USC Marshall

School of Business

DSO 562 Fraud Analytics Project 3
Credit Card Transaction Fraud

Group 1-5: Qian Huang, Junchu Zhang, Yu Liu, Xinjin Liu, Haoyan Zhang

Date: 05/08/2021

Executive Summary

Financial crime is a crime committed against property, involving the unlawful conversion of the ownership of the property to one's own personal use and benefit. Credit card fraud is one type of financial crime. Credit card fraud is the unauthorized use of a credit or debit card, or similar payment tool to fraudulently obtain money or property. Credit and debit card numbers can be stolen from unsecured websites or can be obtained in an identity theft scheme. Credit card fraud losses are increasing year by year, so it is important for card providers and banks to take time and effort to collaborate with investigators to make credit cards more secure.

In the project, we would like to build supervised fraud models on the card transaction data, and to help identify and deduct credit card fraud, thus reducing the loss from transaction fraud and providing a safer transaction environment. We started the process by first looking at the description and visualization of data. Then we cleaned the null values, outliers and unnecessary records and filled in the missing values in the data. We built more than 1000 candidate variables based on our understanding of credit card transaction fraud and used both filter and wrapper methods to select the best subset of 13 variables based on the contribution to the target of interest. We splitted the data into train, test and out-of-time, built different supervised models by tuning the parameters, and selected the model that yields the best fraud detection rate at 3%. The final best model we chose is Random Forest(*n_estimators*=100, *min_samples_leaf*=10, *min_samples_spilit*=20, *max_deplth*=25) that achieved an average FDR of 64.8% at a 3% rejection threshold on the OOT data. Finally, we provided recommendations on the cut-off percentage of fraud detection for credit card bureaus to put the result in real-life application.

1 Data Exploration

There are 96753 records and 10 variables in the data. There is one numerical variable, “Amount”, one datetime variable, “Date”, and seven categorical variables, Cardnum, Merchnum, Merch description, Merch state, Merch zip, Transtype and Fraud.

1.1 Summary Table(Categorical, Numeric)

Table 1.1.1: Summary of the Numeric variables

	# Records	# Zeros	% Populated	Unique Values	Mean	Standard Deviation	Max	Min
Amount	96753	0	100	34909	427.89	10006.14	3102045.53	0.01

Here it's the summary table of the numerical variable. There is only one numerical variable. There are no missing values in the Amount variable, but there are outliers in the variable. The unique value is 34909, mean is 437.89, standard deviation is 10006.14. The maximum and minimum value have significant differences.

Table 1.1.2: Summary of the Categorical and Datetime variables

	# of Records	Missing Values	% Populated	Unique Values	Most Frequent
Date	96753	0	100	365	2010/02/28 0:00
Merchnum	93378	3375	96.51	13091	930090121224
Merch description	96753	0	100	13126	GSA-FSS-ADV
Merch state	95558	1195	98.76	227	TN
Merch zip	96753	0	100	4568	38118
Recnum	96753	0	100	96753	1

Cardnum	92097	4656	95.19	1645	5142148452
Fraud	96753	0	100	2	0
Transtype	96753	0	100	4	P

Here it's the summary table of categorical variables. There are seven categorical variables. Merchnum, Merch State, and Cardnum all have missing values. We need to operate data cleaning for those missing values in the following steps.

1.2 Field Examples

1.2.1 Field “Amount”

Figure 1.2.1(a) Histogram of Field “Amount” (Amount <= 3,000, Log-Scaled)

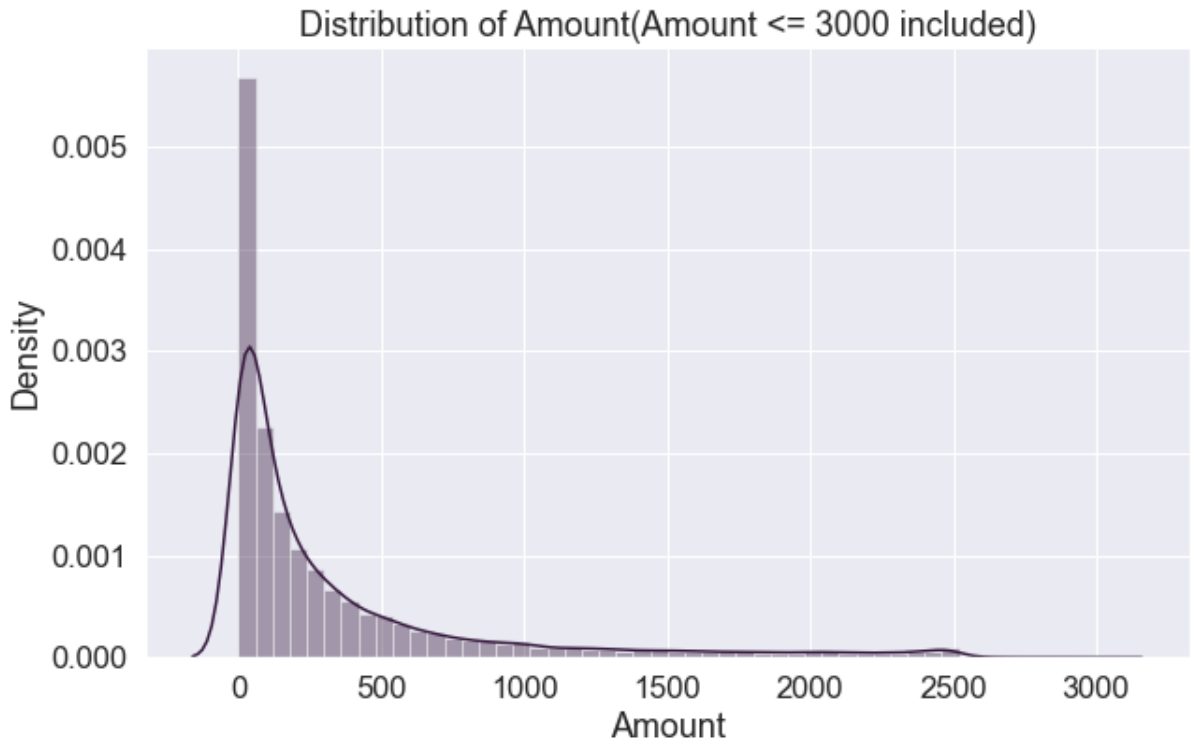


Figure 1.2.1(b) Histogram of Field “Amount”(Amount <= 3,000,000, Log-Scaled)

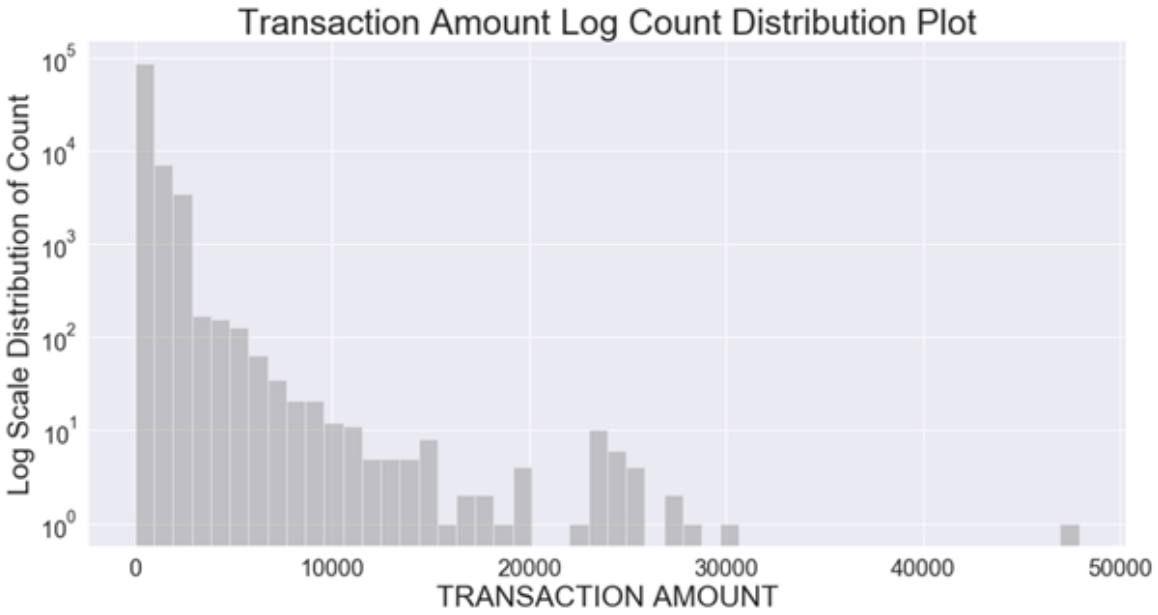


Figure 1.2.1(a) shows the histogram of the numerical field “Amount” (Amount <= 3,000, Log-Scaled), 1.2.1(b) shows a log scale of field “Amount” (Amount <= 3,000,000). Most of the amount values are around 50 dollars, and less than 1000 dollars.

1.2.2 Field “Fraud”

Figure 1.2.2 Distribution of Field “Fraud”

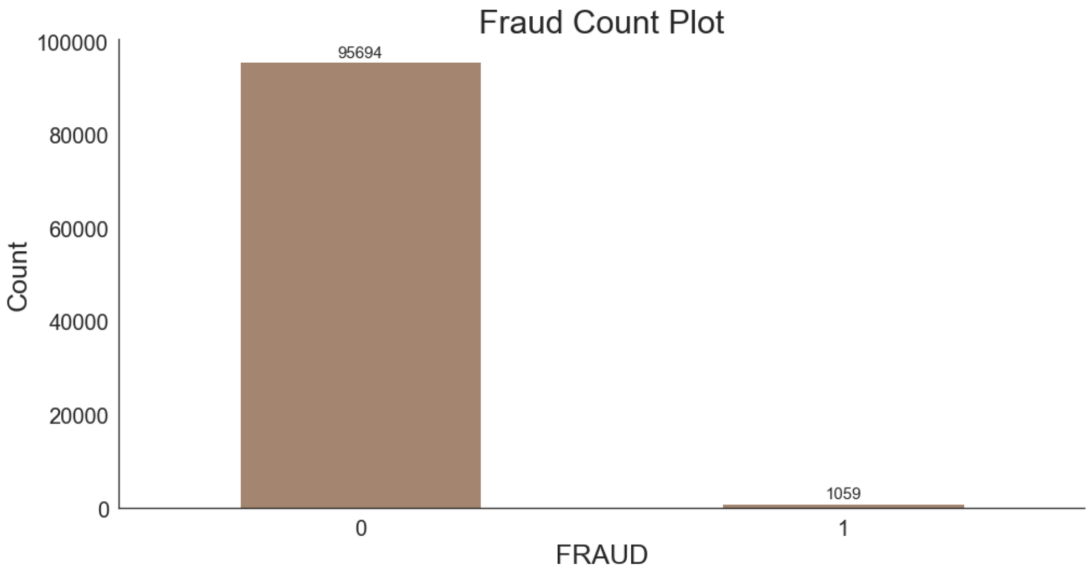


Figure 1.2.2 shows the countplot for the “Fraud” label. The dataset has 96694 non-fraud records, counting for 98.9% of the data, and 1059 fraud records, counting for 1.1% of the

data. The fraud records are unbalanced. Therefore, we need to use 10-folds cross validation to avoid the variation of model results caused by the unbalanced train-test-split.

1.3 Fraud Exploration

1.3.1 Merchant Fraud

Figure 1.3.1 Fraud Distribution of Field “Merchant”

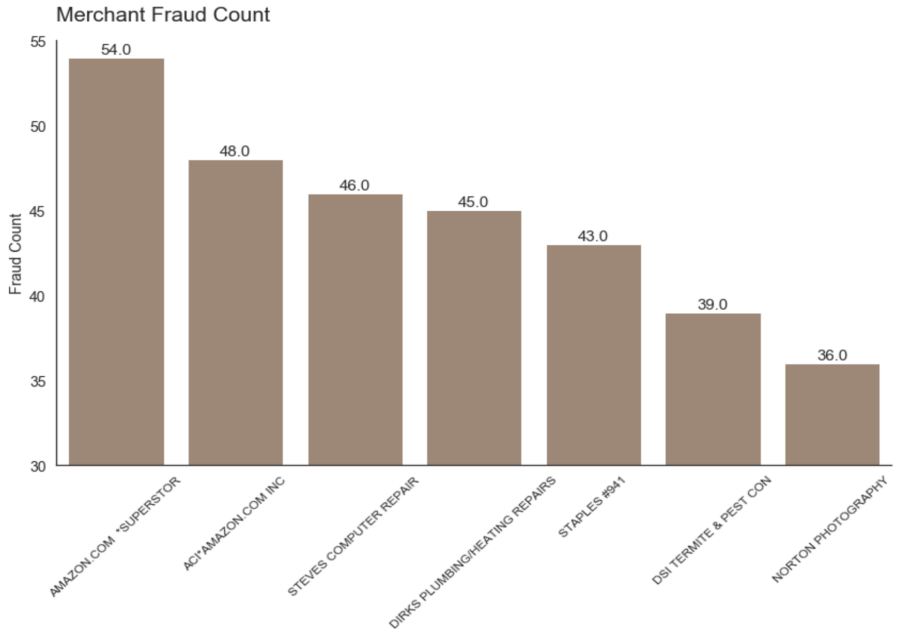


Figure 1.3.1 shows the Fraud distribution plot for the field “Merchant”. There are a large number of fraudsters who are doing transactions at Amazon merchants. “AMAZON.COM *SUPERSTOR” has the highest fraud count of 54.

1.3.2 Cardnum Fraud

Figure 1.3.2 Fraud Distribution of Field “Cardnum”

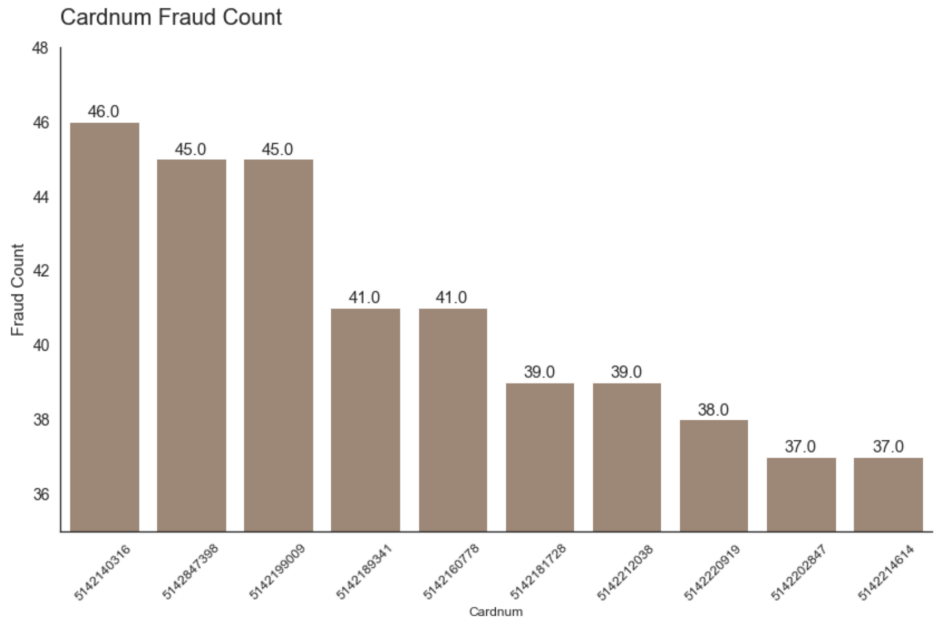


Figure 1.3.2 shows the Fraud distribution plot for the field “Cardnum”. Card number 5142140316 has the highest fraud count of 46.

1.3.3 Merchant State Fraud

Figure 1.3.3 Fraud Distribution of Field “Merchant State”

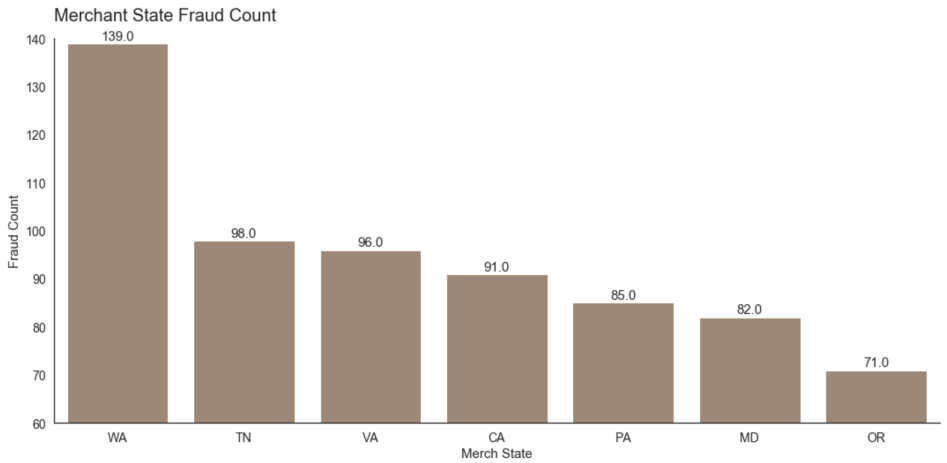


Figure 1.3.3 shows the Fraud distribution plot for the field “Merchant State”. State Washington has the highest fraud count of 139.
The distribution of other fields could be found in the appendix.

2 Data Cleaning

In the original data, only Merchnum, Merch state and Merch zip have missing values. Before we move on to the feature engineering and model building steps, we need to drop out the unnecessary values and fix the missing values.

There is an extreme value, $3,102,045.53 in the Amount field, considered as an outlier. We first removed this value and only kept Transtype “P” because the main focus of analysis is credit card payment. After this step, 356 records are dropped and there are 96397 records in the dataset.

There are 3375 missing Merchnums, 1195 missing Merch states, and 4656 missing Merch zips in the dataset, all of them are related to the Merchant information. If we know some of the merchant information, we may figure out the other merchant related fields. We filled in missing values first based on the existing merchant information and then with the codes we created.

The dataset has 13091 unique Merchnums and 12966 unique Merch descriptions. These two fields could be viewed as a 1-to-1 relationship. Since there are no missing values in the Merch description field, we filled missing Merchnums with Merchnum of the same description and left with 2038 missing Merchnums. For the rest of the missing Merchnums, we filled Merchnum with the previous Merchnum sorted by description + the last two digits of its own description.

Using the similar imputation method, we filled in the missing zip code with the existing zip codes in that state and filled in the missing state with the state of the same zip code. Since there are few missing state records having another record of the same zip code, we also researched missing states with zip code not empty and filled in the exact state of that zip code. Lastly, we filled the rest of the missing states with the first two digits of its own Merch description + “_NaN” and missing zip codes using “0s + Recnum”.

3 Variable Creation

The 9 original fields in the original dataset are insufficient for building a robust fraud prediction model. We want to build as many variables as possible based on our domain knowledge of credit card transaction fraud and use feature selection methods to decide the best variables for model building.

There are many signal activities of transaction fraud, such as burst of activity at different merchants, abnormal purchasing amounts, card usage at different merchants and different geographical locations, purchasing at high-risk merchants, increased usage by card-not-present, intervened transactions, etc. Our goal is to use created features to measure the unusualness of the transaction activities and build models to separate them from regular transactions. The following are the variables we created:

- **Entity Variables**
  There are 4 variables, Cardnum, Merchnum, Merch state, and Merch zip that represent the identity of the transaction entity. We combined two, three and four of these variables together to form a total of 11 entity variables. The more detailed an entity variable is, the more unique it will be to represent a specific entity.
- **Day Since**
  The number of days since an entity was last seen in the data is calculated by finding the last record of the same entity and subtracting the date of last seen from the current record's date. Since credit card fraud risk increases with transaction frequency, the more frequent last same entity was seen, the higher probability it is fraudulent.
- **Frequency**
  The number of times each entity appears over the past 0, 1, 3, 7, 14, and 30 days. Similar to the Day Since variable, this variable also measures the frequency of transaction for a certain entity. The more transactions of the same entity happens in a short period of time, the higher the risk of fraud is.
- **Amount Variables**
  Average, total, actual/average, actual/total and other summary statistics of the Amount variable for each entity in the past n days. We assume that if the transaction amount of an entity is extremely high, it's more likely to be a fraudulent transaction. Thus, we used the summary statistics to represent the unusualness of the transaction amount.
- **Relative Frequency/Amount**
  The ratio of an entity's frequency/amount statistics in the most recent past(past 0 and 1 days) and the same entity's frequency/amount statistics in the past 3, 7, 14, and 30 days. It's very rare for an entity's purchasing habit to change in a short period of time. So we calculated the relative Frequency/Amount variable to compare the change of frequency and amount. If the ratio is farther aways from 1, the entity is more likely to be fraudulent.
- **Fraud Risk for Day of Week**

We extracted Day of Week from the Date variable and calculated the fraud risk for each day of the week. We are curious about whether certain days of the week have a higher fraud risk than the others.

- **Variable Based on Benford's Law**
  Two variables based on Benford's Law are also created to show the unusualness of the transaction amount. Benford's Law indicates that the distribution of the first digit of the transaction amount has a proportional descending pattern from digit 1 to digit 9. The proportion of the transaction amount with first digit 1 and 2 and with 3 and above is around 1.096. If a fraudster is making up the transaction amount number, he or she may not notice this rule and the distribution pattern of the first digit will be different. Thus, we calculated the ratio of a certain card number or merchant number's first digit of transaction amount(First digit 1 and 2/first digit from 3 to 9. If the first digit is 0, find the first non-zero digit). We counted the number of transactions with first digit 1 and 2 as nlow and number of transactions with first digit 3 and above as nhigh. Then, R = 1.096*nlow/nhigh was calculated and U = maximum of R and 1/R was calculated. The smoothed U became the final measurement of Benford's law in our variable.

4 Feature Selection

Having too many features in the data will increase the computational cost of modeling and decrease the accuracy of model performance, so feature selection, which is the process of selecting features that contribute most to the output of interest, is a very important step before model building and model selection.

In this project, filter-based and wrapper-based feature selection methods are both used and eventually 13 final features are selected.

4.1 Preparation for Feature Selection

Before feature selection, the following steps are taken:

I. **Separate Data Set into In-Sample & Out-Sample:** Only use "past" data, the in-sample data that includes January to October for feature selection, leave out-of-time data for model testing

II. **Discard the first two weeks' data:** avoid biased variables created such as *Days Since* because of insufficient previous data records

4.2 Filter

In this project, univariate Kolmogorov-Smirnov (KS) Score and Fraud Detection Rate (FDR) are used to rank all the candidate variables created. Then, the average ranking of these two scores are used to choose top 80 for further feature selection steps.

4.2.1 Univariate Kolmogorov-Smirnov (KS)

For a binary classification, the importance of a variable can be measured by how well it separates bad records from good ones. The figure below shows the curve of bad records against the bad ones. The more different the curves are, the better the variable is to predict the final labels.

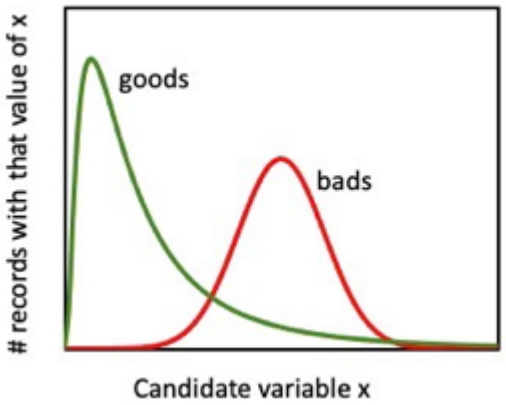


Figure 4.2.1 Illustration of the KS Score

Univariate Kolmogorov-Smirnov (KS) is a statistical measure of how well the two distributions are separated. It calculates the maximum distance between the cumulative goods and bads. The score's formula is shown below:

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

4.2.2 Fraud Detection Rate (FDR)

The second filter-based feature selection method is Fraud Detection Rate (FDR), calculating how much percentage of all the records can be caught at a particular examination cutoff location. For example, FDR 50% at 5% means this predictive model can catch 50% of all the frauds if specific 5% of the population are rejected and in this case, 5% is a rejection rate.

In this project, FDR at 3% is calculated as the percentage of total frauds in the top 3% or bottom 3% records of the total population sorted by the value of variable X.

4.2.3 Average KS and FDR rankings

According to KS and FDR scores, two rankings are given to candidate variables separately. To make the final selection of features more accurate, the average of the two scores is used as the final filter score and the top 80 variables with highest ranking scores are kept.

$$final\ score = \frac{KS\ ranking + FDR\ at\ 3\%\ ranking}{2}$$

4.3 Wrapper

Filter methods often evaluate features individually. In that case, some variables can be useless for prediction in isolation, but they can be quite useful when combined with other variables. To prevent this issue, a wrapper method is used here to select the best feature subsets. Wrapper methods work by evaluating a subset of features using a machine learning algorithm that employs a search strategy to look through the space of possible feature subsets, evaluating each subset based on the quality of the performance of a given algorithm.

In this project, backward feature elimination, a wrapper-type feature selection algorithm, is used. For backward feature selection, the procedure starts with the full set of variables. At each step, it removes the worst variable remaining in the set. To save computation time, the decision tree algorithm is wrapped and FDR score is used as performance metrics.

The final selected 13 features are listed below:

Table 4.3 Final Selected Features

1	Cardnum_zip_sum_Amount_1	The total amount of cardnum in certain zip code in past 1 day
2	Cardnum_state_sum_Amount_3	The total amount of cardnum in certain state in past 3 days
3	Cardnum_state_sum_Amount_7	The total amount of cardnum in certain state in past 7 days
4	Cardnum_state_sum_Amount_1	The total amount of cardnum in certain state in past 1 day
5	Cardnum_state_sum_Amount_0	The total amount of cardnum in certain state in past 0 day
6	Cardnum_Merch_Des_sum_Amount_30	The total amount of cardnum in certain merchant description in past 30 days
7	Cardnum_zip_sum_Amount_30	The total amount of cardnum in certain zip code in past 30 days
8	Cardnum_Merch_Des_sum_Amount_0	The total amount of cardnum in certain merchant description in past 0 day
9	Cardnum_sum_Amount_7	The total amount of cardnum in past 7 days
10	Cardnum_state_max_Amount_14	The maximum amount of cardnum in certain state in past 14 days
11	Cardnum_Merchnum_max_Amount_30	The maximum amount of ardnum of certain merchant in past 30 days
12	num_state_sum_Amount_0	The total amount of merchant in certain state in past 0 day
13	num_state_sum_Amount_1	The total amount of merchant in certain state in past 1 day

5 Model Fitting

5.1 Preparation for Model Fitting

Our selection of modeling data is from January 15th to October 31th, and out-of-time data is from November 1st to December 31st.

Before Model Algorithm, the following steps are taken:

I. **Split data:** We splitted the modeling data into two parts, 80% training data to build the models, and 20% testing data to evaluate the models. .

II. **Cross Validation:** We used modified 10-fold cross validation to evaluate estimator performance.The algorithm runs 10 times total, for each iteration, we randomly splitted the modeling data for training and testing as described in step I. We set the average Out-of-Time FDR at 3% as our performance measurement. For each iteration, we used the train data to construct the model and calculated the Out-of-Time FDR at 3% for train, test and Out-of-Time data. We evaluated the model performance by averaging the score of each iteration.

5.2 Models

We started with a baseline model and increased the model complexity seeking a better performance. For logistic regression models, we got the highest Out-Of-Time FDR at 3% with 42%. Also, We built total five nonlinear models, including three tree ensemble models which are Extreme Gradient Boosting model with 61.3% OOT FDR at 3%, Gradient Boosting Tree with 56.4%, Random Forest with 64.8% which is the highest among all the models, and other two models including Support Vector Machine with 56.3% and Neural Net with 44%. For all the models, we set the parameters to random search to narrow down the range of the best parameter using 10 times random split as described before seeking a better outcome. Then, we adjusted the parameter by hand and finalized our parameters over 10 fold cross validation. The results should be relatively stable after 10 rounds validation. All of our non-linear models perform better than logistic models. The detailed descriptions of each model are illustrated in the following sections.

5.2.1 Logistic

The logistic method is a supervised predictive algorithm using independent variables to predict the dependent variable. It is like Linear Regression, but with a difference that the dependent variable should be categorical variable. Logistic regression is mainly used in classification problems. It uses Logistic function to model the conditional probability.

We used the scikit-learn LogisticRegression package to fit the model with Python. We used 13 candidate variables to fit the model with different hyperparameters including.The following are the groups of parameters we experimented:

Penalty: specify the norm used in the penalization
Solver: Algorithm to use in the optimization problem.
Multi_class: {'auto', 'ovr', 'multinomial'}

We finalized our best model with 13 variables, using L2 Penalty, saga Solver, and Multi_class as auto, which has the highest Out of Time average FDR 0.42 at 3%. (see Table 5.2.1).

Table 5.2.1 Hyper-parameters and Corresponding FDR at 3% of Logistic Regression Models

Model		Parameter			Result		
Logistic	**Number of variables**	**Penalty**	**Solver**	**Multi_class**	**Train**	**Test**	**OOT**
1	13	L2	saga	auto	0.705	0.698	0.420
2	13	none	newton-cg	ovr	0.735	0.737	0.401
3	13	L2	liblinear	ovr	0.717	0.711	0.405
4	13	none	newton-cg	ovr	0.735	0.737	0.385

5.2.2 Neural Network

A neural network is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses. The algorithm maps an input vector to an output scaler, typically a vector of axes into a single dependent variable y, by a series of hidden layers.

We created a deep learning neural network model using MLP Classifier in the sklearn library. We used all 13 candidate variables to fit the model. For each model, we changed the hidden layer sizes, max iteration and batch size parameter seeking for a better performance. The following are the groups of parameters we experimented:

Hidden_layer_sizes: The ith element represents the number of neurons in the ith hidden layer.
Max_iter: Maximum number of iterations.
Learning_rate: Learning rate schedule for weight updates
Batch_size: Size of mini batches for stochastic optimizers

Table 5.2.2 Hyper-parameters and Corresponding FDR at 3% of Neural Network Models

Model		Parameter				Result		
Neural Network	Number of Variable	hidden layer sizes	max_iter	learning_rate	batch_size	Train	Test	OOT
1	13	100,100	60	adaptive	400	0.693	0.65	0.357
2	13	100,100	80	adaptive	600	0.755	0.716	0.372
3	13	200,100	80	adaptive	600	0.782	0.723	0.418
4	13	200,100	100	adaptive	600	0.795	0.747	0.425
5	13	200,150	100	adaptive	600	0.75	0.748	0.44
6	13	200,200	60	adaptive	600	0.789	0.739	0.394
7	13	200,200	100	adaptive	600	0.775	0.724	0.415
8	13	150,150	100	adaptive	600	0.757	0.702	0.412

After we tuned the hyperparameter Hidden Layer Size for combinations of 100, 150 and 200, Maximum Iteration for 60,80, and 100, batch size as 400 or 600, learning rate as “adaptive”, Solver for “adam” as default, Alpha as 0.0001 as default, we finalized the model using hidden layer sizes (200,150), max iteration 100, and batch size 600 with the highest FDR at 3% 0.44 in the Out-of-sample data(see table 5.2.2).

5.2.3 Random Forest

Random forests are a type of ensemble method, meaning using an "average" of many base models to form a single, more accurate model. Each tree is built independently and is a strong, deep tree. Random Forest classifier is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation, jointly referred to as bagging.

We created a deep learning neural network model using Random Forest Classifier in the sklearn library. Our team used all the variables to fit the model, and tried different numbers of trees, minimum sample split size, and minimum sample node sizes.The following are the detailed explanation of the parameter:

Class_weight = Weights associated with classes in the form {class_label: weight}
n_estimators*:* The number of trees in the forest.
max_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
min_samples_split: The minimum number of samples required to split an internal node
min_samples_leaf: The minimum number of samples required to be at a leaf node.

We finalized our best model with n_estimator as 100; maximum depth of the tree as 25; minimum sample split size as 20; minimum sample leaf as 10; n_jobs as -1 as we use all

processors that are parallelized over the trees. The highest FDR at 3% in the Out-of-Sample group is 0.648 (see Table 5.2.3).

Table 5.2.3 Hyper-parameters and Corresponding FDR at 3% of Random Forest Models

Model	Parameter					Result		
Random Forest	Number of variables	Numbers estimators	Min samples leaf	Min samples split	Max depth	Train	Test	OOT
1	13	100	10	20	25	1	0.925	0.648
2	13	100	10	20	50	1	0.923	0.628
3	13	100	10	30	25	1	0.925	0.631
4	13	50	10	20	25	1	0.931	0631
5	13	50	30	60	25	0.996	0.917	0.62
6	13	50	30	60	50	0.998	0.92	0.628
7	13	50	50	100	25	0.974	0.899	0.577

5.2.4 Gradient Boosting Model Using Decision Trees

Boosting is a flexible nonlinear regression procedure that helps improve the accuracy of trees. By sequentially applying weak classification algorithms to the incrementally changed data, a series of decision trees are created that produce an ensemble of weak prediction models.

Our team used GradientBoostingClassifier from sklearn library to implement the model. We used all 13 candidate variables to fit the model with different hyperparameters including different choices on number of trees, maximum depth, and learning rate. Because the learning rate shrinks the contribution of each tree, and there is a trade-off between number of trees and the learning rate, we modified these three parameters to achieve a better model that generates the highest FDR for Out-of-Sample data. The following are the groups of parameters we experiment:

n_estimators: The number of boosting stages to perform
max_depth: The maximum depth of the individual regression estimators.
learing_rate: Boosting learning rate

We finalized our model with n_estimator = 500; maximum depth of tree equals to 2; learning rate as default 0.1; minimum sample leaf as default 1; minimum sample split as default as 2, with the average FDR at 3% of 0.564 for Out-of Sample data (see Table 5.2.4).

Table 5.2.4 Hyper-parameters and Corresponding FDR at 3% of Gradient Boosting Models

Model	Parameter				Result		
Gradient Boosting	Number of variables	n_estimators	max_depth	learning_rate	Train	Test	OOT
1	13	100	2	0.1	0.87	0.857	0.530
2	13	100	5	0.01	0.843	0.845	0.522
3	13	300	2	0.1	0.923	0.896	0.563
4	13	200	5	0.1	0.997	0.939	0.541
5	13	150	3	0.01	0.816	0.795	0.541
6	13	500	3	0.1	0.986	0.939	0.564

5.2.5 Extreme Gradient Boosting Model Using Decision Tree

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting and plits up to the maximum depth specified and prunes the tree backward and removes splits beyond which there is an only negative loss that solves many data science problems in a fast and accurate way.

Our team used XGBoostingClassifier from sklearn library. Total 13 variables were used to fit the model, and we tried different choices on number of boosting rounds, maximum tree depth for base learners, learning rate, and minimum child weight. We tuned different combinations of parameters to achieve a better model that generates the highest FDR for Out-of-Time data. The following are the groups of parameters we experiment:

N_estimators: Number of boosting rounds
Max_depth: Maximum tree depth for base learners
Learing_rate: Boosting learning rate
Min_child_weight: Minimum sum of instance weight(hessian) needed in a child

We finalized our model with numbers of boosting rounds equal to 1300; maximum depth of tree equals to 3; learning rate as 0.04; minimum child weight as 3, which generated the average FDR at 3% of 0.613 for Out-of Sample data (see Table 5.2.4).

Table 5.2.5 Hyper-parameters and Corresponding FDR at 3% of XG Boosting Models

Model		Parameter			Result		
XGBoost	Numbers of variables	Number estimators	Max depth	Learning rate	Train	Test	OOT
1	13	500	5	0.04	0.988	0.941	0.582
2	13	700	3	0.03	0.961	0.904	0.55
3	13	1000	5	0.01	0.975	0.933	0.578
4	13	1000	8	0.02	1	0.945	0.579
5	13	1100	3	0.05	0.971	0.915	0.586
6	13	1250	5	0.04	1	0.955	0.604
7	13	1300	3	0.04	0.99	0.943	0.613

5.2.6 Support Vector Machine

Support-vector machine is an supervised algorithm that plots each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. The algorithm performs classification by finding the hyper-plane that differentiates the two classes very well. The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Support Vectors are the coordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes.

We implemented the model by SVC function from sklearn library, and used all 13 variables to train the model. We modified these parameters to try to achieve a better model performance of FDR for Out-of-Time data. Our team tuned the parameters including gamma, degree and kernel. The following is the detailed explanation of the parameter:
The following are the groups of parameters we experiment:

Gamma: Kernel coefficient
Degree: Degree of the polynomial kernel function
Kernel: Specifies the kernel type to be used in the algorithm

We finalized our model with numbers of gamma equal to auto; degree equals 4; kernel as rbf, which generates the average FDR at 3% of 0.563 for Out-of Sample data (see Table 5.2.6).

Table 5.2.5 Hyper-parameters and Corresponding FDR at 3% of SVM Models

Model		Parameter			Result		
SVM	**Numbers of variables**	**gamma**	**degree**	**kernel**	**Train**	**Test**	**OOT**
1	13	auto	4	rbf	0.76	0.783	0.563
2	13	scale	4	linear	0.703	0.701	0.312
3	13	auto	8	rbf	0.756	0.732	0.539
4	13	auto	3	linear	0.703	0.698	0.312
5	13	auto	6	linear	0.703	0.698	0.316

6 Results

6.1 Summary of Final Best Model Performance

We built all 6 models and tested with 10-fold cross validation. Then we calculated the average FDR at 3% scores for test data, training data and out-of-time data. The general concept is to set a linear model which in this binary classification problem, logistic regression as baseline, and build other non-linear models with simple parameters first, then increase the complexity and tune the hyperparameter for better improvement until we observe over fitting.

Logistic model is the baseline model we built, which has the lowest FDR at 3% scores compared to other models, which is 0.42 for OOT data. The baseline model is still lower than the nonlinear models, that are more complex models and have better performance. The FDR at 3% scores for OOT data are all higher than 0.5.

Among our nonlinear models, we chose random forest as our final model with best OOT performance. The random forest model has the highest FDR at 3% scores for OOT data with 0.648. The random forest model has least standard deviation in test scores among all models, and it also has relatively less difference between train and test performance. Therefore, the random forest model is more stable and stronger compared to other models. That is why we chose the random forest model as our best model for further discussion.

Table 6 Hyper-parameters and average FDR at 3% of Best Neural Network Models

Random Forest	**Number of Variables**	**n_estimater**	**min_samples_leaf**	**min_samples_split**	**max_depth**	**TRAIN FDR at 3%**	**TEST FDR at 3%**	**OOT FDR at 3%**
2	13	100	10	20	25	1	0.925	0.648

The following three tables show the result of our Random Forest model for train data, test data, and out-of-time data. For each dataset, the table shows the number of good data and bad data from the first 1 percent of records to the first 20 percent of records. Also, the cumulative good and cumulative bad, and KS and FPR results.

Train	# Records	# Goods	# Bads	Fraud Rate								
	64505	63811	694	0.010758856								
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goo	Cumulative Bad	% Goods	% Bads (FDR)	KS	FPR
1	645	27	618	4.19%	95.81%	645	27	618	0.04%	89.05%	89.01	0.04
2	645	569	76	88.22%	11.78%	1290	596	694	0.93%	100.00%	99.07	0.86
3	645	645	0	100.00%	0.00%	1935	1241	694	1.95%	100.00%	98.05	1.79
4	645	645	0	100.00%	0.00%	2580	1886	694	2.96%	100.00%	97.04	2.72
5	645	645	0	100.00%	0.00%	3225	2531	694	3.97%	100.00%	96.03	3.65
6	645	645	0	100.00%	0.00%	3870	3176	694	4.98%	100.00%	95.02	4.58
7	645	645	0	100.00%	0.00%	4515	3821	694	5.99%	100.00%	94.01	5.51
8	645	645	0	100.00%	0.00%	5160	4466	694	7.00%	100.00%	93.00	6.44
9	645	645	0	100.00%	0.00%	5805	5111	694	8.01%	100.00%	91.99	7.37
10	645	645	0	100.00%	0.00%	6451	5757	694	9.02%	100.00%	90.98	8.29
11	645	645	0	100.00%	0.00%	7096	6402	694	10.03%	100.00%	89.97	9.22
12	645	645	0	100.00%	0.00%	7741	7047	694	11.04%	100.00%	88.96	10.15
13	645	645	0	100.00%	0.00%	8386	7692	694	12.05%	100.00%	87.95	11.08
14	645	645	0	100.00%	0.00%	9031	8337	694	13.06%	100.00%	86.94	12.01
15	645	645	0	100.00%	0.00%	9676	8982	694	14.08%	100.00%	85.92	12.94
16	645	645	0	100.00%	0.00%	10321	9627	694	15.09%	100.00%	84.91	13.87
17	645	645	0	100.00%	0.00%	10966	10272	694	16.10%	100.00%	83.90	14.80
18	645	645	0	100.00%	0.00%	11611	10917	694	17.11%	100.00%	82.89	15.73
19	645	645	0	100.00%	0.00%	12256	11562	694	18.12%	100.00%	81.88	16.66
20	645	645	0	100.00%	0.00%	12901	12207	694	19.13%	100.00%	80.87	17.59

Testing	# Records	# Goods	# Bads	Fraud Rate								
	16127	15953	174	0.010789359								
	Bin Statistics											
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goods	Cumulative Bads	% Goods	% Bads (FDR)	KS	FPR
1	161	27	134	16.91%	83.09%	161	27	134	0.17%	77.01%	76.84	0.20
2	161	143	18	88.84%	11.16%	323	171	152	1.07%	87.36%	86.29	1.12
3	161	152	9	94.42%	5.58%	484	323	161	2.02%	92.53%	90.51	2.01
4	161	161	0	100.00%	0.00%	645	484	161	3.03%	92.53%	89.49	3.01
5	161	158	3	98.14%	1.86%	806	642	164	4.03%	94.25%	90.23	3.92
6	161	161	0	100.00%	0.00%	968	804	164	5.04%	94.25%	89.22	4.90
7	161	161	0	100.00%	0.00%	1129	965	164	6.05%	94.25%	88.20	5.88
8	161	160	1	99.38%	0.62%	1290	1125	165	7.05%	94.83%	87.77	6.82
9	161	160	1	99.38%	0.62%	1451	1285	166	8.06%	95.40%	87.34	7.74
10	161	161	0	100.00%	0.00%	1613	1447	166	9.07%	95.40%	86.33	8.72
11	161	159	2	98.76%	1.24%	1774	1606	168	10.07%	96.55%	86.48	9.56
12	161	161	0	100.00%	0.00%	1935	1767	168	11.08%	96.55%	85.47	10.52
13	161	161	0	100.00%	0.00%	2097	1929	168	12.09%	96.55%	84.46	11.48
14	161	161	0	100.00%	0.00%	2258	2090	168	13.10%	96.55%	83.45	12.44
15	161	161	0	100.00%	0.00%	2419	2251	168	14.11%	96.55%	82.44	13.40
16	161	160	1	99.38%	0.62%	2580	2411	169	15.12%	97.13%	82.01	14.27
17	161	161	0	100.00%	0.00%	2742	2573	169	16.13%	97.13%	81.00	15.22
18	161	161	0	100.00%	0.00%	2903	2734	169	17.14%	97.13%	79.99	16.18
19	161	161	0	100.00%	0.00%	3064	2895	169	18.15%	97.13%	78.98	17.13
20	161	161	0	100.00%	0.00%	3225	3056	169	19.16%	97.13%	77.97	18.09

OOT	# Records	# Goods	# Bads	Fraud Rate								
	12427	12248	179	0.01440412								
Population Bin	# Records	# Goods	# Bads	% Goods	% Bads	Total # Records	Cumulative Goo	Cumulative Bad	% Goods	% Bads (FDR)	KS	FPR
1	124	53	71	42.87%	57.13%	124	53	71	0.43%	39.66%	39.23	0.75
2	124	96	28	77.47%	22.53%	249	150	99	1.22%	55.31%	54.09	1.51
3	124	107	17	86.32%	13.68%	373	257	116	2.10%	64.80%	62.71	2.21
4	124	112	12	90.34%	9.66%	497	369	128	3.01%	71.51%	68.49	2.88
5	124	119	5	95.98%	4.02%	621	488	133	3.99%	74.30%	70.31	3.67
6	124	121	3	97.59%	2.41%	746	610	136	4.98%	75.98%	71.00	4.48
7	124	118	6	95.17%	4.83%	870	728	142	5.94%	79.33%	73.39	5.13
8	124	122	2	98.39%	1.61%	994	850	144	6.94%	80.45%	73.51	5.90
9	124	124	0	100.00%	0.00%	1118	974	144	7.96%	80.45%	72.49	6.77
10	124	124	0	100.00%	0.00%	1243	1099	144	8.97%	80.45%	71.48	7.63
11	124	123	1	99.20%	0.80%	1367	1222	145	9.98%	81.01%	71.03	8.43
12	124	123	1	99.20%	0.80%	1491	1345	146	10.98%	81.56%	70.58	9.21
13	124	123	1	99.20%	0.80%	1616	1469	147	11.99%	82.12%	70.13	9.99
14	124	122	2	98.39%	1.61%	1740	1591	149	12.99%	83.24%	70.25	10.68
15	124	123	1	99.20%	0.80%	1864	1714	150	13.99%	83.80%	69.80	11.43
16	124	122	2	98.39%	1.61%	1988	1836	152	14.99%	84.92%	69.92	12.08
17	124	123	1	99.20%	0.80%	2113	1960	153	16.00%	85.47%	69.48	12.81
18	124	123	1	99.20%	0.80%	2237	2083	154	17.01%	86.03%	69.03	13.53
19	124	124	0	100.00%	0.00%	2361	2207	154	18.02%	86.03%	68.01	14.33
20	124	124	0	100.00%	0.00%	2485	2331	154	19.03%	86.03%	67.00	15.14

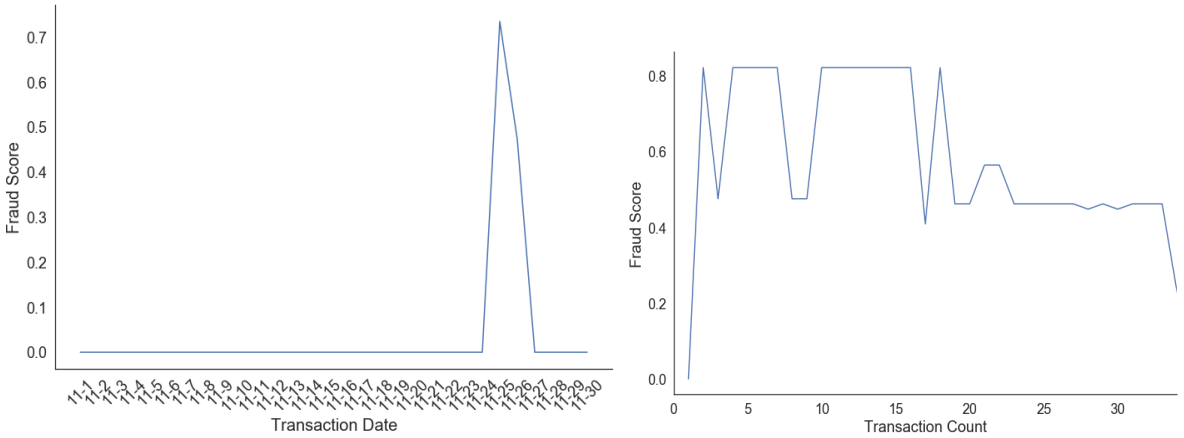
7 Insights & Recommendations

7.1 Insights on Example Cardnum and Merchnum

7.1.1 Cardnum Example

Cardnum = 5142235211

Figure 7.1.1(a), 7.1.1(b) Fraud Score Plot of "Cardnum" 5142235211 by Date and Count

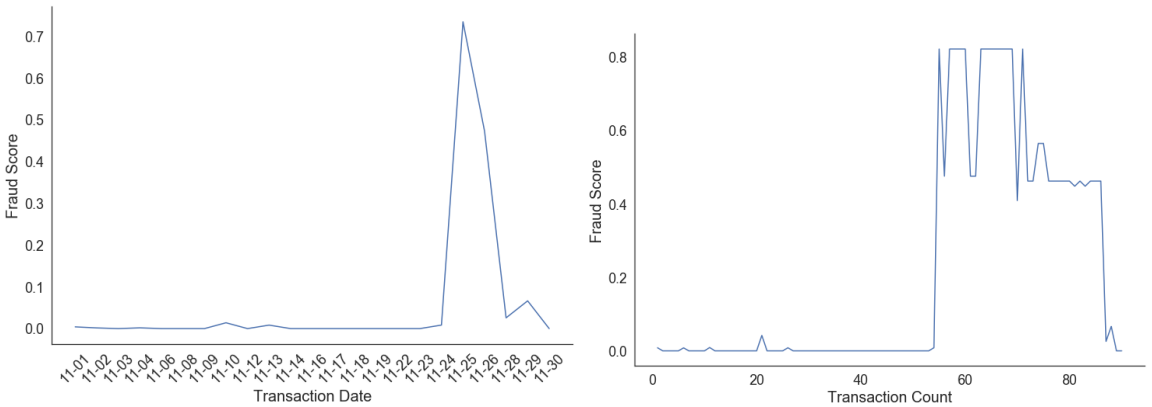


There were about 34 transactions in the month of November. Going by the transaction date, there is a peak on 25th & 26th, which is an abnormal behavior. Similarly, we can see how the fraud score fluctuates for all these amounts transactions between November 25th to 26th.

7.1.2 Merchnum Example

Merchnum = 4353000719908

Figure 7.1.2(a), 7.1.2(b) Score Plot of "Merchnum" 4353000719908 by Date and Count

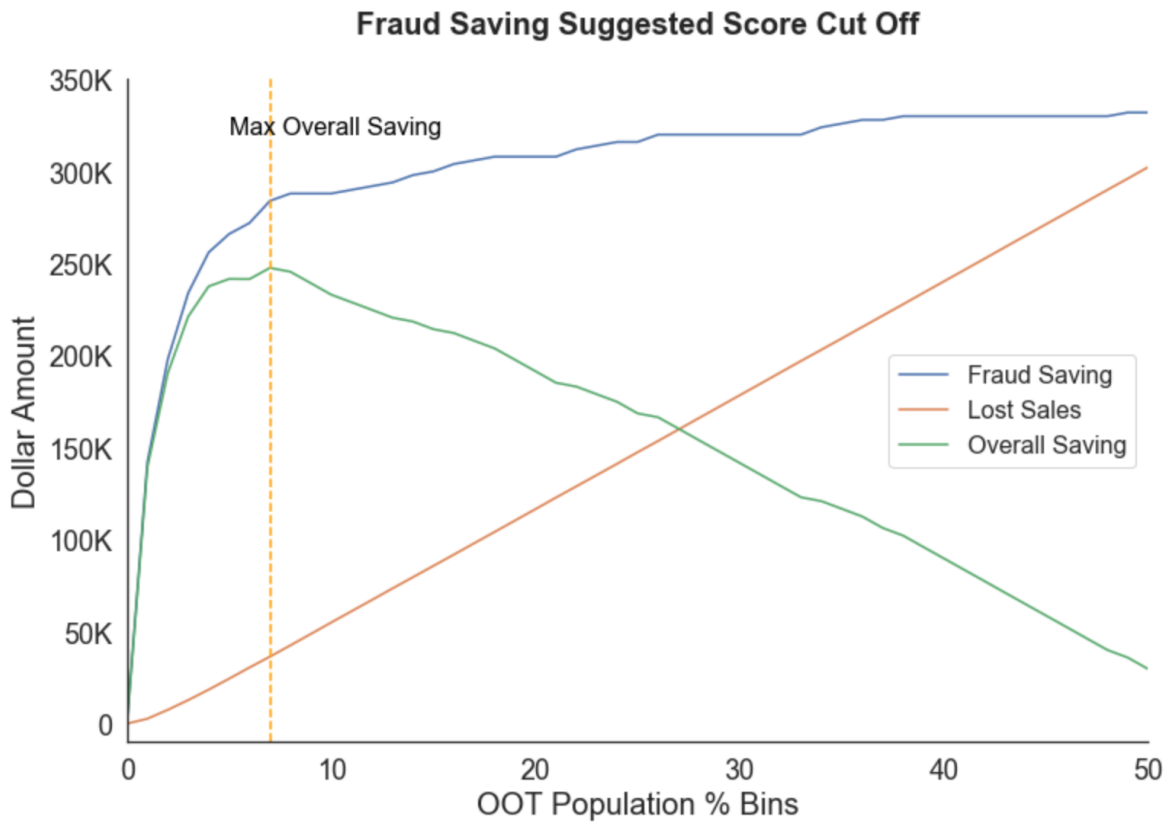


There were about 90 transactions in the month of November. Going by the transaction date, there is a peak also on 25th & 26th, which is an abnormal behavior. Similarly, we can see how the fraud score fluctuates for all these amounts transactions between November 25th to 26th.

7.2 Recommendations

For every new transaction that comes in, we would score them by our model and seek a cut off point to reject or accept that transaction. We assumed for every fraud capture, we are gaining 2000 dollars, and for every false capture we are losing 50 dollars, which means they actually are good transactions but we denied.

Figure 7.2 Fraud Saving and Suggested Score Cut Off



In Figure 7.2, From the green line which represents the overall saving, we can see it reaches a peak and then goes down. It reaches a peak at 7% and there is a maximum gain of $247,600. As it was based only on out of sample data, we multiplied it by 6 and got an annual saving of $1.48 million if we reject the top 7% of the transaction.

8 Conclusions

In this report, we used machine learning models to analyze credit card transaction information and evaluate the data. We first explored the basic statistics of the dataset. There are originally 9 fields excluding the record field, and with a total of 96,753 records. Then we cleaned the data and handled exclusions, outliers, and missing fields. After cleaning the data, we did feature engineering and created candidate variables including amount, frequency, days since, and relative velocity. Then we balanced the dataset, standardized all the features and used KS and FDR methods to do the feature selection. We selected our final 13 variables using Sequential feature selector and recursive feature elimination cross-validation to build machine learning models.

We implemented logistic regression, neural networks, random forest, support vector classification, xg-boosted trees, gradient and boosting. The Logistic algorithm model is the baseline model we first built, which results in a 42% FDR at 3% for out-of-time data. Then we built more complex nonlinear models. After looking at the results, the random forest model has the best performance of 64.8% FDR at 3% for out-of-time data. Therefore, we chose the random forest model as our final model and created summary tables for train data, test data, and out-of-time data.

We provided suggestions on recommended cut-off percentage for fraud detection, for every new incoming transaction, rejecting the record that lies in the top 7% fraud score will save an annual financial benefit of $1.48 million.

We understand that there may be an improvement in space to implement a better model. If we have more time, we would try to make more use of the zip field, add geographical variables. We would also try to create more group identity variables such as industry category. As some models or parameters perform much better than the others, in the future, we will need to learn more about how to deal with overfitting issues so that more models could be trusted. Moreover, we will use more types of machine learning models to handle the data.

Appendix A: Data Quality Report

DATA QUALITY REPORT FOR CARD TRANSACTION DATA

Description

Dataset Name: Card Transaction

Dataset Purpose: Transactions for credit cards, analyzing corresponding fields, for example card number, transaction date, merchandise description, and amount to detect unusual or fraud.

Data Source: Class Material (synthetic U.S. card transaction over 1 years: 2016-01-01 à 2016-12-31)

Time Period: 1 years

Number of Fields: #9 (exclude recnum field)

Number of Records: 96,753

Summary Table

"Recnum" are just unique integer labels for each record.

Numerical Fields:

Column Name	# of Records	% Populated	Unique Values	Mean	Standard Deviation	Minimum Value	Maximum Value	# Zeros
Amount	96753	100	34909	427.8857	10006.14	0.01	3102046	0

Categorical Fields:

Comuln Name	# of Records	% Populated	Unique Values	Most Common Field Value
Cardnum	96753	100.0%	1645	5142148452

Date	96753	100.0%	365	2010-02-28 00:00:00
Merchnum	96753	96.5%	13092	930090121224
Merch description	96753	100.0%	13126	GSA-FSS-ADV
Merch state	96753	98.8%	228	TN
Merch zip	96753	95.2%	9223	38118
Transtype	96753	100.0%	4	P
Fraud	96753	100.0%	2	0

Distribution for all fields except Record:

Field 1

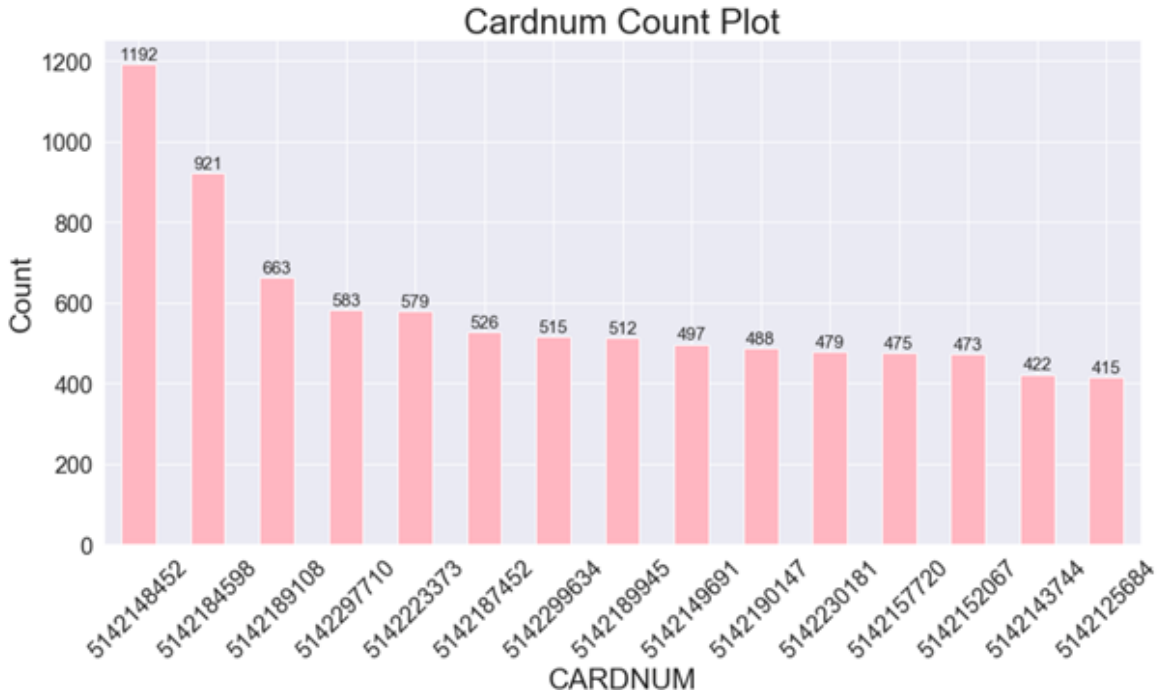
Name: **Recnum**

Description: Unique identifier of each entry in the data, continuous number.

Field 2

Name: **Cardnum**

Description: Card number that used for transaction, top 15.

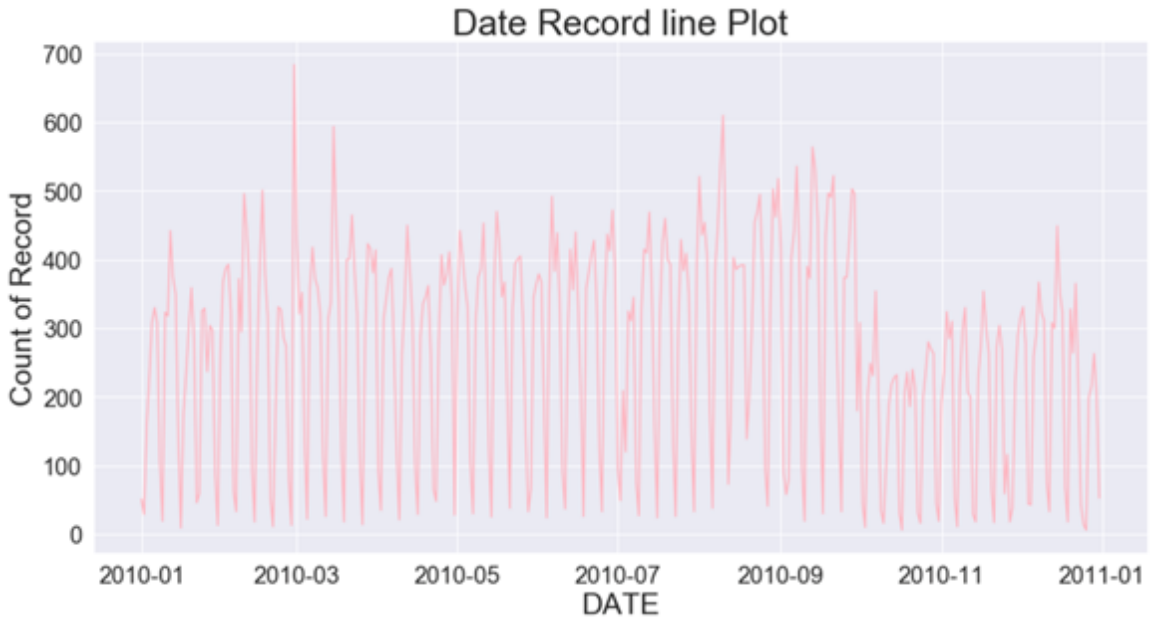


Field 3

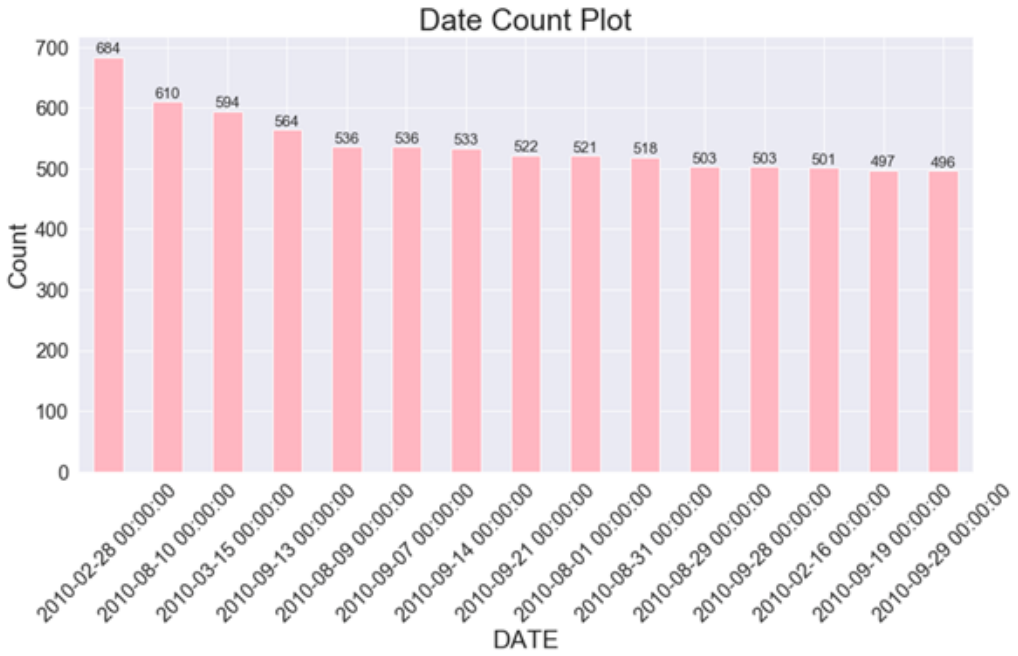
Name: **Date**

Description: The transaction date of all records

1) All days' transaction numbers



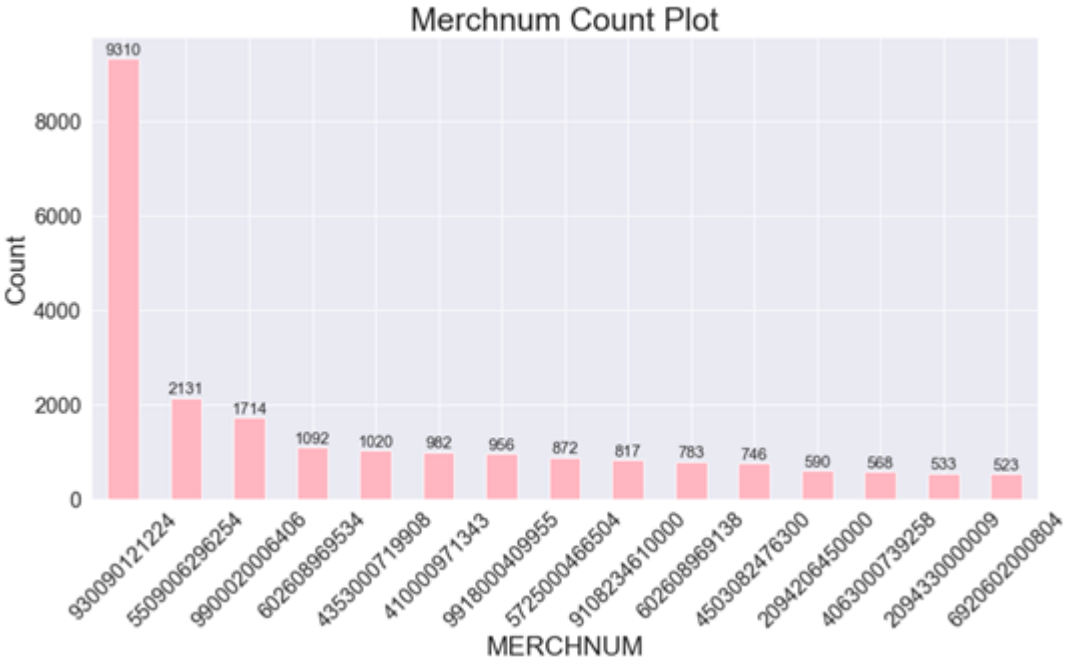
2) Top 15 Dates that have the most card transaction number.



Field 4

Name: **Merchnum**

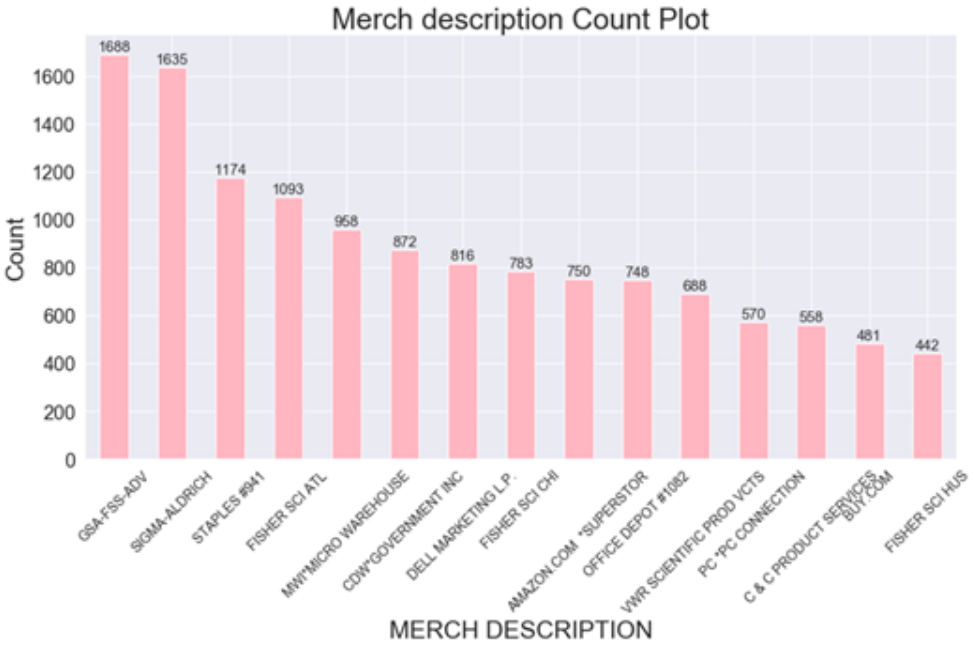
Description: Merchandise number, the number that represent each transaction, top 15.



Field 5

Name: **Merch description**

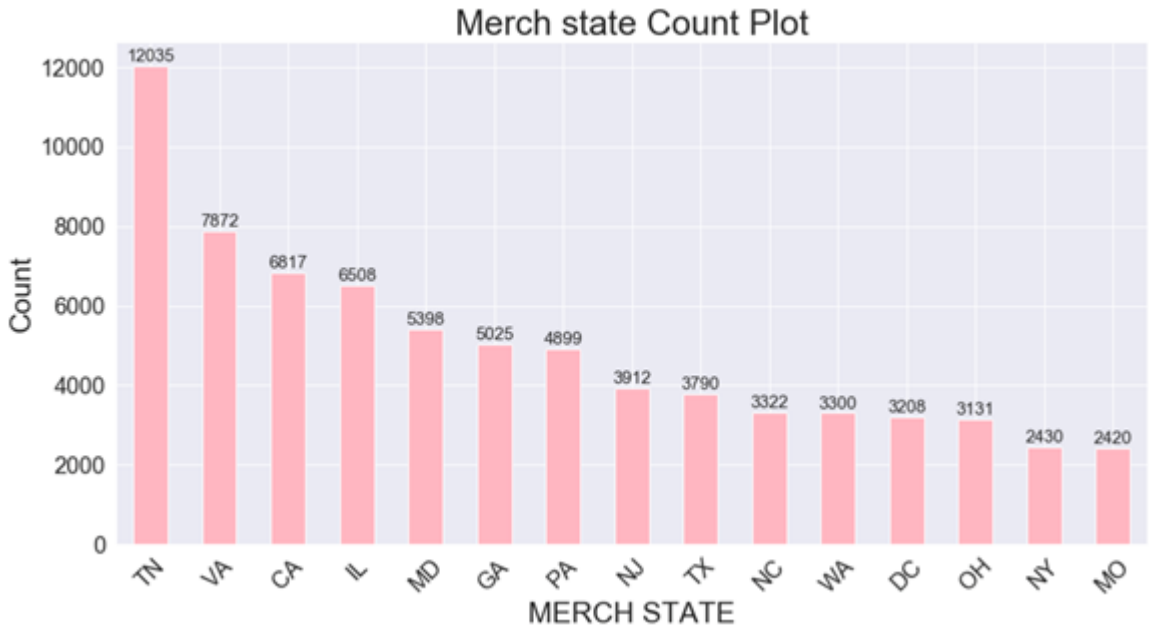
Description: Short description on each transaction's merchandise information, top 15.



Field 6

Name: **Merch state**

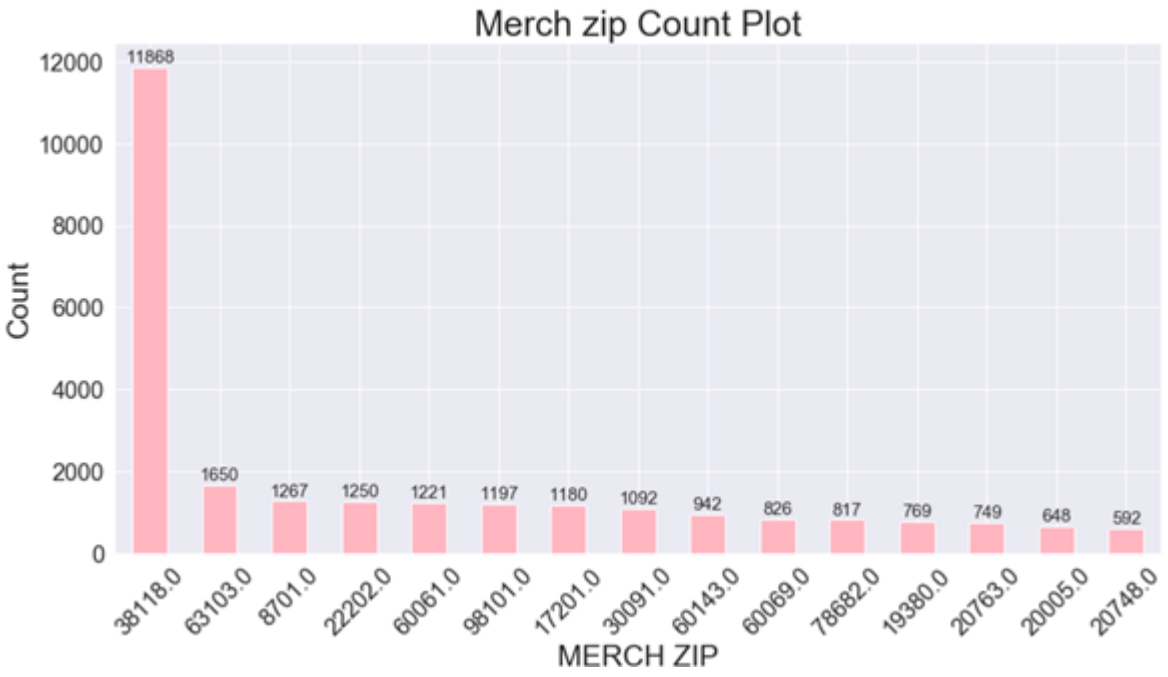
Description: The state for each transaction's merchandise location, top 15.



Field 7

Name: **Merch zip**

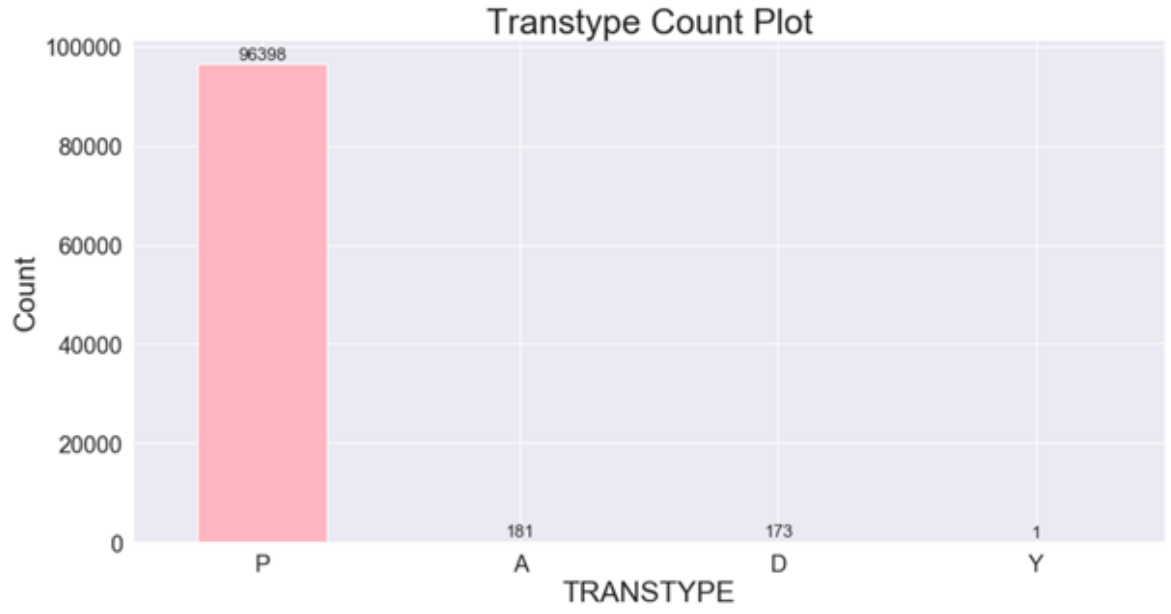
Description: The zip code for each transaction's merchandise location, top 15.



Field 8

Name: **Transtype**

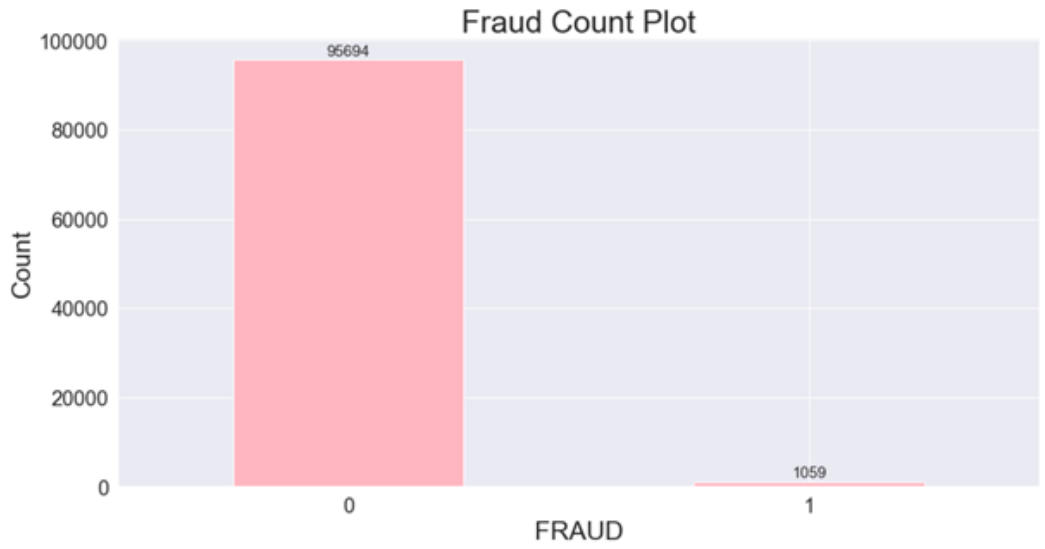
Description: The type for each transaction, P stands for Purchase, top 15.



Field 9

Name: **Fraud**

Description: an indicator that illustrate whether transactions are labeled fraud or not, 1 is for fraud and 0 is no-fraud.



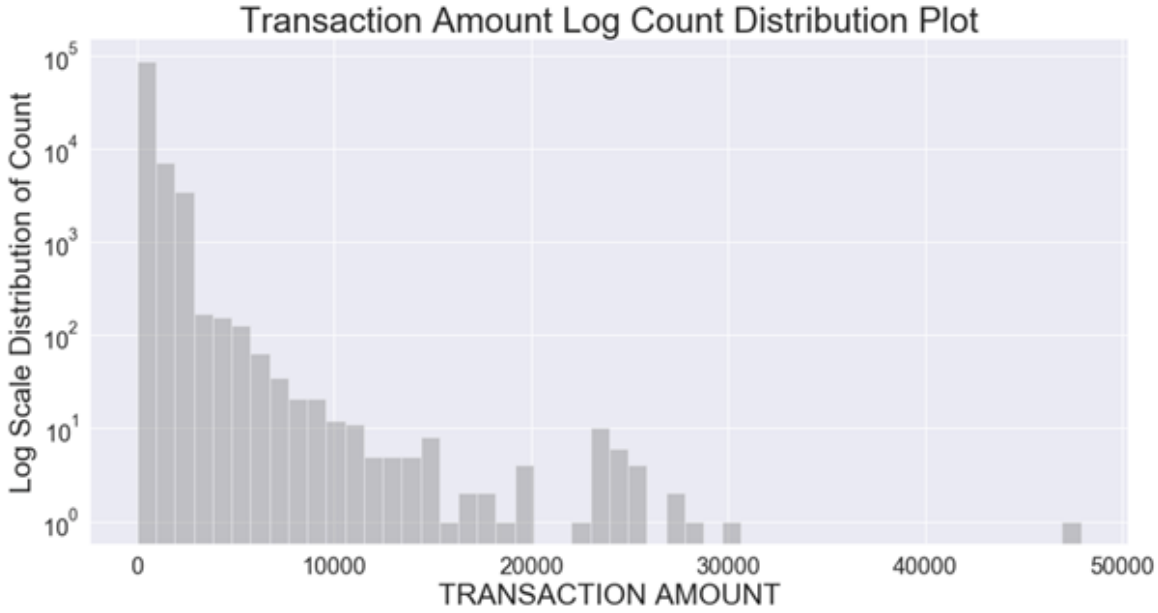
Field 10

Name: **Amount**

Description: The dollar amount for each card transaction

Exclude outliers > 300,000

Data in histogram is nearly 100% populated (exclude one outlier)



Appendix B: List of Variables Created

Original Variables in the Dataset (10 Total)

Recnum	Cardnum	Date	Merchnum	Merch description	Merch state
Merch zip	Transtype	Amount	Fraud		

Variables Based on Benford's Law(2 Total)

Cardnum Benford	Merchnum Benford

Entity Variables(11 Total)

Cardnum_Merchnum	num_description	num_state	num_zip	description_state	description_zip
state_zip	num_description_state	num_description_zip	description_state_zip	num_description_state_zip	

Day of Week (2 Total)

dow	dow_risk

Day Since(18 total)

Cardnum_day_since	Merchnum_day_since	Cardnum_Merchnum_day_since	Merch zip_day_since	Merch state_day_since	num_description_day_since
num_state_day_since	num_zip_day_since	description_state_day_since	description_zip_day_since	state_zip_day_since	num_description_state_day_since
num_description_zip_day_since	description_state_zip_day_since	num_description_state_zip_day_since	Cardnum_zip_day_since	Cardnum_state_day_since	Cardnum_Merch_Des_day_since

Frequency(108 total)

Cardnum_count_0	Cardnum_count_1	Cardnum_count_3	Cardnum_count_7	Cardnum_count_14	Cardnum_count_30
Merchnum_count_0	Merchnum_count_1	Merchnum_count_3	Merchnum_count_7	Merchnum_count_14	Merchnum_count_30

Cardnum_Merchnum_count_0	Cardnum_Merchnum_count_1	Cardnum_Merchnum_count_3	Cardnum_Merchnum_count_7	Cardnum_Merchnum_count_14	Cardnum_Merchnum_count_30
Merch zip_count_0	Merch zip_count_1	Merch zip_count_3	Merch zip_count_7	Merch zip_count_14	Merch zip_count_30
Merch state_count_0	Merch state_count_1	Merch state_count_3	Merch state_count_7	Merch state_count_14	Merch state_count_30
num_description_count_0	num_description_count_1	num_description_count_3	num_description_count_7	num_description_count_14	num_description_count_30
num_state_count_0	num_state_count_1	num_state_count_3	num_state_count_7	num_state_count_14	num_state_count_30
num_zip_count_0	num_zip_count_1	num_zip_count_3	num_zip_count_7	num_zip_count_14	num_zip_count_30
description_state_count_0	description_state_count_1	description_state_count_3	description_state_count_7	description_state_count_14	description_state_count_30
description_zip_count_0	description_zip_count_1	description_zip_count_3	description_zip_count_7	description_zip_count_14	description_zip_count_30
state_zip_count_0	state_zip_count_1	state_zip_count_3	state_zip_count_7	state_zip_count_14	state_zip_count_30
num_description_state_count_0	num_description_state_count_1	num_description_state_count_3	num_description_state_count_7	num_description_state_count_14	num_description_state_count_30
num_description_zip_count_0	num_description_zip_count_1	num_description_zip_count_3	num_description_zip_count_7	num_description_zip_count_14	num_description_zip_count_30
description_state_zip_count_0	description_state_zip_count_1	description_state_zip_count_3	description_state_zip_count_7	description_state_zip_count_14	description_state_zip_count_30
num_description_state_zip_count_0	num_description_state_zip_count_1	num_description_state_zip_count_3	num_description_state_zip_count_7	num_description_state_zip_count_14	num_description_state_zip_count_30
Cardnum_zip_count_0	Cardnum_zip_count_1	Cardnum_zip_count_3	Cardnum_zip_count_7	Cardnum_zip_count_14	Cardnum_zip_count_30
Cardnum_state_count_0	Cardnum_state_count_1	Cardnum_state_count_3	Cardnum_state_count_7	Cardnum_state_count_14	Cardnum_state_count_30
Cardnum_Merch_Des_count_0	Cardnum_Merch_Des_count_1	Cardnum_Merch_Des_count_3	Cardnum_Merch_Des_count_7	Cardnum_Merch_Des_count_14	Cardnum_Merch_Des_count_30

Examples of Amount Variables(900 total, same format for each entity variable)

Cardnum_mean_Amount_0	Cardnum_max_Amount_0	Cardnum_median_Amount_0	Cardnum_sum_Amount_0	Cardnum_freq_0
Cardnum_mean_Amount_1	Cardnum_max_Amount_1	Cardnum_median_Amount_1	Cardnum_sum_Amount_1	Cardnum_freq_1
Cardnum_mean_Amount_3	Cardnum_max_Amount_3	Cardnum_median_Amount_3	Cardnum_sum_Amount_3	Cardnum_freq_3
Cardnum_mean_Amount_7	Cardnum_max_Amount_7	Cardnum_median_Amount_7	Cardnum_sum_Amount_7	Cardnum_freq_7
Cardnum_mean_Amount_14	Cardnum_max_Amount_14	Cardnum_median_Amount_14	Cardnum_sum_Amount_14	Cardnum_freq_14
Cardnum_mean_Amount_30	Cardnum_max_Amount_30	Cardnum_median_Amount_30	Cardnum_sum_Amount_30	Cardnum_freq_30
Actual/Cardnum_mean_Amount_0	Actual/Cardnum_max_Amount_0	Actual/Cardnum_median_Amount_0	Actual/Cardnum_sum_Amount_0	Actual/Cardnum_freq_0
Actual/Cardnum_mean_Amount_1	Actual/Cardnum_max_Amount_1	Actual/Cardnum_median_Amount_1	Actual/Cardnum_sum_Amount_1	Actual/Cardnum_freq_1
Actual/Cardnum_mean_Amount_3	Actual/Cardnum_max_Amount_3	Actual/Cardnum_median_Amount_3	Actual/Cardnum_sum_Amount_3	Actual/Cardnum_freq_3
Actual/Cardnum_mean_Amount_7	Actual/Cardnum_max_Amount_7	Actual/Cardnum_median_Amount_7	Actual/Cardnum_sum_Amount_7	Actual/Cardnum_freq_7
Actual/Cardnum_mean_Amount_14	Actual/Cardnum_max_Amount_14	Actual/Cardnum_median_Amount_14	Actual/Cardnum_sum_Amount_14	Actual/Cardnum_freq_14
Actual/Cardnum_mean_Amount_30	Actual/Cardnum_max_Amount_30	Actual/Cardnum_median_Amount_30	Actual/Cardnum_sum_Amount_30	Actual/Cardnum_freq_30

Examples of Relative Frequency/Amount
(270 Total, same format for each entity variable)

Cardnum_count_0_by_7	Cardnum_count_0_by_14	Cardnum_count_0_by_30	Cardnum_count_1_by_7	Cardnum_count_1_by_14	Cardnum_count_1_by_30
Cardnum_amount_0_by_7	Cardnum_amount_0_by_14	Cardnum_amount_0_by_30	Cardnum_amount_1_by_7	Cardnum_amount_1_by_14	Cardnum_amount_1_by_30

Cardnum_sum_Amount_0_unique_Cardnum_0	Cardnum_sum_Amount_1_unique_Cardnum_1	Cardnum_sum_Amount_3_unique_Cardnum_3	Cardnum_sum_Amount_7_unique_Cardnum_7	Cardnum_sum_Amount_14_unique_Cardnum_14	Cardnum_sum_Amount_30_unique_Cardnum_30