled the model output to go between 1 and 1000 for business convenience.

A logistic regression is $y = \frac{1}{1 + e^{-\sum b_i x_i}}$

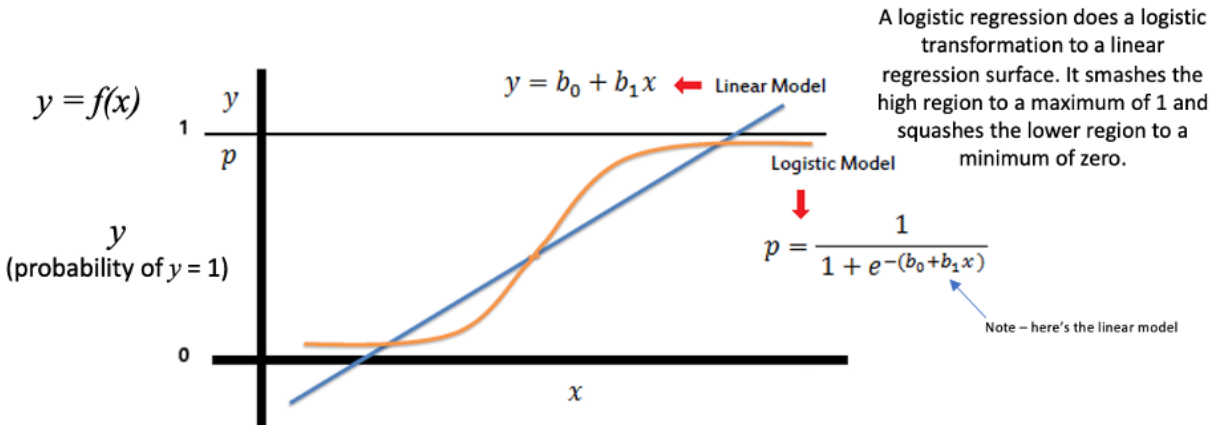



*Figure 14*

Firstly, we tried out logistic regression as our baseline algorithm. The reason why we chose logistic regression is because it's the typical baseline model for binary classification problems. It has a stable performance, and it is hard to screw up. The hyperparameters that we tuned are #variables, penalty, C (regularization strength), solver, and max_iter (max number of iterations).

We explored a range of hyperparameter combinations, including no regularization term, l1 regularization, and l2 regularization. Note that "saga" is the solver for l1 regularization, and "lbfgs" is the solver for l2 regularization or no regularization, and that's why we see the variations here.

**Performance:**

| | | Parameters | | | | | Average FDR at 3% | | |
|---|---|---|---|---|---|---|---|---|---|
| | iteration | #varialbes | penalty | C | solver | max_iter | train | test | oot |
| Logistic Regression | 1 | 15 | l2 | 1 | lbfgs | 1000 | 0.664 | 0.672 | 0.292 |
| | 2 | 5 | l2 | 100 | lbfgs | 1 | 0.592 | 0.595 | 0.352 |
| | 3 | 20 | l2 | 0.1 | lbfgs | 999999 | 0.666 | 0.664 | 0.343 |
| | 4 | 20 | none | | lbfgs | 999999 | 0.665 | 0.674 | 0.373 |
| | 5 | 20 | l1 | 0.001 | saga | 999999 | 0.633 | 0.628 | 0.454 |
| | 6 | 20 | l2 | 0.001 | lbfgs | 999999 | 0.637 | 0.656 | 0.245 |

As we can see from the table, "#variable=20, penalty=l1, C=0.001, solver="saga", and max_iter=9999999" has the highest out-of-time FDR at 3 % while no obvious overfitting seems to have happened.

*Table 5*

## Decision Trees

Decision tree classifier is a flowchart-like tree-based model that makes decisions according to the stepwise results of each branch (sub-tree). From the graph below, we can see each decision node as a feature/attribute where each branch indicates a decision rule and each leaf node indicates an outcome. When structuring the decision tree model, we started with all data in the root node (the top one) and then split them into boxes under different splitting criterion such as gini or entropy which is one of the hyperparameters we need to set for model creation.

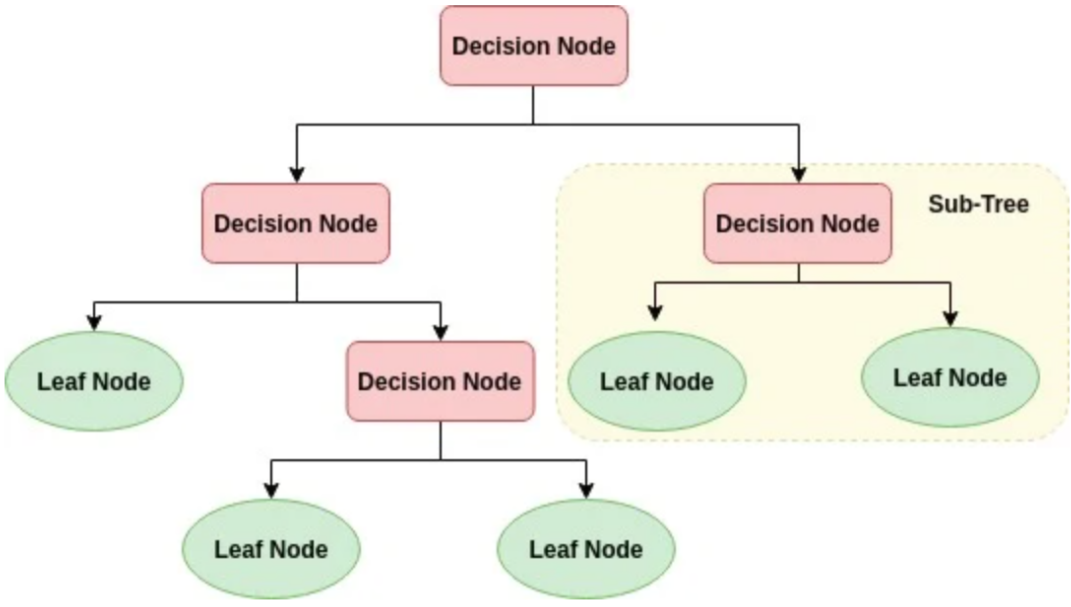



*Figure 15*

**Performance:**

| | | Parameters | | | | | Average FDR at 3% | | |
|---|---|---|---|---|---|---|---|---|---|
| | iteration | #varialbes | criterion | max_depth | min_samples_split | min_samples_leaf | train | test | oot |
| Decision Tree | 1 | 15 | gini | 8 | 100 | 60 | 0.795 | 0.718 | 0.474 |
| | 2 | 5 | gini | 1 | 1000 | 600 | 0.435 | 0.412 | 0.303 |
| | 3 | 20 | gini | 10 | 90 | 60 | 0.830 | 0.770 | 0.412 |
| | 4 | 20 | gini | 9999 | 2 | 1 | 1.000 | 0.602 | 0.274 |
| | 5 | 20 | gini | 10 | 10 | 1 | 0.820 | 0.672 | 0.299 |

*Table 6*

As we can see from the table, "#variable=15, criterion=gini, max_depth=8, min_samples_split=100, and min_samples_leaf=60" has the highest out-of-time FDR at 3%. There might be slight overfitting since the FDR is lower in the test set than it is in the training set. Note that all of the combinations we tried were somewhat overfitted, as decision trees are prone to overfit.

## Random Forest

Random Forest is an extended Bagging version of the Decision Tree algorithm. The ensemble version of the algorithm has a lower risk of overfitting which is a major weakness of the Decision Tree. How Random Forest works is that it consists of a large number of individual decision trees that operate as an ensemble. Each individual tree has some randomness associated with it due to using only a randomly-chosen subset of variables or records and/or for each split iteration within a tree. For classification problems like our case, every single tree in the random forest generates a class prediction and the results from all trees are later combined by voting to become the model's final predictions.

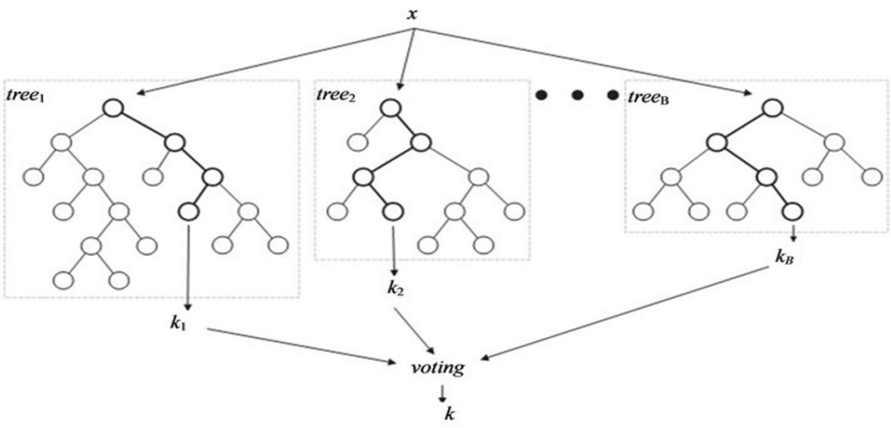



*Figure 16*

**Performance:**

| | | Parameters | | | | | | Average FDR at 3% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | iteration | #variables | n_estimators | max_depth | min_samples_split | min_samples_leaf | max_features | train | test | oot |
| Random Forest | 1 | 15 | 40 | 7 | 100 | 60 | 8 | 0.780 | 0.733 | 0.536 |
| | 2 | 5 | 1 | 1 | 1000 | 1000 | 1 | 0.524 | 0.509 | 0.382 |
| | 3 | 20 | 40 | 7 | 100 | 60 | 8 | 0.798 | 0.764 | 0.551 |
| | 4 | 20 | 1 | 9999 | 2 | 1 | 1 | 0.805 | 0.472 | 0.220 |
| | 5 | 20 | 100 | 9999 | 2 | 1 | 1 | 1.000 | 0.844 | 0.520 |

*Table 7*

As we can see from the table, "#variable=20, n_estimators=40, max_depth=7, min_samples_split=100, min_samples_leaf=60, and max_features=8" has the highest out-of-time FDR at 3% while no obvious overfitting exists. The FDR is significantly higher than the previous two, which is expected.

## Boosted Trees

Tree based models like decision tree and random forest are very popular machine learning models, but they have their limitations. The main drawback of decision trees is that it is prone to overfitting the training data. Random forests are a great alternative as it combines multiple decision trees via a technique called Bagging. However, random forests also have a drawback. Because of parallel learning, if one individual decision tree made a mistake, all the trees could potentially be making the same mistake repeatedly.

The idea of boosting is to avoid those drawbacks. In boosted trees, new trees are formed to only explain the residual errors of trees in previous rounds. Therefore, new trees are created one after another. Each tree is dependent on the previous tree. This type of learning is called sequential learning.

Some of the key characteristics of boosting are:

- Boosting is a way of training a series of weak models (eg. A shallow decision tree) to result in a strong model. Any weak model can be used but trees are the most common.
- The objective of each additional tree is to predict the remainder error of the previous trees.
- Boosting is an iterative process. Each tree is dependent on the previous one. Therefore, it is hard to parallelize the training process of boosting algorithms, hence longer training time.

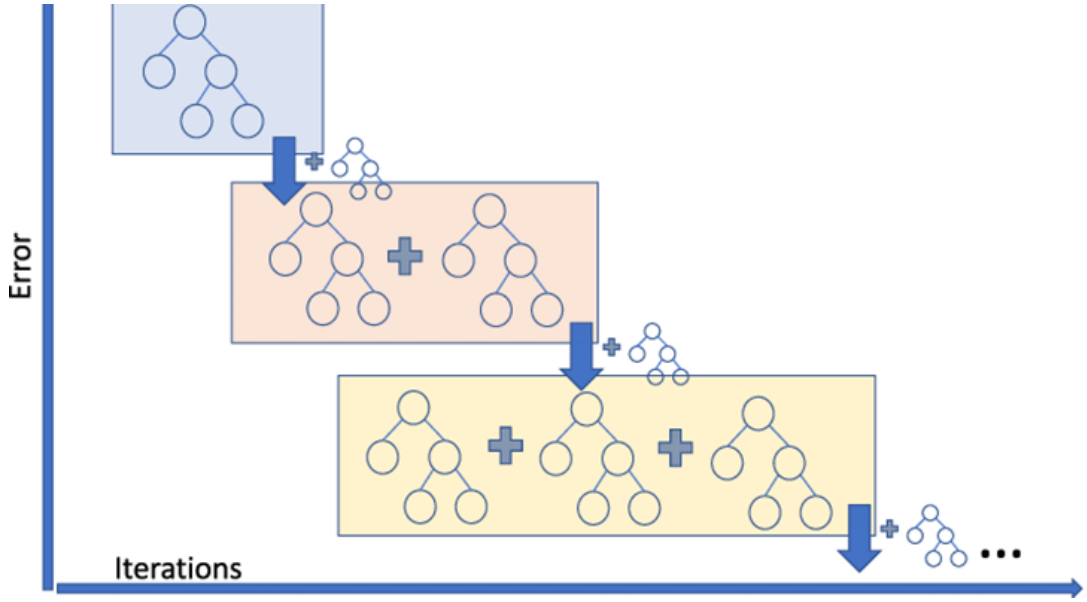



*Figure 17*

Some of the most popular machine learning libraries for boosted trees include AdaBoost, XGBoost, and LightGBM. For this project, we used the LGBMClassifier from the LightGBM library.

**Performance:**

| | | Parameters | | | | | | Average FDR at 3% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | iteration | #variables | num_leaves | n_estimators | max_depth | learning_rate | min_split_gain | train | test | oot |
| LightGBM | 1 | 15 | 3 | 600 | -1 | 0.1 | 0 | 0.869 | 0.818 | 0.498 |
| | 2 | 5 | 2 | 1 | 1 | 1 | 5 | 0.363 | 0.370 | 0.284 |
| | 3 | 20 | 3 | 600 | -1 | 0.1 | 0 | 0.891 | 0.832 | 0.434 |
| | 4 | 20 | 2 | 1 | -1 | 0.1 | 0 | 0.393 | 0.397 | 0.283 |
| | 5 | 20 | 2000 | 1 | -1 | 0.1 | 0 | 0.933 | 0.763 | 0.428 |

*Table 8*

As we can see from the table, "#variable=15, num_leaves=3, n_estimators=600, max_depth=-1 (-1 means there is no restrictions on this parameter), learning_rate=0.1, and min_split_gain=0" has the highest out-of-time FDR at 3% while no obvious overfitting exists. Due to the nature of GBM algorithms, no restrictions on max_depth doesn't really lead to overfitting.

The performance of this algorithm is better than the first two baseline algorithms, but is worse than the random forest.

**Neural Network**

A neural network is a series of algorithms that try to recognize underlying relationships in a set of data through a process that simulates the way the human brain operates. Neural networks take

in data, train themselves to recognize the patterns in this data, and then predict the outputs for a new set of similar data.

Neural Networks have layers of neurons, which are the core processing units of the network. A neural net consists of an input layer, a number of hidden layers, and an output layer. All the independent variables (x's) form the input layer. The dependent variable y is the output layer. The hidden layer is a set of nodes. Each node in the hidden layer receives weighted signals from all the nodes in the previous layer. The nodes then do a transform on this linear combination of signals. In general, the signal received from the preceding layer is a linear combination of the previous layer nodes' outputs. The node receives a combined linear combination signal, which is subsequently transferred via a transfer function, often a sigmoid or logit function. Other transfer functions are employed, but the sigmoid/logit function is the most popular. Logistic activation function $= \frac{1}{1+\exp(-\sum \beta_i x_i)}$.

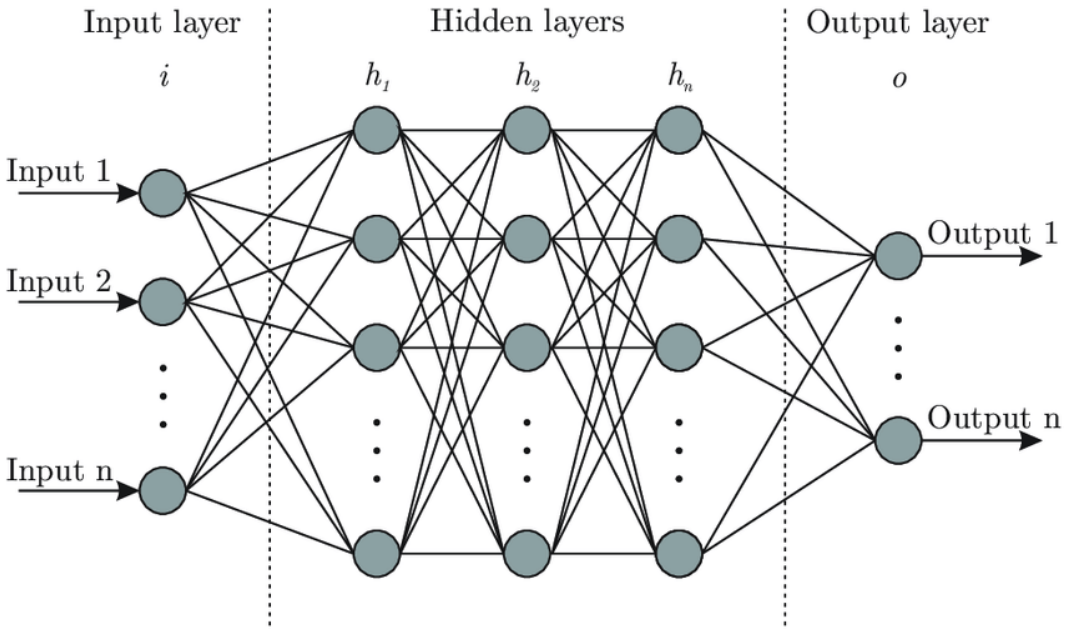



*Figure 18*

In our project, we used a package of *sklearn.neural_network.MLPClassifer*

**Performance:**

| | | | Parameters | | | | Average FDR at 3% | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | iteration | #variables | hidden_layer_sizes | alpha | solver | activation | max_iter | train | test | oot |
| Neural Network | 1 | 15 | (10,10) | 0.005 | adam | relu | 1000 | 0.779 | 0.766 | 0.510 |
| | 2 | 5 | (1,) | 100 | adam | relu | 1 | 0.228 | 0.234 | 0.155 |
| | 3 | 20 | (20,20) | 0.005 | adam | relu | 5000 | 0.861 | 0.821 | 0.450 |
| | 4 | 20 | (1,1,1,1) | 100 | adam | relu | 5000 | 0.026 | 0.030 | 0.089 |
| | 5 | 20 | (10,10,10,10) | 0.1 | adam | relu | 5000 | 0.807 | 0.802 | 0.510 |
| | 6 | 20 | (100,) | 0.001 | adam | relu | 200 | 0.895 | 0.815 | 0.386 |

*Table 9*

Since “adam” and “relu” work well with neural networks that are not as deep, we don’t change the values of these two parameters. We also tried out different numbers of hidden_layers and hidden_nodes, since they are the most important features in neural networks.

As we can see from the table, “#variable=15, hidden_layer_sizes=(10,10,10,10), alpha=0.1, solver=adam, activation=relu, and max_iter=5000” has the highest out-of-time FDR at 3%. The FDR in the training set and the FDR in the test set are almost the same, indicating that there is no overfitting. The FDR at 3% on the out-of-time data is the second highest among all of the algorithms that we have tried so far.

## KNN

KNN (k-nearest-neighbor) is a type of supervised machine learning algorithm. It is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in. KNN is a "lazy learner", meaning that it does not build a model using the training set until a query of the data set is performed. To determine whether a data point is in group A or group B, the algorithm looks at the states of the data points near it. If the majority of data points are in group A, it's very likely that the data point in question is in group A and vice versa. In short, KNN involves classifying a data point by looking at the nearest annotated data point.

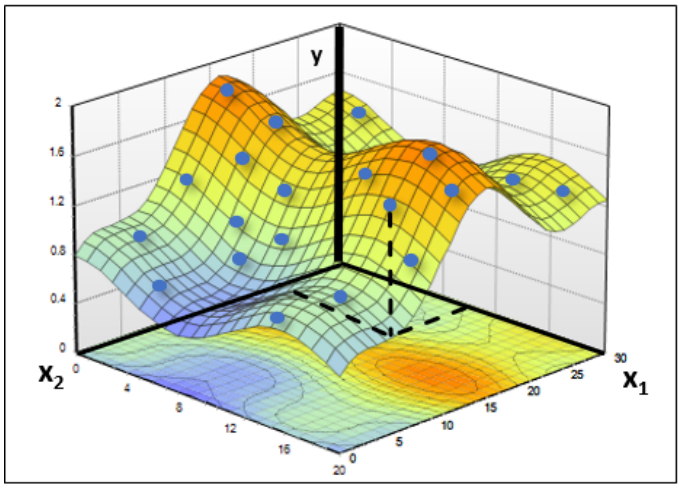



*Figure 19*

**Performance:**

| | | Parameters | | Average FDR at 3% | | |
|---|---|---|---|---|---|---|
| | iteration | #variables | n_neighbors | train | test | oot |
| Knn | 1 | 20 | 300 | 0.708 | 0.699 | 0.390 |
| | 2 | 20 | 1 | 1.000 | 0.624 | 0.285 |
| | 3 | 5 | 1 | 0.996 | 0.521 | 0.325 |
| | 4 | 5 | 300 | 0.686 | 0.669 | 0.563 |
| | 5 | 5 | 500 | 0.651 | 0.638 | 0.545 |

*Table 10*

Interestingly, the model doesn't perform well when #variables is high. Since n_neighbors (the K in KNN) is the only hyperparameter that we can tune for this algorithm, its value deeply influences the performance of the model. When n_neighbors=300, the model has the highest FDR at 3% on out-of-time data of 0.563. Surprisingly, the FDR of KNN is even higher than the random forest model. This implies that complicated models don't guarantee the best performances.

# Results

After comparing the 6 algorithms we built, we concluded that our KNN algorithm with n_neighbors = 300 and 5 variables has the highest out-of-time FDR at 3%. So we decided to use this model as our final model. Although this model is fairly simple compared to others, it does give a good and stable performance in detecting credit card frauds in this case.

We took the first 10 months as our training and test set, and the rest 2 months were chosen as our out-of-time data. We did a 70-30 train test split on the training/test set, so that 70% of the data would be set at our training data, and the rest 30% would be treated as the test set. The performances of our final model on the training set, test set, and the out-of-time set are shown below.

| Training | #Records | #Goods | #Bads | Fraud Rate | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 59010 | 58394 | 616 | 1.0549% | | | | | | | | | |
| | Bin Statistics | | | | | Cumulative Statistics | | | | | | | |
| Population Bin % | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Cumulative Goods | % Bads (FDR) | KS | FPR | |
| 1 | 590 | 298 | 292 | 50.50847 | 49.49153 | 590 | 298 | 292 | 0.510326403 | 47.4025974 | 46.89227 | 1.020548 | |
| 2 | 590 | 512 | 78 | 86.77966 | 13.22034 | 1180 | 810 | 370 | 1.387128815 | 60.06493506 | 58.67781 | 2.189189 | |
| 3 | 590 | 540 | 50 | 91.52542 | 8.474576 | 1770 | 1350 | 420 | 2.311881358 | 68.18181818 | 65.86994 | 3.214286 | |
| 4 | 590 | 563 | 27 | 95.42373 | 4.576271 | 2360 | 1913 | 447 | 3.276021509 | 72.56493506 | 69.28891 | 4.279642 | |
| 5 | 590 | 572 | 18 | 96.94915 | 3.050847 | 2950 | 2485 | 465 | 4.255574203 | 75.48701299 | 71.23144 | 5.344086 | |
| 6 | 591 | 573 | 18 | 96.95431 | 3.045685 | 3541 | 3058 | 483 | 5.236839401 | 78.40909091 | 73.17225 | 6.331263 | |
| 7 | 590 | 577 | 13 | 97.79661 | 2.20339 | 4131 | 3635 | 496 | 6.224954619 | 80.51948052 | 74.29453 | 7.328629 | |
| 8 | 590 | 573 | 17 | 97.11864 | 2.881356 | 4721 | 4208 | 513 | 7.206219817 | 83.27922078 | 76.073 | 8.202729 | |
| 9 | 590 | 581 | 9 | 98.47458 | 1.525424 | 5311 | 4789 | 522 | 8.201185053 | 84.74025974 | 76.53907 | 9.17433 | |
| 10 | 590 | 579 | 11 | 98.13559 | 1.864407 | 5901 | 5368 | 533 | 9.19272528 | 86.52597403 | 77.33325 | 10.07129 | |
| 11 | 590 | 584 | 6 | 98.98305 | 1.016949 | 6491 | 5952 | 539 | 10.19282803 | 87.5 | 77.30717 | 11.04267 | |
| 12 | 590 | 583 | 7 | 98.81356 | 1.186441 | 7081 | 6535 | 546 | 11.19121828 | 88.63636364 | 77.44515 | 11.96886 | |
| 13 | 590 | 579 | 11 | 98.13559 | 1.864407 | 7671 | 7114 | 557 | 12.1827585 | 90.42207792 | 78.23932 | 12.77199 | |
| 14 | 590 | 585 | 5 | 99.15254 | 0.847458 | 8261 | 7699 | 562 | 13.18457376 | 91.23376623 | 78.04919 | 13.69929 | |
| 15 | 591 | 582 | 9 | 98.47716 | 1.522843 | 8852 | 8281 | 571 | 14.1812515 | 92.69480519 | 78.51355 | 14.50263 | |
| 16 | 590 | 588 | 2 | 99.66102 | 0.338983 | 9442 | 8869 | 573 | 15.18820427 | 93.01948052 | 77.83128 | 15.47818 | |
| 17 | 590 | 589 | 1 | 99.83051 | 0.169492 | 10032 | 9458 | 574 | 16.19686954 | 93.18181818 | 76.98495 | 16.47735 | |
| 18 | 590 | 588 | 2 | 99.66102 | 0.338983 | 10622 | 10046 | 576 | 17.20382231 | 93.50649351 | 76.30267 | 17.44097 | |
| 19 | 590 | 585 | 5 | 99.15254 | 0.847458 | 11212 | 10631 | 581 | 18.20563757 | 94.31818182 | 76.11254 | 18.29776 | |
| 20 | 590 | 582 | 8 | 98.64407 | 1.355932 | 11802 | 11213 | 589 | 19.20231531 | 95.61688312 | 76.41457 | 19.03735 | |
| 21 | 590 | 587 | 3 | 99.49153 | 0.508475 | 12392 | 11800 | 592 | 20.20755557 | 96.1038961 | 75.89634 | 19.93243 | |
| 22 | 590 | 588 | 2 | 99.66102 | 0.338983 | 12982 | 12388 | 594 | 21.21450834 | 96.42857143 | 75.21406 | 20.85522 | |
| 23 | 590 | 589 | 1 | 99.83051 | 0.169492 | 13572 | 12977 | 595 | 22.22317361 | 96.59090909 | 74.36774 | 21.81008 | |
| 24 | 590 | 587 | 3 | 99.49153 | 0.508475 | 14162 | 13564 | 598 | 23.22841388 | 97.07792208 | 73.84951 | 22.68227 | |
| 25 | 590 | 590 | 0 | 100 | 0 | 14752 | 14154 | 598 | 24.23879166 | 97.07792208 | 72.83913 | 23.6689 | |
| 26 | 591 | 591 | 0 | 100 | 0 | 15343 | 14745 | 598 | 25.25088194 | 97.07792208 | 71.82704 | 24.65719 | |
| 27 | 590 | 589 | 1 | 99.83051 | 0.169492 | 15933 | 15334 | 599 | 26.25954721 | 97.24025974 | 70.98071 | 25.59933 | |
| 28 | 590 | 589 | 1 | 99.83051 | 0.169492 | 16523 | 15923 | 600 | 27.26821249 | 97.4025974 | 70.13438 | 26.53833 | |

*Table 11*

| Test | #Records | #Goods | #Bads | Fraud Rate | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 25290 | 25026 | 264 | 1.0549% | | | | | | | | | |
| | | | Bin Statistics | | | | | Cumulative Statistics | | | | | |
| Population | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Cumulative Goods | % Bads (FDR) | KS | FPR | |
| 1 | 253 | 119 | 134 | 47.03557 | 52.96443 | 253 | 119 | 134 | 0.475505474 | 50.75757576 | 50.28207 | 0.88806 | |
| 2 | 253 | 231 | 22 | 91.30435 | 8.695652 | 506 | 350 | 156 | 1.398545513 | 59.09090909 | 57.69236 | 2.24359 | |
| 3 | 253 | 231 | 22 | 91.30435 | 8.695652 | 759 | 581 | 178 | 2.321585551 | 67.42424242 | 65.10266 | 3.264045 | |
| 4 | 253 | 247 | 6 | 97.62846 | 2.371542 | 1012 | 828 | 184 | 3.308559099 | 69.6969697 | 66.38841 | 4.5 | |
| 5 | 252 | 246 | 6 | 97.61905 | 2.380952 | 1264 | 1074 | 190 | 4.291536802 | 71.96969697 | 67.67816 | 5.652632 | |
| 6 | 253 | 247 | 6 | 97.62846 | 2.371542 | 1517 | 1321 | 196 | 5.278510349 | 74.24242424 | 68.96391 | 6.739796 | |
| 7 | 253 | 247 | 6 | 97.62846 | 2.371542 | 1770 | 1568 | 202 | 6.265483897 | 76.51515152 | 70.24967 | 7.762376 | |
| 8 | 253 | 243 | 10 | 96.04743 | 3.952569 | 2023 | 1811 | 212 | 7.236474067 | 80.3030303 | 73.06656 | 8.542453 | |
| 9 | 253 | 249 | 4 | 98.41897 | 1.581028 | 2276 | 2060 | 216 | 8.231439303 | 81.81818182 | 73.58674 | 9.537037 | |
| 10 | 253 | 248 | 5 | 98.02372 | 1.976285 | 2529 | 2308 | 221 | 9.222408695 | 83.71212121 | 74.48971 | 10.44344 | |
| 11 | 253 | 245 | 8 | 96.83794 | 3.162055 | 2782 | 2553 | 229 | 10.20139055 | 86.74242424 | 76.54103 | 11.14847 | |
| 12 | 253 | 252 | 1 | 99.60474 | 0.395257 | 3035 | 2805 | 230 | 11.20834332 | 87.12121212 | 75.91287 | 12.19565 | |
| 13 | 253 | 250 | 3 | 98.81423 | 1.185771 | 3288 | 3055 | 233 | 12.2073044 | 88.25757576 | 76.05027 | 13.11159 | |
| 14 | 253 | 250 | 3 | 98.81423 | 1.185771 | 3541 | 3305 | 236 | 13.20626548 | 89.39393939 | 76.18767 | 14.00424 | |
| 15 | 253 | 249 | 4 | 98.41897 | 1.581028 | 3794 | 3554 | 240 | 14.20123072 | 90.90909091 | 76.70786 | 14.80833 | |
| 16 | 252 | 252 | 0 | 100 | 0 | 4046 | 3806 | 240 | 15.20818349 | 90.90909091 | 75.70091 | 15.85833 | |
| 17 | 253 | 251 | 2 | 99.20949 | 0.790514 | 4299 | 4057 | 242 | 16.21114041 | 91.66666667 | 75.45553 | 16.76446 | |
| 18 | 253 | 253 | 0 | 100 | 0 | 4552 | 4310 | 242 | 17.22208903 | 91.66666667 | 74.44458 | 17.80992 | |
| 19 | 253 | 251 | 2 | 99.20949 | 0.790514 | 4805 | 4561 | 244 | 18.22504595 | 92.42424242 | 74.1992 | 18.69262 | |
| 20 | 253 | 253 | 0 | 100 | 0 | 5058 | 4814 | 244 | 19.23599457 | 92.42424242 | 73.18825 | 19.72951 | |
| 21 | 253 | 253 | 0 | 100 | 0 | 5311 | 5067 | 244 | 20.24694318 | 92.42424242 | 72.1773 | 20.76639 | |
| 22 | 253 | 252 | 1 | 99.60474 | 0.395257 | 5564 | 5319 | 245 | 21.25389595 | 92.8030303 | 71.54913 | 21.7102 | |
| 23 | 253 | 252 | 1 | 99.60474 | 0.395257 | 5817 | 5571 | 246 | 22.26084872 | 93.18181818 | 70.92097 | 22.64634 | |
| 24 | 253 | 253 | 0 | 100 | 0 | 6070 | 5824 | 246 | 23.27179733 | 93.18181818 | 69.91002 | 23.6748 | |
| 25 | 252 | 252 | 0 | 100 | 0 | 6322 | 6076 | 246 | 24.2787501 | 93.18181818 | 68.90307 | 24.69919 | |
| 26 | 253 | 252 | 1 | 99.60474 | 0.395257 | 6575 | 6328 | 247 | 25.28570287 | 93.56060606 | 68.2749 | 25.61943 | |
| 27 | 253 | 253 | 0 | 100 | 0 | 6828 | 6581 | 247 | 26.29665148 | 93.56060606 | 67.26395 | 26.64372 | |
| 28 | 253 | 252 | 1 | 99.60474 | 0.395257 | 7081 | 6833 | 248 | 27.30360425 | 93.93939394 | 66.63579 | 27.55242 | |

*Table 12*

| Out-of-time | #Records | #Goods | #Bads | Fraud Rate | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 12097 | 11918 | 179 | 1.5019% | | | | | | | | | |
| | | | Bin Statistics | | | | | Cumulative Statistics | | | | | |
| Population Bi | # Records | # Goods | # Bads | % Goods | % Bads | Total # Records | Cumulative Goods | Cumulative Bads | % Cumulative Goods | % Bads (FDR) | KS | FPR | |
| 1 | 121 | 63 | 58 | 52.06612 | 47.93388 | 121 | 63 | 58 | 0.528612183 | 32.40223464 | 31.87362 | 1.086207 | |
| 2 | 121 | 98 | 23 | 80.99174 | 19.00826 | 242 | 161 | 81 | 1.350897802 | 45.25139665 | 43.9005 | 1.987654 | |
| 3 | 121 | 101 | 20 | 83.47107 | 16.52893 | 363 | 262 | 101 | 2.198355429 | 56.42458101 | 54.22623 | 2.594059 | |
| 4 | 121 | 120 | 1 | 99.17355 | 0.826446 | 484 | 382 | 102 | 3.205235778 | 56.98324022 | 53.778 | 3.745098 | |
| 5 | 121 | 116 | 5 | 95.86777 | 4.132231 | 605 | 498 | 107 | 4.178553449 | 59.77653631 | 55.59798 | 4.654206 | |
| 6 | 121 | 119 | 2 | 98.34711 | 1.652893 | 726 | 617 | 109 | 5.177043128 | 60.89385475 | 55.71681 | 5.66055 | |
| 7 | 121 | 117 | 4 | 96.69421 | 3.305785 | 847 | 734 | 113 | 6.158751468 | 63.12849162 | 56.96974 | 6.495575 | |
| 8 | 121 | 117 | 4 | 96.69421 | 3.305785 | 968 | 851 | 117 | 7.140459809 | 65.36312849 | 58.22267 | 7.273504 | |
| 9 | 121 | 120 | 1 | 99.17355 | 0.826446 | 1089 | 971 | 118 | 8.147340158 | 65.92178771 | 57.77445 | 8.228814 | |
| 10 | 121 | 119 | 2 | 98.34711 | 1.652893 | 1210 | 1090 | 120 | 9.145829837 | 67.03910615 | 57.89328 | 9.083333 | |
| 11 | 121 | 119 | 2 | 98.34711 | 1.652893 | 1331 | 1209 | 122 | 10.14431952 | 68.15642458 | 58.01211 | 9.909836 | |
| 12 | 121 | 117 | 4 | 96.69421 | 3.305785 | 1452 | 1326 | 126 | 11.12602786 | 70.39106145 | 59.26503 | 10.52381 | |
| 13 | 121 | 118 | 3 | 97.52066 | 2.479339 | 1573 | 1444 | 129 | 12.11612687 | 72.06703911 | 59.95091 | 11.1938 | |
| 14 | 121 | 118 | 3 | 97.52066 | 2.479339 | 1694 | 1562 | 132 | 13.10622588 | 73.74301676 | 60.63679 | 11.83333 | |
| 15 | 121 | 120 | 1 | 99.17355 | 0.826446 | 1815 | 1682 | 133 | 14.11310623 | 74.30167598 | 60.18857 | 12.64662 | |
| 16 | 121 | 121 | 0 | 100 | 0 | 1936 | 1803 | 133 | 15.12837724 | 74.30167598 | 59.1733 | 13.55639 | |
| 17 | 120 | 119 | 1 | 99.16667 | 0.833333 | 2056 | 1922 | 134 | 16.12686692 | 74.8603352 | 58.73347 | 14.34328 | |
| 18 | 121 | 121 | 0 | 100 | 0 | 2177 | 2043 | 134 | 17.14213794 | 74.8603352 | 57.7182 | 15.24627 | |
| 19 | 121 | 118 | 3 | 97.52066 | 2.479339 | 2298 | 2161 | 137 | 18.13223695 | 76.53631285 | 58.40408 | 15.77372 | |
| 20 | 121 | 120 | 1 | 99.17355 | 0.826446 | 2419 | 2281 | 138 | 19.1391173 | 77.09497207 | 57.95585 | 16.52899 | |
| 21 | 121 | 121 | 0 | 100 | 0 | 2540 | 2402 | 138 | 20.15438832 | 77.09497207 | 56.94058 | 17.4058 | |
| 22 | 121 | 120 | 1 | 99.17355 | 0.826446 | 2661 | 2522 | 139 | 21.16126867 | 77.65363128 | 56.49236 | 18.14388 | |
| 23 | 121 | 121 | 0 | 100 | 0 | 2782 | 2643 | 139 | 22.17653969 | 77.65363128 | 55.47709 | 19.01439 | |
| 24 | 121 | 120 | 1 | 99.17355 | 0.826446 | 2903 | 2763 | 140 | 23.18342004 | 78.2122905 | 55.02887 | 19.73571 | |
| 25 | 121 | 120 | 1 | 99.17355 | 0.826446 | 3024 | 2883 | 141 | 24.19030039 | 78.77094972 | 54.58065 | 20.44681 | |
| 26 | 121 | 119 | 2 | 98.34711 | 1.652893 | 3145 | 3002 | 143 | 25.18879007 | 79.88826816 | 54.69948 | 20.99301 | |
| 27 | 121 | 121 | 0 | 100 | 0 | 3266 | 3123 | 143 | 26.20406108 | 79.88826816 | 53.68421 | 21.83916 | |
| 28 | 121 | 121 | 0 | 100 | 0 | 3387 | 3244 | 143 | 27.2193321 | 79.88826816 | 52.66894 | 22.68531 | |

*Table 13*

## Conclusions

To sum up, the credit card transaction data with fraudulent labels was used to build machine learning models to predict fraudulent records. The dataset contained transaction details such as data of occurrence, card details, merchant details, and the amount. These fields are processed and cleaned by removing errors, eliminating uninteresting records, outliers and filling missing values.

Around 3000 variables were created to aid the prediction of fraudulent transactions. In time and out of time variables were created by separating the dataset by transitions before and after September 1st. Next, the feature selection method process begins with a univariate filter to get the top, most useful 80 variables, which then gets passed through a stepwise selection wrapper that identifies the final top 20 most useful variables for modeling.

For model tuning, we varied the amounts of these variables used along with hyperparameters test and tried 6 different algorithms to select the best model. The final model is then thoroughly analyzed and used to make predictions for out of time data. It is important for us to make sure the model does not flag clan transactions as fraudulent as it would be poor customer experience, harming the business in the long run. On the other hand, it is important to block as any fraudulent transactions as possible to reduce losses. For these reasons, it is important to strike a balance between the two extremes and choose the right cutoff value. Our estimated saving is $180,000 for a 2-month OOT period.

The next step progression for this project is to further optimize the models measured by higher OOT scores. We would also consult industry experts as well as making the code functional for streamline and possibly industry usage.

# Appendix

## Appendix 1 – Data Quality Report (DQR)

The dataset we used for this project is actual credit card purchases from a US government organization in 2006. It includes 96,753 rows and 10 columns with the data types of float, integer, object, and datetime.

**Field summary table**

Numeric Fields:

| Field Name | % Populated | Min | Max | Mean | Stdev | % Zero |
|---|---|---|---|---|---|---|
| Amount | 100.000% | 0.010 | 3102045.53 | 427.89 | 10006.14 | 0.000 |
| Date | 100.000% | 2006-01-01 | 2006-12-31 | N/A | N/A | N/A |

Categorical Fields:

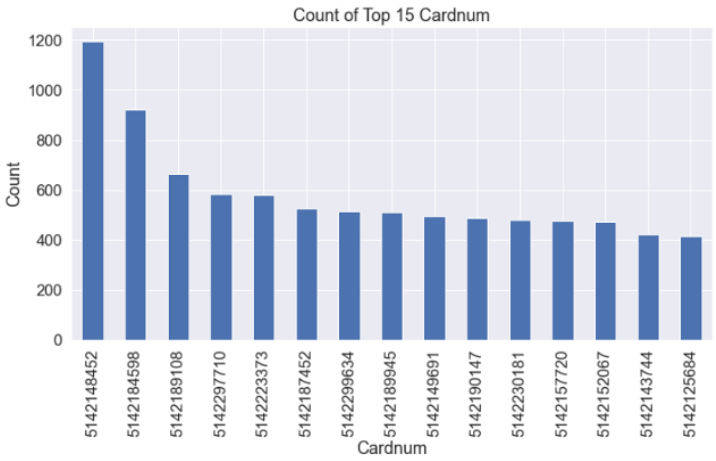
| Field Name | % Populated | # Unique Values | Most Common Value |
|---|---|---|---|
| Recnum | 100.00% | 96753 | N/A |
| Cardnum | 100.00% | 1645 | 5142148452 |
| Merchnum | 100.00% | 13092 | 930090121224 |
| Merch description | 100.00% | 13126 | GSA-FSS-ADV |
| Merch state | 98.77% | 228 | TN |
| Merch zip | 95.19% | 4568 | 38118.0 |
| Transtype | 100.00% | 4 | P |
| Fraud | 100.00% | 2 | 0 |

**Fields Descriptions**

1. Recnum:
For this field, every row has its unique value with a total of 96,753 rows. Therefore, it could be viewed as the id for each observation.

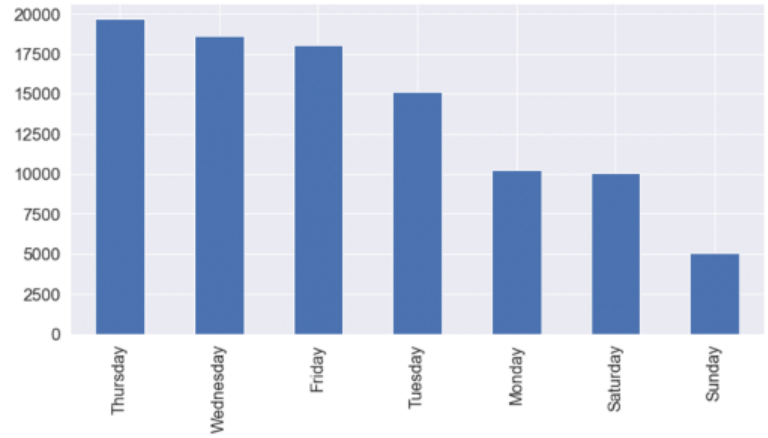
2. Cardnum:

This field represents the card number for each record in a serial number. Each row has a unique non-null value, totaling 1,645 unique values.

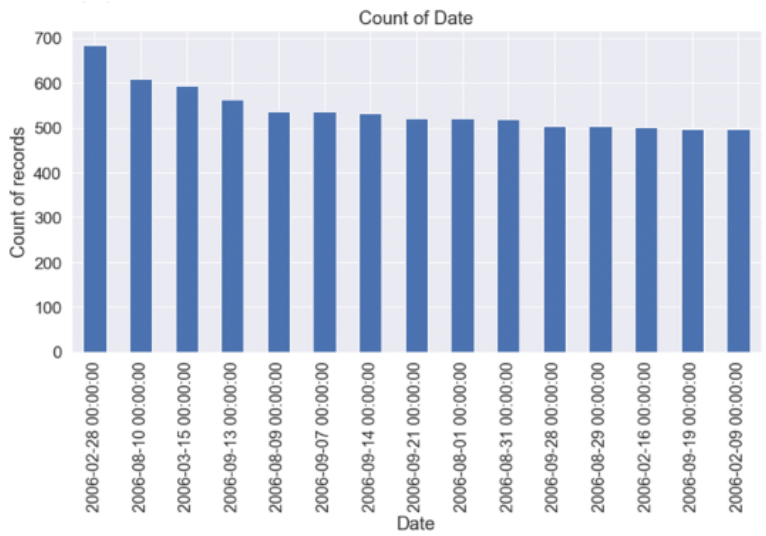



3. Date

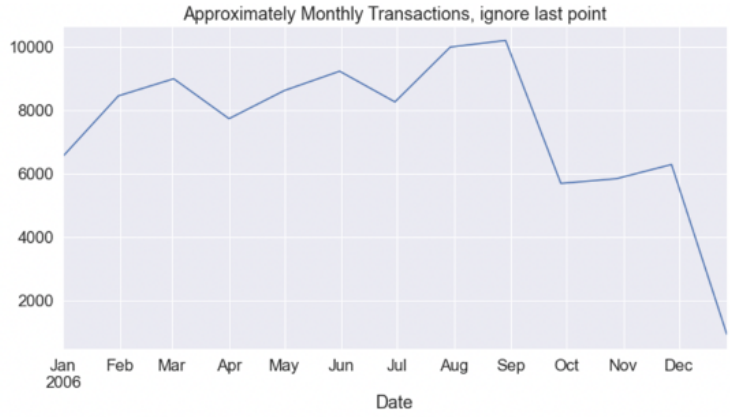
This field represents the date of the transaction with 365 unique values and zero null values. Based on the chart on a level of day of week below, Thursday is the day in which the most of the transactions happen, followed by Wednesday and Friday, while Sunday has the least transactions.



Based on the chart on a level of date across the whole year, 2006-2-28 has the most transactions.

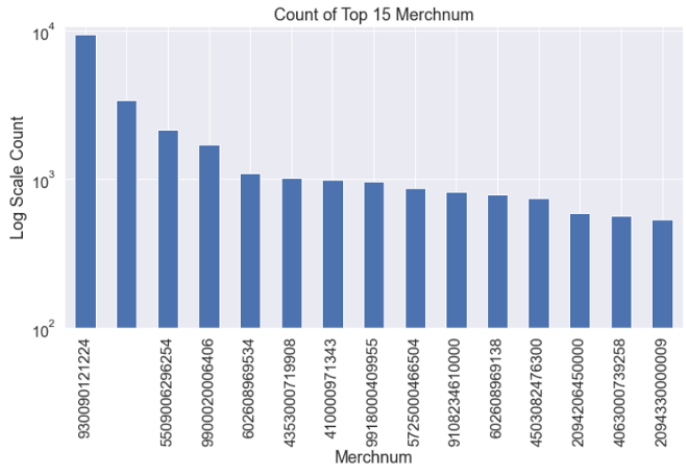



Based on the chart on a level of month, September has the most transactions.



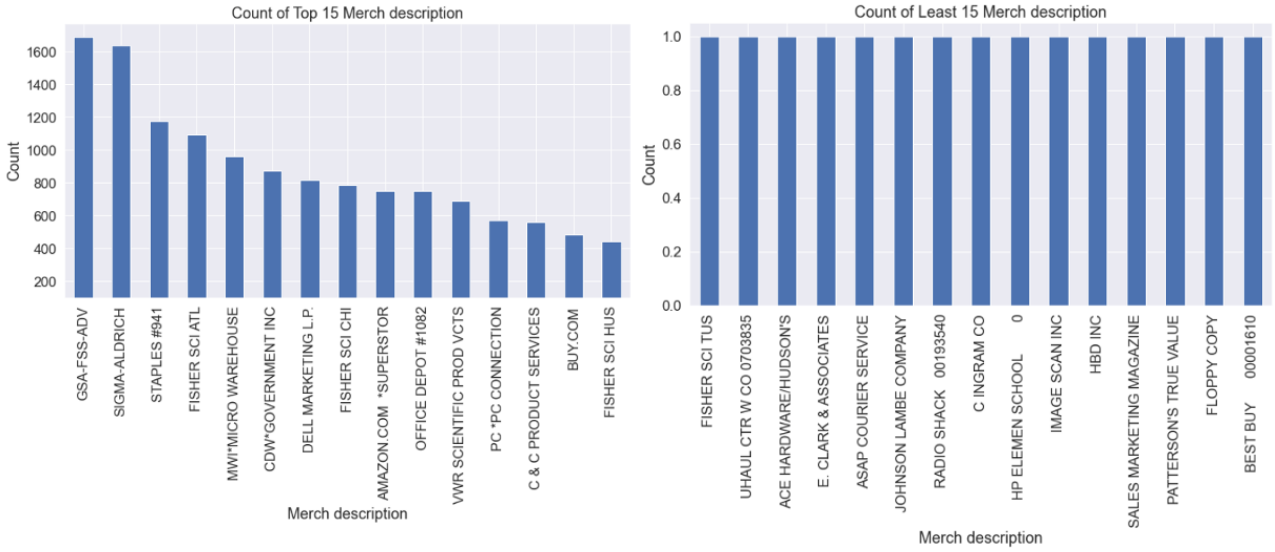

4. Merchnum

This field represents the number of the merchandise in that transaction with 13, 092 unique values and zero missing values. In addition, there are 3,375 transactions that have no information in Merchnum. Furthermore, the merchant number of 930090121224 has the most transactions.




5. Merch description:

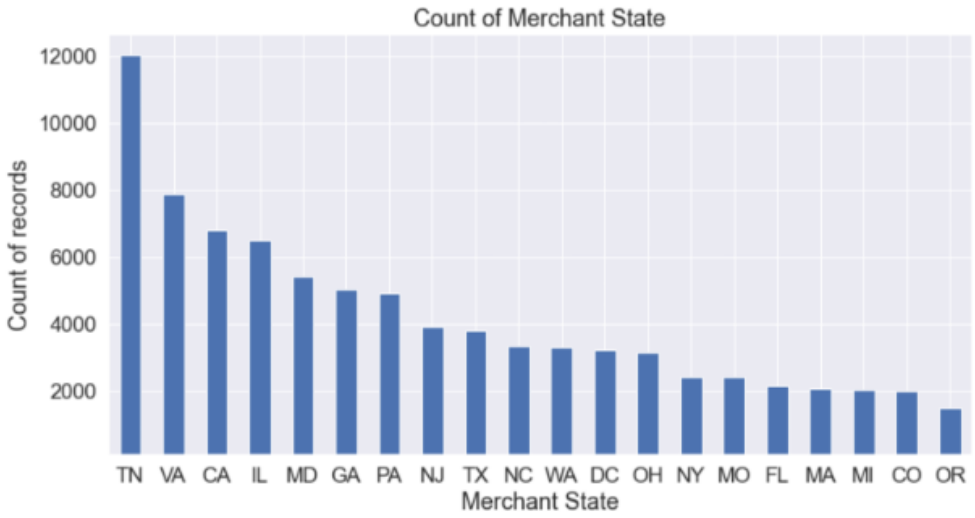
This field represents description of the merchandise purchased in the transaction with 13,126 unique values and zero missing values. Based on the chart below, *GSA-FSS-ADV*, *SIGMA-ALDRICH, STAPLES #941*, and *FISHER SCI ATL* have more than 1000 purchased counts. In addition, there are also merchandise purchased only once.




6. Merch state:

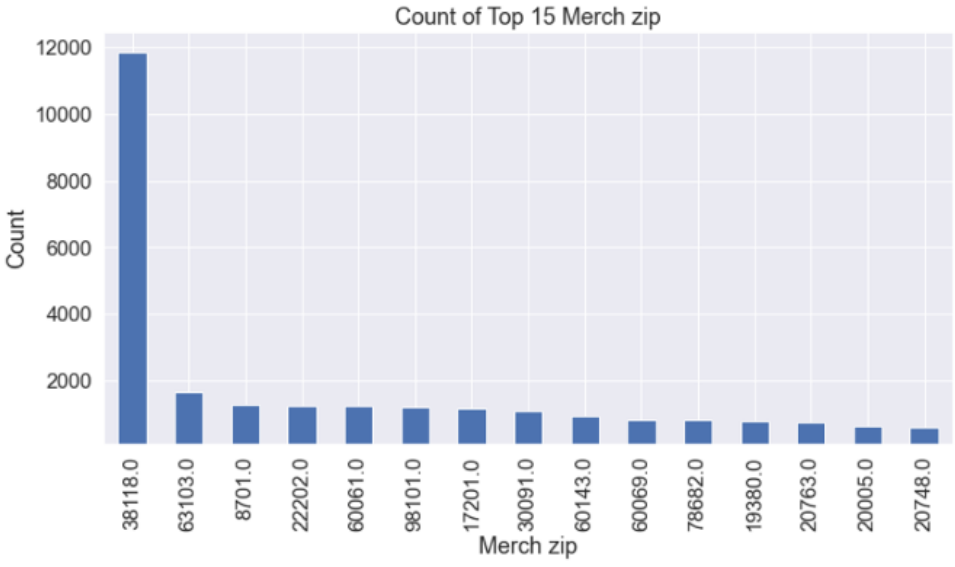
This field represents the state of the transaction that occurred with 228 unique values, which can be weird because there are only 50 states in the U.S.A. In addition, there are more than 100 transactions that have wrong state information. Furthermore, there are 1,195 rows that have

missing value. Lastly, shows that TN has the most merchants, VA ranks the second, followed by CA.



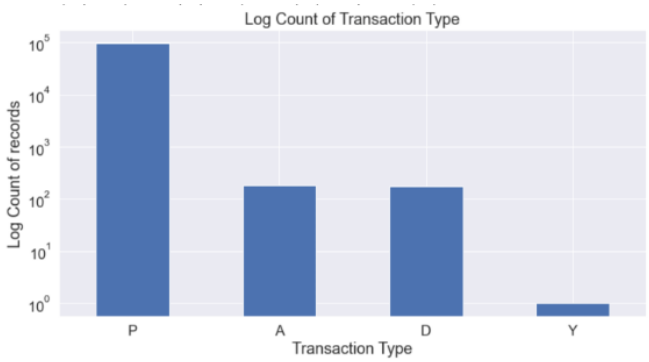

7. Merch zip:

This field represents the zip code of the transaction with 4,568 unique values and around 4% missing values. The chart below shows that zip code 38118 is the place where the most merchants are located.




8. Transtype:

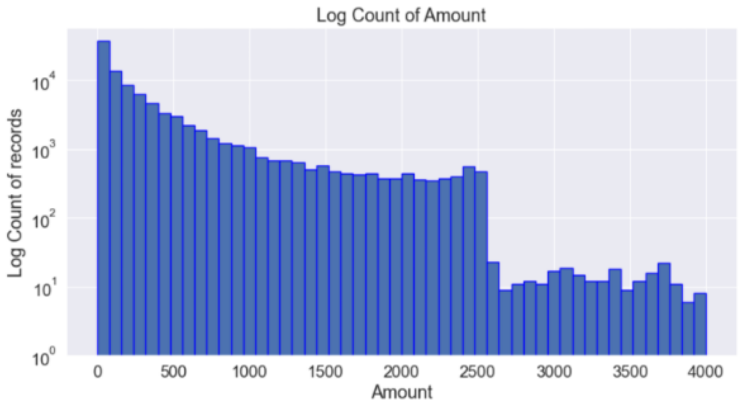
This field represents transaction type with 4 unique classes of P, A, D, Y.




| Class of Transtype | Value Counts |
|---|---|
| P | 96, 398 |
| A | 181 |
| D | 173 |
| Y | 1 |

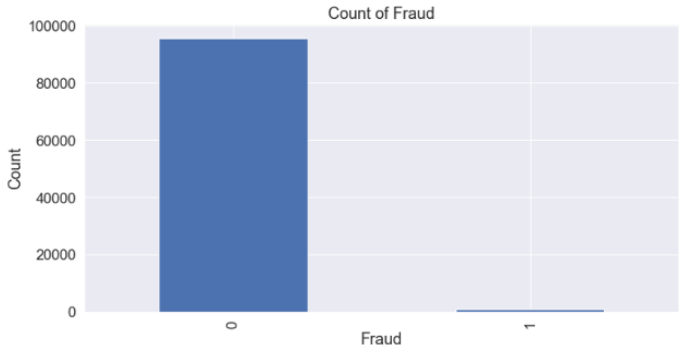
9. Amount:

This field represents the amount of transactions with 34,909 unique values and zero missing value. Based on the chart, most of the amounts are less than 2500. Furthermore, there are 37,586 rows in the first bin, with the amount of $0- $80 and 83 rows have the amount less than $1.




10. Fraud:

This field represents whether it's a fraud transaction or not with 2 classes of 0 and 1. Most of the records (99%) are labeled as not fraud, only 1% records are labeled as fraud.




```
data['Fraud'].value_counts()
```

```
0    95694
1     1059
Name: Fraud, dtype: int64
```

## Appendix 2 – List of Variable Names (examples)

'Recnum', 'Fraud', 'Dow_Risk', 'state_risk', 'Merchnum_desc_State', 'Merchnum_desc_Zip',
'benford_Cardnum', 'benford_Merchnum', 'Cardnum_day_since', 'Cardnum_count_0',
'Cardnum_avg_0', 'Cardnum_max_0', 'Cardnum_med_0', 'Cardnum_total_0',
'Cardnum_actual/avg_0', 'Cardnum_actual/max_0', 'Cardnum_actual/med_0',
'Cardnum_actual/toal_0', 'Cardnum_count_1', 'Cardnum_avg_1', 'Cardnum_max_1',
'Cardnum_med_1', 'Cardnum_total_1', 'Cardnum_actual/avg_1', 'Cardnum_actual/max_1',
'Cardnum_actual/med_1', 'Cardnum_actual/toal_1', 'Cardnum_count_3', 'Cardnum_avg_3',
'Cardnum_max_3', 'Cardnum_med_3', 'Cardnum_total_3', 'Cardnum_actual/avg_3',
'Cardnum_actual/max_3', 'Cardnum_actual/med_3', 'Cardnum_actual/toal_3',
'Cardnum_count_7', 'Cardnum_avg_7', 'Cardnum_max_7', 'Cardnum_med_7',
'Cardnum_total_7', 'Cardnum_actual/avg_7', 'Cardnum_actual/max_7', 'Cardnum_actual/med_7',
'Cardnum_actual/toal_7','Cardnum_count_14', 'Cardnum_avg_14', 'Cardnum_max_14',
'Cardnum_med_14', 'Cardnum_total_14', 'Cardnum_actual/avg_14', 'Cardnum_actual/max_14',
'Cardnum_actual/med_14', 'Cardnum_actual/toal_14', 'Cardnum_count_30',
'Cardnum_avg_30', 'Cardnum_max_30', 'Cardnum_med_30', 'Cardnum_total_30',
'Cardnum_actual/avg_30', 'Cardnum_actual/max_30', 'Cardnum_actual/med_30',
'Cardnum_actual/toal_30', 'Cardnum_count_60','Cardnum_avg_60', 'Cardnum_max_60',
'Cardnum_med_60', 'Cardnum_total_60', 'Cardnum_actual/avg_60', 'Cardnum_actual/max_60',
'Cardnum_actual/med_60', 'Cardnum_actual/toal_60', 'Merchnum_day_since',
'Merchnum_count_0', 'Merchnum_avg_0', 'Merchnum_max_0', 'Merchnum_med_0',
'Merchnum_total_0', 'Merchnum_actual/avg_0', 'Merchnum_actual/max_0',
'Merchnum_actual/med_0', 'Merchnum_actual/toal_0', 'Merchnum_count_1', 'Merchnum_avg_1',
'Merchnum_max_1', 'Merchnum_med_1', 'Merchnum_total_1', 'Merchnum_actual/avg_1',
'Merchnum_actual/max_1', 'Merchnum_actual/med_1', 'Merchnum_actual/toal_1',
'Merchnum_count_3', 'Merchnum_avg_3', 'Merchnum_max_3', 'Merchnum_med_3',
'Merchnum_total_3', 'Merchnum_actual/avg_3', 'Merchnum_actual/max_3',
'Merchnum_actual/med_3', 'Merchnum_actual/toal_3', 'Merchnum_count_7', 'Merchnum_avg_7',
'Merchnum_max_7', 'Merchnum_med_7', 'Merchnum_total_7', 'Merchnum_actual/avg_7',
'Merchnum_actual/max_7', 'Merchnum_actual/med_7', 'Merchnum_actual/toal_7',
'Merchnum_count_14', 'Merchnum_avg_14', 'Merchnum_max_14', 'Merchnum_med_14',
'Merchnum_total_14', 'Merchnum_actual/avg_14', 'Merchnum_actual/max_14',
'Merchnum_actual/med_14', 'Merchnum_actual/toal_14', 'Merchnum_count_30',
'Merchnum_avg_30','Merchnum_max_30','Merchnum_med_30','Merchnum_total_30','Merchnu
m_actual/avg_30','Merchnum_actual/max_30', 'Merchnum_actual/med_30',
'Merchnum_actual/toal_30', 'Merchnum_count_60', 'Merchnum_avg_60', 'Merchnum_max_60',
'Merchnum_med_60', 'Merchnum_total_60', 'Merchnum_actual/avg_60',
'Merchnum_actual/max_60', 'Merchnum_actual/med_60', 'Merchnum_actual/toal_60',
'card_merch_day_since', 'card_merch_count_0', 'card_merch_avg_0', 'card_merch_max_0',
'card_merch_med_0', 'card_merch_total_0', 'card_merch_actual/avg_0',

'card_merch_actual/max_0', 'card_merch_actual/med_0', 'card_merch_actual/toal_0',
'card_merch_count_1', 'card_merch_avg_1', 'card_merch_max_1', 'card_merch_med_1',
'card_merch_total_1', 'card_merch_actual/avg_1', 'card_merch_actual/max_1',
'card_merch_actual/med_1', 'card_merch_actual/toal_1', 'card_merch_count_3',
'card_merch_avg_3', 'card_merch_max_3', 'card_merch_med_3', 'card_merch_total_3',
'card_merch_actual/avg_3', 'card_merch_actual/max_3', 'card_merch_actual/med_3',
'card_merch_actual/toal_3', 'card_merch_count_7', 'card_merch_avg_7', 'card_merch_max_7',
'card_merch_med_7', 'card_merch_total_7', 'card_merch_actual/avg_7',
'card_merch_actual/max_7', 'card_merch_actual/med_7', 'card_merch_actual/toal_7',
'card_merch_count_14', 'card_merch_avg_14', 'card_merch_max_14', 'card_merch_med_14',
'card_merch_total_14', 'card_merch_actual/avg_14', 'card_merch_actual/max_14',
'card_merch_actual/med_14', 'card_merch_actual/toal_14', 'card_merch_count_30',
'card_merch_avg_30', 'card_merch_max_30', 'card_merch_med_30', 'card_merch_total_30',
'card_merch_actual/avg_30', 'card_merch_actual/max_30', 'card_merch_actual/med_30',
'card_merch_actual/toal_30', 'card_merch_count_60', 'card_merch_avg_60',
'card_merch_max_60', 'card_merch_med_60', 'card_merch_total_60',
'card_merch_actual/avg_60', 'card_merch_actual/max_60', 'card_merch_actual/med_60',
'card_merch_actual/toal_60', 'card_zip_day_since', 'card_zip_count_0', 'card_zip_avg_0',
'card_zip_max_0', 'card_zip_med_0', 'card_zip_total_0', 'card_zip_actual/avg_0',
'card_zip_actual/max_0', 'card_zip_actual/med_0', 'card_zip_actual/toal_0', 'card_zip_count_1',
'card_zip_avg_1', 'card_zip_max_1', 'card_zip_med_1', 'card_zip_total_1',
'card_zip_actual/avg_1', 'card_zip_actual/max_1', 'card_zip_actual/med_1',
'card_zip_actual/toal_1', 'card_zip_count_3', 'card_zip_avg_3', 'card_zip_max_3',
'card_zip_med_3', 'card_zip_total_3', 'card_zip_actual/avg_3', 'card_zip_actual/max_3',
'card_zip_actual/med_3', 'card_zip_actual/toal_3', 'card_zip_count_7', 'card_zip_avg_7',
'card_zip_max_7', 'card_zip_med_7', 'card_zip_total_7', 'card_zip_actual/avg_7',
'card_zip_actual/max_7', 'card_zip_actual/med_7', 'card_zip_actual/toal_7', 'card_zip_count_14',
'card_zip_avg_14', 'card_zip_max_14', 'card_zip_med_14', 'card_zip_total_14',
'card_zip_actual/avg_14', 'card_zip_actual/max_14', 'card_zip_actual/med_14',
'card_zip_actual/toal_14', 'card_zip_count_30', 'card_zip_avg_30', 'card_zip_max_30',
'card_zip_med_30', 'card_zip_total_30', 'card_zip_actual/avg_30', 'card_zip_actual/max_30',
'card_zip_actual/med_30', 'card_zip_actual/toal_30', 'card_zip_count_60', 'card_zip_avg_60',
'card_zip_max_60', 'card_zip_med_60', 'card_zip_total_60',
'card_zip_actual/avg_60', 'card_zip_actual/max_60', 'card_zip_actual/med_60',
'card_zip_actual/toal_60', 'card_state_day_since', 'card_state_count_0', 'card_state_avg_0',
'card_state_max_0', 'card_state_med_0', 'card_state_total_0', 'card_state_actual/avg_0',
'card_state_actual/max_0', 'card_state_actual/med_0', 'card_state_actual/toal_0',
'card_state_count_1', 'card_state_avg_1', 'card_state_max_1', 'card_state_med_1',
'card_state_total_1', 'card_state_actual/avg_1', 'card_state_actual/max_1',
'card_state_actual/med_1', 'card_state_actual/toal_1', 'card_state_count_3', 'card_state_avg_3',

'card_state_max_3', 'card_state_med_3','card_state_total_3','card_state_actual/avg_3',
'card_state_actual/max_3', 'card_state_actual/med_3', 'card_state_actual/toal_3',
'card_state_count_7', 'card_state_avg_7','card_state_max_7', 'card_state_med_7',
'card_state_total_7', 'card_state_actual/avg_7', 'card_state_actual/max_7',
'card_state_actual/med_7', 'card_state_actual/toal_7', 'card_state_count_14',
'card_state_avg_14', 'card_state_max_14', 'card_state_med_14',
'card_state_total_14', 'card_state_actual/avg_14', 'card_state_actual/max_14',
'card_state_actual/med_14', 'card_state_actual/toal_14',
'card_state_count_30', 'card_state_avg_30', 'card_state_max_30', 'card_state_med_30',
'card_state_total_30', 'card_state_actual/avg_30', 'card_state_actual/max_30',
'card_state_actual/med_30', 'card_state_actual/toal_30', 'card_state_count_60',
'card_state_avg_60', 'card_state_max_60', 'card_state_med_60', 'card_state_total_60',
'card_state_actual/avg_60', 'card_state_actual/max_60', 'card_state_actual/med_60',
'card_state_actual/toal_60', 'merch_zip_day_since', 'merch_zip_count_0', 'merch_zip_avg_0',
'merch_zip_max_0', 'merch_zip_med_0', 'merch_zip_total_0', 'merch_zip_actual/avg_0',
'merch_zip_actual/max_0', 'merch_zip_actual/med_0', 'merch_zip_actual/toal_0',
'merch_zip_count_1', 'merch_zip_avg_1', 'merch_zip_max_1', 'merch_zip_med_1',
'merch_zip_total_1', 'merch_zip_actual/avg_1', 'merch_zip_actual/max_1',
'merch_zip_actual/med_1', 'merch_zip_actual/toal_1', 'merch_zip_count_3',
'merch_zip_avg_3', 'merch_zip_max_3', 'merch_zip_med_3','merch_zip_total_3',
'merch_zip_actual/avg_3', 'merch_zip_actual/max_3', 'merch_zip_actual/med_3',
'merch_zip_actual/toal_3', 'merch_zip_count_7', 'merch_zip_avg_7',
'merch_zip_max_7', 'merch_zip_med_7', 'merch_zip_total_7',
'merch_zip_actual/avg_7', 'merch_zip_actual/max_7', 'merch_zip_actual/med_7',
'merch_zip_actual/toal_7', 'merch_zip_count_14', 'merch_zip_avg_14', 'merch_zip_max_14',
'merch_zip_med_14', 'merch_zip_total_14', 'merch_zip_actual/avg_14',
'merch_zip_actual/max_14', 'merch_zip_actual/med_14',
'merch_zip_actual/toal_14', 'merch_zip_count_30', 'merch_zip_avg_30', 'merch_zip_max_30',
'merch_zip_med_30', 'merch_zip_total_30', 'merch_zip_actual/avg_30',
'merch_zip_actual/max_30', 'merch_zip_actual/med_30', 'merch_zip_actual/toal_30',
'merch_zip_count_60', 'merch_zip_avg_60', 'merch_zip_max_60', 'merch_zip_med_60',
'merch_zip_total_60', 'merch_zip_actual/avg_60', 'merch_zip_actual/max_60',
'merch_zip_actual/med_60', 'merch_zip_actual/toal_60', 'merch_state_day_since',
'merch_state_count_0', 'merch_state_avg_0', 'merch_state_max_0', 'merch_state_med_0',
'merch_state_total_0', 'merch_state_actual/avg_0', 'merch_state_actual/max_0',
'merch_state_actual/med_0', 'merch_state_actual/toal_0', 'merch_state_count_1',
'merch_state_avg_1', 'merch_state_max_1', 'merch_state_med_1',
'merch_state_total_1', 'merch_state_actual/avg_1', 'merch_state_actual/max_1',
'merch_state_actual/med_1', 'merch_state_actual/toal_1', 'merch_state_count_3',
'merch_state_avg_3', 'merch_state_max_3', 'merch_state_med_3', 'merch_state_total_3',

'merch_state_actual/avg_3', 'merch_state_actual/max_3', 'merch_state_actual/med_3',
'merch_state_actual/toal_3', 'merch_state_count_7', 'merch_state_avg_7', 'merch_state_max_7',
'merch_state_med_7', 'merch_state_total_7', 'merch_state_actual/avg_7',
'merch_state_actual/max_7', 'merch_state_actual/med_7', 'merch_state_actual/toal_7',
'merch_state_count_14', 'merch_state_avg_14', 'merch_state_max_14', 'merch_state_med_14',
'merch_state_total_14','merch_state_actual/avg_14', 'merch_state_actual/max_14',
'merch_state_actual/med_14', 'merch_state_actual/toal_14', 'merch_state_count_30',
'merch_state_avg_30', 'merch_state_max_30', 'merch_state_med_30', 'merch_state_total_30',
'merch_state_actual/avg_30', 'merch_state_actual/max_30', 'merch_state_actual/med_30',
'merch_state_actual/toal_30', 'merch_state_count_60', 'merch_state_avg_60',
'merch_state_max_60', 'merch_state_med_60', 'merch_state_total_60',
'merch_state_actual/avg_60', 'merch_state_actual/max_60', 'merch_state_actual/med_60',
'merch_state_actual/toal_60', 'state_des_day_since', 'state_des_count_0', 'state_des_avg_0',
'state_des_max_0', 'state_des_med_0', 'state_des_total_0', 'state_des_actual/avg_0',
'state_des_actual/max_0', 'state_des_actual/med_0', 'state_des_actual/toal_0',
'state_des_count_1', 'state_des_avg_1', 'state_des_max_1', 'state_des_med_1', 'state_des_total_1',
'state_des_actual/avg_1', 'state_des_actual/max_1', 'state_des_actual/med_1',
'state_des_actual/toal_1', 'state_des_count_3', 'state_des_avg_3', 'state_des_max_3',
'state_des_med_3', 'state_des_total_3', 'state_des_actual/avg_3', 'state_des_actual/max_3',
'state_des_actual/med_3', 'state_des_actual/toal_3', 'state_des_count_7', 'state_des_avg_7',
'state_des_max_7', 'state_des_med_7', 'state_des_total_7', 'state_des_actual/avg_7',
'state_des_actual/max_7', 'state_des_actual/med_7', 'state_des_actual/toal_7',
'state_des_count_14', 'state_des_avg_14', 'state_des_max_14', 'state_des_med_14',
'state_des_total_14', 'state_des_actual/avg_14', 'state_des_actual/max_14',
'state_des_actual/med_14', 'state_des_actual/toal_14', 'state_des_count_30', 'state_des_avg_30',
'state_des_max_30', 'state_des_med_30', 'state_des_total_30', 'state_des_actual/avg_30',
'state_des_actual/max_30', 'state_des_actual/med_30', 'state_des_actual/toal_30',
'state_des_count_60', 'state_des_avg_60', 'state_des_max_60', 'state_des_med_60',
'state_des_total_60', 'state_des_actual/avg_60', 'state_des_actual/max_60',
'state_des_actual/med_60', 'state_des_actual/toal_60', 'state_zip_day_since', 'state_zip_count_0',
'state_zip_avg_0', 'state_zip_max_0', 'state_zip_med_0', 'state_zip_total_0',
'state_zip_actual/avg_0', 'state_zip_actual/max_0', 'state_zip_actual/med_0',
'state_zip_actual/toal_0', 'state_zip_count_1', 'state_zip_avg_1', 'state_zip_max_1',
'state_zip_med_1', 'state_zip_total_1', 'state_zip_actual/avg_1', 'state_zip_actual/max_1',
'state_zip_actual/med_1', 'state_zip_actual/toal_1', 'state_zip_count_3',
'state_zip_avg_3', 'state_zip_max_3', 'state_zip_med_3', 'state_zip_total_3',
'state_zip_actual/avg_3', 'state_zip_actual/max_3', 'state_zip_actual/med_3',
'state_zip_actual/toal_3', 'state_zip_count_7', 'state_zip_avg_7', 'state_zip_max_7',
'state_zip_med_7', 'state_zip_total_7', 'state_zip_actual/avg_7', 'state_zip_actual/max_7',
'state_zip_actual/med_7', 'state_zip_actual/toal_7', 'state_zip_count_14',

'state_zip_avg_14', 'state_zip_max_14', 'state_zip_med_14', 'state_zip_total_14',
'state_zip_actual/avg_14', 'state_zip_actual/max_14', 'state_zip_actual/med_14',
'state_zip_actual/toal_14', 'state_zip_count_30', 'state_zip_avg_30',
'state_zip_max_30', 'state_zip_med_30', 'state_zip_total_30', 'state_zip_actual/avg_30',
'state_zip_actual/max_30', 'state_zip_actual/med_30', 'state_zip_actual/toal_30',
'state_zip_count_60', 'state_zip_avg_60', 'state_zip_max_60', 'state_zip_med_60',
'state_zip_total_60', 'state_zip_actual/avg_60', 'state_zip_actual/max_60',
'state_zip_actual/med_60', 'state_zip_actual/toal_60', 'card_zip3_day_since', 'card_zip3_count_0',
'card_zip3_avg_0', 'card_zip3_max_0', 'card_zip3_med_0', 'card_zip3_total_0',
'card_zip3_actual/avg_0', 'card_zip3_actual/max_0', 'card_zip3_actual/med_0',
'card_zip3_actual/toal_0', 'card_zip3_count_1', 'card_zip3_avg_1', 'card_zip3_max_1',
'card_zip3_med_1', 'card_zip3_total_1', 'card_zip3_actual/avg_1', 'card_zip3_actual/max_1',
'card_zip3_actual/med_1', 'card_zip3_actual/toal_1', 'card_zip3_count_3',
'card_zip3_avg_3', 'card_zip3_max_3', 'card_zip3_med_3', 'card_zip3_total_3',
'card_zip3_actual/avg_3', 'card_zip3_actual/max_3', 'card_zip3_actual/med_3',
'card_zip3_actual/toal_3', 'card_zip3_count_7', 'card_zip3_avg_7',
'card_zip3_max_7', 'card_zip3_med_7', 'card_zip3_total_7', 'card_zip3_actual/avg_7',
'card_zip3_actual/max_7', 'card_zip3_actual/med_7', 'card_zip3_actual/toal_7',
'card_zip3_count_14', 'card_zip3_avg_14', 'card_zip3_max_14', 'card_zip3_med_14',
'card_zip3_total_14', 'card_zip3_actual/avg_14', 'card_zip3_actual/max_14',
'card_zip3_actual/med_14', 'card_zip3_actual/toal_14', 'card_zip3_count_30',
'card_zip3_avg_30', 'card_zip3_max_30', 'card_zip3_med_30', 'card_zip3_total_30',
'card_zip3_actual/avg_30', 'card_zip3_actual/max_30', 'card_zip3_actual/med_30',
'card_zip3_actual/toal_30', 'card_zip3_count_60', 'card_zip3_avg_60', 'card_zip3_max_60',
'card_zip3_med_60', 'card_zip3_total_60', 'card_zip3_actual/avg_60', 'card_zip3_actual/max_60',
'card_zip3_actual/med_60', 'card_zip3_actual/toal_60', 'merchnum_zip_day_since',
'merchnum_zip_count_0', 'merchnum_zip_avg_0', 'merchnum_zip_max_0',
'merchnum_zip_med_0', 'merchnum_zip_total_0', 'merchnum_zip_actual/avg_0',
'merchnum_zip_actual/max_0', 'merchnum_zip_actual/med_0', 'merchnum_zip_actual/toal_0',
'merchnum_zip_count_1', 'merchnum_zip_avg_1', 'merchnum_zip_max_1',
'merchnum_zip_med_1', 'merchnum_zip_total_1', 'merchnum_zip_actual/avg_1',
'merchnum_zip_actual/max_1', 'merchnum_zip_actual/med_1', 'merchnum_zip_actual/toal_1',
'merchnum_zip_count_3', 'merchnum_zip_avg_3', 'merchnum_zip_max_3',
'merchnum_zip_med_3', 'merchnum_zip_total_3', 'merchnum_zip_actual/avg_3',
'merchnum_zip_actual/max_3', 'merchnum_zip_actual/med_3', 'merchnum_zip_actual/toal_3',
'merchnum_zip_count_7', 'merchnum_zip_avg_7',
'merchnum_zip_max_7','merchnum_zip_med_7', 'merchnum_zip_total_7',
'merchnum_zip_actual/avg_7', 'merchnum_zip_actual/max_7', 'merchnum_zip_actual/med_7',
'merchnum_zip_actual/toal_7', 'merchnum_zip_count_14', 'merchnum_zip_avg_14',
'merchnum_zip_max_14', 'merchnum_zip_med_14', 'merchnum_zip_total_14',

'merchnum_zip_actual/avg_14', 'merchnum_zip_actual/max_14',
'merchnum_zip_actual/med_14', 'merchnum_zip_actual/toal_14',
'merchnum_zip_count_30','merchnum_zip_avg_30', 'merchnum_zip_max_30',
'merchnum_zip_med_30', 'merchnum_zip_total_30', 'merchnum_zip_actual/avg_30',
'merchnum_zip_actual/max_30', 'merchnum_zip_actual/med_30',
'merchnum_zip_actual/toal_30', 'merchnum_zip_count_60', 'merchnum_zip_avg_60',
'merchnum_zip_max_60', 'merchnum_zip_med_60', 'merchnum_zip_total_60',
'merchnum_zip_actual/avg_60', 'merchnum_zip_actual/max_60','merchnum_zip_actual/med_60',
'merchnum_zip_actual/toal_60', 'merchnum_zip3_day_since', 'merchnum_zip3_count_0',
'merchnum_zip3_avg_0', 'merchnum_zip3_max_0', 'merchnum_zip3_med_0',
'merchnum_zip3_total_0', 'merchnum_zip3_actual/avg_0', 'merchnum_zip3_actual/max_0',
'merchnum_zip3_actual/med_0', 'merchnum_zip3_actual/toal_0', 'merchnum_zip3_count_1',
'merchnum_zip3_avg_1', 'merchnum_zip3_max_1', 'merchnum_zip3_med_1',
'merchnum_zip3_total_1', 'merchnum_zip3_actual/avg_1', 'merchnum_zip3_actual/max_1',
'merchnum_zip3_actual/med_1', 'merchnum_zip3_actual/toal_1', 'merchnum_zip3_count_3',
'merchnum_zip3_avg_3', 'merchnum_zip3_max_3', 'merchnum_zip3_med_3',
'merchnum_zip3_total_3', 'merchnum_zip3_actual/avg_3', 'merchnum_zip3_actual/max_3',
'merchnum_zip3_actual/med_3', 'merchnum_zip3_actual/toal_3',
'merchnum_zip3_count_7', 'merchnum_zip3_avg_7', 'merchnum_zip3_max_7',
'merchnum_zip3_med_7', 'merchnum_zip3_total_7', 'merchnum_zip3_actual/avg_7',
'merchnum_zip3_actual/max_7', 'merchnum_zip3_actual/med_7',
'merchnum_zip3_actual/toal_7', 'merchnum_zip3_count_14', 'merchnum_zip3_avg_14',
'merchnum_zip3_max_14', 'merchnum_zip3_med_14', 'merchnum_zip3_total_14',
'merchnum_zip3_actual/avg_14','merchnum_zip3_actual/max_14','merchnum_zip3_actual/med_
14', 'merchnum_zip3_actual/toal_14', 'merchnum_zip3_count_30', 'merchnum_zip3_avg_30',
'merchnum_zip3_max_30', 'merchnum_zip3_med_30', 'merchnum_zip3_total_30',
'merchnum_zip3_actual/avg_30', 'merchnum_zip3_actual/max_30',
'merchnum_zip3_actual/med_30', 'merchnum_zip3_actual/toal_30', 'merchnum_zip3_count_60',
'merchnum_zip3_avg_60', 'merchnum_zip3_max_60', 'merchnum_zip3_med_60',
'merchnum_zip3_total_60', 'merchnum_zip3_actual/avg_60', 'merchnum_zip3_actual/max_60',
'merchnum_zip3_actual/med_60', 'merchnum_zip3_actual/toal_60','Card_Merchdesc_day_since',
'Card_Merchdesc_count_0', 'Card_Merchdesc_avg_0', 'Card_Merchdesc_max_0',
'Card_Merchdesc_med_0', 'Card_Merchdesc_total_0', 'Card_Merchdesc_actual/avg_0',
'Card_Merchdesc_actual/max_0', 'Card_Merchdesc_actual/med_0',
'Card_Merchdesc_actual/toal_0', 'Card_Merchdesc_count_1',
'Card_Merchdesc_avg_1', 'Card_Merchdesc_max_1', 'Card_Merchdesc_med_1',
'Card_Merchdesc_total_1', 'Card_Merchdesc_actual/avg_1', 'Card_Merchdesc_actual/max_1',
'Card_Merchdesc_actual/med_1', 'Card_Merchdesc_actual/toal_1', 'Card_Merchdesc_count_3',
'Card_Merchdesc_avg_3', 'Card_Merchdesc_max_3', 'Card_Merchdesc_med_3',
'Card_Merchdesc_total_3', 'Card_Merchdesc_actual/avg_3', 'Card_Merchdesc_actual/max_3',

'Card_Merchdesc_actual/med_3', 'Card_Merchdesc_actual/toal_3', 'Card_Merchdesc_count_7',
'Card_Merchdesc_avg_7', 'Card_Merchdesc_max_7', 'Card_Merchdesc_med_7',
'Card_Merchdesc_total_7', 'Card_Merchdesc_actual/avg_7', 'Card_Merchdesc_actual/max_7',
'Card_Merchdesc_actual/med_7', 'Card_Merchdesc_actual/toal_7', 'Card_Merchdesc_count_14',
'Card_Merchdesc_avg_14', 'Card_Merchdesc_max_14', 'Card_Merchdesc_med_14',
'Card_Merchdesc_total_14', 'Card_Merchdesc_actual/avg_14',
'Card_Merchdesc_actual/max_14', 'Card_Merchdesc_actual/med_14',
'Card_Merchdesc_actual/toal_14', 'Card_Merchdesc_count_30', 'Card_Merchdesc_avg_30',
'Card_Merchdesc_max_30', 'Card_Merchdesc_med_30', 'Card_Merchdesc_total_30',
'Card_Merchdesc_actual/avg_30', 'Card_Merchdesc_actual/max_30',
'Card_Merchdesc_actual/med_30', 'Card_Merchdesc_actual/toal_30',
'Card_Merchdesc_count_60', 'Card_Merchdesc_avg_60', 'Card_Merchdesc_max_60',
'Card_Merchdesc_med_60', 'Card_Merchdesc_total_60', 'Card_Merchdesc_actual/avg_60',
'Card_Merchdesc_actual/max_60', 'Card_Merchdesc_actual/med_60',
'Card_Merchdesc_actual/toal_60', 'Card_dow_day_since', 'Card_dow_count_0',
'Card_dow_avg_0', 'Card_dow_max_0', 'Card_dow_med_0', 'Card_dow_total_0',
'Card_dow_actual/avg_0', 'Card_dow_actual/max_0', 'Card_dow_actual/med_0',
'Card_dow_actual/toal_0', 'Card_dow_count_1', 'Card_dow_avg_1', 'Card_dow_max_1',
'Card_dow_med_1', 'Card_dow_total_1', 'Card_dow_actual/avg_1', 'Card_dow_actual/max_1',
'Card_dow_actual/med_1', 'Card_dow_actual/toal_1', 'Card_dow_count_3', 'Card_dow_avg_3',
'Card_dow_max_3', 'Card_dow_med_3', 'Card_dow_total_3', 'Card_dow_actual/avg_3',
'Card_dow_actual/max_3', 'Card_dow_actual/med_3', 'Card_dow_actual/toal_3',
'Card_dow_count_7', 'Card_dow_avg_7', 'Card_dow_max_7', 'Card_dow_med_7',
'Card_dow_total_7', 'Card_dow_actual/avg_7', 'Card_dow_actual/max_7',
'Card_dow_actual/med_7', 'Card_dow_actual/toal_7', 'Card_dow_count_14',
'Card_dow_avg_14', 'Card_dow_max_14', 'Card_dow_med_14', 'Card_dow_total_14',
'Card_dow_actual/avg_14', 'Card_dow_actual/max_14', 'Card_dow_actual/med_14',
'Card_dow_actual/toal_14', 'Card_dow_count_30', 'Card_dow_avg_30', 'Card_dow_max_30',
'Card_dow_med_30', 'Card_dow_total_30', 'Card_dow_actual/avg_30',
'Card_dow_actual/max_30', 'Card_dow_actual/med_30', 'Card_dow_actual/toal_30',
'Card_dow_count_60', 'Card_dow_avg_60', 'Card_dow_max_60', 'Card_dow_med_60',
'Card_dow_total_60', 'Card_dow_actual/avg_60', 'Card_dow_actual/max_60',
'Card_dow_actual/med_60', 'Card_dow_actual/toal_60', 'Merchnum_desc_day_since',
'Merchnum_desc_count_0', 'Merchnum_desc_avg_0', 'Merchnum_desc_max_0',
'Merchnum_desc_med_0', 'Merchnum_desc_total_0', 'Merchnum_desc_actual/avg_0',
'Merchnum_desc_actual/max_0', 'Merchnum_desc_actual/med_0',
'Merchnum_desc_actual/toal_0', 'Merchnum_desc_count_1', 'Merchnum_desc_avg_1',
'Merchnum_desc_max_1', 'Merchnum_desc_med_1', 'Merchnum_desc_total_1',
'Merchnum_desc_actual/avg_1','Merchnum_desc_actual/max_1','Merchnum_desc_actual/med_1'
,'Merchnum_desc_actual/toal_1', 'Merchnum_desc_count_3', 'Merchnum_desc_avg_3',

'Merchnum_desc_max_3', 'Merchnum_desc_med_3', 'Merchnum_desc_total_3',
'Merchnum_desc_actual/avg_3', 'Merchnum_desc_actual/max_3',
'Merchnum_desc_actual/med_3', 'Merchnum_desc_actual/toal_3', 'Merchnum_desc_count_7',
'Merchnum_desc_avg_7','Merchnum_desc_max_7', 'Merchnum_desc_med_7',
'Merchnum_desc_total_7', 'Merchnum_desc_actual/avg_7', 'Merchnum_desc_actual/max_7',
'Merchnum_desc_actual/med_7', 'Merchnum_desc_actual/toal_7', 'Merchnum_desc_count_14',
'Merchnum_desc_avg_14', 'Merchnum_desc_max_14', 'Merchnum_desc_med_14',
'Merchnum_desc_total_14', 'Merchnum_desc_actual/avg_14', 'Merchnum_desc_actual/max_14',
'Merchnum_desc_actual/med_14', 'Merchnum_desc_actual/toal_14',
'Merchnum_desc_count_30', 'Merchnum_desc_avg_30', 'Merchnum_desc_max_30',
'Merchnum_desc_med_30', 'Merchnum_desc_total_30', 'Merchnum_desc_actual/avg_30',
'Merchnum_desc_actual/max_30', 'Merchnum_desc_actual/med_30',
'Merchnum_desc_actual/toal_30', 'Merchnum_desc_count_60', 'Merchnum_desc_avg_60',
'Merchnum_desc_max_60', 'Merchnum_desc_med_60', 'Merchnum_desc_total_60',
'Merchnum_desc_actual/avg_60', 'Merchnum_desc_actual/max_60',
'Merchnum_desc_actual/med_60', 'Merchnum_desc_actual/toal_60',
'Merchnum_dow_day_since', 'Merchnum_dow_count_0', 'Merchnum_dow_avg_0',
'Merchnum_dow_max_0', 'Merchnum_dow_med_0', 'Merchnum_dow_total_0',
'Merchnum_dow_actual/avg_0', 'Merchnum_dow_actual/max_0',
'Merchnum_dow_actual/med_0', 'Merchnum_dow_actual/toal_0', 'Merchnum_dow_count_1',
'Merchnum_dow_avg_1', 'Merchnum_dow_max_1', 'Merchnum_dow_med_1',
'Merchnum_dow_total_1', 'Merchnum_dow_actual/avg_1', 'Merchnum_dow_actual/max_1',
'Merchnum_dow_actual/med_1', 'Merchnum_dow_actual/toal_1', 'Merchnum_dow_count_3',
'Merchnum_dow_avg_3', 'Merchnum_dow_max_3', 'Merchnum_dow_med_3',
'Merchnum_dow_total_3', 'Merchnum_dow_actual/avg_3', 'Merchnum_dow_actual/max_3',
'Merchnum_dow_actual/med_3', 'Merchnum_dow_actual/toal_3', 'Merchnum_dow_count_7',
'Merchnum_dow_avg_7', 'Merchnum_dow_max_7',
'Merchnum_dow_med_7',
...]