Project3: Supervised Model on Card Transactions Data

Group 27

Shiru Xu, Zhihuan Cui, Lin Liu, Huiyu Huang, Ningxi Wang

05/09/2021

TABLE OF CONTENTS

Executive Summary

Nowadays, credit card usage has significantly increased especially in the modern era, as almost all transactions can easily be processed online by entering credit card information. This has become a serious problem because a non negligible amount of credit card transactions are fraudulent and caused over $40 million lost per year worldwide. Fraud can be committed in different ways. Merchants and banks can use the address, transaction amount, location and device identification and etc. and apply a set of business rules, statistical rules and algorithms to detect as many frauds as possible while denying as few transactions as possible.

This report provides an analysis of credit card's transaction during the account usage process. The primary aim was to examine the best supervised fraud model to detect credit card transaction fraud and denying as few transactions as possible. This can be achieved by bringing together features of credit card transaction information, such as the amount, zip code, state, card number, merchant number, transaction type and merchant description. This information can go through the well-trained model that finds the patterns and rules of fraud transactions so that it can detect as many fraudulent transactions as possible while denying as few non-fraudulent transactions as possible and set a score cutoff that will divert records above the cutoff.

Models of analytics include logistic regression, neural network, boosted trees(XGBoost) and random forest. Methods include data cleaning, variable creation, filter and wrapper. Calculations include univariate Kolmogorov-Smirnov score and model performance measure (e.g. FDR). More details about methods used can be found in the feature selection section. Specific details about models and calculations can be found in the model algorithms and results section.

The final best model we chose to analyze is RandomForestClassifier with 150 trees and 20 max depths. The dataset is splitted into three parts including training, testing and out-of-time. The final chosen model achieved 79.6% FDR at 3% on the out-of-time data. The final performance tables illustrate that the average results of 3% FDR rate and KS value for each dataset are in a relatively good range. We also illustrated the fraud savings calculation with recommended cutoff scores and two detailed plots for a card number and a merchant number.

Description of Data

The dataset is called “Card Transactions”. It contains actual credit cards transactions information from January 1, 2010 to December 31, 2010. It has 10 fields and 96,753 records which come from a US government organization.

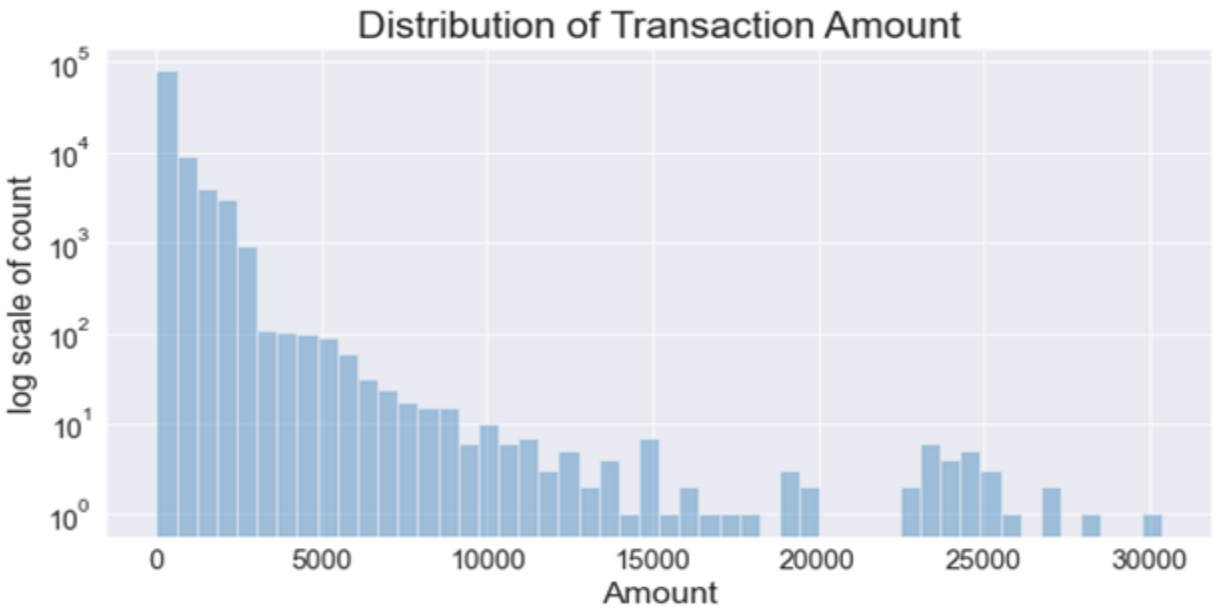
The following table is the summary statistics table of categorical fields in the dataset. It shows that there are missing values in the fields ‘Merchnum’, ‘Merch state’ and ‘Merch zip’.

Column Name	# of Records	% Populated	# Zeros	Unique Values	Most Common Field Value
Recnum	96753	100	0	96753	N/A
Cardnum	96753	100	0	1645	5142148452
Merchnum	96753	96.51	0	13092	930090121224
Merch description	96753	100	0	13126	GSA-FSS-ADV
Merch state	96753	98.76	0	228	TN
Merch zip	96753	95.19	0	4568	38118
Transtype	96753	100	0	4	P
Fraud	96753	100	95694	2	0

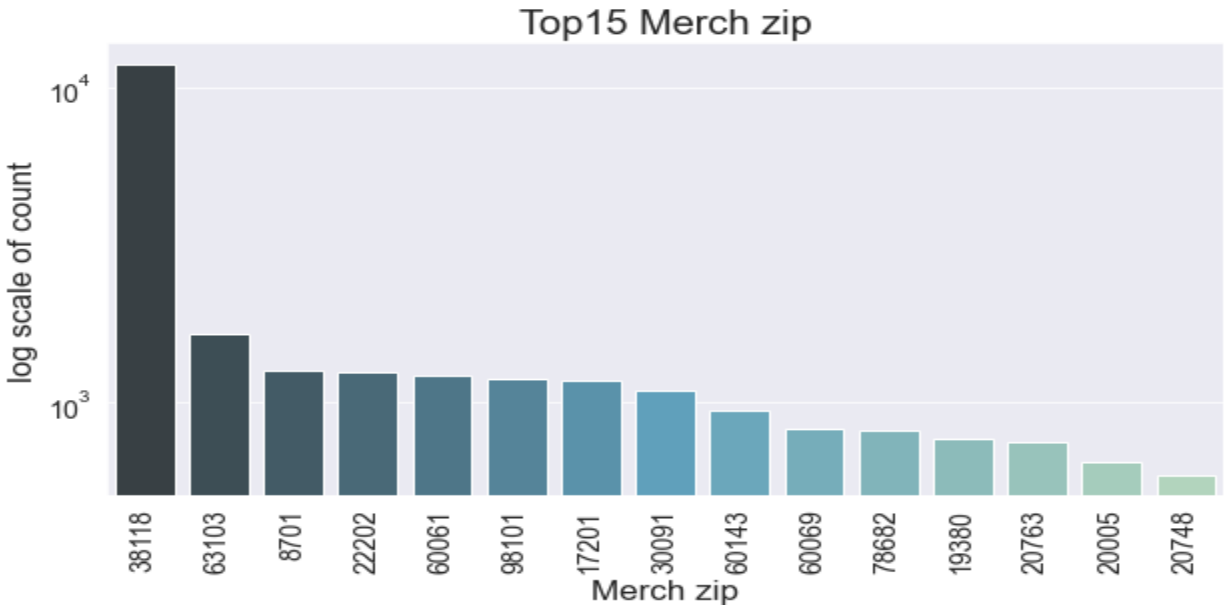
The following table is the summary statistics table of the numerical field in the dataset. It shows that the maximum value of the field ‘Amount’ is way higher than the mean value of the field ‘Amount’. And, it means that we should pay attention to some records in this dataset.

Column Name	% Populated	Unique Values	Most Common Field Value	Mean	Std	Min	Max
Amount	100	34909	3.62	427.89	10006.14	0.01	3102045.53

The following plot is the log scale distribution of transaction amount. It’s the dollar amount of the transaction. In this plot, Record 52714 is excluded because it is an extreme value of 3102045.53. And we can see that the amount of most transactions is less than 2500.



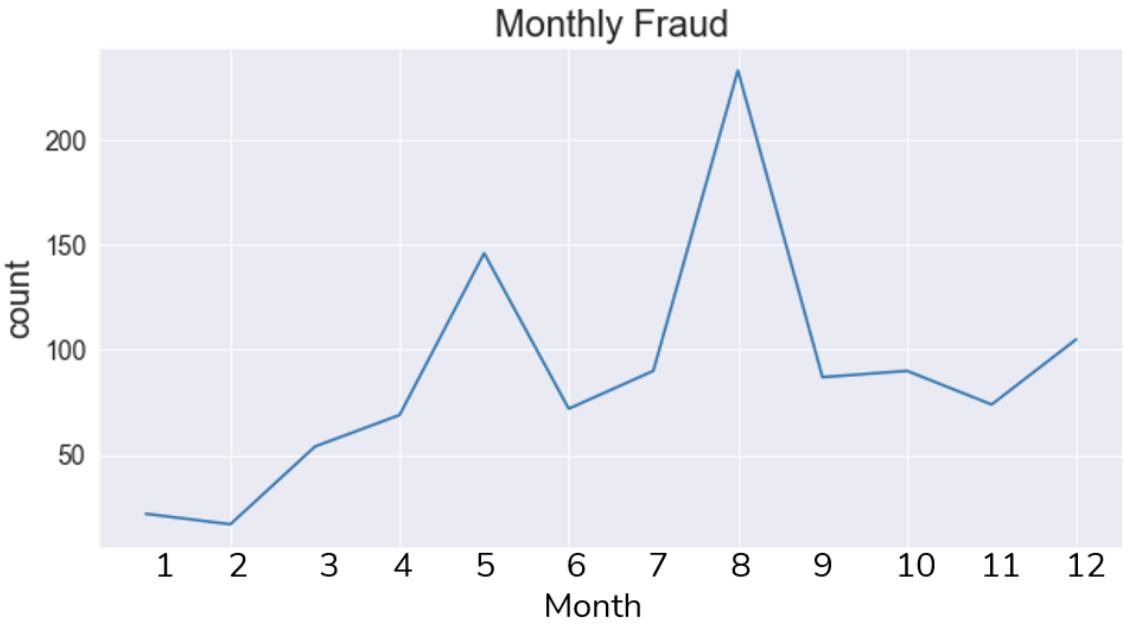
Next plot is for the field ‘Merh zip’. It’s the zip code of the place where the transaction happened. Or, the zip code of the place that the headquarter of the merchant’s company is in. In this field, we get 4568 unique values. And, 38118 is the most common field value. For the top 15 merch zip, we can see that except for the zip code 38118, other zip codes are nearly evenly distributed.



The following table is the log scale distribution of ‘Fraud’. There are only values of 0 and 1 in this field. It means whether the record is fraudulent or not. 0 represents that the record is not fraudulent while 1 represents that the record is fraudulent. In this dataset, 95694 (98.91%)

records have a value of 0 and 1059 (1.09%) records have a value of 1.

We also looked at the number of 'Fraud' that happened each month and got the following table. We can see that the largest number of 'Fraud' transactions happened in August and this is consistent with the monthly number of transactions. The largest number of transactions also happened in August.



Data Cleaning

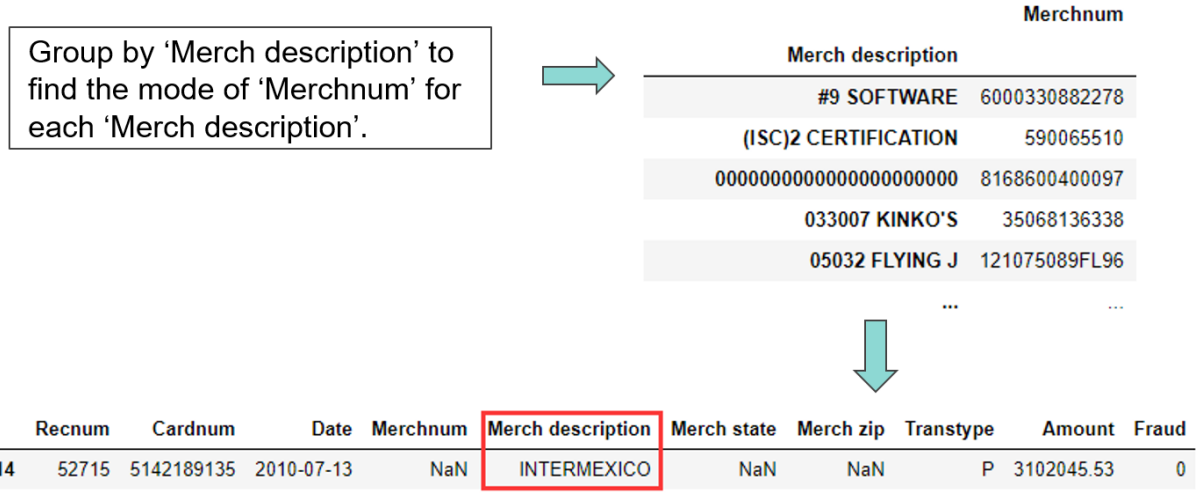
After looking at the data, we found that there is an outlier for the field 'Amount'. The following picture is the record with the extreme value of the field 'Amount'.

	Recnum	Cardnum	Date	Merchnum	Merch description	Merch state	Merch zip	Transtype	Amount	Fraud	month
52715	5142189135	2010-07-13	NaN	INTERMEXICO	NaN	NaN	P	3102045.53	0	7

We investigated this record and found that this is not fraudulent. Thus, it would not influence the quality of dataset samples if we drop this record. So, to avoid the impact of this extreme value on our model performance, we dropped this record.

We also found that three fields have missing values. They are 'Merchnum', 'Merch state' and 'Merch zip'.

To fill in the missing values in 'Merchnum', we first grouped by 'Merch description' to find the mode of 'Merchnum' for each 'Merch description'. Then, we got a table for 'Merch description' and 'Merchnum'. In the table, every 'Merch description' has a corresponding 'Merchnum'. Then, for records that have missing values in 'Merchnum', we looked at its 'Merch description' and went to the above table to find the corresponding 'Merchnum'. If found, we would fill the missing value with the corresponding 'Merchnum'. And if not found, we ignored them temporarily.



Then, we did the same to 'Merch state' and 'Merch zip'. We always first grouped by 'Merch description' to find the mode of 'Merch state' and 'Merch zip' for each 'Merch description' and got tables. Then, we went to the record which has missing merch value, found the merch description and the corresponding value in tables and filled in the missing merch value. If not found, also ignored temporarily.

After these filling, there were still many missing values in ‘Merch state’. So, we did further filling. We did the similar process. However, this time, to fill in the missing values in ‘Merch state’, we grouped by ‘Merch zip’ to find the mode of ‘Merch state’ for each ‘Merch zip’. Then, we did the same as the above filling process.

Finally, for the remaining missing values, we filled them with unknown. And now, we got our data prepared.

Feature Engineering

In the feature engineering process, we built candidate variables to capture signals of credit card transaction fraud. In this project, we focused on fraud signals including: 1) burst of activity, both in terms of amount and frequency; 2) use in first-occurred merchant or geography; 3) use by high-risk card or merchant; 4) fictitious transaction patterns. Based on these signals, we created six categories of variables, and built 358 candidate variables in total.

Firstly, we decided five entities to build our variables on. They are 'card', 'merchant', 'card at this merchant', 'card in this zip code', and 'card in this state'. We used 'Cardnum' and 'Merchnum' for the first two, and combined 'Cardnum' with 'Merchnum', 'Merch zip' and 'Merch state' for the other three. We thought a fraudster is likely to commit fraud at each entity level.

The first set of variables we built is amount variables. They record the mean, maximum, median, total, actual/mean, actual/maximum, actual/median, actual/total, of the transaction amount of each entity over the past 0, 1, 3, 7, 14, 30 days. There are 240 amount variables. Their names take the form '{agg}_amount_{entity}_count_{pastdays}'. We built them to capture unusual transaction amounts. When the aggregated amount is unusually high, this could be an indicator of fraud.

Next, we built 30 frequency variables and 5 days since variables to capture unusual transaction frequencies. The frequency variables measure the number of transactions with the same entity seen over the past 0, 1, 3, 7, 14, 30 days. They have names in the form '{entity}_count_{pastdays}'. When there are a lot of transactions happening within a very few days for the same entity, it is very risky. The days since variables measure the number of days since we last saw a transaction with the same entity. They are called '{entity}_day_since'. If the entity only appears once, we fill the value with 365. When we only see the same entity involved in many transactions, it is more likely to be held by a fraudster.

Derived from the amount and frequency variables, we created another set of velocity change variables. They measure the relative transaction amount or frequency for a certain entity over different time periods. Specifically, we calculated the transaction amount/frequency with a certain entity over the past {0, 1} days, divided by the average daily transaction/frequency for the same entity over the past {3, 7, 14, 30} days. There are 80 combinations of them and they are named after '{entity}_count_{d}_by_{dd}' and '{entity}_{agg}_amount_{d}_by_{dd}'. We used them to capture sudden increase in activities over a very short period.

In addition, we create a risk table variable for the 'day of week', called 'dow_risk'. It represents the likelihood of fraud for that day of the week. We first build a variable called 'dow' for the day of week when each record happens. Then, we target-encoded it by the proportion of fraud that happened on that day of week. We excluded the out-of-time data to calculate the proportion and

transformed it using a statistically smoothing formula: y_avg + (y_dow - y_avg)/(1 + np.exp(-(num - nmid)/c)) with c = 4 and nmid = 20.

Last but not least, we create two Benford's Law variables for cardholder and merchant. Benford's Law states that the first digit of many measurements is not uniformly distributed, with the first digit '1' appearing about 30% of the time. For transaction fraud, we assumed that the fraudsters do not know about Benford's law when they are making up transactions. Thus, the first digit of the transaction amounts should not follow Benford's Law. To build the variables, we extracted the first digit from the amount, and summed up the number of times each digit appears. We did this separately for each 'Cardnum' and 'Merchnum'. We set the number of times digit 1 or 2 appears to 'n_low', and the number of times digit 3 to 9 appear to 'n_high'. According to Benford's Law distribution, n_low/n_high should be around 1.096. Thus, when we set R = 1.096*n_low/n_high, and it should be close to 1. If R significantly deviates from 1, then the maximum of R or 1/R should be very big, indicating substantial violation of Benford' Law. We set U=max(R, 1/R) to denote the level of unusualness. We also used a smoothing formula to transform U to U*: U* = 1+(U-1)/(1+exp^(-t)), where t = (n-nmid)/c, nmid=15, c=3. U* for each cardholder is called {benford_card}, and U* for each merchant is called {benford_merch}. As U* gets higher, the cardholder or merchant is riskier. Thus, these two Benford's Law variables could capture whether a transaction involves a high-risk card or merchant.

Feature Selection

Feature selections are intended to reduce the number of input variables which are irrelevant or less important features in order to achieve better accuracy of our model. First, it reduces overfitting because less redundant data means less opportunity to predict models based on noise. Then, it can improve accuracy by removing misleading data. At last it reduces training time because fewer input variables can reduce algorithm complexity and train the model faster.

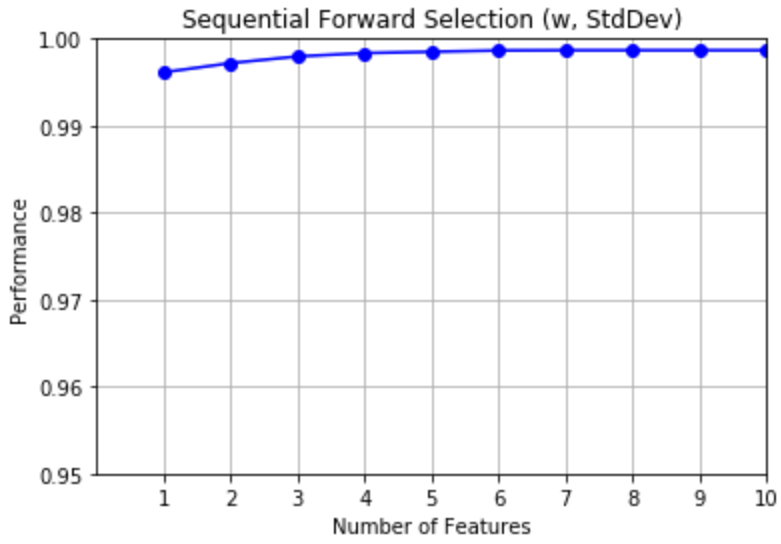
In this project, we decided to use filter and wrapper methods. We started with 358 variables. And we used the filter method to select the top 80 variables. And then move on to the wrapper method and get the most relevant 30 variables for our final modeling process.

In the filter method, the selection of features is independent of any modeling method. Instead, features are selected on the basis of their univariate scores. In this case, the univariate Kolmogorov-Smirnov (KS) score and the Fraud Detection Rate (FDR) at 3%, for their correlation with the outcome variable. KS measures how well the goods curve and the bads curve are separated between each other. We calculated the univariate KS score for each variable and all candidate variables acquired a KS score lower than 0.57.

FDR @3% shows the proportion of total frauds getting caught if we reject the top 3% of records. We chose 3% because companies that use a fraud score to deny a small fraction of requests typically set the cutoff at 3,4,5%. We sorted the dataset by the values of each variable and got the top3% of data. We added all the frauds and divided it by the total number of frauds. Thus, we calculated the FDR@3 % for each variable. The highest FDR we got is 0.508. Finally, we computed the rank by KS and FDR@3 %. We summed up the two ranks and divided by 2. We sorted our final list by the average rank of the two scores and took the top 80 variables.

And then we moved on to wrapper and got the best top 30 variables from the 80 variables selected from the filter method. For the wrapper method, we would like to get a list of 30 variables in the order of multivariate importance, which means that correlations are taken into account. In wrapper methods, the feature selection process is based on a specific learning algorithm that we are trying to fit on a given dataset and it follows a greedy search approach by evaluating all possible combinations of features against the evaluation criterion. In this case, we used forward selection based on decision tree model and accuracy as the evaluation criterion. In forward selection, it starts with no variables selected and tries adding in each individual variable that we don't already have and chooses the next best variables given all the ones we already have based on the overall model performance.

Thus, we got the good subsets of 30 variables and removed correlations. The graph below evaluates variable performance.



List of Final Variables

'Median_amount_card_state_count_1': the median amount spent by the same card in the state over the past 1 day

'ave_amount_card_state_count_1': the average amount spent by the same card in the state over the past 1 day

'ave_amount_card_zip_count_7': the average amount spent by the same card in this zip code over the past 7 days

'ave_amount_card_zip_count_3': the average amount spent by the same card in this zip code over the past 3 days

'ave_amount_card_zip_count_1': the average amount spent by the same card in this zip code over the past 1 day

'ave_amount_card_state_count_3': the average amount spent by the same card in this state over the past 3 days

'ave_amount_card_state_count_30': the average amount spent by the same card in this state over the past 30 days

'ave_amount_card_zip_count_0': the average amount spent by the same card in this zip code over the past 0 day

'ave_amount_card_merchnum_count_1': the average amount spent by the same card at this merchant over the past 1 day

'ave_amount_card_state_count_7': the average amount spent by the same card in this state over the past 7 days

'ave_amount_card_zip_count_30': the average amount spent by the same card in this zip code over the past 20 days

'ave_amount_card_zip_count_14': the average amount spent by the same card in this zip code over the past 14 days

'ave_amount_card_merchnum_count_3': the average amount spent by the same card in this zip code over the past 1 day

'median_amount_card_state_count_0': the median amount spent by the same card in this state over the past 0 day

'median_amount_card_zip_count_14': the median amount spent by the same card in this zip code over the past 14 days

'max_amount_card_state_count_7': the maximum amount spent by the same card in this state over the past 7 days

'median_amount_card_merchnum_count_3': the median amount spent by the same card in this merchant over the past 3 days

'sum_amount_card_zip_count_0': the total amount spent by the same card in this zip code over the past 0 day

'sum_amount_card_zip_count_1': the total amount spent by the same card in this zip code over the past 1 day

'median_amount_card_merchnum_count_30': the median amount spent by the same card in this merchant over the past 30 days

'ave_amount_Cardnum_count_1': the average amount spent by the same card over the past 1 day

'ave_amount_Cardnum_count_14': the average amount spent by the same card over the past 14 days

'ave_amount_Cardnum_count_3': the average amount spent by the same card over the past 3 days

'ave_amount_Cardnum_count_0': the average amount spent by the same card over the past 0 day

'ave_amount_Cardnum_count_30': the average amount spent by the same card over the past 30 days

'sum_amount_card_state_count_0': the total amount spent by the same card in this state over the past 0 day

'act/median_amount_Merchnum_count_30': the actual amount over the median amount spent by the same merchant over the past 30 days

'sum_amount_card_state_count_1': the total amount spent by the same card in this state over the past 1 day

'act/mean_amount_Merchnum_count_7': the actual amount over the mean amount spent by the same merchant over the past 7 days

'act/mean_amount_Merchnum_count_30': the actual amount over the mean amount spent by the same merchant over the past 30 days

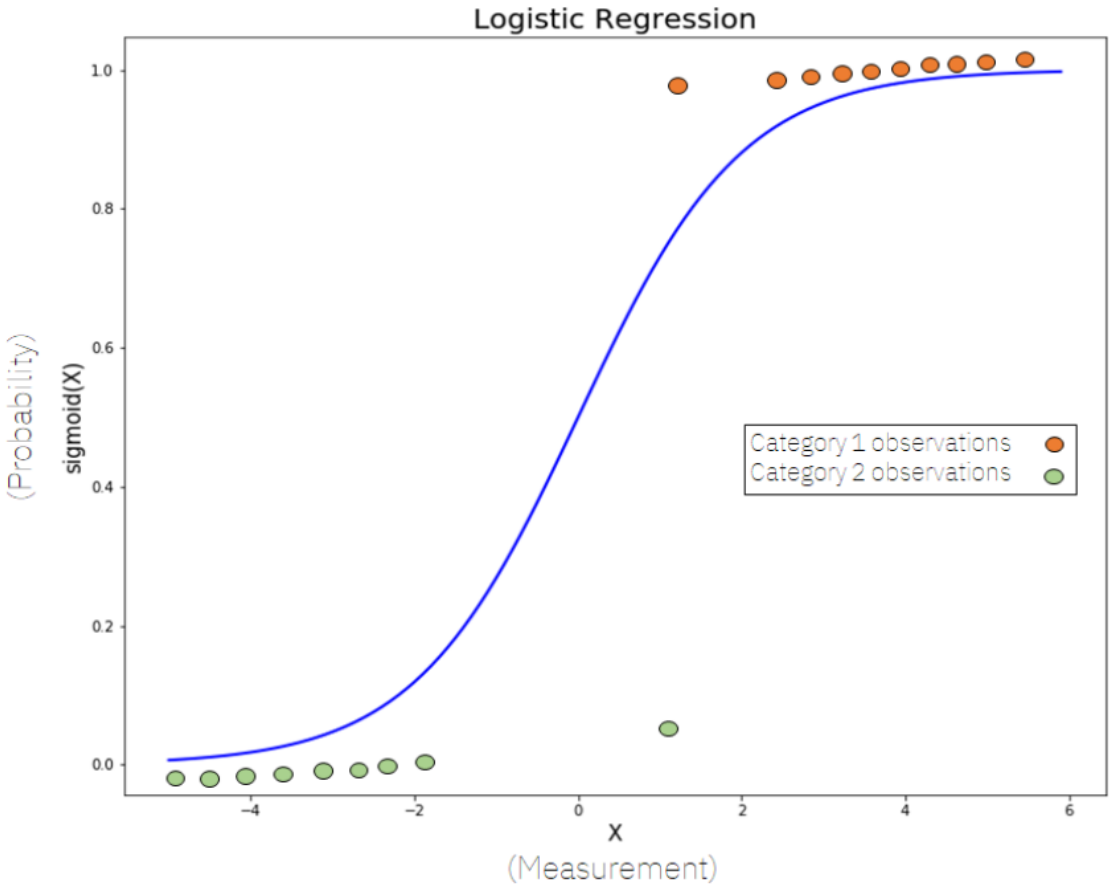
Model Explanations

In this project, we used **logistic regression** as the baseline model to give us a picture about what the minimum result could look like.

Logistic Regression models are classification models; specifically binary classification models

In Logistic Regression, we don't directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called Sigmoid, to our observations. as we can see, the Y-axis goes from 0 to 1. This fits very well to our goal of classifying samples in two different categories. By computing the sigmoid function of X, we get a probability of an observation belonging to one of the two categories

To make a classifier, we need to choose a cutoff value and classify inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier.

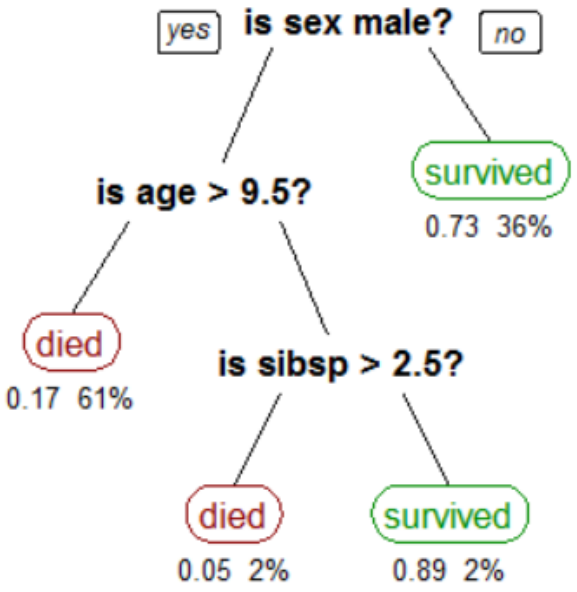


The next series of models we implemented is the tree methods. Even though we did not use the decision tree algorithm as our algorithm, it is not explanatory to talk about other tree methods with touching upon the decision tree algorithm.

A **decision tree** is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

A decision tree is drawn upside down with its root at the top. In the image at the bottom, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

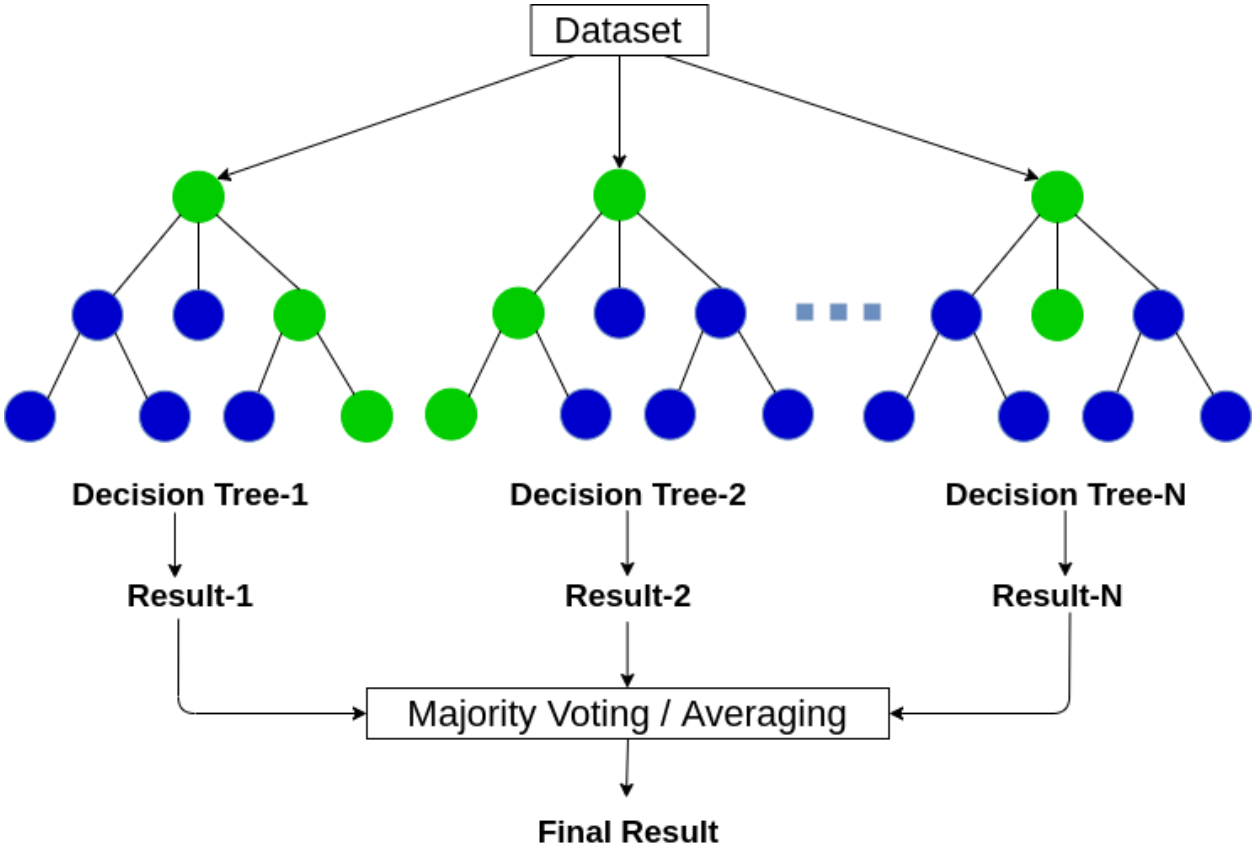
So how exactly do we prune the tree? The decision tree finds a cut point which minimizes the node impurity using gini index or entropy.



Random forest is one of the tree methods we tried. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or average prediction of the individual trees.

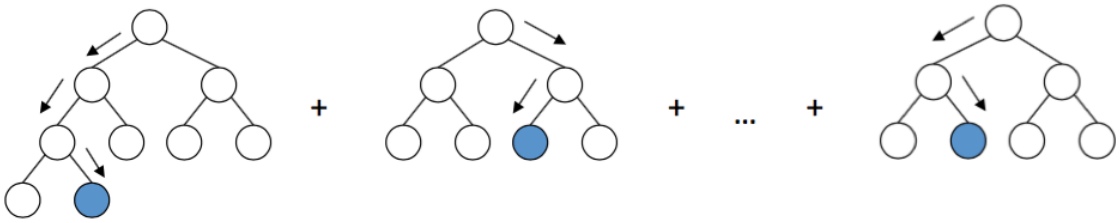
Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors.



Boosting means combining a learning algorithm in series to achieve a strong learner from many sequentially connected weak learners.Each tree attempts to minimize the errors of previous tree. Trees in boosting are weak learners but adding many trees in series and each focusing on the errors from the previous one make boosting a highly efficient and accurate model. Unlike bagging, boosting does not involve bootstrap sampling. Everytime a new tree is added, it fits on a modified version of the initial dataset.Since trees are added sequentially, boosting algorithms learn slowly. In statistical learning, models that learn slowly perform better.

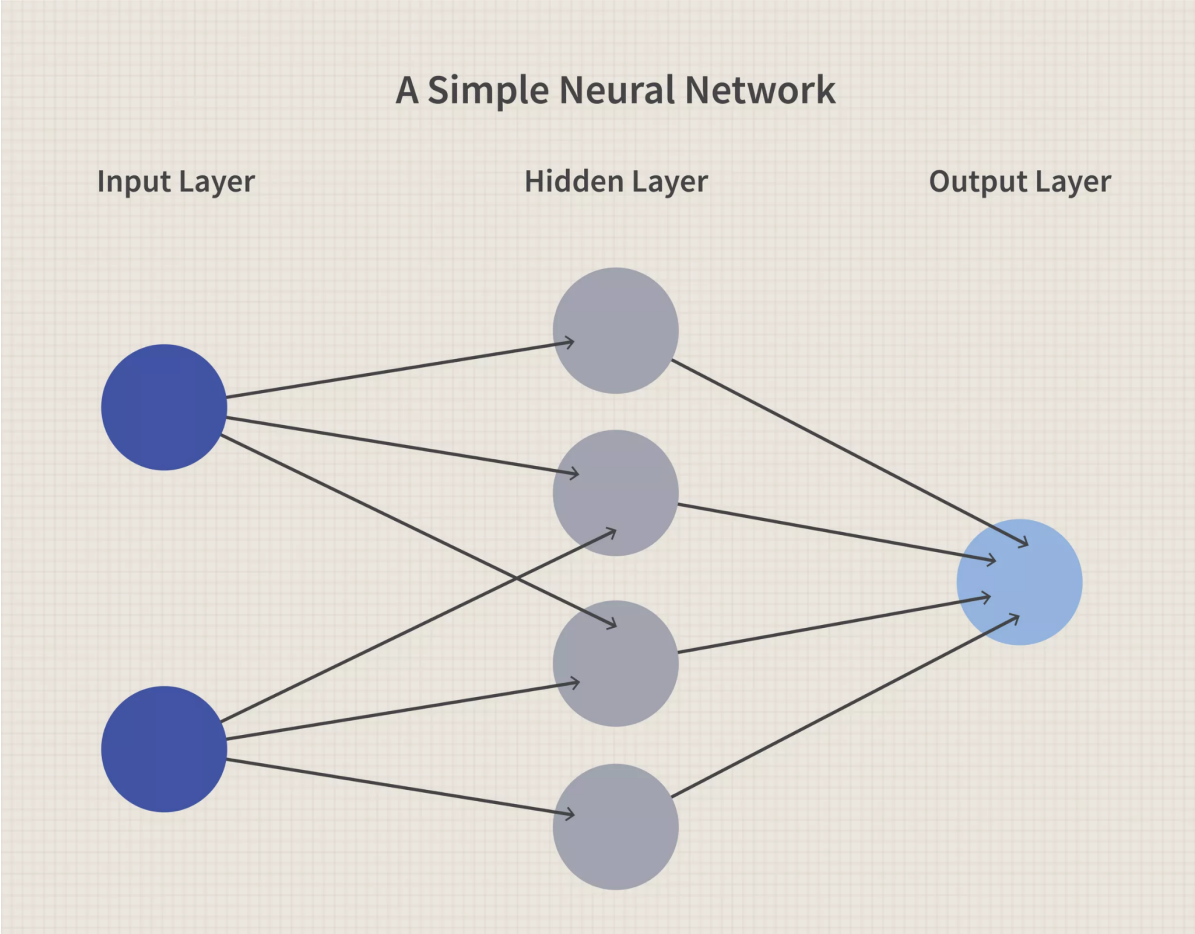
The boosted tree python library we used is XGBoost. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on a major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.



The last model we attempted is the Neural Network.

Machine learning algorithms that use neural networks generally do not need to be programmed with specific rules that define what to expect from the input. The neural net learning algorithm instead learns from processing many labeled examples that are supplied during training and using this answer key to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results. The more examples and variety of inputs the program sees, the more accurate the results typically become because the program learns with experience.

A Neural Network is made of layers and the layers are made of nodes. A node is just a place where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn.



Model Results

Since different variables are on different scales, we zscaled our data via StandardScaler from the Scikit-Learn library in python before modelling. We selected 4 months data as the OOT data and randomly split the training and testing data with a 40% test size.

We ran different methods and did hyperparameter tunings by grid-search. We got average 3% FDRs from 10 runs on train, test and OOT data.

First, We tried Logistic Regression, which has the best performance of 54% on oot. It represents the baseline performance for our modelling process.

Model	Parameter		Average FDR (%) at 3%		
Logistic Regression	Total Variables	Number of Varaibles Selected	Train	Test	OOT
1	30	30	0.582	0.58	0.54
2	30	25	0.559	0.565	0.516
3	30	20	0.551	0.541	0.537
4	30	15	0.542	0.552	0.527
5	30	10	0.483	0.474	0.496

Then we moved to tree-based methods. Since the decision tree is easy to interpret but weaker at prediction accuracies compared to xgboost and random forest, we didn't try the decision tree. Here is the result of XGBoost, and its best performance is 0.649 on oot.

Model	Parameter				Average FDR (%) at 3%		
XGBoost	*Number of Variables*	*# of Trees*	*Max Depth*	*Learning Rate*	*Train*	*Test*	*OOT*
1	30	500	0.01	2	0.780	0.743	0.546
2	30	500	0.01	3	0.809	0.777	0.552
3	30	500	0.01	4	0.891	0.881	0.578
4	30	500	0.01	5	0.952	0.933	0.620
5	30	500	0.01	2	0.845	0.821	0.551
6	30	500	0.01	3	0.933	0.913	0.558
7	30	500	0.01	4	0.968	0.935	0.585
8	30	500	0.01	5	0.986	0.955	0.634
9	30	500	0.02	2	0.910	0.900	0.583
10	30	500	0.02	3	0.975	0.948	0.592
11	30	500	0.02	4	0.994	0.970	0.619
12	30	500	0.02	5	0.999	0.973	0.610
13	30	1000	0.01	2	0.838	0.820	0.544
14	30	1000	0.01	3	0.927	0.901	0.558
15	30	1000	0.01	4	0.972	0.933	0.572
16	30	1000	0.01	5	0.985	0.951	0.639
17	30	1000	0.01	2	0.907	0.888	0.591
18	30	1000	0.01	3	0.975	0.941	0.594
19	30	1000	0.01	4	0.994	0.959	0.613
20	30	1000	0.01	5	1.000	0.972	0.644
21	30	1000	0.02	2	0.972	0.943	0.649
22	30	1000	0.02	3	0.997	0.966	0.617
23	30	1000	0.02	4	1.000	0.978	0.638
24	30	1000	0.02	5	1.000	0.980	0.644

Here is the result of Random Forest, and its best performance is 0.796. It is higher than what Logistic Regression performed and it didn't show overfitting. In the end, we chose it as our final model.

Model	Parameter				Average FDR (%) at 3%		
Random Forest	Number of Variables	# of Trees	n_jobs	max_depth	Train	Test	OOT
1	30	200	10	30	1.000	0.986	0.726
2	30	200	5	20	1.000	0.980	0.758
3	30	150	10	30	1.000	0.989	0.790
4	30	150	5	20	1.000	0.988	0.748
5	30	100	10	30	1.000	0.986	0.734
6	30	100	5	20	1.000	0.985	0.748
7	25	200	10	30	1.000	0.982	0.731
8	25	200	5	20	1.000	0.985	0.754
9	25	150	10	30	1.000	0.984	0.727
10	25	150	5	20	1.000	0.987	0.796
11	25	100	10	30	1.000	0.983	0.781
12	25	100	5	20	1.000	0.988	0.766

Besides, we also tried the Neural Network. Since the method does feature engineering by itself, we were curious about its performance. We wanted to see if it would outperform tree-based methods.

However, it didn't perform as good as the tree-based methods. Maybe we needed more hyperparameter tuning.

Model	Parameter				Average FDR (%) at 3%		
Neural Net	Number of Variables	Layer	Node	Epoch	Train	Test	OOT
1	30	1	5	20	0.849	0.822	0.490
2	30	1	5	50	0.835	0.799	0.501
3	30	1	10	20	0.924	0.879	0.479
4	30	1	10	50	0.937	0.906	0.495
5	30	1	15	20	0.942	0.896	0.531
6	30	1	15	50	0.963	0.914	0.537
7	30	1	20	20	0.955	0.920	0.513
8	30	1	20	50	0.976	0.918	0.531

In the end of our model exploration, we chose to use:

model = RandomForestClassifier(n_estimators=150,n_jobs =5,max_depth=20, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None) with 30 selected variables.

Final Model Statistics

Our final chosen model is a RandomForestClassifier with 150 trees and 20 max depths. Tables below are result summaries on 3 datasets from top 20% ranked population: training, testing and OOT. The areas shaded in green are results within certain population bins, and the area shaded in blue are cumulative results.

The model fits the train and test data almost perfectly. 100% of bads are caught in the 1% bin on train data and none of the goods are miscaught. For the test data, 99.62% bads are caught within the top 3% bin. However, there is 1 bad never caught within 20% bins.

Moreover, the top 1% bin always catches the highest rate of % bad. For oot, 52% of bads are caught in the 1% bin, and only 0.32% of goods are miscaught. Using the top 10% bin, almost 90% of bads are caught, and only 8.95% goods are miscaught.

Training	# Records	# Goods	# Bads	Fraud Rate
	39424	38995	429	0.010881696

	Bin Statistis					Cumulative Statistis						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Good	% Bads (FDR)	KS	FPR
1	394	0	394	0.00%	100.00%	394	0	394	0.00%	91.84%	91.84	0.00
2	394	359	35	91.12%	8.88%	788	359	429	0.92%	100.00%	99.08	0.46
3	394	394	0	100.00%	0.00%	1182	753	429	1.93%	100.00%	98.07	0.64
4	394	394	0	100.00%	0.00%	1576	1147	429	2.94%	100.00%	97.06	0.73
5	395	395	0	100.00%	0.00%	1971	1542	429	3.95%	100.00%	96.05	0.78
6	394	394	0	100.00%	0.00%	2365	1936	429	4.96%	100.00%	95.04	0.82
7	394	394	0	100.00%	0.00%	2759	2330	429	5.98%	100.00%	94.02	0.84
8	394	394	0	100.00%	0.00%	3153	2724	429	6.99%	100.00%	93.01	0.86
9	395	395	0	100.00%	0.00%	3548	3119	429	8.00%	100.00%	92.00	0.88
10	394	394	0	100.00%	0.00%	3942	3513	429	9.01%	100.00%	90.99	0.89
11	394	394	0	100.00%	0.00%	4336	3907	429	10.02%	100.00%	89.98	0.90
12	394	394	0	100.00%	0.00%	4730	4301	429	11.03%	100.00%	88.97	0.91
13	395	395	0	100.00%	0.00%	5125	4696	429	12.04%	100.00%	87.96	0.92
14	394	394	0	100.00%	0.00%	5519	5090	429	13.05%	100.00%	86.95	0.92
15	394	394	0	100.00%	0.00%	5913	5484	429	14.06%	100.00%	85.94	0.93
16	394	394	0	100.00%	0.00%	6307	5878	429	15.07%	100.00%	84.93	0.93
17	395	395	0	100.00%	0.00%	6702	6273	429	16.09%	100.00%	83.91	0.94
18	394	394	0	100.00%	0.00%	7096	6667	429	17.10%	100.00%	82.90	0.94
19	394	394	0	100.00%	0.00%	7490	7061	429	18.11%	100.00%	81.89	0.94
20	394	394	0	100.00%	0.00%	7884	7455	429	19.12%	100.00%	80.88	0.95

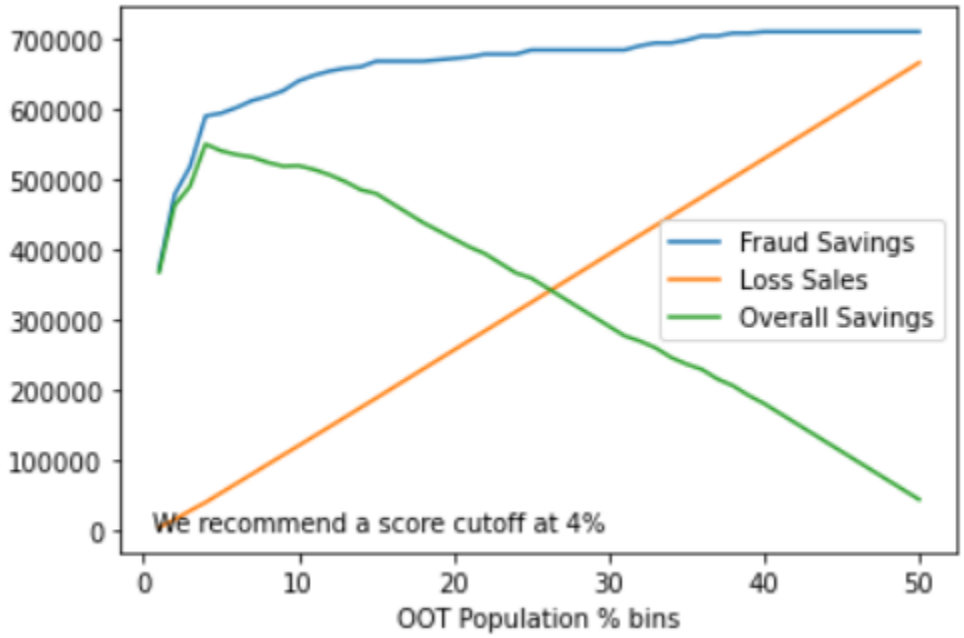
Testing	# Records	# Goods	# Bads	Fraud Rate
	26284	26022	262	0.009968041

	Bin Statistis					Cumulative Statistis						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Good	% Bads (FDR)	KS	FPR
1	262	19	243	7.25%	92.75%	262	19	243	0.07%	92.75%	92.68	0.07
2	263	246	17	93.54%	6.46%	525	265	260	1.02%	99.24%	98.22	0.50
3	263	262	1	99.62%	0.38%	788	527	261	2.03%	99.62%	97.59	0.67
4	263	263	0	100.00%	0.00%	1051	790	261	3.04%	99.62%	96.58	0.75
5	263	263	0	100.00%	0.00%	1314	1053	261	4.05%	99.62%	95.57	0.80
6	263	263	0	100.00%	0.00%	1577	1316	261	5.06%	99.62%	94.56	0.83
7	262	262	0	100.00%	0.00%	1839	1578	261	6.06%	99.62%	93.55	0.86
8	263	263	0	100.00%	0.00%	2102	1841	261	7.07%	99.62%	92.54	0.88
9	263	263	0	100.00%	0.00%	2365	2104	261	8.09%	99.62%	91.53	0.89
10	263	263	0	100.00%	0.00%	2628	2367	261	9.10%	99.62%	90.52	0.90
11	263	263	0	100.00%	0.00%	2891	2630	261	10.11%	99.62%	89.51	0.91
12	263	263	0	100.00%	0.00%	3154	2893	261	11.12%	99.62%	88.50	0.92
13	262	262	0	100.00%	0.00%	3416	3155	261	12.12%	99.62%	87.49	0.92
14	263	263	0	100.00%	0.00%	3679	3418	261	13.14%	99.62%	86.48	0.93
15	263	263	0	100.00%	0.00%	3942	3681	261	14.15%	99.62%	85.47	0.93
16	263	263	0	100.00%	0.00%	4205	3944	261	15.16%	99.62%	84.46	0.94
17	263	263	0	100.00%	0.00%	4468	4207	261	16.17%	99.62%	83.45	0.94
18	263	263	0	100.00%	0.00%	4731	4470	261	17.18%	99.62%	82.44	0.94
19	262	262	0	100.00%	0.00%	4993	4732	261	18.18%	99.62%	81.43	0.95
20	263	263	0	100.00%	0.00%	5256	4995	261	19.20%	99.62%	80.42	0.95

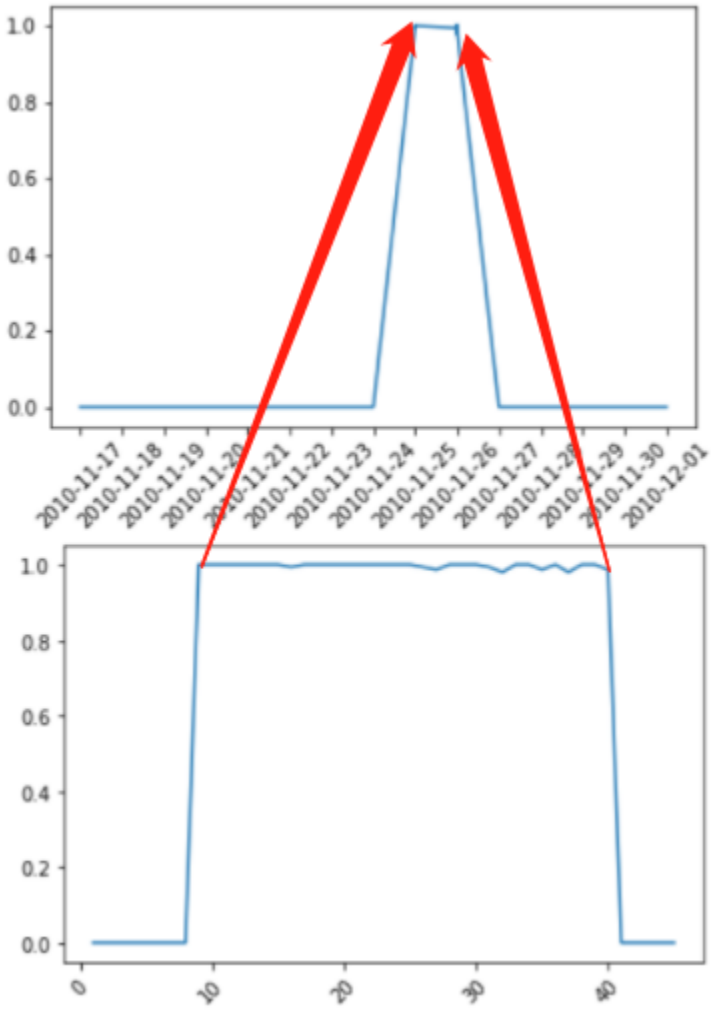
OOT	# Records	# Goods	# Bads	Fraud Rate
	27351	26995	356	0.013015977

	Bin Statistis					Cumulative Statistis						
Population Bin%	# Records	# Goods	# Bads	% Goods	% Bads	Total Records	Cumulative Goods	Cumulative Bads	% Good	% Bads (FDR)	KS	FPR
1	273	87	186	31.87%	68.13%	273	87	186	0.32%	52.25%	51.92	0.32
2	274	221	53	80.66%	19.34%	547	308	239	1.14%	67.13%	65.99	0.56
3	273	253	20	92.67%	7.33%	820	561	259	2.08%	72.75%	70.67	0.68
4	274	238	36	86.86%	13.14%	1094	799	295	2.96%	82.87%	79.91	0.73
5	273	271	2	99.27%	0.73%	1367	1070	297	3.96%	83.43%	79.46	0.78
6	274	270	4	98.54%	1.46%	1641	1340	301	4.96%	84.55%	79.59	0.82
7	273	268	5	98.17%	1.83%	1914	1608	306	5.96%	85.96%	80.00	0.84
8	274	271	3	98.91%	1.09%	2188	1879	309	6.96%	86.80%	79.84	0.86
9	273	269	4	98.53%	1.47%	2461	2148	313	7.96%	87.92%	79.96	0.87
10	274	267	7	97.45%	2.55%	2735	2415	320	8.95%	89.89%	80.94	0.88
11	273	269	4	98.53%	1.47%	3008	2684	324	9.94%	91.01%	81.07	0.89
12	274	271	3	98.91%	1.09%	3282	2955	327	10.95%	91.85%	80.91	0.90
13	273	271	2	99.27%	0.73%	3555	3226	329	11.95%	92.42%	80.47	0.91
14	274	273	1	99.64%	0.36%	3829	3499	330	12.96%	92.70%	79.73	0.91
15	273	269	4	98.53%	1.47%	4102	3768	334	13.96%	93.82%	79.86	0.92
16	274	274	0	100.00%	0.00%	4376	4042	334	14.97%	93.82%	78.85	0.92
17	273	273	0	100.00%	0.00%	4649	4315	334	15.98%	93.82%	77.84	0.93
18	274	274	0	100.00%	0.00%	4923	4589	334	17.00%	93.82%	76.82	0.93
19	273	272	1	99.63%	0.37%	5196	4861	335	18.01%	94.10%	76.09	0.94
20	274	273	1	99.64%	0.36%	5470	5134	336	19.02%	94.38%	75.36	0.94

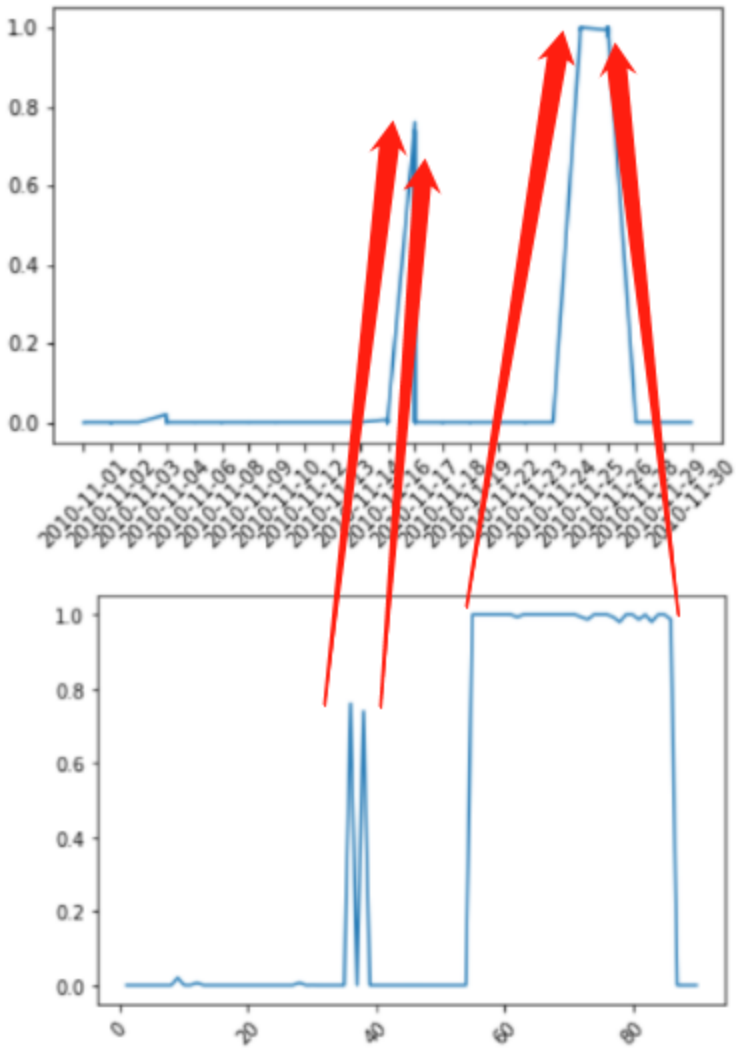
Model Result and Mechanism Explanation



We assumed that every fraud we deny can earn us $2000 and every good record we reject will cause us $50 loss. This graph conveys three types of information: fraud savings, loss sales and overall savings over different percent of OOT population. The overall savings is maximized at 4% population bin with a value of $550050. Therefore, we recommend setting up the cutoff of 4% OOT population.



The graph above shows the fraud analysis about the transactions that happen with a specific credit card. The upper plot describes the fraud score change over time and the lower plot shows the change of the fraud score over different number of transactions. The card number is 5142235211. We found in the data that there were a large number of transactions with this card on November 25th and 26th in 2010, about 17 on the 25th and 15 on the 26th. The fraud score grows rapidly as the number of transactions rises. It is likely to be a stolen credit card. As shown in the lower graph, as the number of transactions passes a threshold, the fraud score is highered by the model and many of the transactions are marked as fraudulent. As we previously stated, we included 30 amount variables in our model. Those variables, especially the 'act/mean' ones, measure the irregularity about the amount of transactions of a credit card over a past time window. Since a large number of transactions gather around these 2 days, the variables detect that the amount of transactions is unusual compared to that during the normal time.



The graph above shows the fraud analysis about the transactions that happen with a specific merchant. The merchant number is 4353000719908. The upper plot describes the fraud score change over time and the lower plot shows the change of the fraud score over different number of transactions. As shown in the upper plot, the fraud score has two peaks around two time periods. We took a look at the data and found a large number of transactions happened with different credit cards during these two days. About 90 transactions happened this month. 17 were on the 25th and 15 were on the 26th. It is possible that some employees of the merchant secretly swiped different credit cards. The lower plot indicates that as the number of transactions falls into a certain range, the transactions get marked as fraudulent. Different amount variables across different time windows examine the record and detect the irregularity of the amount during these time periods.

Conclusions

For this project, before we start, we first did a data quality report to explore how the data looks like. Then, we developed a high-level plan for this project. To begin with the project, we found a mechanism based on the one-to-one relationship between the three missing fields and the merchant description field to fill in the missing fields. After we got a complete dataset, we created 358 candidate variables according to our knowledge of the fraudulent behavior related to credit card transactions. Subsequent to feature engineering, we filtered all the candidate variables through the average rank of their univariate KS score and the FDR at 3% and acquired top 80 ranked variables. We put the 80 variables in a wrapper process, which forward selects the variables using the Decision Tree Classifier as the base model and the Sequential Feature Selection. After the feature selection procedures, we finalize our 30 selected variables for model building. We splitted the dataset into 3 parts: training, testing, and the OOT. To balance the classes in these 3 samples, we chose 4 months data as the OOT and used 40% test size to randomly split the training and testing data. We tried Logistic Regression, Random Forest, Boosted Tree(XGBoost), and Neural Network algorithms. Comparing different performance across different model and hyperparameter choices, we chose the Random Forest model with 150 trees and 20 max depths as our final model. It hits 79.6% OOT FDR at 3%.

For the next step, we will investigate more into real-world credit card fraudulent problems. Therefore, we will be able to create better candidate variables that better understand the nature of credit card fraud. In addition, we will research more about machine learning algorithms and try more hyperparameter tuning.

Appendix
Data Quality Report

High-level Description

The dataset is called “Card Transactions”. It contains actual credit cards transactions information from January 1, 2010 to December 31, 2010. It has 10 fields and 96,753 records which come from a US government organization.

Summary Table

Categorical Fields:

Column Name	# of Records	% Populated	# Zeros	Unique Values	Most Common Field Value
Recnum	96753	100	0	96753	N/A
Cardnum	96753	100	0	1645	5142148452
Merchnum	96753	96.51	0	13092	930090121224
Merch description	96753	100	0	13126	GSA-FSS-ADV
Merch state	96753	98.76	0	228	TN
Merch zip	96753	95.19	0	4568	38118
Transtype	96753	100	0	4	P
Fraud	96753	100	95694	2	0

Numerical Fields:

Column Name	% Populated	Unique Values	Most Common Field Value	Mean	Std	Min	Max
Amount	100	34909	3.62	427.89	10006.14	0.01	3102045.53

Datetime:

Column Name	# of Records	% Populated	# Zeros	Min	Max	Most Common
Date	96753	100	0	20100101	20101231	20100228

Data Field Exploration:

Field 1:

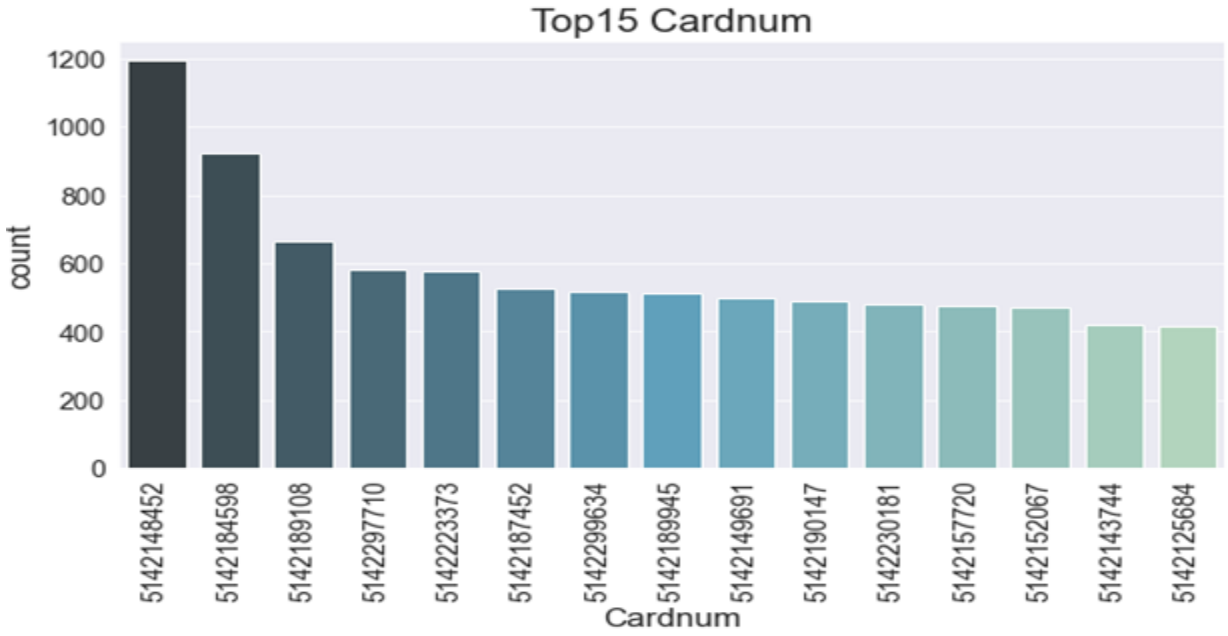
Name: Recnum

Description: Unique identifier of each entry in the data.

Field 2:

Name: Cardnum

Description: The identifier of a specific credit card.

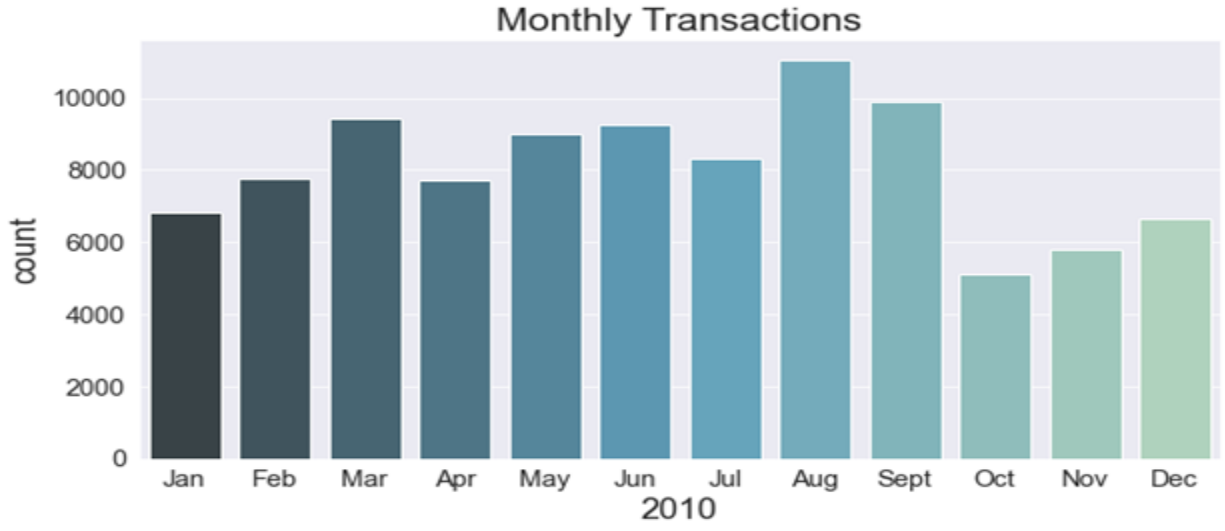


Field 3:

Name: Date

Description: The specific date when the credit card transaction happened.

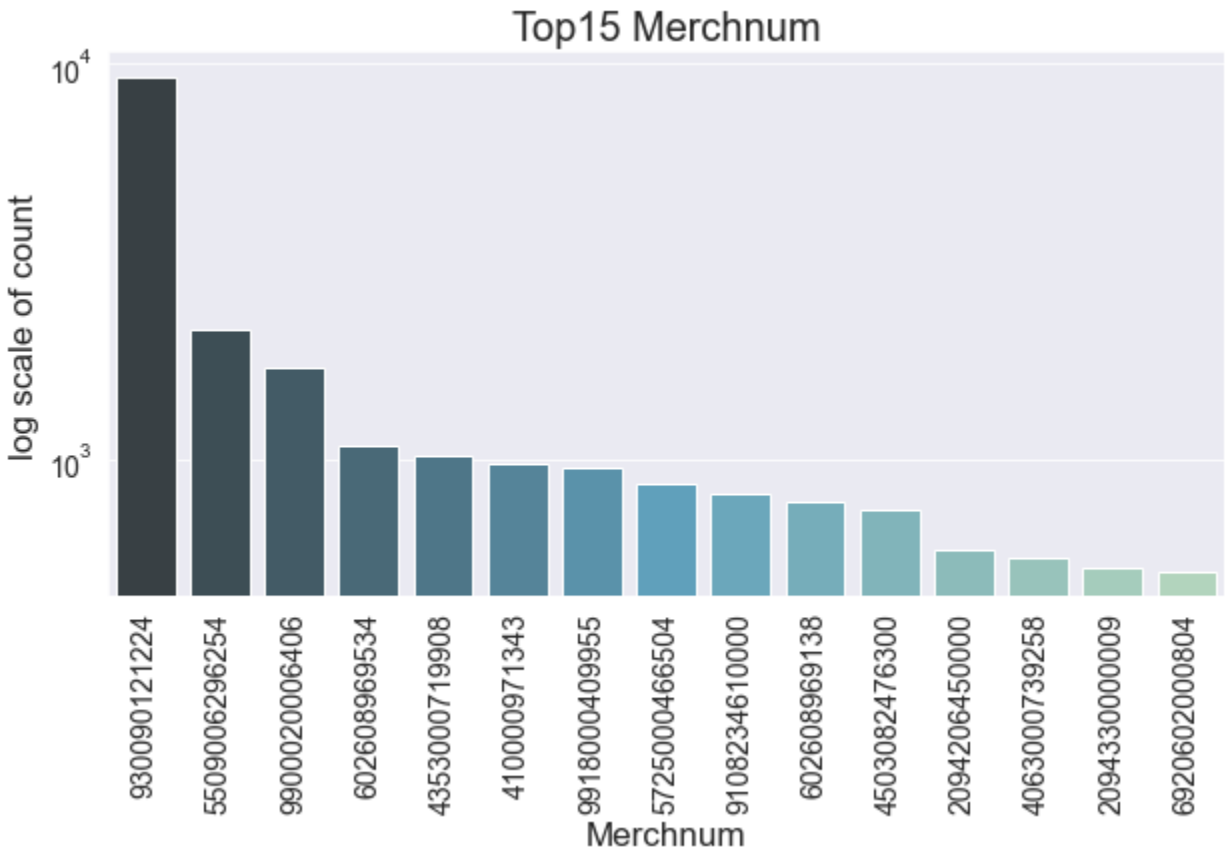
The following bar chart is grouped by months.



Field 4:

Name: Merchnum

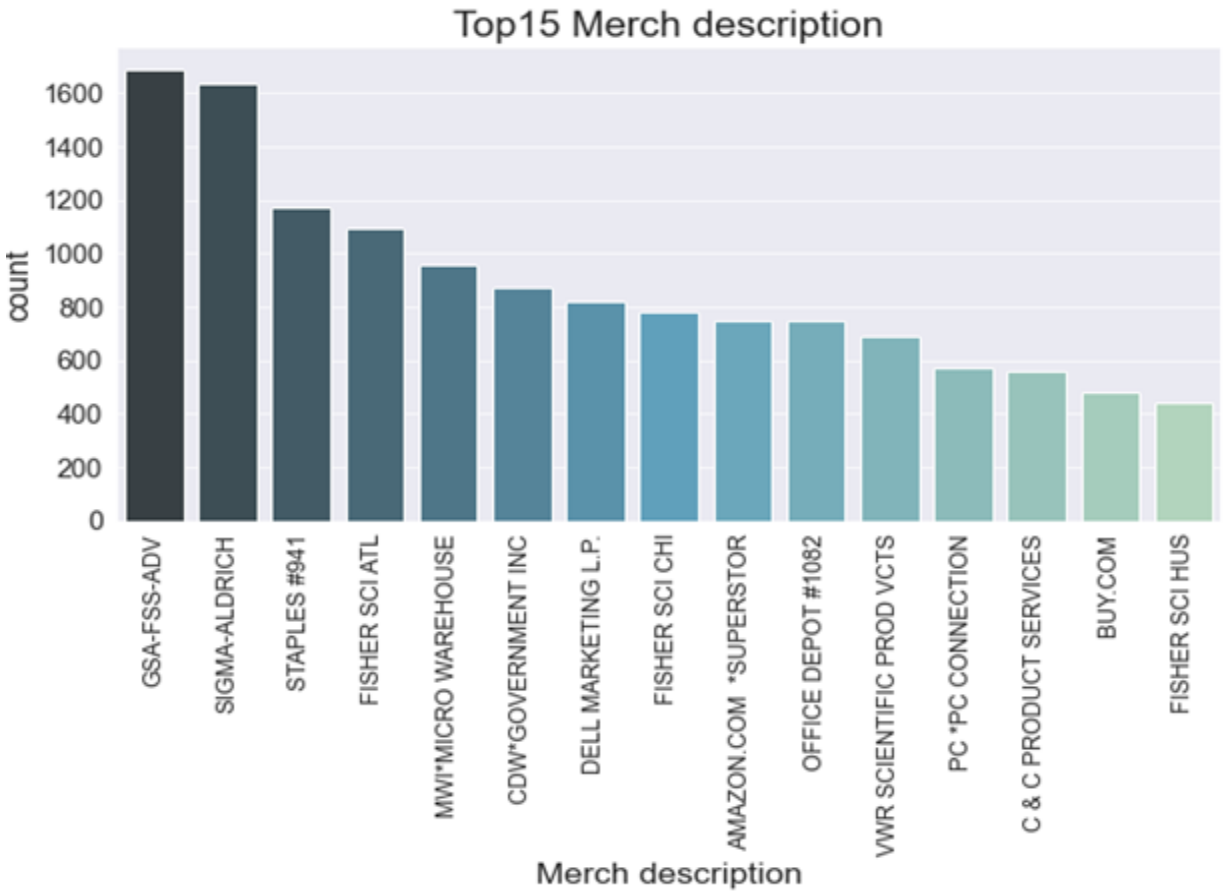
Description: The identifier of a specific merchant.



Field 5:

Name: Merch description

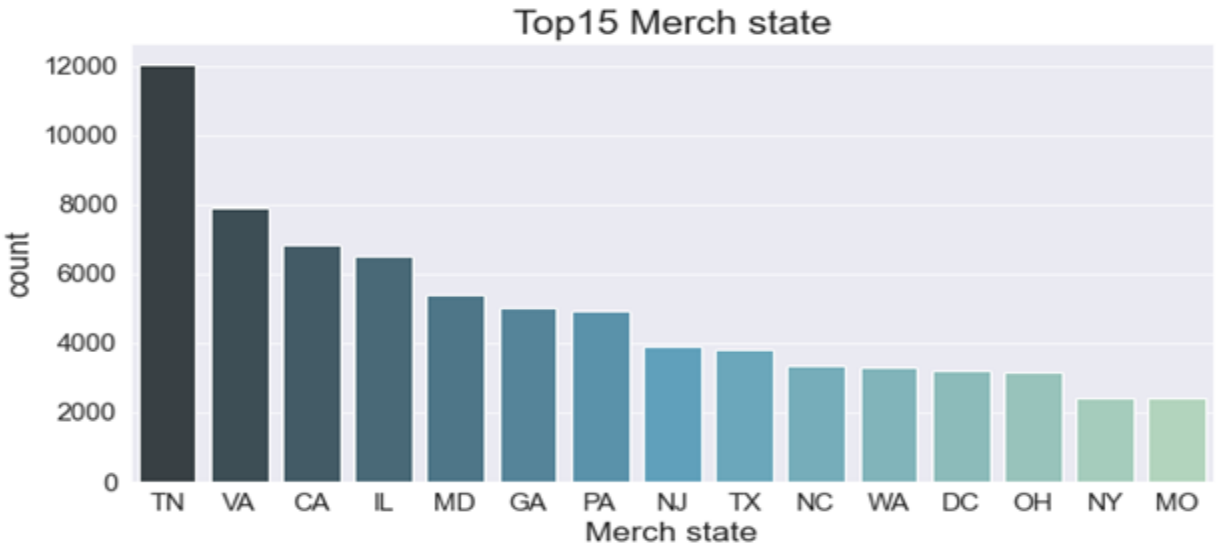
Description: The text description of the merchant.



Field 6:

Name: Merch state

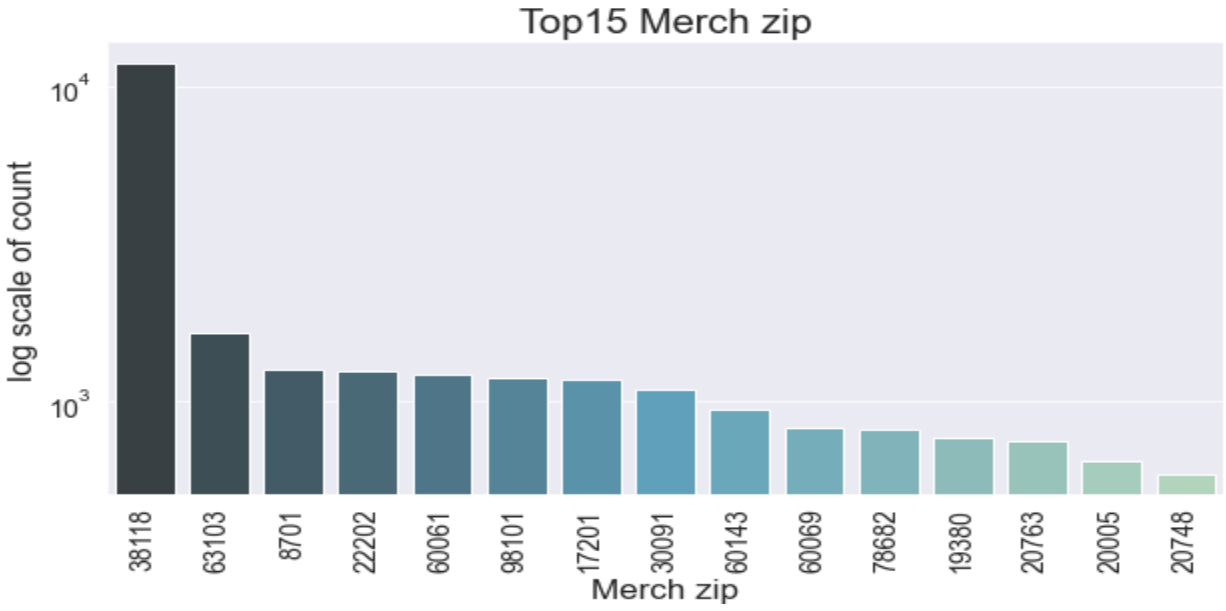
Description: The abbreviation of the state where the transaction happened. Or, the abbreviation of the state that the headquarter of the merchant's company is in.



Field 7:

Name: Merch zip

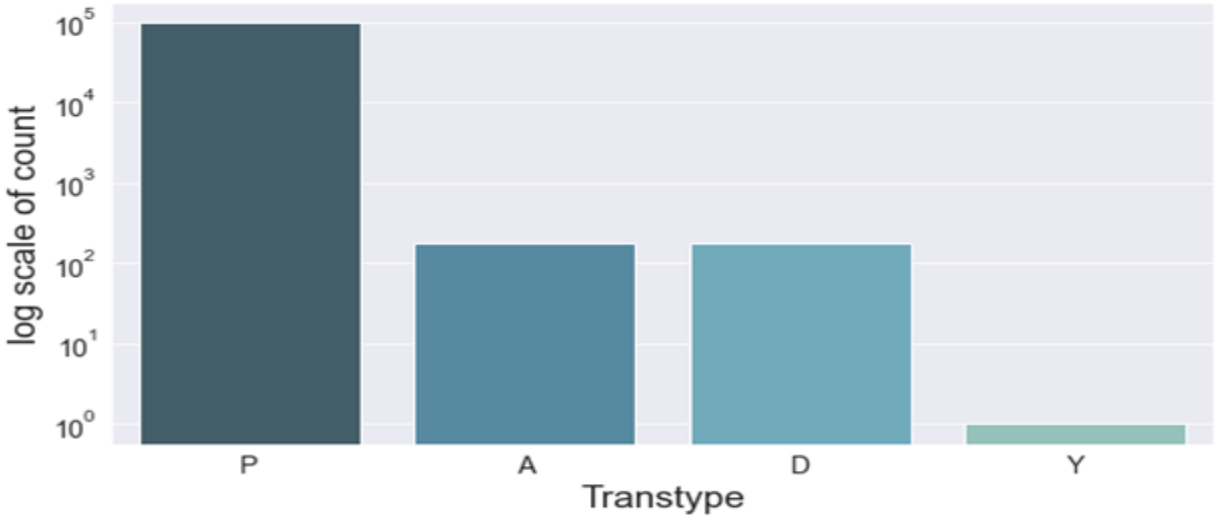
Description: The zip code of the place where the transaction happened. Or, the zip code of the place that the headquarter of the merchant's company is in.



Field 8:

Name: Transtype

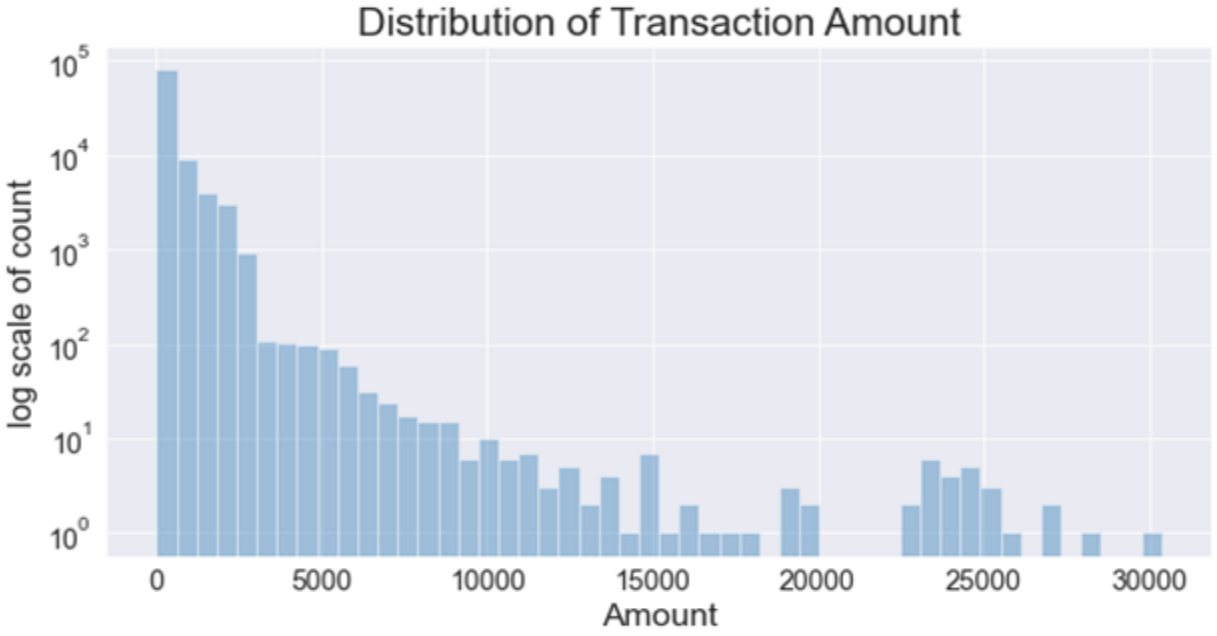
Description: The type of the transaction.



Field 9:

Name: Amount

Description: The dollar amount of the transaction. Record 52714 is excluded because it is an extreme value of 3102045.53. And, record 47339 is also excluded because it is an outlier.



Field 10:

Name: Fraud

Description: Whether the record is fraudulent or not. 0 represents that the record is not fraudulent while 1 represents that the record is fraudulent.

In this dataset, 95694 (98.91%) records have a value of 0 and 1059 (1.09%) records have a value of 1.

Top20 cardholder with the highest U*:

	Cardnum	U*
0	5142253356	13.371
1	5142299705	9.028
2	5142197563	8.151
3	5142194617	6.020
4	5142288241	5.534
5	5142239140	4.835
6	5142144931	4.559
7	5142192606	4.062
8	5142204384	4.039
9	5142284940	3.785
10	5142189113	3.632
11	5142225308	3.535
12	5142116864	3.532
13	5142293257	3.465
14	5142173286	3.465
15	5142246929	3.463
16	5142224699	3.251
17	5142847398	3.193
18	5142273608	3.154
19	5142147267	3.152

Top20 merchant with the highest U*:

	Merchnum	U*
0	991808369338	165.146
1	8078200641472	64.664
2	308904389335	48.358
3	3523000628102	37.200
4	808998385332	32.826
5	55158027	29.222
6	8916500620062	28.154
7	3910694900001	26.742
8	8889817332	25.432
9	881145544	25.104
10	5600900060992	24.329
11	6844000608436	23.635
12	5803301245621	21.070
13	92891948003	20.907
14	3433000017263	19.363
15	467615916337	18.387
16	817004638227	17.675
17	2376700063599	16.387
18	993620816222	14.741
19	993620810220	13.869

List of Candidate Variables:

Cardnum_day_since
Cardnum_count_0
ave_amount_Cardnum_count_0
max_amount_Cardnum_count_0
sum_amount_Cardnum_count_0
median_amount_Cardnum_count_0
Cardnum_count_1
ave_amount_Cardnum_count_1
max_amount_Cardnum_count_1
sum_amount_Cardnum_count_1
median_amount_Cardnum_count_1
Cardnum_count_3
ave_amount_Cardnum_count_3
max_amount_Cardnum_count_3
sum_amount_Cardnum_count_3
median_amount_Cardnum_count_3
Cardnum_count_7
ave_amount_Cardnum_count_7
max_amount_Cardnum_count_7
sum_amount_Cardnum_count_7
median_amount_Cardnum_count_7
Cardnum_count_14
ave_amount_Cardnum_count_14
max_amount_Cardnum_count_14
sum_amount_Cardnum_count_14
median_amount_Cardnum_count_14
Cardnum_count_30
ave_amount_Cardnum_count_30
max_amount_Cardnum_count_30
sum_amount_Cardnum_count_30
median_amount_Cardnum_count_30
Merchnum_day_since
Merchnum_count_0
ave_amount_Merchnum_count_0
max_amount_Merchnum_count_0
sum_amount_Merchnum_count_0
median_amount_Merchnum_count_0
Merchnum_count_1
ave_amount_Merchnum_count_1
max_amount_Merchnum_count_1
sum_amount_Merchnum_count_1
median_amount_Merchnum_count_1
Merchnum_count_3
ave_amount_Merchnum_count_3

max_amount_Merchnum_count_3
sum_amount_Merchnum_count_3
median_amount_Merchnum_count_3
Merchnum_count_7
ave_amount_Merchnum_count_7
max_amount_Merchnum_count_7
sum_amount_Merchnum_count_7
median_amount_Merchnum_count_7
Merchnum_count_14
ave_amount_Merchnum_count_14
max_amount_Merchnum_count_14
sum_amount_Merchnum_count_14
median_amount_Merchnum_count_14
Merchnum_count_30
ave_amount_Merchnum_count_30
max_amount_Merchnum_count_30
sum_amount_Merchnum_count_30
median_amount_Merchnum_count_30
card_merchnum_day_since
card_merchnum_count_0
ave_amount_card_merchnum_count_0
max_amount_card_merchnum_count_0
sum_amount_card_merchnum_count_0
median_amount_card_merchnum_count_0
card_merchnum_count_1
ave_amount_card_merchnum_count_1
max_amount_card_merchnum_count_1
sum_amount_card_merchnum_count_1
median_amount_card_merchnum_count_1
card_merchnum_count_3
ave_amount_card_merchnum_count_3
max_amount_card_merchnum_count_3
sum_amount_card_merchnum_count_3
median_amount_card_merchnum_count_3
card_merchnum_count_7
ave_amount_card_merchnum_count_7
max_amount_card_merchnum_count_7
sum_amount_card_merchnum_count_7
median_amount_card_merchnum_count_7
card_merchnum_count_14
ave_amount_card_merchnum_count_14
max_amount_card_merchnum_count_14
sum_amount_card_merchnum_count_14
median_amount_card_merchnum_count_14
card_merchnum_count_30
ave_amount_card_merchnum_count_30

max_amount_card_merchnum_count_30
sum_amount_card_merchnum_count_30
median_amount_card_merchnum_count_30
card_zip_day_since
card_zip_count_0
ave_amount_card_zip_count_0
max_amount_card_zip_count_0
sum_amount_card_zip_count_0
median_amount_card_zip_count_0
card_zip_count_1
ave_amount_card_zip_count_1
max_amount_card_zip_count_1
sum_amount_card_zip_count_1
median_amount_card_zip_count_1
card_zip_count_3
ave_amount_card_zip_count_3
max_amount_card_zip_count_3
sum_amount_card_zip_count_3
median_amount_card_zip_count_3
card_zip_count_7
ave_amount_card_zip_count_7
max_amount_card_zip_count_7
sum_amount_card_zip_count_7
median_amount_card_zip_count_7
card_zip_count_14
ave_amount_card_zip_count_14
max_amount_card_zip_count_14
sum_amount_card_zip_count_14
median_amount_card_zip_count_14
card_zip_count_30
ave_amount_card_zip_count_30
max_amount_card_zip_count_30
sum_amount_card_zip_count_30
median_amount_card_zip_count_30
card_state_day_since
card_state_count_0
ave_amount_card_state_count_0
max_amount_card_state_count_0
sum_amount_card_state_count_0
median_amount_card_state_count_0
card_state_count_1
ave_amount_card_state_count_1
max_amount_card_state_count_1
sum_amount_card_state_count_1
median_amount_card_state_count_1
card_state_count_3

ave_amount_card_state_count_3
max_amount_card_state_count_3
sum_amount_card_state_count_3
median_amount_card_state_count_3
card_state_count_7
ave_amount_card_state_count_7
max_amount_card_state_count_7
sum_amount_card_state_count_7
median_amount_card_state_count_7
card_state_count_14
ave_amount_card_state_count_14
max_amount_card_state_count_14
sum_amount_card_state_count_14
median_amount_card_state_count_14
card_state_count_30
ave_amount_card_state_count_30
max_amount_card_state_count_30
sum_amount_card_state_count_30
median_amount_card_state_count_30
act/mean_amount_Cardnum_count_0
act/max_amount_Cardnum_count_0
act/sum_amount_Cardnum_count_0
act/median_amount_Cardnum_count_0
act/mean_amount_Cardnum_count_1
act/max_amount_Cardnum_count_1
act/sum_amount_Cardnum_count_1
act/median_amount_Cardnum_count_1
act/mean_amount_Cardnum_count_3
act/max_amount_Cardnum_count_3
act/sum_amount_Cardnum_count_3
act/median_amount_Cardnum_count_3
act/mean_amount_Cardnum_count_7
act/max_amount_Cardnum_count_7
act/sum_amount_Cardnum_count_7
act/median_amount_Cardnum_count_7
act/mean_amount_Cardnum_count_14
act/max_amount_Cardnum_count_14
act/sum_amount_Cardnum_count_14
act/median_amount_Cardnum_count_14
act/mean_amount_Cardnum_count_30
act/max_amount_Cardnum_count_30
act/sum_amount_Cardnum_count_30
act/median_amount_Cardnum_count_30
act/mean_amount_Merchnum_count_0
act/max_amount_Merchnum_count_0
act/sum_amount_Merchnum_count_0

act/median_amount_Merchnum_count_0
act/mean_amount_Merchnum_count_1
act/max_amount_Merchnum_count_1
act/sum_amount_Merchnum_count_1
act/median_amount_Merchnum_count_1
act/mean_amount_Merchnum_count_3
act/max_amount_Merchnum_count_3
act/sum_amount_Merchnum_count_3
act/median_amount_Merchnum_count_3
act/mean_amount_Merchnum_count_7
act/max_amount_Merchnum_count_7
act/sum_amount_Merchnum_count_7
act/median_amount_Merchnum_count_7
act/mean_amount_Merchnum_count_14
act/max_amount_Merchnum_count_14
act/sum_amount_Merchnum_count_14
act/median_amount_Merchnum_count_14
act/mean_amount_Merchnum_count_30
act/max_amount_Merchnum_count_30
act/sum_amount_Merchnum_count_30
act/median_amount_Merchnum_count_30
act/mean_amount_card_merchnum_count_0
act/max_amount_card_merchnum_count_0
act/sum_amount_card_merchnum_count_0
act/median_amount_card_merchnum_count_0
act/mean_amount_card_merchnum_count_1
act/max_amount_card_merchnum_count_1
act/sum_amount_card_merchnum_count_1
act/median_amount_card_merchnum_count_1
act/mean_amount_card_merchnum_count_3
act/max_amount_card_merchnum_count_3
act/sum_amount_card_merchnum_count_3
act/median_amount_card_merchnum_count_3
act/mean_amount_card_merchnum_count_7
act/max_amount_card_merchnum_count_7
act/sum_amount_card_merchnum_count_7
act/median_amount_card_merchnum_count_7
act/mean_amount_card_merchnum_count_14
act/max_amount_card_merchnum_count_14
act/sum_amount_card_merchnum_count_14
act/median_amount_card_merchnum_count_14
act/mean_amount_card_merchnum_count_30
act/max_amount_card_merchnum_count_30
act/sum_amount_card_merchnum_count_30
act/median_amount_card_merchnum_count_30
act/mean_amount_card_zip_count_0

act/max_amount_card_zip_count_0
act/sum_amount_card_zip_count_0
act/median_amount_card_zip_count_0
act/mean_amount_card_zip_count_1
act/max_amount_card_zip_count_1
act/sum_amount_card_zip_count_1
act/median_amount_card_zip_count_1
act/mean_amount_card_zip_count_3
act/max_amount_card_zip_count_3
act/sum_amount_card_zip_count_3
act/median_amount_card_zip_count_3
act/mean_amount_card_zip_count_7
act/max_amount_card_zip_count_7
act/sum_amount_card_zip_count_7
act/median_amount_card_zip_count_7
act/mean_amount_card_zip_count_14
act/max_amount_card_zip_count_14
act/sum_amount_card_zip_count_14
act/median_amount_card_zip_count_14
act/mean_amount_card_zip_count_30
act/max_amount_card_zip_count_30
act/sum_amount_card_zip_count_30
act/median_amount_card_zip_count_30
act/mean_amount_card_state_count_0
act/max_amount_card_state_count_0
act/sum_amount_card_state_count_0
act/median_amount_card_state_count_0
act/mean_amount_card_state_count_1
act/max_amount_card_state_count_1
act/sum_amount_card_state_count_1
act/median_amount_card_state_count_1
act/mean_amount_card_state_count_3
act/max_amount_card_state_count_3
act/sum_amount_card_state_count_3
act/median_amount_card_state_count_3
act/mean_amount_card_state_count_7
act/max_amount_card_state_count_7
act/sum_amount_card_state_count_7
act/median_amount_card_state_count_7
act/mean_amount_card_state_count_14
act/max_amount_card_state_count_14
act/sum_amount_card_state_count_14
act/median_amount_card_state_count_14
act/mean_amount_card_state_count_30
act/max_amount_card_state_count_30
act/sum_amount_card_state_count_30

act/median_amount_card_state_count_30
Cardnum_count_0_by_3
Cardnum_amount_0_by_3
Cardnum_count_0_by_7
Cardnum_amount_0_by_7
Cardnum_count_0_by_14
Cardnum_amount_0_by_14
Cardnum_count_0_by_30
Cardnum_amount_0_by_30
Cardnum_count_1_by_3
Cardnum_amount_1_by_3
Cardnum_count_1_by_7
Cardnum_amount_1_by_7
Cardnum_count_1_by_14
Cardnum_amount_1_by_14
Cardnum_count_1_by_30
Cardnum_amount_1_by_30
Merchnum_count_0_by_3
Merchnum_amount_0_by_3
Merchnum_count_0_by_7
Merchnum_amount_0_by_7
Merchnum_count_0_by_14
Merchnum_amount_0_by_14
Merchnum_count_0_by_30
Merchnum_amount_0_by_30
Merchnum_count_1_by_3
Merchnum_amount_1_by_3
Merchnum_count_1_by_7
Merchnum_amount_1_by_7
Merchnum_count_1_by_14
Merchnum_amount_1_by_14
Merchnum_count_1_by_30
Merchnum_amount_1_by_30
card_merchnum_count_0_by_3
card_merchnum_amount_0_by_3
card_merchnum_count_0_by_7
card_merchnum_amount_0_by_7
card_merchnum_count_0_by_14
card_merchnum_amount_0_by_14
card_merchnum_count_0_by_30
card_merchnum_amount_0_by_30
card_merchnum_count_1_by_3
card_merchnum_amount_1_by_3
card_merchnum_count_1_by_7
card_merchnum_amount_1_by_7
card_merchnum_count_1_by_14

card_merchnum_amount_1_by_14
card_merchnum_count_1_by_30
card_merchnum_amount_1_by_30
card_zip_count_0_by_3
card_zip_amount_0_by_3
card_zip_count_0_by_7
card_zip_amount_0_by_7
card_zip_count_0_by_14
card_zip_amount_0_by_14
card_zip_count_0_by_30
card_zip_amount_0_by_30
card_zip_count_1_by_3
card_zip_amount_1_by_3
card_zip_count_1_by_7
card_zip_amount_1_by_7
card_zip_count_1_by_14
card_zip_amount_1_by_14
card_zip_count_1_by_30
card_zip_amount_1_by_30
card_state_count_0_by_3
card_state_amount_0_by_3
card_state_count_0_by_7
card_state_amount_0_by_7
card_state_count_0_by_14
card_state_amount_0_by_14
card_state_count_0_by_30
card_state_amount_0_by_30
card_state_count_1_by_3
card_state_amount_1_by_3
card_state_count_1_by_7
card_state_amount_1_by_7
card_state_count_1_by_14
card_state_amount_1_by_14
card_state_count_1_by_30
card_state_amount_1_by_30
dow_risk
benford_card
benford_merch