FRAUD DETECTION ALGORITHMS

Credit Card Transaction Dataset

Group 205
Seng Fong Chang, Xiaohan Chen (Miranda), Yiyang Duanmu
Muchan (Catherine) Gao, Derek He

May 04, 2022

Table of Contents

Executive Summary

Ever since credit cards were introduced in the 1950s, card fraud has increased to a large scale that needs attention from individuals, organizations, and governments. Per 2019 The Nilson Report, credit card transaction fraud will reach approximately $40 billion per year in 2027. Aite Group has also reported in 2016 that 47% of consumers have experienced card fraud in the past five years. Although, with the introduction of chips, card-present fraud losses have declined, card-not-present fraud has increased dramatically since then. In this project, we intend to build a supervised machine learning model to identify credit card transaction fraud to help stakeholders prevent future financial losses.

Our original data consists of information about 96,753 transactions happening between January 1st, 2006, to December 31st, 2006. There are in total 10 fields for each transaction including information about the credit card and merchant involved in the transaction.

We start off by an exploratory data analysis and identify outliers and missing values from there. We then clean and prepare the data for building our models by filtering out all the non-purchase transactions and filling in missing values using mode of other available data.

With the cleaned data, we create four types of variables: amount variables, frequency variables, days-since variables and velocity change variables using the entities and combinations of entities from the original data or built from the original data. In the end of the feature engineering step, we have created a total of 1,058 variables. In the feature selection step, we use a univariate filter followed by a wrapper. The filter helps us filter out highly correlated variables so that we can use wrapper on a much smaller pool of variables. We then use LightGBM when building the wrapper and only kept 30 variables that have the highest score and are useful for our analysis. We then elaborately select our final 6 variables from them.

Using our 6 variables, we explore several models including Logistics Regression, Decision Tree, Random Forest, LightGBM, XGBoost, CatBoost and Neural Network to evaluate the performance of each model on training, testing, and out-of-time (OOT) data with adjustments in hyperparameters. After evaluating the fraud detection rate at 3% of all models with different adjusted hyperparameters, we decide to use CatBoost as our final model considering its accuracy in predicting OOT data.

Description of Data

The card transaction dataset contains real credit card transaction records from U.S. government institutions. The dataset covers the time from January 1st, 2006 to December 31st, 2006, with ten fields and in total 96,753 records.

Following tables show a preliminary summary statistic of the original fields.

Numerical Fields

Field Name	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100	2006-01-01	2006-12-31	N/A	N/A	0
Amount	100	0.01	3,102,045.53	427.89	10,006.14	0

Categorical Fields

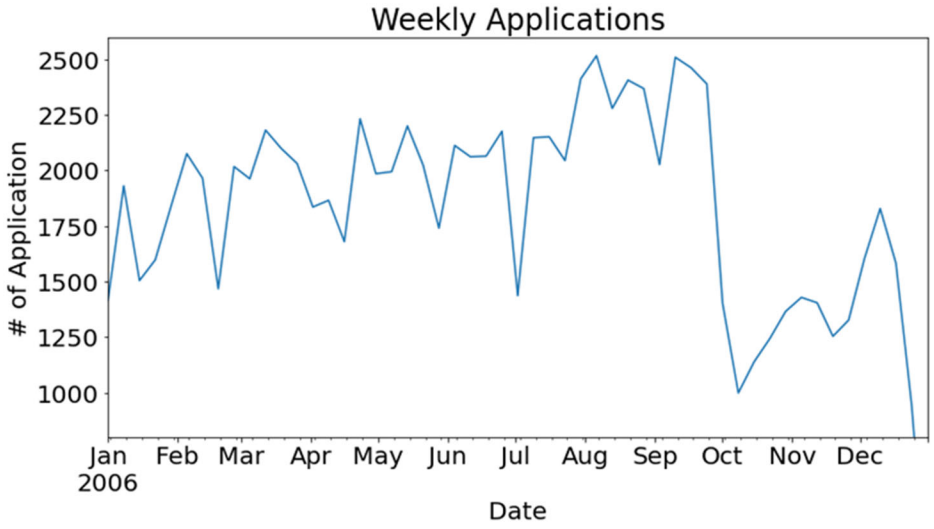
Field Name	% Populated	# Unique Values	Most Common Value
Recnum	100	96,753	N/A
Cardnum	100	1,645	5142148452
Merchnum	96.51	13,092	930090121224
Merch description	100	13,126	GSA-FSS-ADV
Merch state	98.76	228	TN
Merch zip	95.19	4,568	38118
Transtype	100	4	P
Fraud	100	2	0

Distributions

Following are some key distributions of the original dataset.

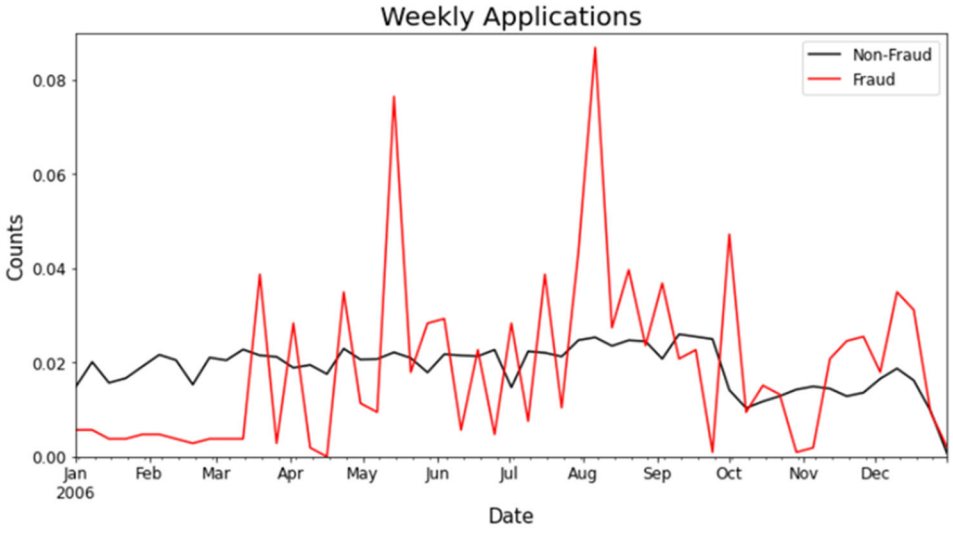
Date

Represents the date of the application. In this dataset, the date ranges from 2006-01-01 to 2006-12-31. Graph below shows the number of transactions per week.



Fraud

Represents the fraud label of the transaction and ranges from 2006-01-01 to 2006-12-31. Graph below shows the number of fraudulent activities per week.



Data Cleaning

In the data cleaning section, we do data manipulations after completing exploratory data analysis to handle exclusions, missing values, outliers, and frivolous values.

We first build the '**cleaning**' function to fill null values in several columns in the following step. The function groups the data by 'i' column, finds mode of 'j' column where the records share the same 'i' column value and fills null values in 'j' column with the mode.

Date

In the dataset, the date value is defined as object type, and our approach is to convert them into datetime datatype. We use the pandas to _datetime function with a lambda function to achieve the datatype conversion.

Cardnum

The cardnum variable in the dataset refers to the credit card number. It is defined as int64, and our approach is to convert them into string datatype. We use '. astype' to achieve the datatype conversion.

Merchnum

The merchnum variable in the dataset refers to the merchant number. We notice many null values, so we replace zeros as nan (not a number) using the '. replace' function. Before we fill in the null values with "unknown", we use the cleaning function first. Hence, we can find a mode of merchnum and fill in the null values with it. For example, we have many merchnum values for the same merch description, and these values are filled with the mode. Then, for the rest of the null values, we replace them with "unknown".

Merch state

The merch state variable in the dataset refers to the state the merchant located. We implement the cleaning function to find the mode of merch state and fill in the null values with the mode. We first group by the merch zip and find the mode of the merch state to fill in the null value. We then manually fill in the null values where zip code is available to identify the state. There are four zip codes (86899, 23080, 48700, 50823) out of the United States, so we classify them as 'Foreign'. We repeat the imputation using merchnum and merch description to fill the rest of the null values. Finally, we replace the rest of the null values with "unknown".

Transtype

The transtype variable in the dataset refers to transaction type, and we filter out other transactions and keep only purchase, represented by transaction type 'P', for the purpose of analysis.

Merch zip

The merch zip variable in the dataset refers to the merchant zip code. We implement the cleaning function to find the mode of merch zip and fill in the null values with the mode. We repeat the imputation for merchnum and merch description in order and replace the rest of the null values with “unknown”.

Candidate Variables

Credit card transaction fraud often happens in three scenarios. First, someone steals a credit card or finds and uses a lost card. Second, someone at a merchant could "steal" many credit cards by making several counterfeit cards or having skimmers at gas stations, ATMs, and/or other merchants. Third, someone gets the username and password for an online account and therefore gets the credit card information. Based on these scenarios, we identify several characteristics of potential transaction fraud including but not limited to the burst of activities, larger than normal purchase amounts, increased usage in card-not-present, activities at new merchants, high-risk merchant or new geography, and fictitious transactions or merchants.

Based on the above logic, we create four types of candidate variables:

- **Amount** variables indicate differently calculated amounts by an entity or combination of the entities over the past n days where n = 0, 1, 3, 7, 14, 30 and amount is average, maximum, median, total, actual/average, actual/maximum, actual/median and actual/total,
- **Frequency** variables indicate the number of transactions with an entity or combination of the entities over the past n days where n = 0, 1, 3, 7, 14, 30,
- **Days-since** variables indicate how many days since an entity or combination of the entities last appeared in the dataset, and
- **Velocity** change variables indicate the ratio of the short-term frequency where n = 0, 1 to a longer-term frequency where n = 3, 7, 14, 30.

To simplify the code, we rename all the variables. We rename 'Date', 'Recnum', 'Cardnum', 'Merchnum', 'Merch description', 'Merch state', 'Merch zip' and 'Amount' to 'date', 'recnum', 'cardnum', 'merchnum', 'description', 'state', 'zip5' and 'amount' respectively.

Entities

Specifically, the basic identity information from the dataset includes cardnum, merchnum, description, state and zip5. We first create another basic characteristic variable – country – based on state. Countries in the dataset include "US" for the United States, "Canada " for Canada, "Foreign" for any other country and "unknown" for missing values.

We then create the following relevant fields by combining the original data to have more entities representing the transactions' characteristics:

- 'state_zip5_country': combination of state, zip5, and country
- 'merchnum_country': combination of merchnum and country
- 'description_country': combination of description and country
- 'description_state_zip5_country': combination of description, state, zip5, and country
- 'cardnum_merchnum': combination of cardnum and merchnum

Additionally, we combine cardnum and merchnum with description, state, country, state_zip5_country, merchnum_country, description_country and description_state_zip5_country respectively.

Risk Table Variables

We also create two risk table variables, one for day of week and one for state, based on the date and state of the transaction respectively using data before 2006-11-01 (training data) to transform the categorical field to a numerical field. Specifically, we use the Bayesian method to encode these two fields and use the average of the Fraud for all records in each day of week/state as the numeric representative of the categorical field. We finalize these two fields by smoothing the value to make sure it smoothly transits between the low number to the high number.

Benford's Law variables

Our last type of variable is Benford's Law variable where we quantify the difference of the first digit distribution for each cardholder and merchant from Benford's Law Probabilities. Benford's Law is the observation that the first digit of many measurements is not uniformly distributed and follows a specific distribution. We use Benford's Law to detect fraud where a potential fraudster is making up a large amount of numbers.

Since we may not have enough records for every cardholder or merchant to have ten bins, we divided the first digit distribution into two bins, a low bin with 1 and 2 and a high bin with 3 through 9. We then calculate a measure of unusualness U based on Benford's Law distribution and smoothed the value as U*. We exclude transactions from Fedex in this process since they violate Benford's Law but are not unusual. The top 40 potential cardholder fraudsters and merchant fraudsters based on Benford's law are listed in Appendix II for future reference.

Summary of All Candidates Variables

The final entity list is ['cardnum', 'merchnum', 'state_zip5_country', 'merchnum_country', 'description_country', 'description_state_zip5_country', 'cardnum_merchnum', 'cardnum_country', 'cardnum_state_zip5_country', 'cardnum_merchnum_country', 'cardnum_description_country', 'cardnum_description_state_zip5_country', 'merchnum_description', 'merchnum_state', 'merchnum_state_zip5_country', 'merchnum_description_country', 'merchnum_description_state_zip5_country']. For each entity in this list, we created the amount, frequency, day-since and velocity change variables.

Following table summarizes all the candidate variables we created.

Description	# Of Variables Created
Basic characteristics of transactions	17
Amount by/at that entity over the past n days. Amounts are [average, maximum, median, total, actual/average, actual/maximum, actual/median, actual/total]; n is {0,1,3,7,14,30}.	816
# Transactions with that entity over the past n days. n is {0,1,3,7,14,30}	102
# Days since a transaction with that entity has been seen.	17
Ratio of # transactions / amount at that entity over the past n days to # transactions / amount at the same entity over the past m days. n is {0,1}, m is {7,14,30}	102
Risk table variable for day of week and state using likelihood of fraud for that day of week and state respectively	2
Benford's law variables for cardnum and merchnum represent a smoothed measure of unusualness of the transaction amount by each cardnum/merchnum	2
Total Variables Created	1,058

Feature Selection

In the feature selection step, we utilize a univariate filter followed by a wrapper. We store our candidate variables in three separate files in the step above. We set the balance equal to 0 so that all variables in each of the files will be used. The filter runs separately on each file of variables we created and keeps the top 100 variables from each file. When creating the filter, we remove the last two months' data as out-of-time data, and we also remove the first two weeks of records since the variables are not well-formed. We add a continuous random number variable that is uniformly distributed to make sure it does not come up as important when filtering. For each of the variables, we calculate a Kolmogorov-Smirnov (KS) score using pre-assigned fraud labels (goods and bads). After we finish calculating all the scores for each of the candidate variables, we sort them by descending order and choose the top 100 variables with the highest scores. This filter process helps us to get rid of highly correlated variables.

For the wrapper, we choose to use forward selection and set the number of variables we wanted to keep in the end to be 30. We use the LightGBM model as our final model for the wrapper to wrap around the 100 candidate variables we get from the filtering process. We choose to keep the top 30 variables with highest average scores (0.722). This wrapper process helps us to further remove correlated variables and only leave the ones that are truly important and useful for our analysis.

Following table shows a list of the final top 30 variables sorted by wrapper.

Variable	Score
cardnum_country_total_3	0.604
cardnum_description_country_max_3	0.650
cardnum_description_country_max_30	0.655
cardnum_description_country_max_7	0.657
cardnum_description_country_total_14	0.666
cardnum_description_country_total_3	0.662
cardnum_description_country_total_30	0.657
cardnum_description_state_zip5_country_max_3	0.647
cardnum_description_state_zip5_country_max_30	0.651
cardnum_description_state_zip5_country_max_7	0.653
cardnum_description_state_zip5_country_total_14	0.661

cardnum_description_state_zip5_country_total_3	0.658
cardnum_description_state_zip5_country_total_30	0.652
cardnum_merchnum_country_max_3	0.645
cardnum_merchnum_country_max_30	0.651
cardnum_merchnum_country_max_7	0.651
cardnum_merchnum_country_total_14	0.676
cardnum_merchnum_country_total_30	0.660
cardnum_merchnum_max_14	0.655
cardnum_merchnum_max_30	0.650
cardnum_merchnum_max_7	0.651
cardnum_merchnum_total_14	0.676
cardnum_state_zip5_country_max_14	0.656
cardnum_state_zip5_country_max_3	0.648
cardnum_state_zip5_country_max_7	0.656
cardnum_state_zip5_country_total_7	0.685
cardnum_total_3	0.602
description_country_total_1	0.612
description_state_zip5_country_max_1	0.599
merchnum_max_0	0.602

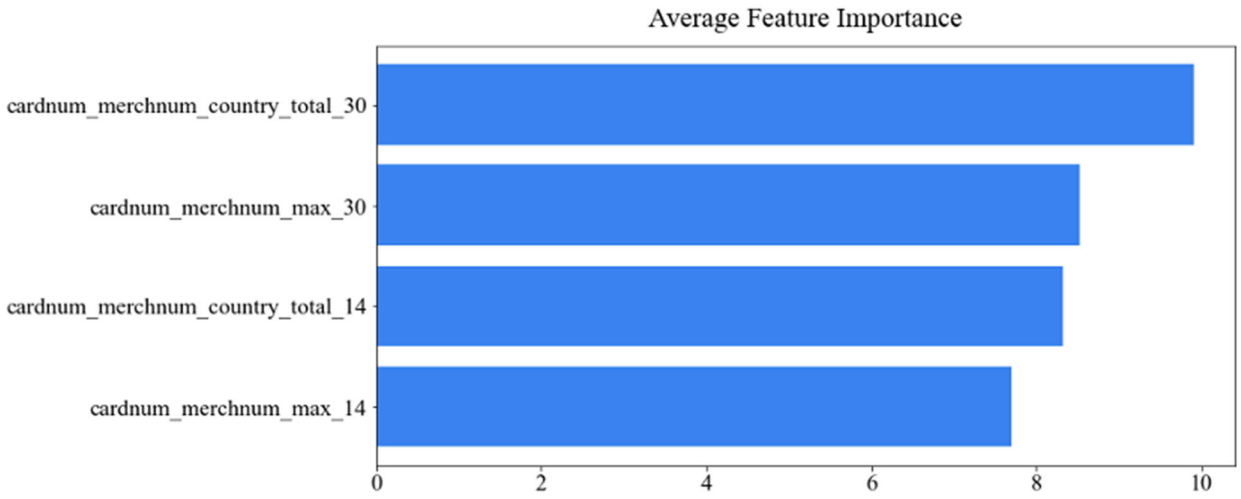
Logic/Steps of Final Variables Selection

1. Run the forward selection for the wrapper and choose top 30 variables from 100 candidate variables

2. Split these 30 variables into two sets (one includes the top 20 variables because their average score can reach 0.712 and the other includes the rest 10 variables.)

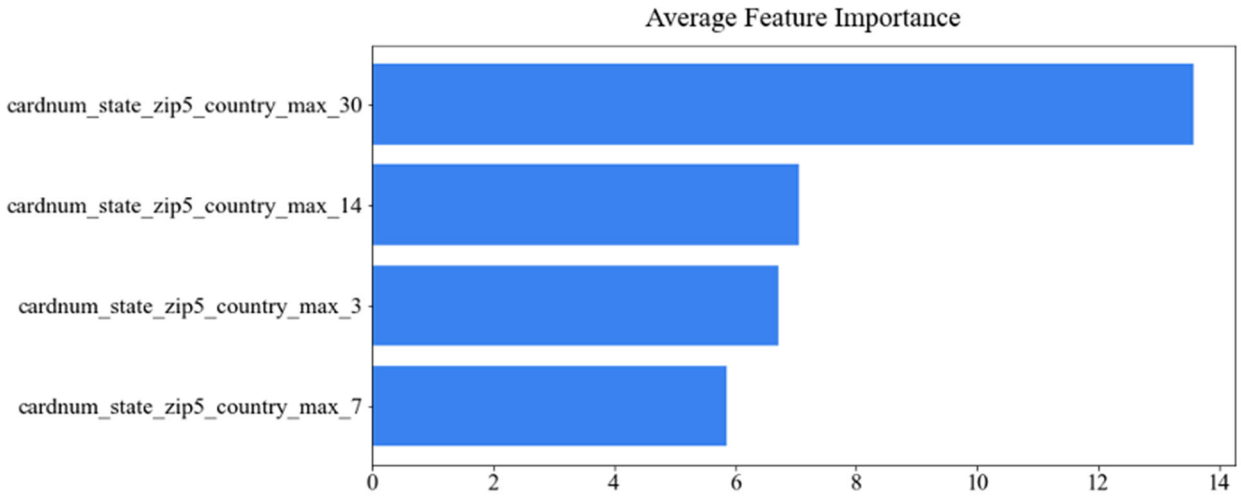
3. Run the model to compare the performance of these two sets of variables. As a result, the set of 10 variables performs better with fewer variables.

4. Study on the 10 variables selected above, remove duplicates such as 'cardnum_merchnum_country_total_14' and 'cardnum_merchnum_max_14', and only keep 'cardnum_merchnum_country_total_30' and 'cardnum_merchnum_max_30'

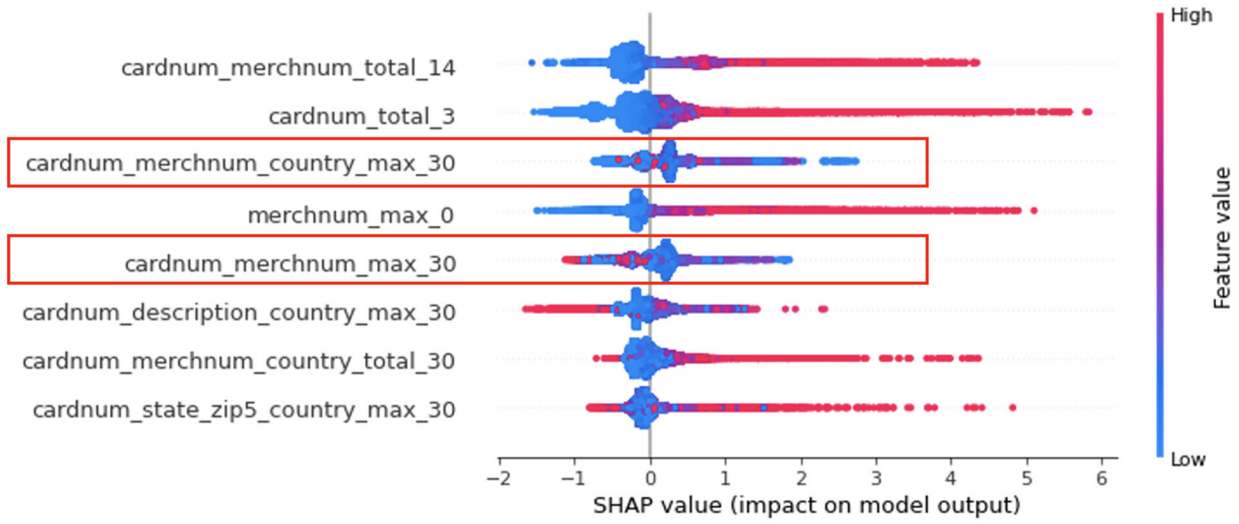


5. Examine performance of ‘cardnum_state_zip5_country_max_3' because its importance in the model is not stable.

6. Replace 'cardnum_state_zip5_country_max_3' by variable 'cardnum_state_zip5_country_max_30' which has the highest and most stable importance and is in a similar form ('cardnum_state_zip5_country_max_#').



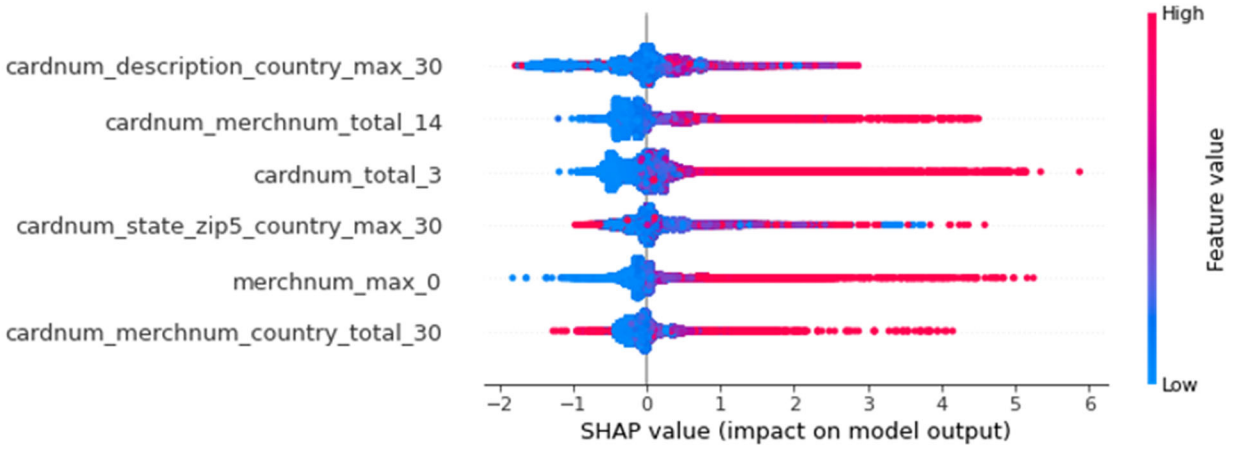
7. Use Shapley Additive Explanations (SHAP) Algorithm to find relatively weaker variables and remove them which are 'cardnum_merchnum_country_max_30' and 'cardnum_merchnum_max_30'.



SHAP algorithm is an advanced explanation algorithm of boosting tree models. It helps recognize relatively weaker variables with artificially high importance. From graph above, we can see that, for the two variables highlighted in the red box, their scatters are nearly all blue no matter if the SHAP value is high or low. This means that how the low values of these variables impact the output is unknown. However, the other variables have an obvious impact on the model output when their values change. Therefore, we remove these two variables and get our final six variables.

The Five Most Important Variables

We use SHAP Algorithm to find the five strongest variables with high importance and clear impact.



We can see that 'cardnum_description_country_max_30', 'cardnum_merchnum_total_14', 'cardnum_state_zip5_country_max_3', 'cardnum_total_3', and 'merchnum_max_0' are the 5 strongest variables.

Final Variables

Following table shows a list of the final six variables with description.

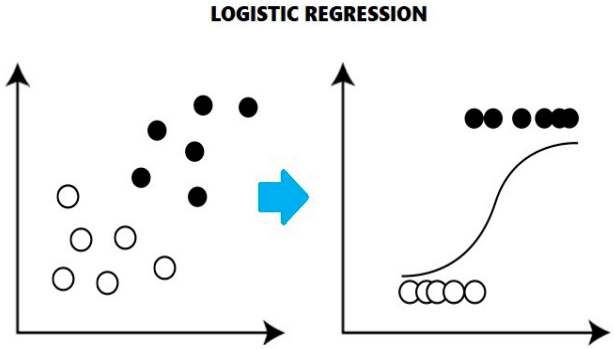
Final Variables	Description
cardnum_description_country_max_30	Maximum amount of all the transactions by this combination of cardnum, merch description, and country over the past 30 days
cardnum_merchnum_country_total_30	Total amount of all the transactions by this combination of cardnum, merchnum, and country over the past 30 days
cardnum_merchnum_total_14	Total amount of all the transactions by this combination of cardnum and merchnum over the past 14 days
cardnum_state_zip5_country_max_30	Maximum amount of all the transactions by this combination of cardnum, state, zip5 and country over the past 30 days
cardnum_total_3	Total amount of all the transactions by this cardnum over the past 3 days
merchnum_max_0	Maximum amount of all the transactions by this merchnum over the past 0 days

Model Algorithms

With the six variables chosen from the previous steps, we have run different models with parameter tuning to evaluate the fraud detection rate (FDR) on the Train, Test, and OOT dataset.

We have chosen logistic regression as our baseline model with all values of parameters as default (Average FDR: Train 67.8%, Test 65.7%, OOT 37.7%).

Logistics Regression (Baseline model)



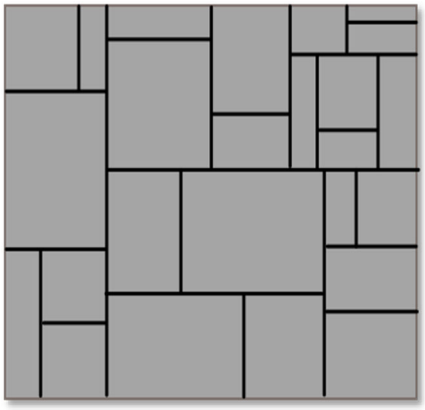
Some background information

While doing logistic regression, we are actually fitting our data into a logistic (a.k.a sigmoid) function f(x) = e^(b0+b1*x)/(1+e^(b0+b1*x), where f(x)=P (Y=1 | X=x) by using the maximum likelihood approach. In other words, in logistic regression, we are predicting the probability of a point to be in class 0 or 1, and we set a threshold to classify it to class 0 or 1.

Result

We build this model to evaluate the importance of our final variables. This model is the baseline model, therefore, we decide not to exhaustively adjust the parameters many times.

Decision Tree (Results after tuning parameters are improved)



Background Information

The Decision tree algorithm searches for the best cut point which results in the lowest impurity after the cut for every dimension. The most common measures of impurity are:

Variance: $I = \Sigma(y_i - < y_i >)^2$ and

Gini index: $I = 1 - \sum p_i^2 = 1 - \left(\frac{n_g}{n}\right)^2 - \left(\frac{n_b}{n}\right)^2$

Hyperparameters

criterion*:* The function to measure the quality of a split.

splitter: The strategy used to choose the split at each node.

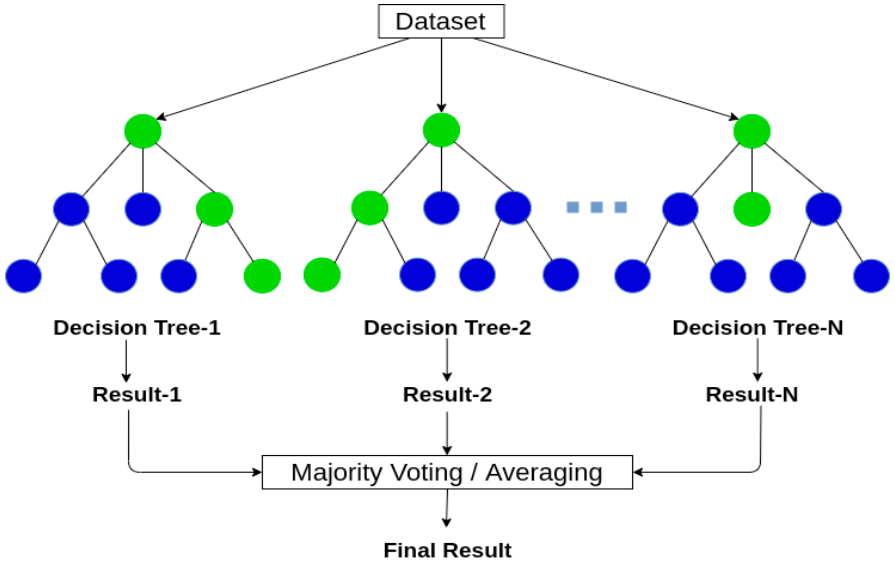
max_depth*:* The maximum depth of the tree.

min_samples_split*:* The minimum number of samples required to split an internal node

Result

The FDR of this model with default value of parameters shows that it is obviously overfitting because FDR of the training dataset is 100% which is much larger than FDR of testing dataset (61.2%) and FDR of OOT dataset (23.5%). Therefore, we tuned the parameters (criterion, max_depth, min_samples_leaf, splitter) to solve the problem of overfitting and the results have improved substantially.

Random Forest (Exhaustively tuned model to avoid overfitting)



Background Information

Random Forest is an example of bagging, the intuition is that we create a lot of strong decision trees, and the final output is an average or vote across all the strong models. The steps while performing the random forest algorithm are as follows:

1. We pick K random records from our dataset having a total of M records.
2. We build and train a decision tree model on these K records.
3. Choose how many trees you want in your random forest model and repeat steps1 and 2.

Hyperparameter

Bootstrap: Whether bootstrap samples are used when building trees.

n_estimators: The number of trees in the forest.

max_depth: The maximum depth of the tree.

min_samples_leaf*:* The minimum number of samples required to be at a leaf node.

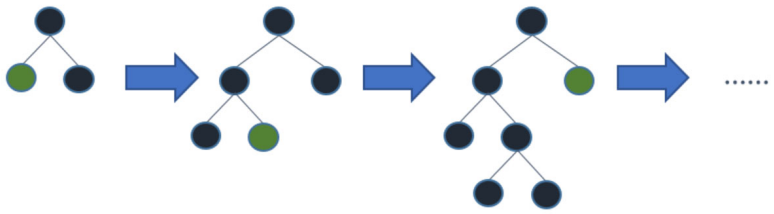
max_features*:* The number of features to consider when looking for the best split.

criterion: The function to measure the quality of a split.

Result

Random Forest model with default choice of parameters is also overfitting like Decision Tree but it is a bit better than Decision Tree. The average FDR of Training is 100% which is larger than FDR of Test (82.5%) and FDR of OOT (59.2%). After tuning parameters (max_depth, max_features, min_samples_leaf, criterion) exhaustively, the results are no longer overfitting, but the average FDR decreased by 1.62%. The best run has an average FDR of Training (71.6%), FDR of Test (70.1%), and FDR of OOT (57.6%).

LightGBM (One of the best models)



Leaf-wise tree growth

Background Information

LightGBM is an example of bagging. It trains a series of weak models to result in a strong model and each weak model is trained to predict the residual error of the current sum. The special thing about LightGbm is that it expands in a vertical direction which means it grows leaf-wise while others expand level-wise. It selects the leaf which produces the least error and maximum efficiency.

Hyperparameters

boosting_type: 'gbdt', traditional Gradient Boosting Decision Tree. 'dart', Dropouts meet Multiple Additive Regression Trees. 'goss', Gradient-based One-Side Sampling. 'rf', Random Forest.

num_leaves: Maximum tree leaves for base learners.

max_depth: Maximum tree depth for base learners

learning_rate: Boosting learning rate.

n_estimators : Number of boosted trees to fit.

colsample_bytree: Subsample ratio of columns when constructing each tree.

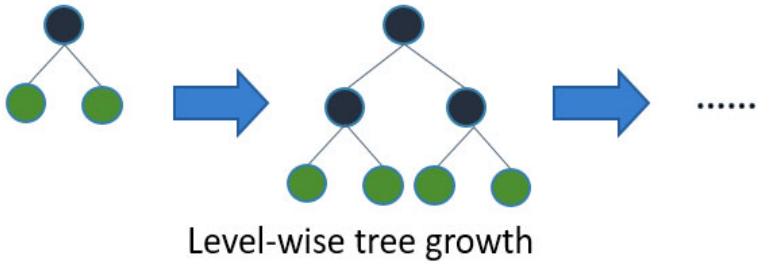
subsample: Subsample ratio of the training instance.

eval_metric: If str, it should be a built-in evaluation metric to use. If callable, it should be a custom evaluation metric, see note below for more details. If list, it can be a list of built-in metrics, a list of custom evaluation metrics, or a mix of both. In either case, the metric from the model parameters will be evaluated and used as well. Default: 'l2' for LGBMRegressor, 'logloss' for LGBMClassifier, 'ndcg' for LGBMRanker.

Result

LightGBM gives better results compared with the previous models tried. The best result we get after turning from this model has an average FDR on Train of 75.1%, Test of 73.7% and OOT of 58.2%. There is no obvious overfitting problem, and its running time is not too long (around 50 seconds).

XGBoost (Similar performance as LightGBM)



Background Information

XGBoost is also an example of boosting. It trains a series of weak models to result in a strong model and each weak model is trained to predict the residual error of the current sum. It grows level-wise and takes more time to run compared to LightGBM.

Hyperparameters

n_estimators: Number of gradient boosted trees.

max_depth: Maximum depth of a tree.

tree_method string: The tree construction algorithm used in XGBoost.

min_child_weight: Minimum sum of instance weight (hessian) needed in a child.

colsample_bytree: Subsample ratio of columns when constructing each tree.

subsample: Subsample ratio of the training instances.

eta: Step size shrinkage used in update to prevents overfitting.

eval_metric: If str, it should be a built-in evaluation metric to use. If callable, it should be a custom evaluation metric, see note below for more details. If list, it can be a list of built-in metrics, a list of custom evaluation metrics, or a mix of both.

scale_pos_weight: Control the balance of positive and negative weights.

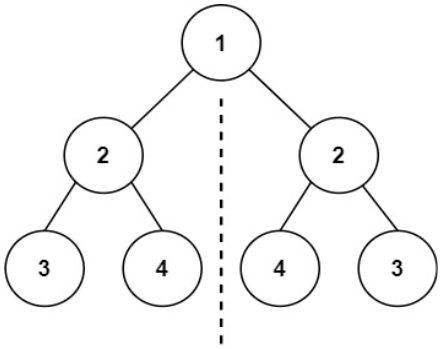
reg_alpha: L1 regularization term on weights. Increasing this value will make model more conservative.

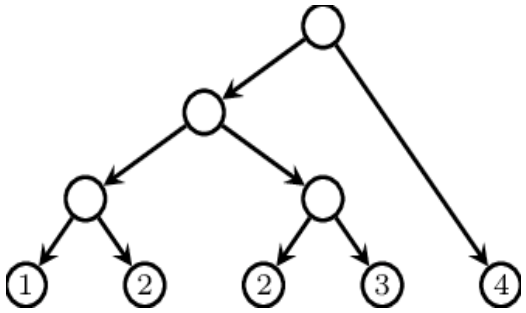
reg_lambda: L2 regularization term on weights. Increasing this value will make model more conservative.

Result

XGBoost is the fastest model with a running time of only 33 seconds. It gives similar results as LightGBM with lower average FDR on OOT dataset of 54.1%. After tuning the hyperparameters, the best result achieved is an average FDR on Train of 76.0%, Test of 72.9% and OOT of 54.1%.

CatBoost (Best performed model)





Background Information

CatBoost is also an example of boosting.

It does especially well with data containing "categorical variables." It trains a series of weak models to result in a strong model and each weak model is trained to predict the residual error of the current sum. This CatBoost model proceeds by building **"symmetric binary trees"** for each permutation of the data. It does especially well with data containing "categorical variables."

Hyperparameters

bootstrap_type: Defines the method for sampling the weights of objects.

depth: Depth of the tree.

iterations: The maximum number of trees that can be built when solving machine learning problems.

l2_leaf_reg: Coefficient at the L2 regularization term of the cost function.

learning_rate: The learning rate.

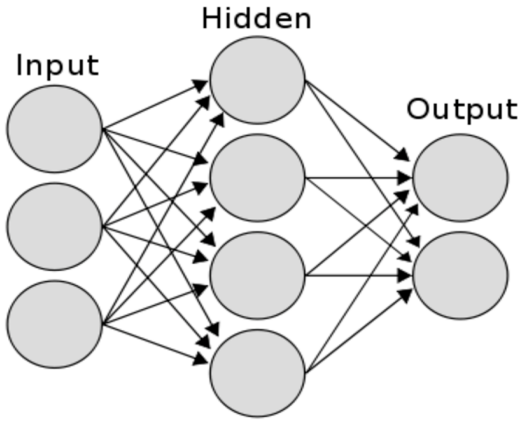
scale_pos_weight: Control the balance of positive and negative weights.

random_state: The random seed used for training.

Result

CatBoost performs the best among all the models tried. Even with the default parameter values, the average FDR of the OOT dataset is above 60%. After tuning the parameters, the best results achieved is an average FDR on Train of 78.3%, Test of 75.2% and OOT of 64.4% when the bootstrap type is Bayesian.

Neural Network (Performs well with no obvious overfitting problem)



Background Information

The invention of the Neural Network is inspired by observing how the human brain works. A neural net consists of an input layer, some number of hidden layers and an output layer. All the independent variables (x's) form the input layer, the dependent variable y is the output layer, and the hidden layer is a set of nodes with weighted signals trained by backpropagating in the previous layer. The model then finally does a transform on this linear combination of signals. The most common transform(activation) function is Rectified Linear Unit (ReLu), sigmoid function, and softplus activation function.

Hyperparameters

activation*:* Activation function for the hidden layer.

solver*:* The solver for weight optimization.

learning_rate*:* Learning rate schedule for weight updates.

hidden_layer_sizes: The ith element represents the number of neurons in the ith hidden layer.

max_iter: Maximum number of iterations.

alpha: L2 penalty (regularization term) parameter.

Result

Neural Network also performs well with no obvious overfitting problem before and after tuning hyperparameters. The average FDR on OOT dataset ranges from 57.9% to 60.5%. The best result from this model is an average FDR on Train of 70.9%, Test of 69.1% and OOT of 60.5%.

Following table shows our model explorations and their performance.

Model			Parameters									Average FDR at 3%		
Logistics Regression	Iteration	# of Variables	Penalty	C	solver	l1_ratio						Train	Test	OOT
	1 (default)	30	N/A	1	lbfgs	None						67.8	65.7	33.7
	2	20	N/A	1	lbfgs	None						68.4	67.5	39.5
	3	10	N/A	1	lbfgs	None						64.5	65.7	40.2
	4	6	N/A	1	lbfgs	None						64.9	64.3	44.1
	5	6	l1	1	liblinear	None						64.1	64.2	41.3
	6	6	l2	0.1	saga	None						63.5	63.0	42.7
	7	8	ElasticNet	1	saga	0.4						63.7	62.8	44.9
	8	6	ElasticNet	1	saga	0.4						65.2	63.5	45.1
Decision Tree	Iteration	# of Variables	criterion	max_depth	min_samples_leaf	splitter						Train	Test	OOT
	1 (default)	6	gini	None	2	best						100.0	61.2	23.5
	2	6	gini	20	60	best						81.4	77.6	51.4
	3	6	gini	20	80	best						80.3	76.6	51.8
	4	6	entropy	20	60	best						82.7	74.4	45.8
	5	6	entropy	20	60	random						69.0	67.1	44.0
Random Forest	Iteration	# of Variables	bootstrap	n_estimators	max_depth	max_features	min_samples_leaf	criterion				Train	Test	OOT
	1 (default)	6	TRUE	100	None	auto	1	gini				100.0	82.5	59.2
	2	6	TRUE	50	20	auto	20	gini				91.9	78.6	60.2
	3	6	TRUE	100	24	auto	36	gini				85.8	78.3	59.2
	4	6	TRUE	80	25	auto	50	entropy				83.3	78.2	49.8
	5	6	TRUE	80	15	sqrt	30	gini				85.9	80.6	58.8
	6	6	TRUE	150	10	auto	80	entropy				78.4	73.9	48.9
	7	6	TRUE	50	10	sqrt	40	entropy				78.3	74.4	49.4
	8	6	TRUE	80	5	auto	30	gini				71.20	71.33	57.32
	9	6	TRUE	100	5	sqrt	50	gini				71.83	69.19	57.15
	10	6	TRUE	80	3	log2	40	gini				68.34	67.74	53.41
	11	6	TRUE	30	5	log2	15	gini				71.6	70.1	57.6
	12	6	TRUE	20	5	log2	15	gini				71.98	70.69	55.87
LightGBM	Iteration	# of Variables	boosting_type	max_depth	n_estimators	num_leaves	colsample_bytree	subsample	learning_rate	eval_metric		Train	Test	OOT
	1 (default)	6	gbdt	-1	100	31	1	1	0.1	None		98.9	80.9	47.0
	2	6	GOSS	5	800	22	0.8	0.8	0.03	auc		98.0	77.2	50.1
	3	6	GOSS	5	2000	22	0.8	0.8	0.01	auc		96.2	83.3	54.4
	4	6	GOSS	7	1000	20	0.8	0.8	0.01	auc		95.1	78.8	60.2
	5	6	GOSS	7	1000	22	0.8	0.8	0.003	auc		87.7	82.0	61.1
	6	6	GOSS	10	2000	22	0.8	0.8	0.003	auc		92.9	81.3	59.2
	7	6	gbdt	15	1000	25	0.6	0.6	0.001	logloss		85.0	78.3	58.3
	8	6	gbdt	10	500	15	1	1	0.001	logloss		76.7	73.8	53.7
	9	6	gbdt	8	100	15	0.6	0.6	0.001	None		75.1	73.7	58.2
	10	6	goss	5	80	20	0.5	0.5	0.01	None		79.1	75.1	55.6
	11	6	goss	-1	50	10	0.5	0.5	0.01	None		70.9	68.6	56.8
	12	6	GOSS	7	2000	22	0.8	0.8	0.003	logloss		88.0	79.6	62.2
XGBoost	Iteration	# of Variables	n_estimators	max_depth	tree_method	min_child_weight	colsample_bytree	subsample	eta	eval_metric	scale_pos_weight	reg_alpha	reg_lambda	Train	Test	OOT
	1 (default)	6	100	6	auto	1	1	1	0.3	logloss	1	1	1	99.1	84.4	57.9
	2	6	500	5	auto	100	0.8	0.8	0.2	logloss	1	1	1	70.3	72.3	43.2
	3	6	2000	10	exact	10	0.8	0.8	0.025	logloss	1	1	1	97.1	81.3	60.0
	4	6	1000	10	approx	10	0.8	0.8	0.03	logloss	1	1	1	91.5	83.0	60.3
	5	6	800	10	approx	10	0.8	0.8	0.03	mlogloss	0.5	1	1	84.4	78.4	56.3
	6	6	1200	8	hist	10	0.6	0.2	0.2	logloss	1	7	8	76.5	73.6	52.8
	7	6	1000	10	approx	12	0.9	0.2	0.03	mlogloss	1	1	1	76.0	72.9	54.1
	8	6	500	10	hist	10	0.8	0.2	0.07	logloss	0.9	1	1	77.1	72.2	53.5
	9	6	800	10	hist	12	0.9	0.2	0.06	logloss	0.9	1	1	76.8	72.7	54.5
	10	6	1500	7	hist	10	0.8	0.15	0.2	logloss	1	7	8	74.6	72.6	51.6
	11	6	1200	5	hist	10	0.8	0.15	0.2	logloss	1	6	6	75.2	74.1	52.0
	12	6	1200	7	hist	12	0.8	0.15	0.2	logloss	1	6	8	74.6	72.0	51.4
CatBoost	Iteration	# of Variables	boostrap_type	depth	iterations	l2_leaf_reg	learning_rate	scale_pos_weight	random_state					Train	Test	OOT
	1 (default)	6	Bayesian	6	1000	3	0.03	1	None					93.8	80.2	60.2
	2	6	Bayesian	7	3000	12	0.01	1	10					88.1	82.3	64.3
	3	6	Bernoulli	5	1000	12	0.01	1	10					82.3	77.5	62.8
	4	6	MVS	7	1000	8	0.03	1	10					87.9	81.8	66.5
	5	6	Bayesian	7	1000	20	0.025	1	10					81.8	78.0	63.7
	6	6	Bayesian	7	1000	4	0.025	1	10					89.8	79.9	66.7
	7	6	MVS	7	600	1	0.01	0.3	3					76.7	74.0	62.7
	8	6	MVS	7	700	1	0.025	0.3	3					84.5	78.5	64.6
	9	6	Bayesian	7	500	1	0.02	0.3	3					80.9	76.9	63.8
	10	6	Bayesian	7	1000	4	0.01	0.7	3					78.3	75.2	64.4
	11	6	Bayesian	7	700	1	0.01	0.3	3					78.8	74.7	63.4
	12	6	Bayesian	7	500	0.5	0.01	0.7	3					78.6	77.0	63.5
Neural Network	Iteration	# of Variables	activation	solver	learning_rate	hidden_layer_sizes	max_iter	alpha						Train	Test	OOT
	1 (default)	6	relu	adam	constant	100	200	0.0001						70.9	69.1	60.5
	2	6	relu	lbfgs	constant	30	500	0.15						70.6	69.3	57.9
	3	6	relu	adam	invscaling	30	400	0.0004						69.0	69.5	58.3
	4	6	tanh	adam	invscaling	120	300	0.00003						71.5	67.1	60.5
	5	6	tanh	adam	adaptive	200	100	0.0001						69.0	71.2	59.6

Results

After comparing multiple algorithms, we decide to use CatBoost as our final algorithm because it predicts OOT data well and runs fast. We set our final model hyperparameters as bootstrap_type ='Bayesian', learning_rate=0.01, iterations=1000, loss_function='Logloss', depth= 7, od_type='Iter', random_state=3, l2_leaf_reg=4, scale_pos_weight=0.7, verbose=False and other hyperparameters as default.

Note that we filter out the last two months' data as OOT data. The rest of the data is our train and test data (83,970 records). We then split this data into 70% training data and 30% testing data randomly and run the CatBoost model.

Below is the summary table for the training dataset.

Training	# Records	# Goods	# Bads	Fraud Rate
	58779	58168	611	0.010394869

	Bin Statistics					Cumulative Statistics						
Population Bin %	#Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	588	189	399	32.1	67.9	588	189	399	0.3	65.3	65.0	0.5
2	588	532	56	90.5	9.5	1176	721	455	1.2	74.5	73.2	1.6
3	587	560	27	95.4	4.6	1763	1281	482	2.2	78.9	76.7	2.7
4	588	573	15	97.4	2.6	2351	1854	497	3.2	81.3	78.2	3.7
5	588	571	17	97.1	2.9	2939	2425	514	4.2	84.1	80.0	4.7
6	588	580	8	98.6	1.4	3527	3005	522	5.2	85.4	80.3	5.8
7	588	582	6	99.0	1.0	4115	3587	528	6.2	86.4	80.2	6.8
8	587	574	13	97.8	2.2	4702	4161	541	7.2	88.5	81.4	7.7
9	588	581	7	98.8	1.2	5290	4742	548	8.2	89.7	81.5	8.7
10	588	582	6	99.0	1.0	5878	5324	554	9.2	90.7	81.5	9.6
11	588	583	5	99.1	0.9	6466	5907	559	10.2	91.5	81.3	10.6
12	587	587	0	100.0	0.0	7053	6494	559	11.2	91.5	80.3	11.6
13	588	584	4	99.3	0.7	7641	7078	563	12.2	92.1	80.0	12.6
14	588	582	6	99.0	1.0	8229	7660	569	13.2	93.1	80.0	13.5
15	588	587	1	99.8	0.2	8817	8247	570	14.2	93.3	79.1	14.5
16	588	584	4	99.3	0.7	9405	8831	574	15.2	93.9	78.8	15.4
17	587	586	1	99.8	0.2	9992	9417	575	16.2	94.1	77.9	16.4
18	588	586	2	99.7	0.3	10580	10003	577	17.2	94.4	77.2	17.3
19	588	588	0	100.0	0.0	11168	10591	577	18.2	94.4	76.2	18.4
20	588	587	1	99.8	0.2	11756	11178	578	19.2	94.6	75.4	19.3

Below is the summary table for the testing dataset.

Testing	# Records	# Goods	# Bads	Fraud Rate
	25191	24922	269	0.010678417

	Bin Statistics					Cumulative Statistics						
Population Bin %	#Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	252	85	167	33.7	66.3	252	85	167	0.3	62.1	61.7	0.5
2	252	220	32	87.3	12.7	504	305	199	1.2	74.0	72.8	1.5
3	252	236	16	93.7	6.3	756	541	215	2.2	79.9	77.8	2.5
4	252	245	7	97.2	2.8	1008	786	222	3.2	82.5	79.4	3.5
5	252	248	4	98.4	1.6	1260	1034	226	4.1	84.0	79.9	4.6
6	251	248	3	98.8	1.2	1511	1282	229	5.1	85.1	80.0	5.6
7	252	250	2	99.2	0.8	1763	1532	231	6.1	85.9	79.7	6.6
8	252	251	1	99.6	0.4	2015	1783	232	7.2	86.2	79.1	7.7
9	252	250	2	99.2	0.8	2267	2033	234	8.2	87.0	78.8	8.7
10	252	252	0	100.0	0.0	2519	2285	234	9.2	87.0	77.8	9.8
11	252	251	1	99.6	0.4	2771	2536	235	10.2	87.4	77.2	10.8
12	252	251	1	99.6	0.4	3023	2787	236	11.2	87.7	76.5	11.8
13	252	252	0	100.0	0.0	3275	3039	236	12.2	87.7	75.5	12.9
14	252	251	1	99.6	0.4	3527	3290	237	13.2	88.1	74.9	13.9
15	252	247	5	98.0	2.0	3779	3537	242	14.2	90.0	75.8	14.6
16	252	251	1	99.6	0.4	4031	3788	243	15.2	90.3	75.1	15.6
17	251	250	1	99.6	0.4	4282	4038	244	16.2	90.7	74.5	16.5
18	252	250	2	99.2	0.8	4534	4288	246	17.2	91.4	74.2	17.4
19	252	252	0	100.0	0.0	4786	4540	246	18.2	91.4	73.2	18.5
20	252	250	2	99.2	0.8	5038	4790	248	19.2	92.2	73.0	19.3

Below is the summary table for the OOT dataset.

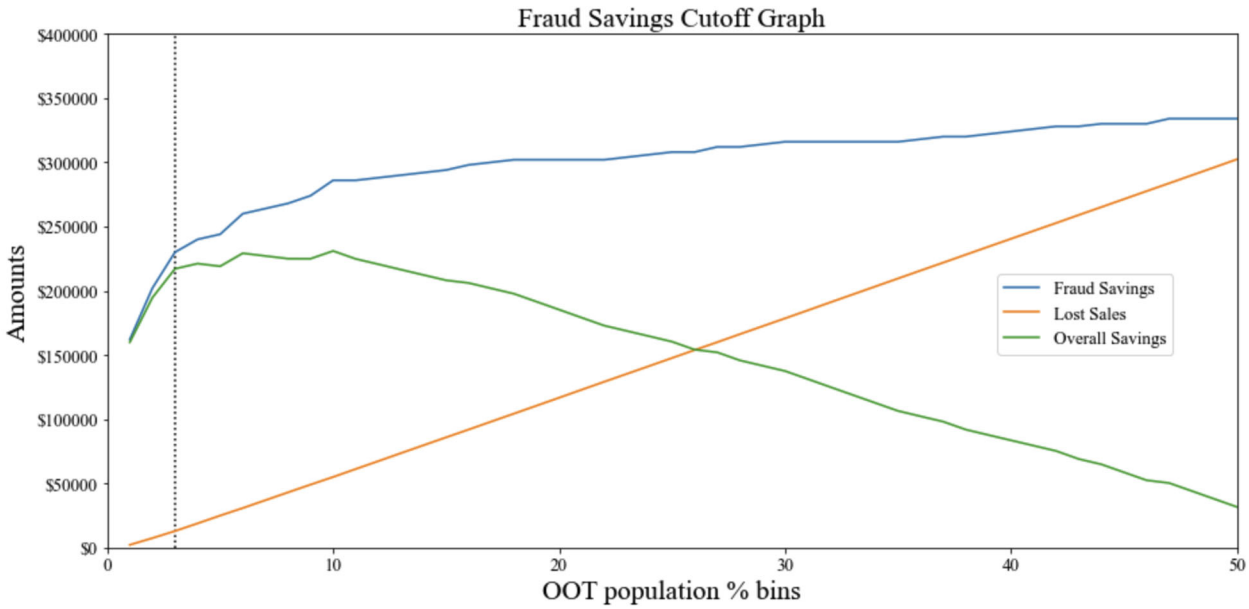
OOT	# Records		# Goods		# Bads		Fraud Rate					
	12427		12248		179		0.01440412					
	Bin Statistics					Cumulative Statistics						
Population Bin %	#Records	# Goods	# Bads	%Goods	%Bads	Total # Records	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	124	43	81	34.7	65.3	124	43	81	0.4	45.3	44.9	0.5
2	125	105	20	84.0	16.0	249	148	101	1.2	56.4	55.2	1.5
3	124	110	14	88.7	11.3	373	258	115	2.1	64.2	62.1	2.2
4	124	119	5	96.0	4.0	497	377	120	3.1	67.0	64.0	3.1
5	124	122	2	98.4	1.6	621	499	122	4.1	68.2	64.1	4.1
6	125	117	8	93.6	6.4	746	616	130	5.0	72.6	67.6	4.7
7	124	122	2	98.4	1.6	870	738	132	6.0	73.7	67.7	5.6
8	124	122	2	98.4	1.6	994	860	134	7.0	74.9	67.8	6.4
9	124	121	3	97.6	2.4	1118	981	137	8.0	76.5	68.5	7.2
10	125	119	6	95.2	4.8	1243	1100	143	9.0	79.9	70.9	7.7
11	124	124	0	100.0	0.0	1367	1224	143	10.0	79.9	69.9	8.6
12	124	123	1	99.2	0.8	1491	1347	144	11.0	80.4	69.4	9.4
13	125	124	1	99.2	0.8	1616	1471	145	12.0	81.0	69.0	10.1
14	124	123	1	99.2	0.8	1740	1594	146	13.0	81.6	68.5	10.9
15	124	123	1	99.2	0.8	1864	1717	147	14.0	82.1	68.1	11.7
16	124	122	2	98.4	1.6	1988	1839	149	15.0	83.2	68.2	12.3
17	125	124	1	99.2	0.8	2113	1963	150	16.0	83.8	67.8	13.1
18	124	123	1	99.2	0.8	2237	2086	151	17.0	84.4	67.3	13.8
19	124	124	0	100.0	0.0	2361	2210	151	18.0	84.4	66.3	14.6
20	124	124	0	100.0	0.0	2485	2334	151	19.1	84.4	65.3	15.5

Overview of the OOT table

This is the summary table to show the performance of our model in our OOT dataset. We have in total 12,417 records in the OOT dataset. The table in pink is our Bin Statistics. We divide the records into 100 equal bins, so approximately each bins contain 1% of the OOT data. In this report, we focused on the top 20% data only. The table in green is our cumulative statistics. It contains all the statistics information up to and including each bin.

Moreover, in bin statistics table, % Goods is calculated by # Goods divided by # Records in that specific bin, % Bads is calculated by # Bads divided by # Records in that specific bin. In cumulative statistics table, Cumulative Goods is calculated by adding # Goods up to and including that bin, Cumulative Bads is calculated by adding # Bads up to and including that bin, % Goods is calculated by Cumulative goods divided by total number of goods, % Bads (FDR) is calculated by Cumulative Bads divided by the total number of bads. KS is calculated by % Bads - % Goods. FPR is the false positive ratio which is calculated by Cumulative Goods divided by Cumulative Bads.

Below is the Fraud Savings Calculation and Suggests Score Cut-off plot



The graph above is a line plot that shows the amount of the fraud savings (blue), lost sales (yellow), and overall savings (green) for one percent increase in the number of credit card transactions declined in our OOT population.

Based on the assumption that we gain $2,000 for every fraud that is caught, $50 loss for every good we labeled as bad (false positive). We recommend a score cut-off at 3% population since it can decline 64.2% of the frauds. We will catch 115 frauds, and have $230,000 fraud savings, $12,900 lost sales, and result in $217,100 overall savings for the organization while maintaining a good and satisfying experience for customers.

Conclusion

In this project, we develop a credit card fraud transaction algorithm utilizing 96,753 real credit card transactions data with ten fields. We begin our investigation of the dataset by summarizing statistics from numerical and categorical variables respectively and doing exploratory data analysis. During this process, we find outliers and missing values and replaced these records with appropriate imputation for building model algorithms. From the cleaned data, we then create over 1,000 candidate variables based on scenarios where transaction fraud can happen. These variables mainly fall into four categories: amount variables, frequency variables, days-since variables, and velocity change variables. Besides, we create two risk table variables to convert day of week and state into numerical fields. Last but not least, we utilize Benford's Law distribution and identified top 40 card holder and merchant fraudster based on their unusualness.

We then utilize a univariate filter to narrow down our variables to 100 variables followed by a LightGBM with forward selection wrapper to further narrow them down to the top 30 variables. Finally, we select the six variables with highest feature importance and low correlation with each other.

Using our final six variables we run several models. We filter out the first two weeks' data as they are not complete and set aside the last two months' data as OOT data. We randomly split the left data into 70% as training and 30% as testing and run models with adjustments in hyperparameters. We evaluate the fraud detection rate (FDR) on the training, testing, and OOT datasets for Logistics Regression, Decision Tree, Random Forest, LightGBM, XGBoost, CatBoost and Neural Network. With mean FDR of 78.3% for training, 75.2% for testing, and 64.4% for OOT, we choose CatBoost with Bayesian bootstrap as our final algorithm since it accurately predicts OOT data and runs efficiently.

There are certain potential areas for improvement in the future. First, the dataset only contains 96,753 records with 1,059 fraudulent records. Over-fitting is harder to avoid when exploring and tuning different models. Even with our best performance model, its performance fluctuates. We will retrain our model in the future with more input data to stabilize the performance of our optimal model. Second, the dataset only contains ten fields with nonnegligible number of missing values. With more complete data and more fields we may build a stronger model in the future.

As the credit card transaction increases, so does the transaction fraud. The good news is that we have been evolved from rule-based expert systems to real-time fraud detection algorithms. With more efficient processor and larger computation power, organizations are able to build more complicated and stronger fraud detection systems. With careful examination of trade-off between lost sales and savings from rejected frauds, organizations will achieve more and more savings from implementing effective fraud detection systems in the future.

Appendix I Data Quality Report

Description

The credit card transaction data consists of information of credit card transactions throughout year 2006. In the dataset, each record represents one transaction. There are, in total, 96,753 records with 10 fields.

Field Summary Tables

Numeric Fields

	% Populated	Min	Max	Mean	Stdev	% Zero
Date	100.0	2006-01-01	2006-12-31	NaN	NaN	0.0
Amount	100.0	0.01	3,102,045.53	427.89	10,006.14	0.0

Categorical Fields

	% Populated	# Unique Values	Most Common Value
Cardnum	100.0	1,645	5142148452
Merchnum	100.0	13,092	930090121224
Merch description	100.0	13,126	GSA-FSS-ADV
Merch state	98.76	227	TN
Merch zip	100.0	2,692	38118
Transtype	100.0	4	P
Fraud	100.0	2	0

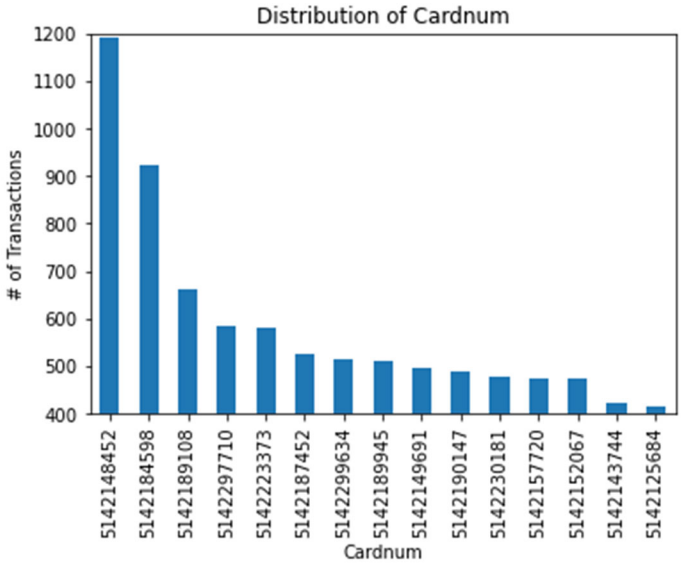
Description and Distribution of Field

Recnum

This is unique to each record and works as an identifier.

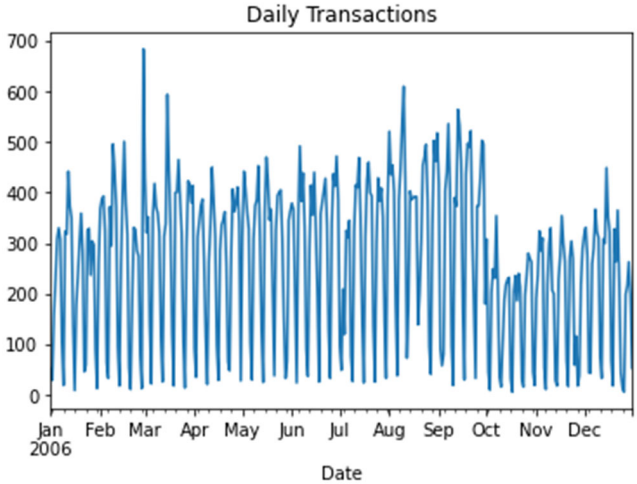
Cardnum

A categorical field represents the credit card number of the transaction. This field has 1,645 unique values and the graph below shows the top 15 with highest frequency.



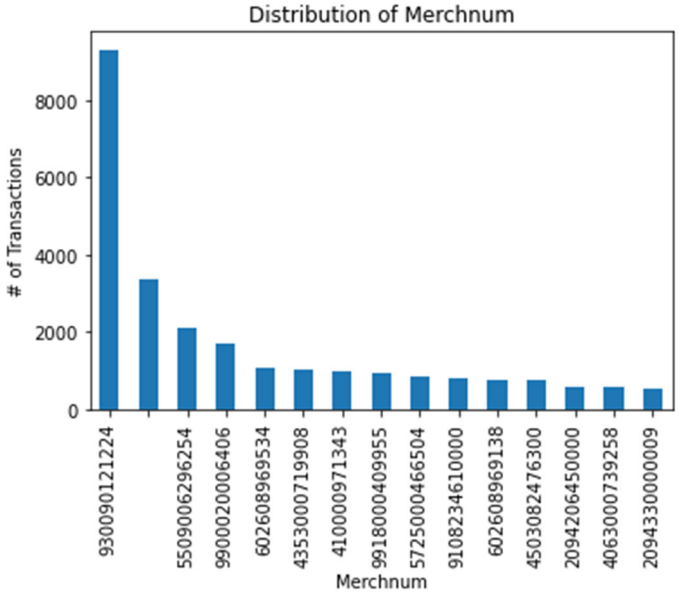
Date

Treated as a numerical field represents the date of the transaction. In this dataset, date is ranged from 2006-01-01 to 2006-12-31. Graph below shows number of transaction per day. We can see a weekly seasonality in the graph with approximately 52 peaks.



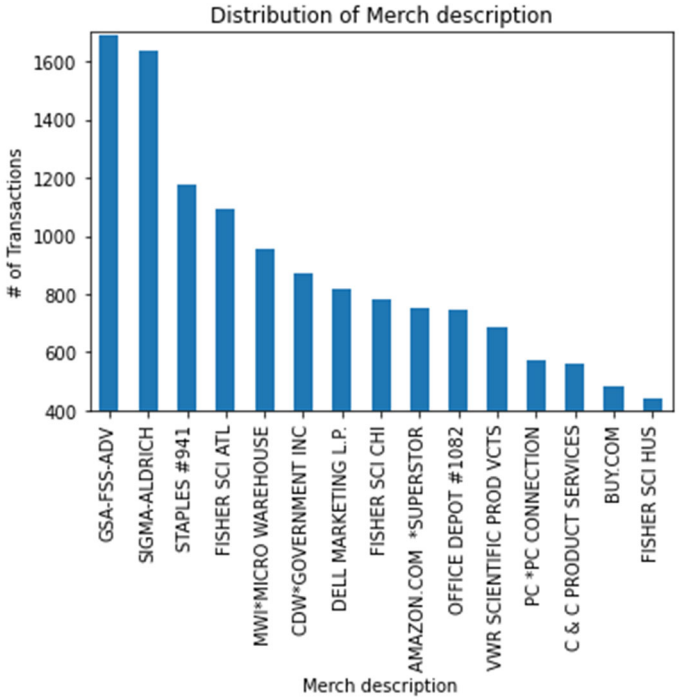
Merchnum

A categorical field represents the merchant number of the transaction. This field has 13,092 unique values and the graph below shows the top 15 with highest frequency. The second bar represents the numebr of missing values and need to be further discussed in the future.



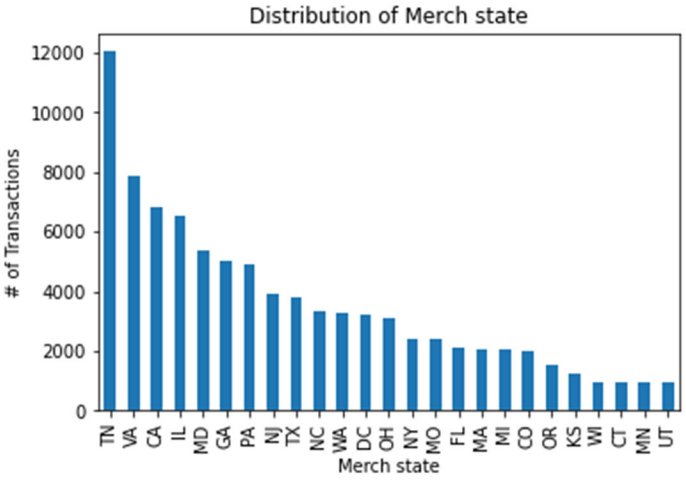
Merch description

A categorical field represents the description of the merchant involved in the transaction. This field has 13,126 unique values and the graph below shows the top 15 with highest frequency.



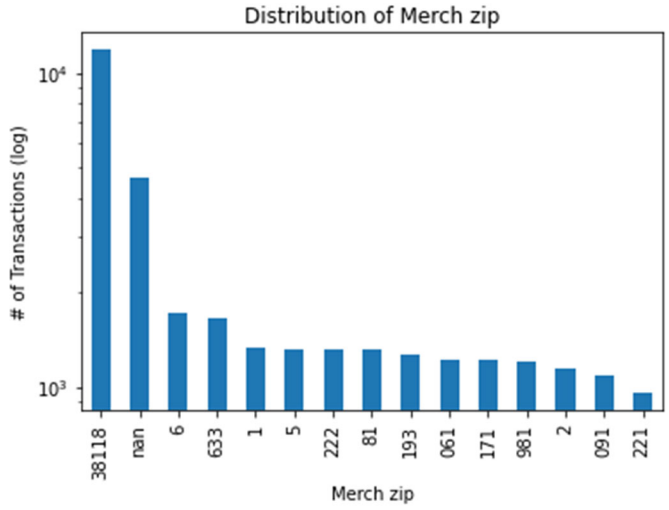
Merch state

A categorical field represents the state abbreviation of the merchant involved in the transaction. This field has 228 unique values which is unusual. Many of the field has value of a 3-digit number and we need to discuss it in the future analysis. The graph below shows the top 25 with highest frequency.



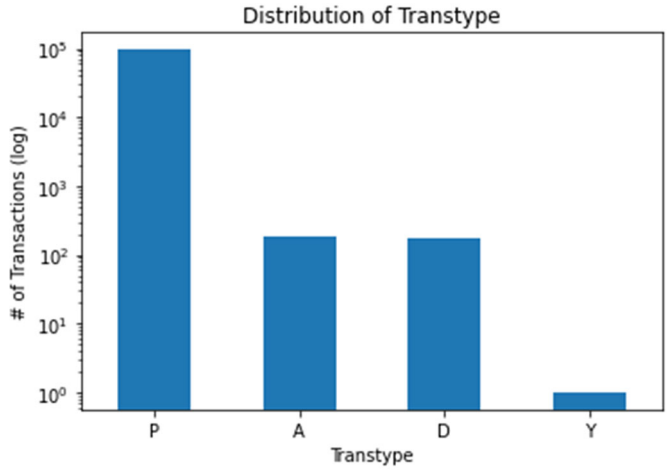
Merch zip

A categorical field represents the 5-digit zip code of the merchant involved in the transaction. This field has 2,692 unique values. The graph below shows the top 15 with highest frequency.



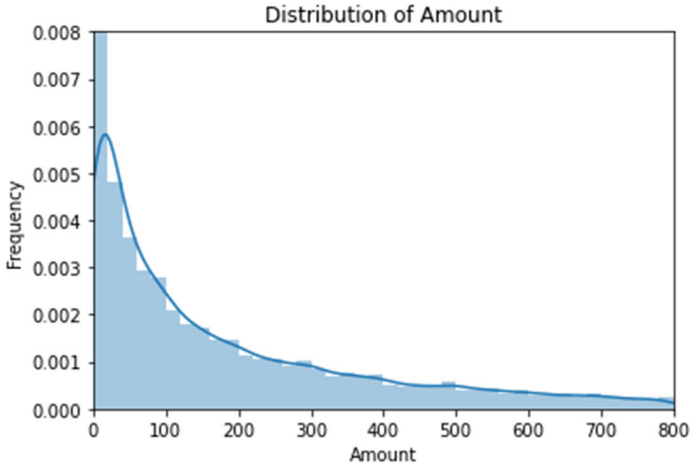
Transtype

A categorical field represents the type the transaction. There are four types of transactions – “P”, “A”, “D” and “Y”. P stands for purchase and we do not know what the other three stands for at current stage. The graph below shows the distrbution of the transaction types.



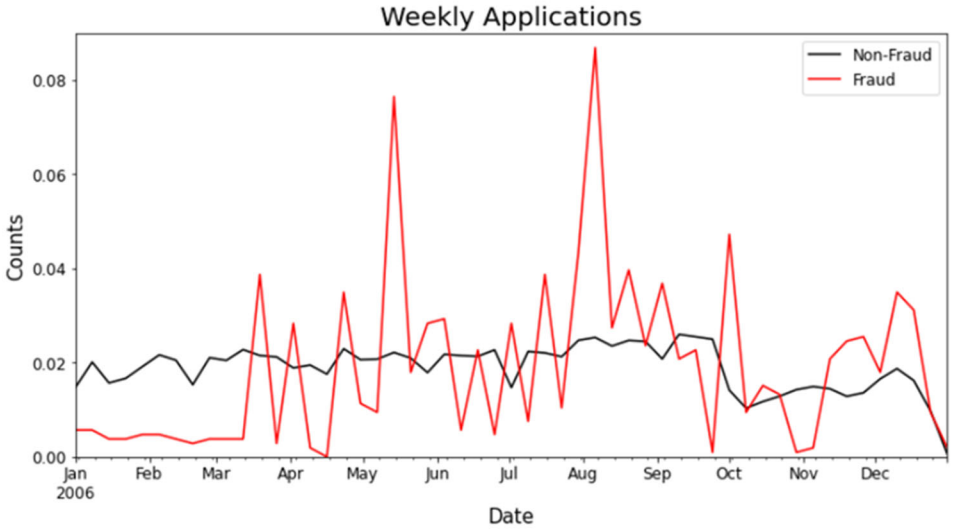
Amount

A numerical field represents the amount the transaction. The following graph shows distribution of the amount below 800, which represents more than 85% of the data. There are extremely high value in this field which is not showed in the graph and needs further analysis. The graph focus on frequncy below 0.08 to show more information.



Fraud

A categorical field represents the fraud label of the transaction. This field has 2 unique values 0 and 1, where 1 means suspicious. There are 1,059 suspicious records and 95,694 non-suspicious records. Graph below shows number of suspicious and non-suspicious transactions per week where green line is for suspicous records and red line is for non-suspicious records.



Appendix II Benford's Law Variables

Top 40 Cardnum (Potential Fraud Based on Benford's Law)

cardnum	n	n_low	n_high	R	1/R	U	t	U*
5142253356	66	61	5	13.37	0.07	13.37	17.00	13.37
5142299705	28	25	3	9.13	0.11	9.13	4.33	9.03
5142197563	149	15	134	0.12	8.15	8.15	44.67	8.15
5142194617	38	5	33	0.17	6.02	6.02	7.67	6.02
5142288241	14	1	13	0.08	11.86	11.86	-0.33	5.53
5142239140	19	16	3	5.85	0.17	5.85	1.33	4.83
5142144931	36	6	30	0.22	4.56	4.56	7.00	4.56
5142192606	15	13	2	7.12	0.14	7.12	0.00	4.06
5142204384	253	199	54	4.04	0.25	4.04	79.33	4.04
5142284940	27	21	6	3.84	0.26	3.84	4.00	3.78
5142189113	30	6	24	0.27	3.65	3.65	5.00	3.63
5142225308	21	4	17	0.26	3.88	3.88	2.00	3.53
5142116864	76	58	18	3.53	0.28	3.53	20.33	3.53
5142293257	15	2	13	0.17	5.93	5.93	0.00	3.47
5142173286	15	2	13	0.17	5.93	5.93	0.00	3.47
5142246929	104	79	25	3.46	0.29	3.46	29.67	3.46
5142224699	32	7	25	0.31	3.26	3.26	5.67	3.25
5142847398	45	10	35	0.31	3.19	3.19	10.00	3.19
5142273608	27	6	21	0.31	3.19	3.19	4.00	3.15
5142147267	98	22	76	0.32	3.15	3.15	27.67	3.15
5142224769	20	15	5	3.29	0.30	3.29	1.67	2.92
5142242241	67	16	51	0.34	2.91	2.91	17.33	2.91
5142260984	366	265	101	2.88	0.35	2.88	117.00	2.88
5142113192	14	2	12	0.18	5.47	5.47	-0.33	2.87
5142191416	25	18	7	2.82	0.35	2.82	3.33	2.76
5142308889	13	11	2	6.03	0.17	6.03	-0.67	2.71
5142194228	13	11	2	6.03	0.17	6.03	-0.67	2.71
5142195887	15	12	3	4.38	0.23	4.38	0.00	2.69
5142212038	15	12	3	4.38	0.23	4.38	0.00	2.69
5142225184	38	27	11	2.69	0.37	2.69	7.67	2.69
5142257356	200	142	58	2.68	0.37	2.68	61.67	2.68
5142216493	19	14	5	3.07	0.33	3.07	1.33	2.64
5142239106	31	8	23	0.38	2.62	2.62	5.33	2.62
5142144593	18	4	14	0.31	3.19	3.19	1.00	2.60
5142126842	54	38	16	2.60	0.38	2.60	13.00	2.60
5142117315	27	7	20	0.38	2.61	2.61	4.00	2.58
5142218798	30	21	9	2.56	0.39	2.56	5.00	2.55
5142180432	83	58	25	2.54	0.39	2.54	22.67	2.54
5142264155	39	27	12	2.47	0.41	2.47	8.00	2.47
5142294614	20	5	15	0.37	2.74	2.74	1.67	2.46

Top 40 Merchnum (Potential Fraud Based on Benford's Law)

merchnum	n	n_low	n_high	R	1/R	U	t	U*
991808369338	181	1	181	0.01	165.15	165.15	55.33	165.15
8078200641472	60	59	1	64.66	0.02	64.66	15.00	64.66
308904389335	53	1	53	0.02	48.36	48.36	12.67	48.36
3523000628102	34	34	1	37.26	0.03	37.26	6.33	37.20
808998385332	37	1	36	0.03	32.85	32.85	7.33	32.83
55158027	28	27	1	29.59	0.03	29.59	4.33	29.22
8916500620062	31	1	31	0.04	28.28	28.28	5.33	28.15
3910694900001	26	25	1	27.40	0.04	27.40	3.67	26.74
8889817332	25	24	1	26.30	0.04	26.30	3.33	25.43
881145544	24	24	1	26.30	0.04	26.30	3.00	25.10
5600900060992	28	1	27	0.04	24.64	24.64	4.33	24.33
6844000608436	23	23	1	25.21	0.04	25.21	2.67	23.64
5803301245621	22	21	1	23.02	0.04	23.02	2.33	21.07
92891948003	24	1	24	0.05	21.90	21.90	3.00	20.91
3433000017263	56	53	3	19.36	0.05	19.36	13.67	19.36
467615916337	22	1	22	0.05	20.07	20.07	2.33	18.39
817004638227	20	19	1	20.82	0.05	20.82	1.67	17.67
2376700063599	32	30	2	16.44	0.06	16.44	5.67	16.39
993620816222	20	1	19	0.06	17.34	17.34	1.67	14.74
993620810220	81	5	76	0.07	13.87	13.87	22.00	13.87
8999000079657	19	1	18	0.06	16.42	16.42	1.33	13.21
465614140337	19	1	18	0.06	16.42	16.42	1.33	13.21
8317600900099	26	24	2	13.15	0.08	13.15	3.67	12.85
5000006000095	276	253	23	12.06	0.08	12.06	87.00	12.06
5186264200136	18	1	17	0.06	15.51	15.51	1.00	11.61
9420966064460	18	1	17	0.06	15.51	15.51	1.00	11.61
600000201284	51	4	47	0.09	10.72	10.72	12.00	10.72
7080606900600	17	1	16	0.07	14.60	14.60	0.67	9.99
5600000060302	17	1	16	0.07	14.60	14.60	0.67	9.99
6070095870009	29	26	3	9.50	0.11	9.50	4.67	9.42
999960264339	31	3	28	0.12	8.52	8.52	5.33	8.48
555400670006	16	1	15	0.07	13.69	13.69	0.33	8.39
881894855	16	1	15	0.07	13.69	13.69	0.33	8.39
1960400470068	26	23	3	8.40	0.12	8.40	3.67	8.22
993620559229	48	5	43	0.13	7.85	7.85	11.00	7.85
2644006060269	14	13	1	14.25	0.07	14.25	-0.33	6.53
6000330043193	14	13	1	14.25	0.07	14.25	-0.33	6.53
8124906575841	34	29	5	6.36	0.16	6.36	6.33	6.35
6880098906148	180	23	157	0.16	6.23	6.23	55.00	6.23
6020094951312	24	3	21	0.16	6.39	6.39	3.00	6.13