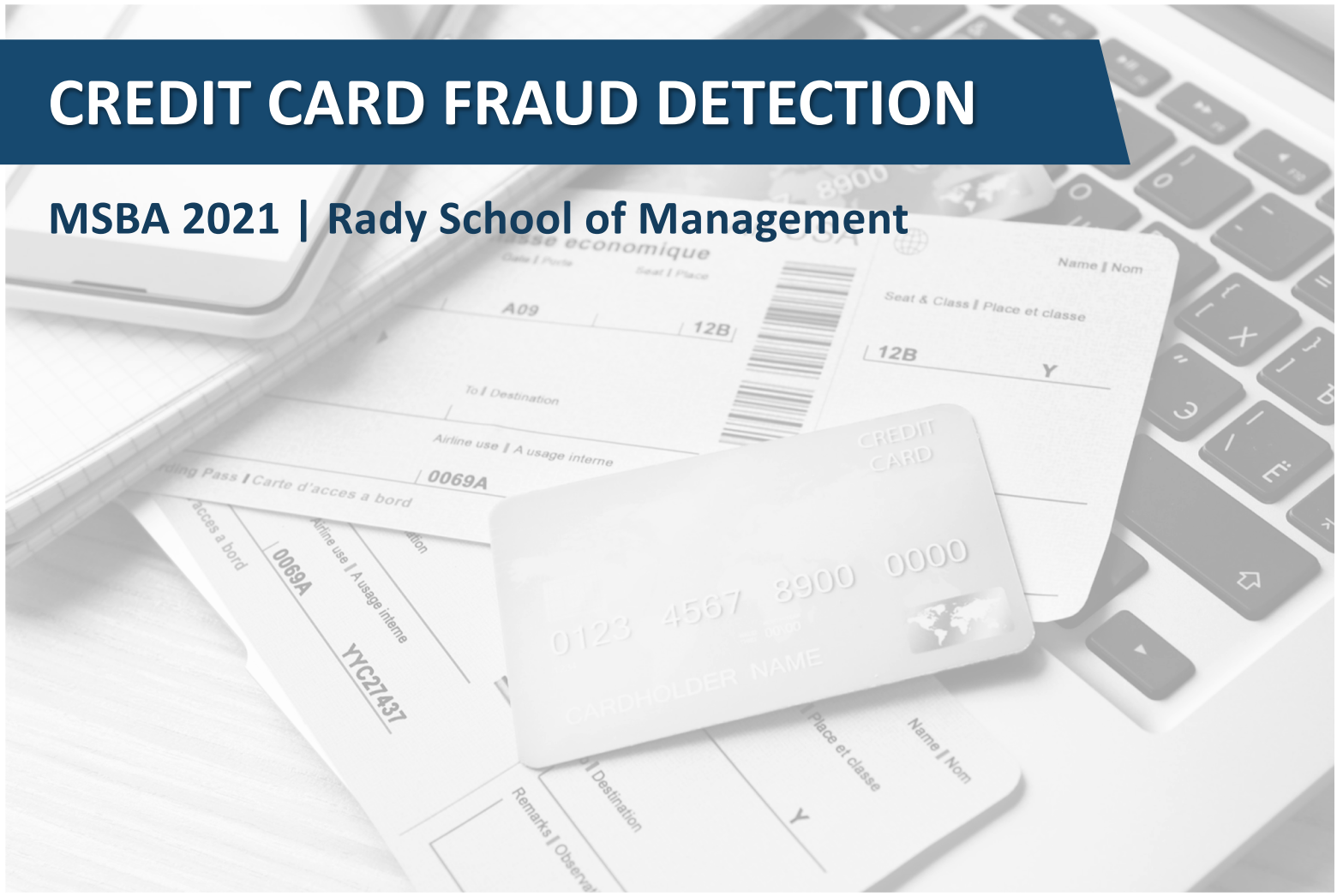
EAST ASIA INTELLIGENCE CONSULTING

CREDIT CARD FRAUD DETECTION

MSBA 2021 | Rady School of Management



Group 11

Seyoung Ahn, Wil Son Chuah, Li Du
Chia-Hsien Ho, Yuying Liu, Yuxiang Zhou

MGTA 463 | Fraud Analytics
June 11, 2021

Table of Contents

Executive Summary

Credit card fraud is a huge problem for small businesses, banks and card issuers as well as cardholders. Global card fraud losses reached $28.65 billion in 2019 alone, and the U.S. accounts for more than a third of the total losses, according to the Nilson Report[1]. The COVID-19 pandemic has fueled the growth of credit card frauds in 2020, and the loss of $11 billion is expected in the U.S.[2] With card frauds rapidly increasing and patterns constantly changing, banks and card issuers have been using technological solutions to detect more frauds, faster.

The goal of this project was to build an effective and efficient model that can identify fraudulent credit card transactions in real time. A real-word dataset that contained 96,753 transactions in 2010 was used, and multiple supervised machine learning algorithms were built and tested to find the best model. The process of developing the models is as follows:

Data Cleaning. We performed EDA to identify any idiosyncrasies in the data. An outlier in the 'amount' field and transactions other than the type 'P' were removed, and missing values in 'Merchnum', 'Merch description', 'Merch state', and 'Merch zip' fields were imputed.
Feature Engineering. We attempted to create as many candidate variables as possible. A total of 551 variables were created using 9 different entities, and they help analyze 5 different aspects of card transactions: amount, frequency, recency, velocity change, and fraud risks (state / weekday).
Feature Selection. We used filter and wrapper methods to eliminate less important variables which may cause noise when included in the model. KS (Kolmogorov-Smirnov) and FDR (Fraud Detection Rate) at 3% were used to measure variables' importance and selected the top 30 variables.
Modeling. We built a baseline linear model using logistic regression and several nonlinear models using algorithms including Neural Network, Random Forest and Gradient Boosted Tree (XGBoost). Data was divided into train, test and OOT(Out-Of-Time) sets for model development and evaluation. For each algorithm, we built multiple models using several combinations of hyperparameters, and each model was tested 10 times with a different train/test split per hyperparameter combination. Models were compared using the measurement of FDR at 3%. Among all the models we built, Gradient Boosted Tree using XGBoost (*learning_rate=0.1, max_depth=6, min_child_weight=5*) achieved FDR scores of 93.46%, 88.65% and 61.97% on train, test and OOT data respectively and was selected as our final, best model.
Results. The final model was tested one last time to confirm that the model is performing well on the OOT set and without any overfitting issues. Three performance tables were created to show the results in detail. We also performed fraud savings calculations and analyzed fraud score patterns over time.

This report details the process of developing a fraud detection model and high-level explanations of statistical methods and algorithms used. It also presents results from the final model.

1. https://nilsonreport.com/content_promo.php?id_promo=16
2. https://www.cnbc.com/2021/01/27/credit-card-fraud-is-on-the-rise-due-to-covid-pandemic.html

1. Data Description

1.1 File description

The file “card transactions.csv” contains information of actual credit card purchases provided by a US government organization. 96,753 records covered information of 10 fields, ranging across the full year of 2010. There are only 1059 records labeled as fraudulent cases (including a few artificial frauds), taking up a small fraction of all transactions. This is a small dataset of imbalanced data, which will bring challenges in later analysis.

1.2 Field description

1.2.1 Numeric Field

Field Name	Count	Mean	Std	Min	Max	Populated(%)
Amount	96,753	427.89	10,006.14	0.01	3,102,045.53	100

1.2.2 Categorical Fields

Field Name	Counts	Populated (%)	Unique Records	Most Common Values	Most Common Values
Recnum	96,753	100	96753	2049	--
Cardnum	96,753	100	1645	5142148452	1192
Date	96,753	100	365	2/28/10	684
Merchnum	93,378	96.51	13092	930090121224	9310
Merch description	96,753	100	12850	FEDEX SHP	11767
Merch state	95,558	98.76	228	TN	12035
Merch zip	92,097	95.19	4568	38118	11868
Transtype	96,753	100	4	P	96398
Fraud	96,753	100	2	0	95694

1.3 Field Examples

1.3.1 Field name: Amount

Description: The amount of each transaction.
Type: Numeric
This is the only numeric field in the dataset. Most transactions are below $5,000, and the rest are below $50,000, except one outlier of $3102045.53. This outlier is excluded in later analysis.

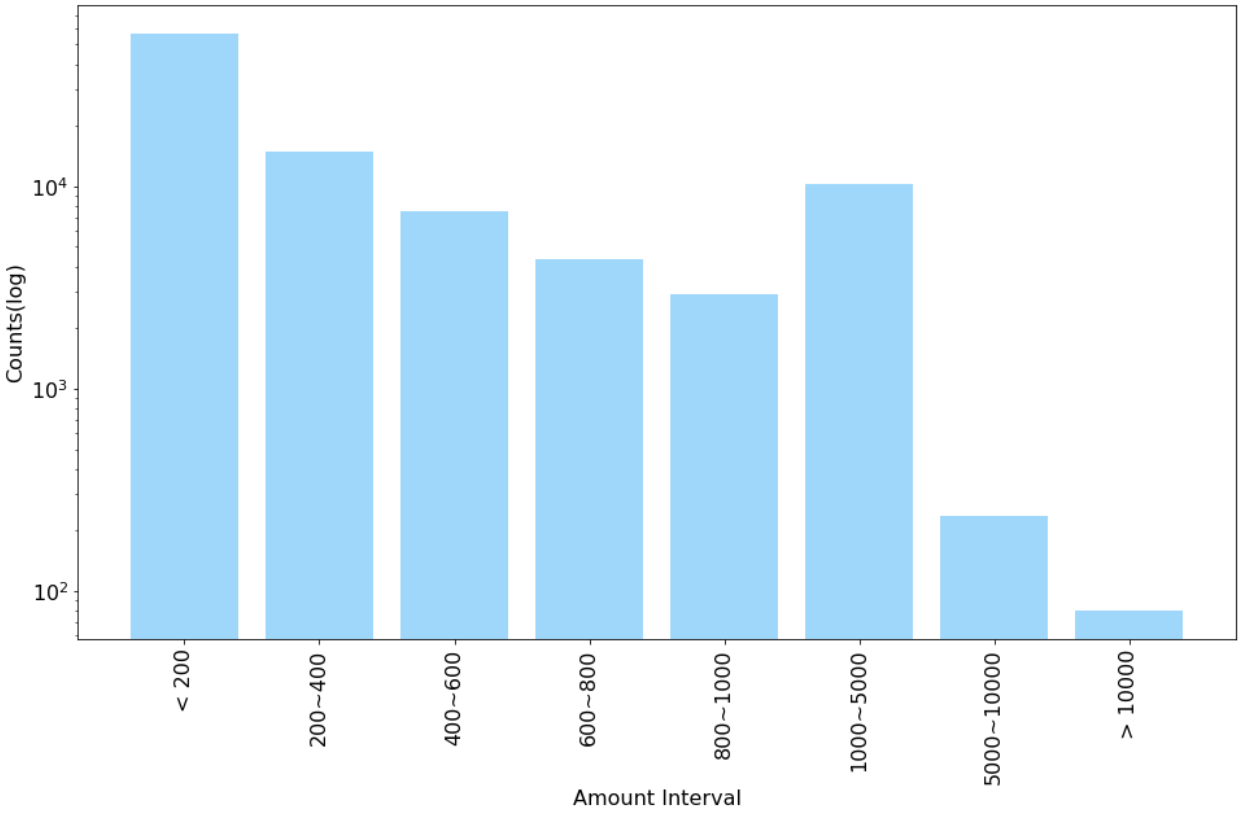


Figure 1.3.1 Frequency distribution of transactions amount

1.3.2 Field name: Date

Description: date of each transaction

On average, there are more transactions on workdays than on weekends.

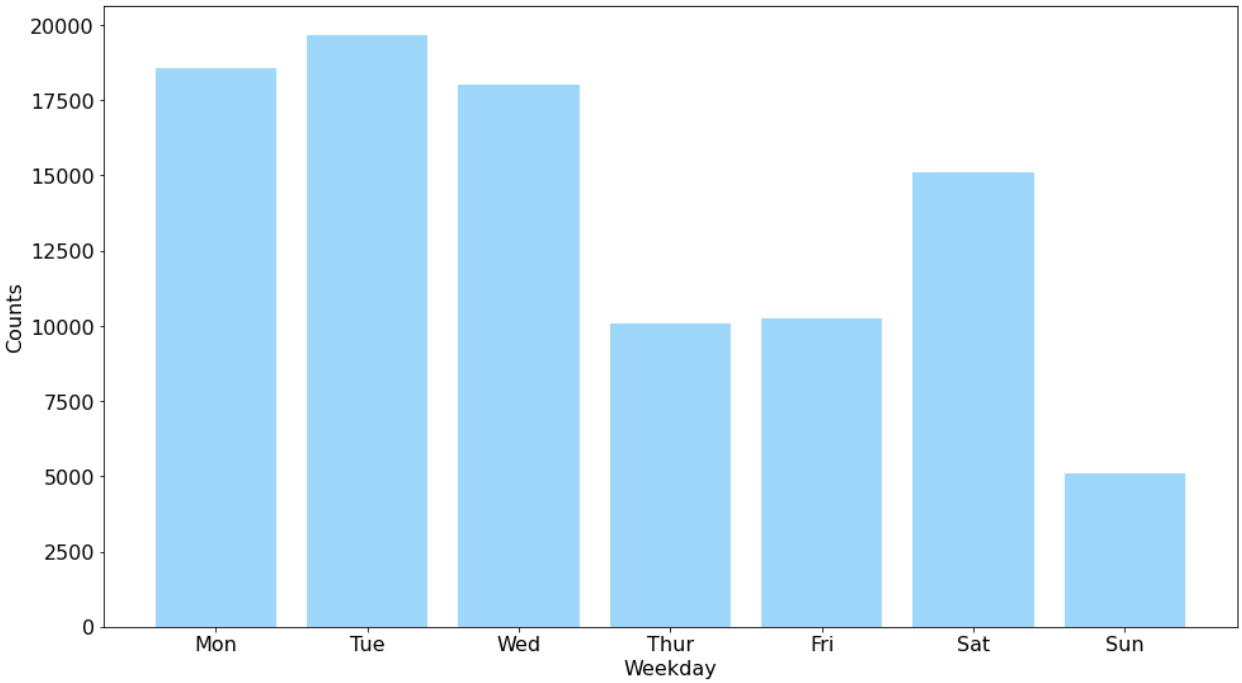


Figure 1.3.2 - 1 Frequency distribution of transactions on weekdays

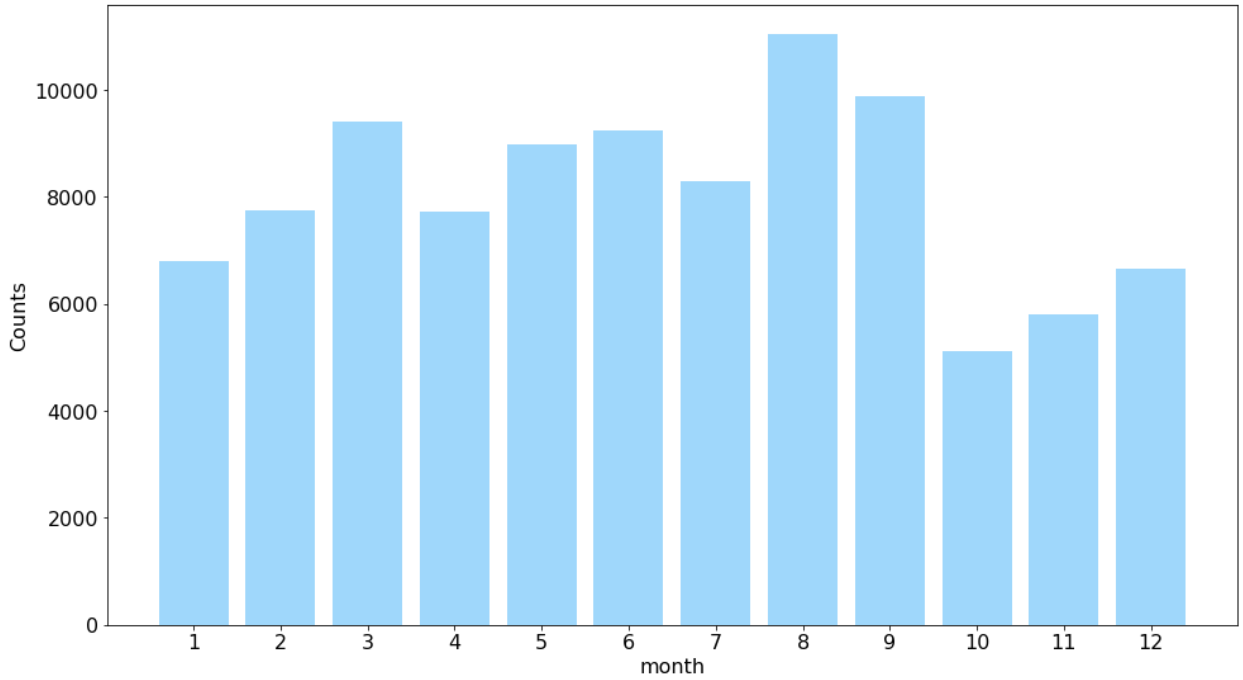


Figure 1.3.2 - 2 Frequency distribution of transactions in each month

1.3.3 Field name: Merch Description

Description: Merchant description.

Type: Categorical

(All records of Fedex Shipping service, regardless of date, are aggregated into 'FEDEX SHP'.)

This plot shows the frequency distribution of merchants whose number of transactions is greater than 200. 51 merchants contributed 35,342 transactions, 36.53% of total transactions. Fedex shipping service alone took up more than 12% of total transactions.

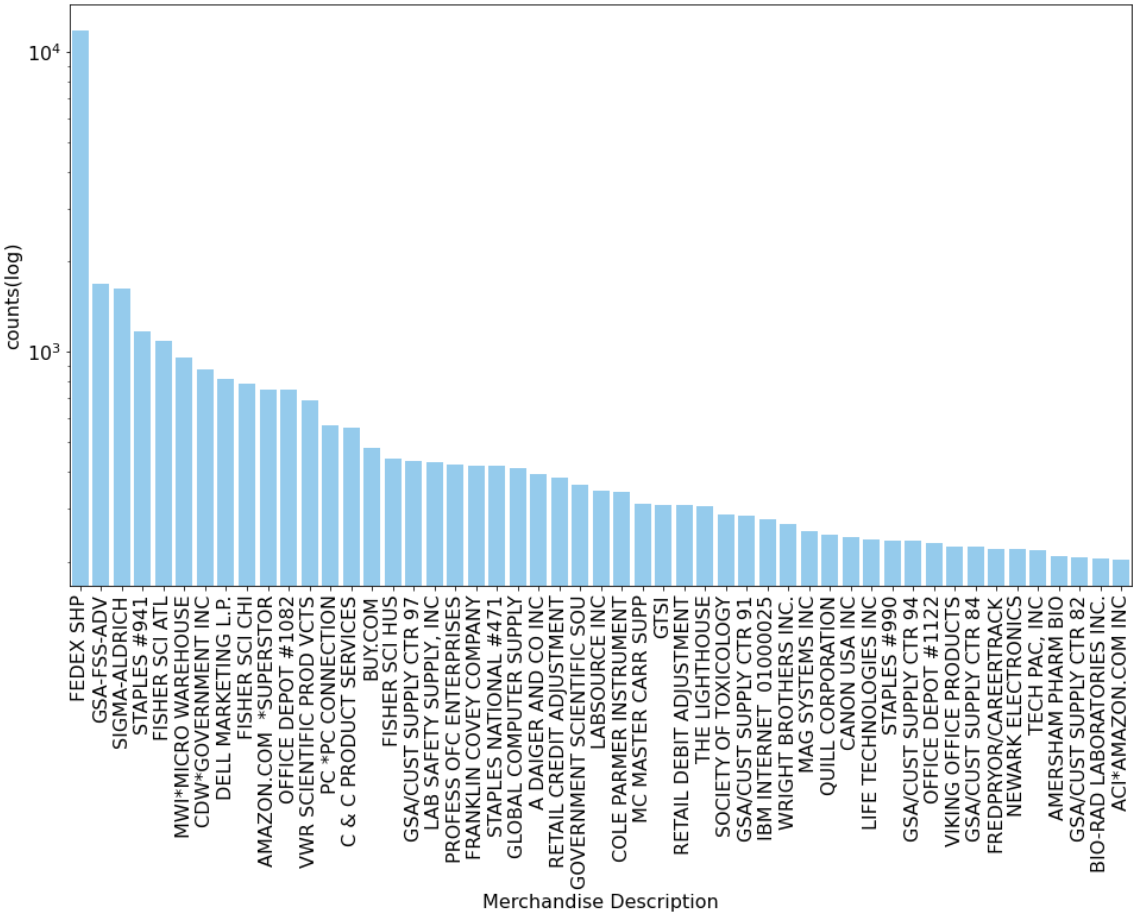


Figure 1.3.3 Frequency distribution of merchants with more than 200 transactions

1.3.4 Field name: Merch state

Description: The state where the transaction happened.

Type: Categorical

This field is 98.76% populated. A small number (273) of transactions which had missing merchant state, and a part of them happened outside the US.

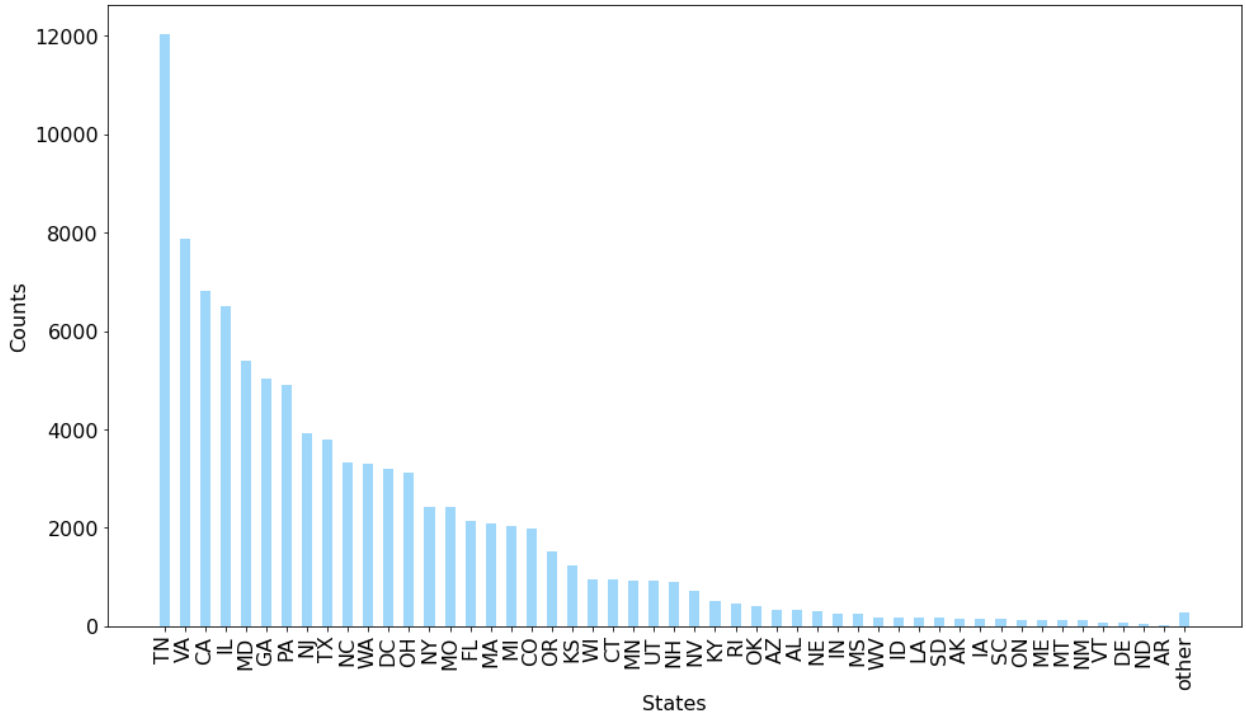


Figure 1.3.4 Frequency distribution of merchants in different states

1.3.5 Field name: Transaction type

Description: The state where the transaction happened.

Type: Categorical

99.63% of transactions were type P, which is also the only type that contains frauds.

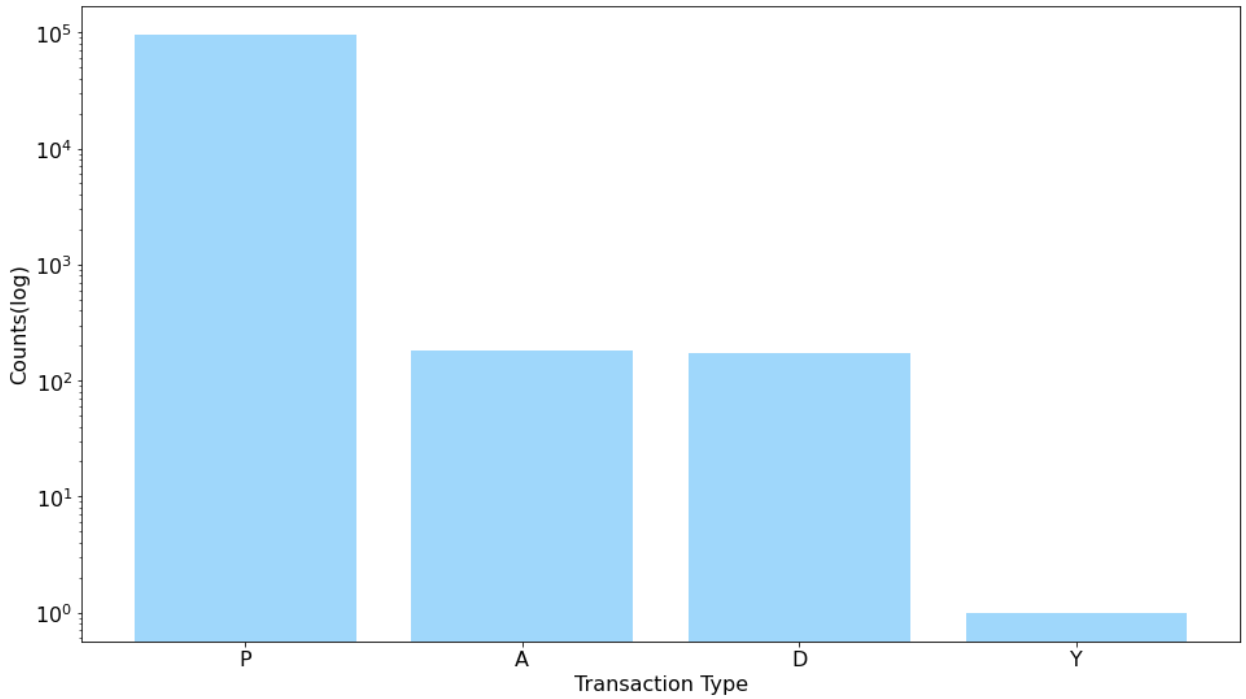


Figure 1.3.5 Frequency distribution of transaction types

1.3.6 Field name: Fraud

Description: Binary variable where 1 means a fraud, and 0 means a normal transaction.

Type: Categorical

Among 96,753 transaction records, only 1059 entries are labeled as fraud, taking up 1.09% of all transactions.

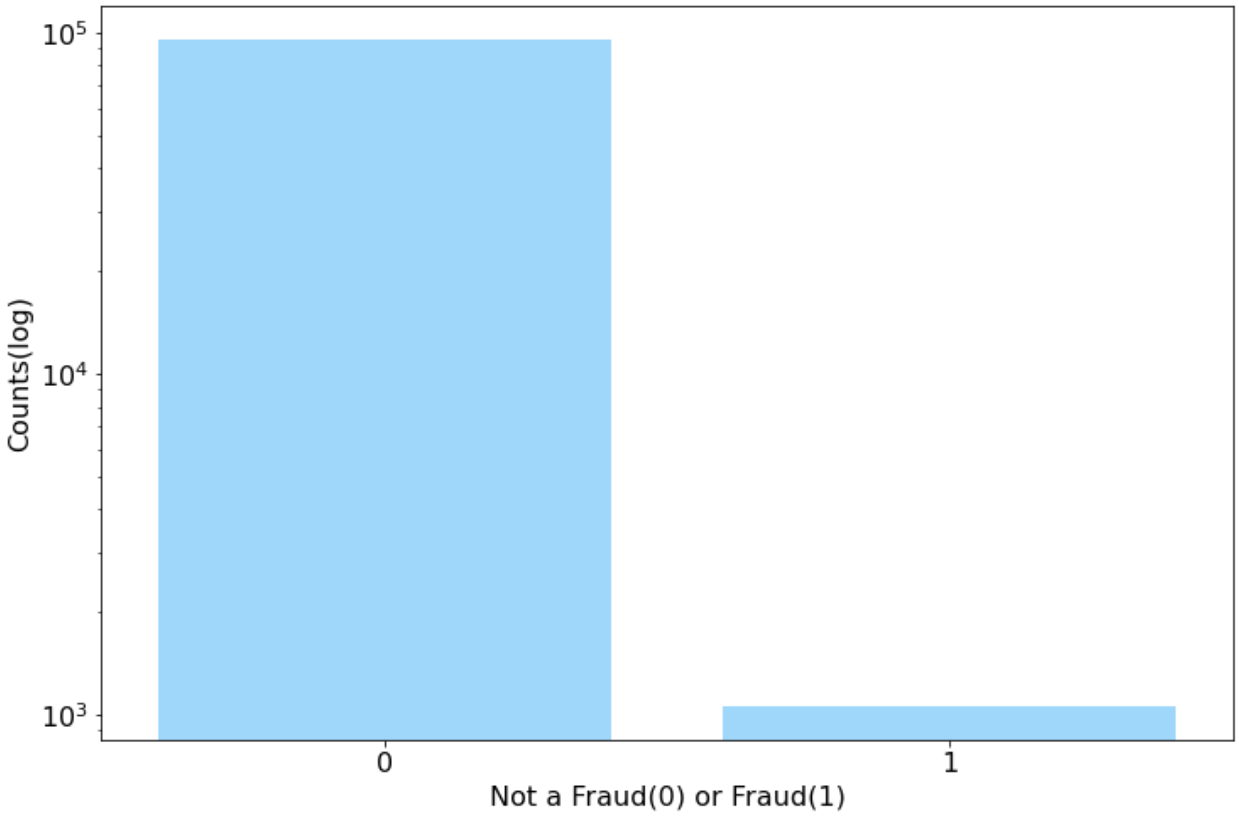


Figure 1.3.6 Frequency distribution of fraud and non-fraud transactions

2. Data Cleaning

We found three fields with missing values during the data exploration stage, which will have an effect on model efficiency. In order to resolve this problem, these missing values have to be replaced with reasonable values.

The following table shows how the missing values were replaced:

Field	Missing Value	Replaced With
Merch State	1. NaN with a zip code 2. Range 00600 - 00799, 00900 - 00999 3. NaN without a zip code 4. NaN	1. State for that specific zip 2. “PR” 3. Mode of the Merchnum or Merch description 4. “Unk”
Merchnum	1. 0 2. NaN 3. NaN without Merch description	1. NaN 2. Mode of Merch Description 3. “Unk”
Merch Zip	NaN	Mode of merchant number

3. Feature Engineering

A total of 551 variables were created based on the existing P1 field **(card, merchant, card at this merchant, card in this zip code, card in this state, merchant in this zip code, and merchant in this state, merchant and description,** and **card and description variables)** to describe the structure inherent in the data, thus better quantifying the characteristics of fraud behaviors. Table 3.1 summarizes the description and the number of variables created in each category.

Table 3.1: Summary of 551 Candidate variables

Variable	Description	# of variables created
Amount	Aggregate functions the value of transaction and their relative ratio to the actual value of transaction	432
Frequency	Number of transaction over a past particular time frame	54
Day Since	Number of transaction records of the same PI showed over a specific time before	9
Velocity change	Measure of the momentum change of number of transaction of the same PI on the same day or past 1 day by comparing with average momentum change of number of transaction of the same PI over the past specific time frame	52
Target Encoding	Measure of fraud risk across the weekday and the state	2

3.1 Amount

We try to explore different type of aggregate functions (**average, maximum, median, total**) and also taking ratio of the actual value to the aggregate functions (**actual/average, actual/maximum, actual/median, actual/total**) of amount at this **card, merchant, card at this merchant, card in this zip code, card in this state, merchant in this zip code, and merchant in this state, merchant and description,** and **card and description variables** over a past amount of period (**0 days, 1 day, 3 days, 7 days, 14 days, 30 days**). In total, we obtain 8 x 9 x 6 (432) variables.

$$\begin{bmatrix} Average \\ Maximum \\ Median \\ Total \\ Actual/average \\ Actual/maximum \\ Actual/median \\ Actual/total \end{bmatrix} \text{ amount by/at this } \begin{bmatrix} 'Card' \\ 'Merchant' \\ 'Card\ at\ this\ merchant' \\ 'Card\ in\ this\ zip\ code' \\ 'Card\ in\ this\ state' \\ 'Merchant\ in\ this\ zip\ code' \\ 'Merchant\ in\ this\ state' \\ 'Merchant\ and\ description' \\ 'Card\ and\ description' \end{bmatrix} \text{ over the past } \begin{bmatrix} 0\ days \\ 1\ day \\ 3\ days \\ 7\ days \\ 14\ days \\ 30\ days \end{bmatrix}$$

3.2 Frequency

We would like to observe the number of transactions with this **card** and **merchant** in general over the past certain period of time windows. For our case, we decided to fix the time windows over the past **0 days**, **1 day**, **3 days**, **7 days** (1 week), **14 days** (1 fortnight), and **30 days** (1 month). On top of the general case (group by card or group by merchant), we also took these variables to create combinations among themselves and obtain **card at this merchant variable**. We then use card variables to combine with zip code, state and description variables to create a **card in this zip code, card in this state**, and **card and description** variables. For the merchant variable, we use it to combine with zip code, state, and description as well to create **merchant in this zip code**, **merchant in this state**, and **merchant and description** variables. Number of transactions of all these new variables we created are also observed over the past time frame that we set. In total, we would get 9 x 6(54) of frequency variables.

$$\text{Number of transactions with this } \begin{bmatrix} \text{'Card'} \\ 'Merchant' \\ 'Card\ at\ this\ merchant' \\ 'Card\ at\ this\ zip\ code' \\ 'Card\ in\ this\ state' \\ 'Merchant\ in\ this\ zip\ code' \\ 'Merchant\ in\ this\ state' \\ 'Merchant\ and\ description' \\ 'Card\ and\ description' \end{bmatrix} \text{ over the past } \begin{bmatrix} 0\ days \\ 1\ day \\ 3\ days \\ 7\ days \\ 14\ days \\ 30\ days \end{bmatrix}$$

3.3 Days since last seen

The logic we create the days since variables is because if the same record appeared too frequently in the short period of time, it has a high possibility that it is a fraud. We are interested to find the duration of the most current date and the most recent transaction with the same **card, merchant, card at this merchant, card in this zip code, card in this state, merchant in this zip code, and merchant in this state, merchant and description,** and **card and description variables**. If the record is first time seen, we would use 365 days for the day since variables. We created a total of 9 days-since variables.

$$\text{Current date minus date of most recent with the same} \begin{bmatrix} \text{'Card'} \\ 'Merchant' \\ 'Card\ at\ this\ merchant' \\ 'Card\ at\ this\ zip\ code' \\ 'Card\ in\ this\ state' \\ 'Merchant\ in\ this\ zip\ code' \\ 'Merchant\ in\ this\ state' \\ 'Merchant\ and\ description' \\ 'Card\ and\ description' \end{bmatrix}$$

* Used 365 days-since if it is first time seen

3.4 Velocity change

Velocity change is another way to capture the **frequency** of transactions of the same **card, merchant, card at this merchant, card in this zip code, card in this state, merchant in this zip code, and merchant in this state, merchant and description,** and **card and description variables**, showed over certain past time window of **0 days** and **1 day** divided by the average daily **frequency** of transactions with the same **card, merchant, card at this merchant, card in this zip code, card in this state, merchant in this zip code, and merchant in this state, merchant and description,** and **card and description variables** over the particular past time window of **7days**(1 week), **14 days**(1 fortnight), **30 days**(1 month). A bigger value means frequent indicates a higher likelihood of fraud as when it compares to average number of transactions in a week, a fortnight and a month. The past time window set for the base here is 7,14,30 days, and in the numerator the past time frame is 0 days and 1 day.0 days meaning the same day of the last application. As a result, we obtain 1 x 9 x 2 x 3 = 54 variables.

$$[Number] \text{ of transactions with same } \begin{bmatrix} 'Card' \\ 'Merchant' \\ 'Card\ at\ this\ merchant' \\ 'Card\ in\ this\ zip\ code' \\ 'Card\ in\ this\ state' \\ 'Merchant\ in\ this\ zip\ code' \\ 'Merchant\ in\ this\ state' \\ 'Merchant\ and\ description' \\ 'Card\ and\ description' \end{bmatrix} \text{ over the past } \begin{bmatrix} 0\ days \\ 1\ day \end{bmatrix}$$

$$\text{Avearge daily } [Number] \text{ of transactions with same } \begin{bmatrix} 'Card' \\ 'Merchant' \\ 'Card\ at\ this\ merchant' \\ 'Card\ in\ this\ zip\ code' \\ 'Card\ in\ this\ state' \\ 'Merchant\ in\ this\ zip\ code' \\ 'Merchant\ in\ this\ state' \\ 'Merchant\ and\ description' \\ 'Card\ and\ description' \end{bmatrix} \text{ over the past } \begin{bmatrix} 7\ days \\ 14\ days \\ 30\ days \end{bmatrix}$$

3.5 Target Encoding

We compute another 2 special variables which are fraud risk over the weekday of the week and different states. The reason for observing the fraud risk over the weekday of the week is because it contains information of fraudsters' preference of committing fraud on a particular weekday. Besides that, by creating fraud risk over different states will give us information of demographic behavior of particular states.

$$\text{Fraud risk over } \begin{bmatrix} 'the\ weekday\ of\ the\ week' \\ 'that\ state' \end{bmatrix}$$

4. Feature Selection

With over 560 candidate variables for the supervised fraud algorithms, it is important to implement feature selection to minimize dimensionality of the model and include only variables with substantial information. To begin with, we narrow the variables down to 80 most favorable candidates with highest rankings by choices of measures. The next step for feature selection is to conduct forward selection on the 80 variables using a simple nonlinear model and a predefined wrapper function. After forward selection, the resulting top 30 variables will be used for the following supervised fraud models.

4.1 Filter

The first step is to filter out 80 variables by the rankings of all candidates using measures of goodness. The most common measures of goodness for fraud variables include Kolmogorov-Smirnov (KS), Fraud Detection Rate (FDR), and False Positives. For the present project, the KS and FDR at 3% for each candidate variable are calculated and their arithmetic averages are used as the index by which all candidate variables are ranked.

4.1.1 Kolmogorov-Smirnov (KS)

By definition, KS is the maximum difference between the cumulatives of the distributions of the two levels (i.e. goods & bads) in the **fraud** label in the card transactions data. It is a robust measure of how well these two distributions are separated and therefore how well the variable classifies data into fraudulent and non-fraudulent transactions. Mathematically, KS is calculated as follows:

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

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

4.1.2 Fraud Detection Rate (FDR)

The fraud detection rate at 3% is the percentage of detected frauds among all the fraudulent transactions in the top 3% of the population. It is commonly used in business for its robustness and meaningfulness compared to other measures like False Positives.

4.1.3 KSFDR

To integrate KS and FDR into one score, take the arithmetic average of the two, KSFDR. Then, rank the candidates by KSFDR and select the top 80 variables as filtered candidates.

4.2 Wrapper

The second step is to implement forward selection with a random forest model using the wrapper function. After the forward selection process is finished, choose the 30 first selected variables as the final variables for the supervised fraud models.
The major method employed is Forward Selection, which go as follows:
Given N candidates variables:

1. Build N separate one-dimensional models.
2. Keep the best variable, and continue to build (N-1) separate two-dimensional models, all including the first best variable.
3. Keep the second best variable.
4. Recursively repeat the steps above until desirable numbers of variables have been selected.

The advantage of forward selection is that it will find good subsets of all candidate variables and avoid correlations between variables in the model. On the other hand, a simple random forest classifier was used as the nonlinear wrapper in the forward selection process.

4.3 Top 30 Variables

The table below displays the resulting top 30 variables that will be used in the fraud models:

1	card_desc_total_3	16	merch_desc_max_3
2	card_state_max_30	17	card_state_total_30
3	card_state_total_7	18	merch_zip_max_3
4	card_zip_max_1	19	Merchnum_total_1
5	merch_desc_total_0	20	merch_zip_total_1
6	card_desc_total_30	21	card_state_max_7
7	card_desc_total_14	22	card_zip_total_1
8	Cardnum_total_0	23	merch_desc_total_1
9	merch_state_total_1	24	merch_state_total_0
10	card_merch_max_1	25	Merchnum_total_3
11	merch_state_total_3	26	card_merch_total_30
12	card_zip_total_14	27	merch_zip_max_0
13	card_zip_max_14	28	card_zip_max_30
14	merch_desc_max_0	29	Cardnum_total_7
15	merch_state_max_1	30	merch_desc_max_1

Please see Appendix B for a list of top 80 variables with KS, FDR and KS FDR scores.

5. Model Algorithms

5.1 Model Summary

To get a more robust model, we compared different algorithms with several combinations of parameters, including Logistic Regression, Random Forest, XGBoost, and Neural Network. To prevent overfitting, we randomly split the dataset into ⅔ training and ⅓ testing 10 times and calculate the average FDR at a 3% rejection rate to compare different models. XGBoost performed best and was selected as the final model. Details of different models are shown in the model performance table.

Table 5.1 Model Performance Summary

Model	Parameters				Average FDR(%) at 3%		
Logistic Regression	iteration	C	penalty	solver	Train	Test	OOT
	1	1	l1	liblinear	69.90	67.24	54.47
	2	10	l1	liblinear	69.88	69.11	54.13
	3	100	l1	liblinear	70.23	67.32	55.34
	4	0.1	l2	lbfgs	67.76	67.55	54.47
	5	1	l2	lbfgs	70.00	67.43	55.31
	6	10	l2	lbfgs	70.21	67.30	54.72
	7	100	l2	lbfgs	69.88	67.63	54.47
	8	1	l2	newton-cg	69.71	68.89	55.39
	9	10	l2	newton-cg	69.03	69.47	54.30
	10	100	l2	newton-cg	69.69	68.08	54.10
Random Forest	iteration	n_estimators	max_depth	criterion	Train	Test	OOT
	1	30	10	gini	90.44	83.24	64.02
	2	60	15	gini	99.89	85.90	64.63
	3	60	30	entropy	100.00	86.85	62.30
	4	80	15	entropy	100.00	88.13	62.50
	5	80	30	gini	100.00	88.59	63.34
	6	100	15	gini	99.98	86.87	64.75
	7	100	30	entropy	100.00	88.32	62.48
	8	120	15	gini	99.98	87.69	64.89
	9	120	30	entropy	100.00	88.17	62.13
	10	150	30	entropy	100.00	89.06	62.89
Gradient Boosting Tree	iteration	learning_rate	max_depth	min_child_weight	Train	Test	OOT
	1	0.1	3	1	89.81	85.73	60.37
	2	0.01	5	3	76.92	73.17	62.50
	3	0.05	6	5	90.14	85.36	61.77
	4	0.1	5	7	91.60	88.03	61.24
	5	0.01	6	1	77.60	76.47	63.03
	6	0.05	3	3	82.51	79.05	60.56
	7	0.1	6	5	93.46	88.65	61.97
	8	0.01	3	7	73.04	69.32	54.55
	9	0.05	5	5	88.93	84.45	60.39
	10	0.05	6	7	89.12	83.79	61.18
Neural Network	iteration	solver	alpha	hidden_layer_sizes	Train	Test	OOT
	1	adam	0.0001	(5,)	74.74	72.68	54.24
	2	sgd	0.05	(10,)	64.78	65.90	50.28
	3	lbfgs	0.0001	(20,)	93.14	79.01	45.20
	4	adam	0.05	(5, 10)	75.00	72.65	55.20
	5	sgd	0.0001	(5, 20)	65.46	63.80	52.53
	6	lbfgs	0.05	(10, 10)	90.69	79.43	48.15
	7	adam	0.0001	(10, 20)	86.58	79.71	54.69
	8	sgd	0.05	(15, 10)	65.47	65.61	51.77
	9	lbfgs	0.0001	(15, 20)	94.76	77.74	43.31
	10	adam	0.05	(20, 20)	83.82	78.80	57.11

5.2 Model Training

To begin with, we applied manual cross validation on each model. Take note that we did not incorporate the OOT data in cross validation, so that it will not cause data leakage and overfitting. The general process of this cross validation is as follows:

1. Randomly shuffle the dataset and then choose a ratio of 0.34 to split it into training sets and testing sets 10 times without setting a random seed, which will give us 10 different combinations of training and testing sets.
2. We fit the model on the training data and evaluate the model on testing data.
3. Check the evaluation score in each of the 10 rounds and make sure that they do not have great variance. (We use FDR at 3% rejection here.)
4. Calculate the average of the performance metrics (FDRs) over the 10 runs for training, testing and OOT dataset.

There are two reasons for performing manual cross-validation:

1. We can evaluate the model each time and then get the average performance of the OOT data at the same time.
2. Standard CV divides the modeling data into 10 distinct parts which will reduce the size of testing sets, while manual cross-validation could solve this problem.

We performed the steps above on various algorithms with many combinations of parameters, including Logistic Regression, Random Forest, XGBoost, and Neural Network. Eventually, XGBoost had the best performance.

5.2.1 Logistic Regression

Logistic regression (also known as logit regression) is a statistical model that uses the logistic function to estimate the probability of an output given two alternative outcomes. When the dependent variable is binary, it is a great regression analysis model to utilize. The model takes into account a number of independent variables as well as a binary dependent variable of 0 or 1. The label “1” generally indicates a happening event or the observation of an item that belongs to a specific category. The label “0” denotes the opposite. Below is a graph that depicts the fundamental of logistic regression:

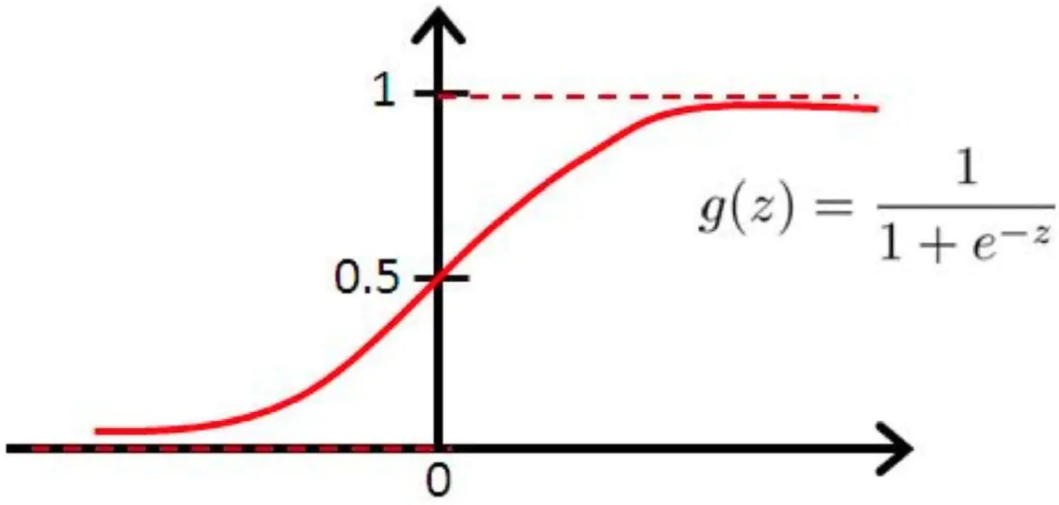


Figure 5.2.1: Logistic Model Illustration

Logistic regression models are commonly employed in classification problems due to their simplicity and efficiency. We decided to use the binary logistic regression model as a baseline model in this fraud project, with all useful variables from our feature selection process as independent variables and fraud labels indicating whether the application in the dataset is fraudulent or non-fraudulent as our dependent variable.

Given the model's versatility, we tuned the following parameters:

Package: sklearn.linear_model.LogisticRegression

Table 5.2.1 Essential Parameters in Logistic Regression

Name	Description
C	Used to specify the norm used in the penalization. We used "l1" and "l2".
penality	Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization. We used 0.1, 1, 10, and 100
solver	Algorithm to use in the optimization problem. We used liblinear, lbfgs, and newton-cg

5.2.2 Random Forest

Random Forest creates a collection of numerous strong, independent trees and averages them. A predictor is a node within a decision tree, and the leaf denotes the number of samples in each class. The model only employs a randomly chosen subset of variables or records for each tree and/or split iteration inside a tree. The model then integrates all of the results by averaging for regression or voting for classification when it has finished creating trees.

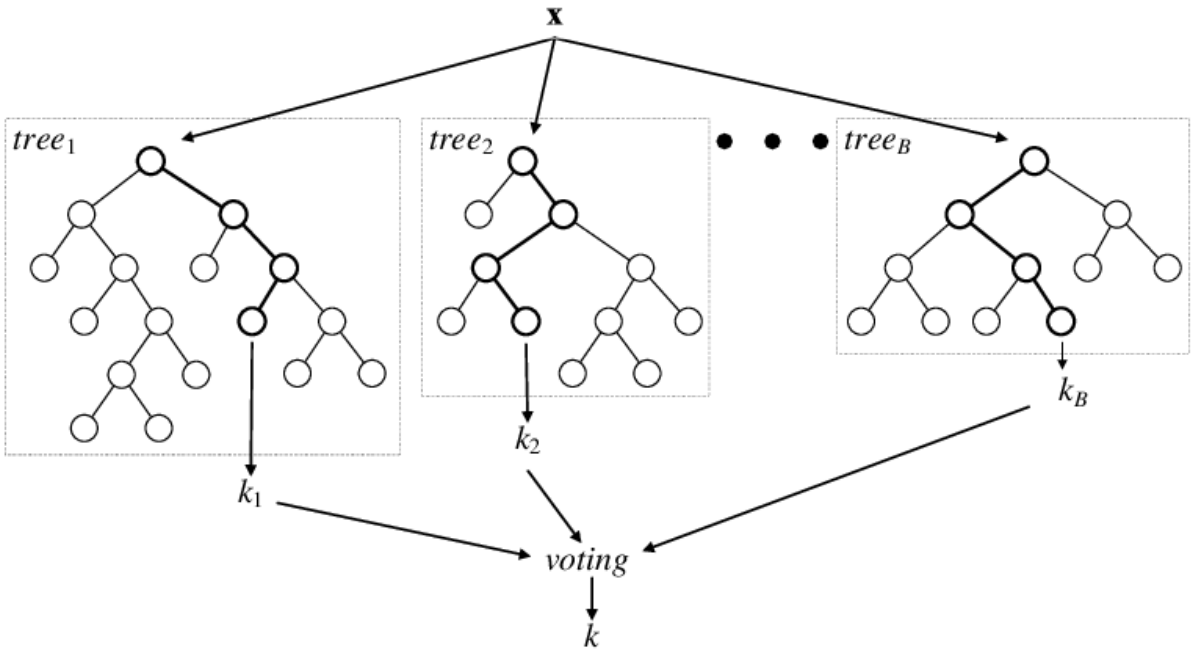


Figure 5.2.2: Random Forest Model Illustration

Given the model's versatility, we tuned the following parameters:

Package: sklearn.ensemble.RandomForestClassifier

Table 5.2.2: Essential Parameters in Random Forest

Name	Description
n_estimators	The number of trees in the forest. At each iteration, we choose a number between 30 and 150.
max_depth	The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. We used 10, 15, and 30.
criterion	The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

5.2.3 XGBoost

Boosted trees classifiers predict the outcome variable by combining the outcomes of a sequence of simple tree models. The goal of each succeeding tree in the series is to capture the preceding tree's residual errors.

XGBoost is a distributed gradient boosting toolkit that has been tuned for efficiency, flexibility and portability. It uses the gradient boosting framework to create machine learning algorithms. XGBoost is a parallel tree boosting (also known as GBDT or GBM) algorithm that solves a variety of data science problems quickly and accurately.

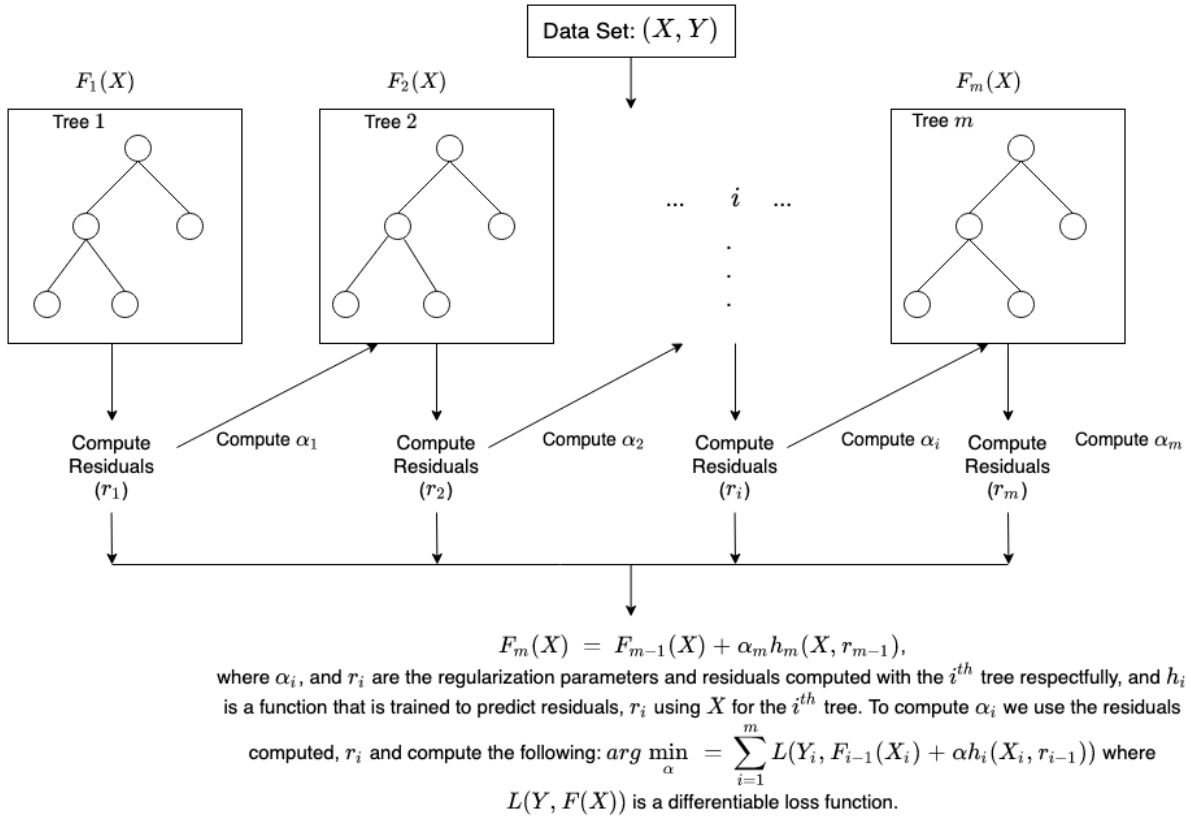


Figure 5.2.3: XGBoost Model Illustration

Given the model's versatility, we tuned the following parameters:

Package: xgboost.XGBClassifier

Table 5.2.3 : Essential Parameters in XGBoost

Name	Description
learning_rate	Size of shrinkage, and it is also named as 'eta'. After each boosting step, we can get the weights of new features, and the learning rate shrinks the feature weights to prevent overfitting.
max_depth	Maximum depth of the tree. Increasing this parameter will also increase the complexity of the model and the likelihood of overfitting.
min_child_weight	Minimum sum of instance weight needed in the child. If a tree partitioning step results in a tree node with the sum of instance weight less than this parameter, it will give up partitioning. Setting this parameter can help to prevent overfitting.

5.2.4 Neural Network

An artificial neural network (ANN) is an algorithm that has the similar simulative structure as the human brain's neural network. The overall network architecture consists of an input layer, hidden layers and an output layer. Each of these layers contains nodes or neurons that collect numerical signals from the previous to the next level for receipt and transmission. Each neuron in the structure receives a signal from the previous layer nodes and applies a signal transmission function and then sends out a new signal to the next layer nodes. A generally linear combination of the outputs of the previous layer nodes is the signal received from the previous layer. A transfer function is then carried out, typically a sigmoid or logit function, through this combined linear combination signal, received by the node. There are also other transfer functions, but the sigmoid/logit is the most common.

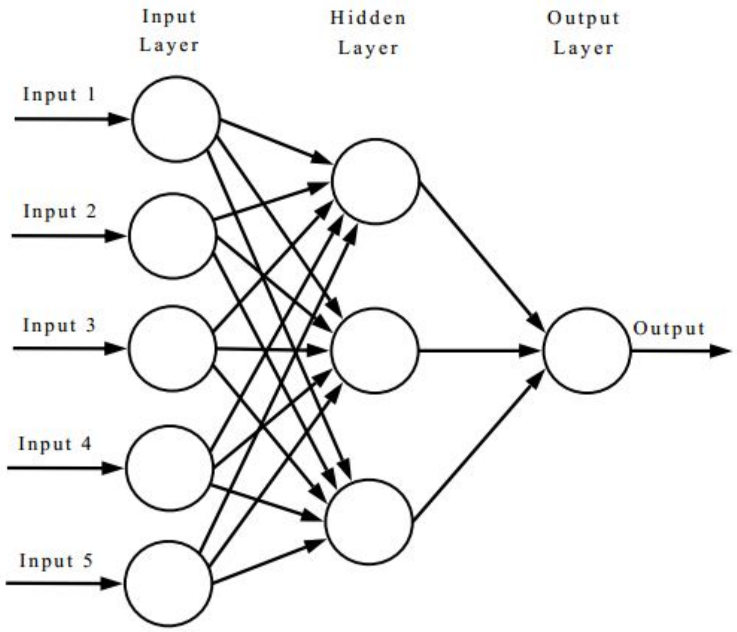


Figure 5.2.4: Neural Network Model Illustration

Given the model's versatility, we tuned the following parameters:

Package: sklearn.neural_network.MLPClassifier

Table 5.2.4: Essential Parameters in Neural Network

Name	Description
solver	The solver for weight optimization. We used the values 'adam', 'sgd' and 'lbfgs'
alpha	L2 penalty (regularization term) parameter.
hidden_layer_sizes	The ith element represents the number of neurons in the ith hidden layer. We use value nodes = 5, 10, 20 and layers = 1, 2

6. Results

6.1 Final Model

We chose the Gradient Boosting Tree algorithm using XGBoost (*learning_rate=0.1, max_depth=6, min_child_weight=5*) as our final model, since it reached the highest FDR at 3% rejection rate on the testing data and a reasonable FDR for OOT data. It achieved an FDR at 3% rejection rate of 93.46% on the training data, 88.65% on the test data, and 61.97% on the OOT data. The performance between training data and test data is less than 5%, which indicates that the model is not overfitting.

We ran the final model again (which may yield different results since we're not using the random seed) and predicted the probabilities of being fraud, and then sorted the observations in descending order. Then we split each dataset into 100 bins, and the statistics of the top 20 bins are shown in the table below. The green part in the left shows the statistics of each population bin, while the blue part in the right shows the cumulative statistics including the current bin and all previous bins.

In the tables for training and testing data, the number of bads detected in the first population bin is the highest among all 100 bins. Moreover, within this 1% population bin with the most number of frauds, the model is able to capture 87.19% of the records with actual frauds in the training data and 73.56% of the records in the testing data. This indicates that the model has a good performance in detecting fraud. Also, the cumulative KS scores in these two tables are mostly above 80%, which indicates that the model can effectively differentiate between the goods and bads.

As for OOT data, it has a slightly worse performance compared to training and test data. It is possibly because the data from the last three months are typically substantially different due to the start of a new fiscal year on October 1st. However, it shouldn't be a concern since we're using the OOT data to observe what we can expect going forward after we fit our model, and the variation of the OOT performance from various models and different sets of parameters does not fluctuate a lot.

Training	# Records	# Bads	# Goods	Fraud Rate								
	48332	494	47838	0.0102								
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Bads	# Goods	% Bads	% Goods	Total # Records	Cumulative Bad	Cumulative Good	% Bad(FDR)	% Good	KS	FPR
1	484	422	62	87.19%	12.81%	484	422	62	85.43%	0.13%	85%	0.15
2	483	28	455	5.80%	94.20%	967	450	517	91.09%	1.08%	90%	1.15
3	483	11	472	2.28%	97.72%	1450	461	989	93.32%	2.07%	91%	2.15
4	484	1	483	0.21%	99.79%	1934	462	1472	93.52%	3.08%	90%	3.19
5	483	4	479	0.83%	99.17%	2417	466	1951	94.33%	4.08%	90%	4.19
6	483	6	477	1.24%	98.76%	2900	472	2428	95.55%	5.08%	90%	5.14
7	484	2	482	0.41%	99.59%	3384	474	2910	95.95%	6.08%	90%	6.14
8	483	2	481	0.41%	99.59%	3867	476	3391	96.36%	7.09%	89%	7.12
9	483	4	479	0.83%	99.17%	4350	480	3870	97.17%	8.09%	89%	8.06
10	477	4	473	0.84%	99.16%	4827	484	4343	97.98%	9.08%	89%	8.97
11	490	1	489	0.20%	99.80%	5317	485	4832	98.18%	10.10%	88%	9.96
12	481	0	481	0.00%	100.00%	5798	485	5313	98.18%	11.11%	87%	10.95
13	486	1	485	0.21%	99.79%	6284	486	5798	98.38%	12.12%	86%	11.93
14	483	1	482	0.21%	99.79%	6767	487	6280	98.58%	13.13%	85%	12.90
15	483	0	483	0.00%	100.00%	7250	487	6763	98.58%	14.14%	84%	13.89
16	435	1	434	0.23%	99.77%	7685	488	7197	98.79%	15.04%	84%	14.75
17	532	0	532	0.00%	100.00%	8217	488	7729	98.79%	16.16%	83%	15.84
18	483	0	483	0.00%	100.00%	8700	488	8212	98.79%	17.17%	82%	16.83
19	481	0	481	0.00%	100.00%	9181	488	8693	98.79%	18.17%	81%	17.81
20	486	1	485	0.21%	99.79%	9667	489	9178	98.99%	19.19%	80%	18.77

Table 6.1.1: Key Statistics of Top 20 Bins in Training Data

Testing	# Records	# Bads	# Goods	Fraud Rate								
	20714	209	20505	0.0101								
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Bads	# Goods	% Bads	% Goods	Total # Records	Cumulative Bad	Cumulative Good	% Bad(FDR)	% Good	KS	FPR
1	208	153	55	73.56%	26.44%	208	153	55	73.21%	0.27%	72.94%	0.36
2	207	26	181	12.56%	87.44%	415	179	236	85.65%	1.15%	84.49%	1.32
3	207	8	199	3.86%	96.14%	622	187	435	89.47%	2.12%	87.35%	2.33
4	207	1	206	0.48%	99.52%	829	188	641	89.95%	3.13%	86.83%	3.41
5	207	2	205	0.97%	99.03%	1036	190	846	90.91%	4.13%	86.78%	4.45
6	206	4	202	1.94%	98.06%	1242	194	1048	92.82%	5.11%	87.71%	5.40
7	208	1	207	0.48%	99.52%	1450	195	1255	93.30%	6.12%	87.18%	6.44
8	208	0	208	0.00%	100.00%	1658	195	1463	93.30%	7.13%	86.17%	7.50
9	207	0	207	0.00%	100.00%	1865	195	1670	93.30%	8.14%	85.16%	8.56
10	207	1	206	0.48%	99.52%	2072	196	1876	93.78%	9.15%	84.63%	9.57
11	207	0	207	0.00%	100.00%	2279	196	2083	93.78%	10.16%	83.62%	10.63
12	207	0	207	0.00%	100.00%	2486	196	2290	93.78%	11.17%	82.61%	11.68
13	206	1	205	0.49%	99.51%	2692	197	2495	94.26%	12.17%	82.09%	12.66
14	206	1	205	0.49%	99.51%	2898	198	2700	94.74%	13.17%	81.57%	13.64
15	209	0	209	0.00%	100.00%	3107	198	2909	94.74%	14.19%	80.55%	14.69
16	117	0	117	0.00%	100.00%	3224	198	3026	94.74%	14.76%	79.98%	15.28
17	298	0	298	0.00%	100.00%	3522	198	3324	94.74%	16.21%	78.53%	16.79
18	206	2	204	0.97%	99.03%	3728	200	3528	95.69%	17.21%	78.49%	17.64
19	208	1	207	0.48%	99.52%	3936	201	3735	96.17%	18.22%	77.96%	18.58
20	207	0	207	0.00%	100.00%	4143	201	3942	96.17%	19.22%	76.95%	19.61

Table 6.1.2: Key Statistics of Top 20 Bins in Testing Data

Out of Time	# Records	# Bads	# Goods	Fraud Rate								
	27351	356	26995	0.0130								
	Bin Statistics					Cumulative Statistics						
Population Bin %	# Records	# Bads	# Goods	% Bads	% Goods	Total # Records	Cumulative Bad	Cumulative Good	% Bad(FDR)	% Good	KS	FPR
1	274	161	113	58.76%	41.24%	274	161	113	45.22%	0.42%	44.81%	0.70
2	273	48	225	17.58%	82.42%	547	209	338	58.71%	1.25%	57.46%	1.62
3	274	15	259	5.47%	94.53%	821	224	597	62.92%	2.21%	60.71%	2.67
4	273	15	258	5.49%	94.51%	1094	239	855	67.13%	3.17%	63.97%	3.58
5	274	10	264	3.65%	96.35%	1368	249	1119	69.94%	4.15%	65.80%	4.49
6	273	9	264	3.30%	96.70%	1641	258	1383	72.47%	5.12%	67.35%	5.36
7	274	7	267	2.55%	97.45%	1915	265	1650	74.44%	6.11%	68.33%	6.23
8	273	8	265	2.93%	97.07%	2188	273	1915	76.69%	7.09%	69.59%	7.01
9	274	9	265	3.28%	96.72%	2462	282	2180	79.21%	8.08%	71.14%	7.73
10	242	4	238	1.65%	98.35%	2704	286	2418	80.34%	8.96%	71.38%	8.45
11	305	9	296	2.95%	97.05%	3009	295	2714	82.87%	10.05%	72.81%	9.20
12	273	2	271	0.73%	99.27%	3282	297	2985	83.43%	11.06%	72.37%	10.05
13	274	6	268	2.19%	97.81%	3556	303	3253	85.11%	12.05%	73.06%	10.74
14	270	2	268	0.74%	99.26%	3826	305	3521	85.67%	13.04%	72.63%	11.54
15	277	1	276	0.36%	99.64%	4103	306	3797	85.96%	14.07%	71.89%	12.41
16	256	4	252	1.56%	98.44%	4359	310	4049	87.08%	15.00%	72.08%	13.06
17	290	1	289	0.34%	99.66%	4649	311	4338	87.36%	16.07%	71.29%	13.95
18	253	1	252	0.40%	99.60%	4902	312	4590	87.64%	17.00%	70.64%	14.71
19	295	2	293	0.68%	99.32%	5197	314	4883	88.20%	18.09%	70.11%	15.55
20	273	2	271	0.73%	99.27%	5470	316	5154	88.76%	19.09%	69.67%	16.31

Table 6.1.3: Key Statistics of Top 20 Bins in Out of Time Data

6.2 Fraud Savings Calculation

After building a model to predict the probability of fraud for each transaction, we then calculate the expected annual savings that we could achieve and recommend a score cutoff that can lead to the highest savings. The blue curve represents the fraud savings, assuming a $2,000 gain for every fraudulent transaction the model caught. The orange curve shows the lost sales, assuming a $50 loss for every false positive prediction from our model. And the green curve shows the overall savings by subtracting the lost sales from the fraud savings.

From the figure below, we achieved a maximum annual savings of $1,365,000 by setting a score cutoff at 9% population bin. However, there are almost 2500 transactions within the 9% population bin. In order to deny the transactions as few as possible, we could adjust the threshold leftward so that the model still detects most of the fraud while remaining a high overall savings.

Please see the plot on the next page.

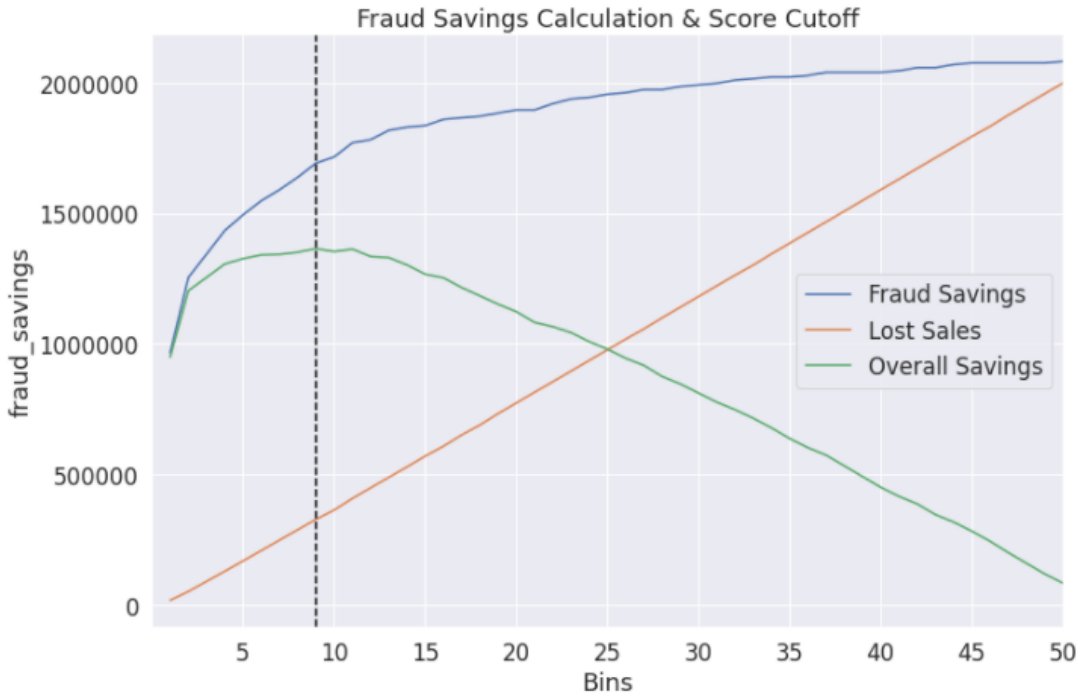


Figure: Fraud Savings Calculation and Score Cutoff Recommendation

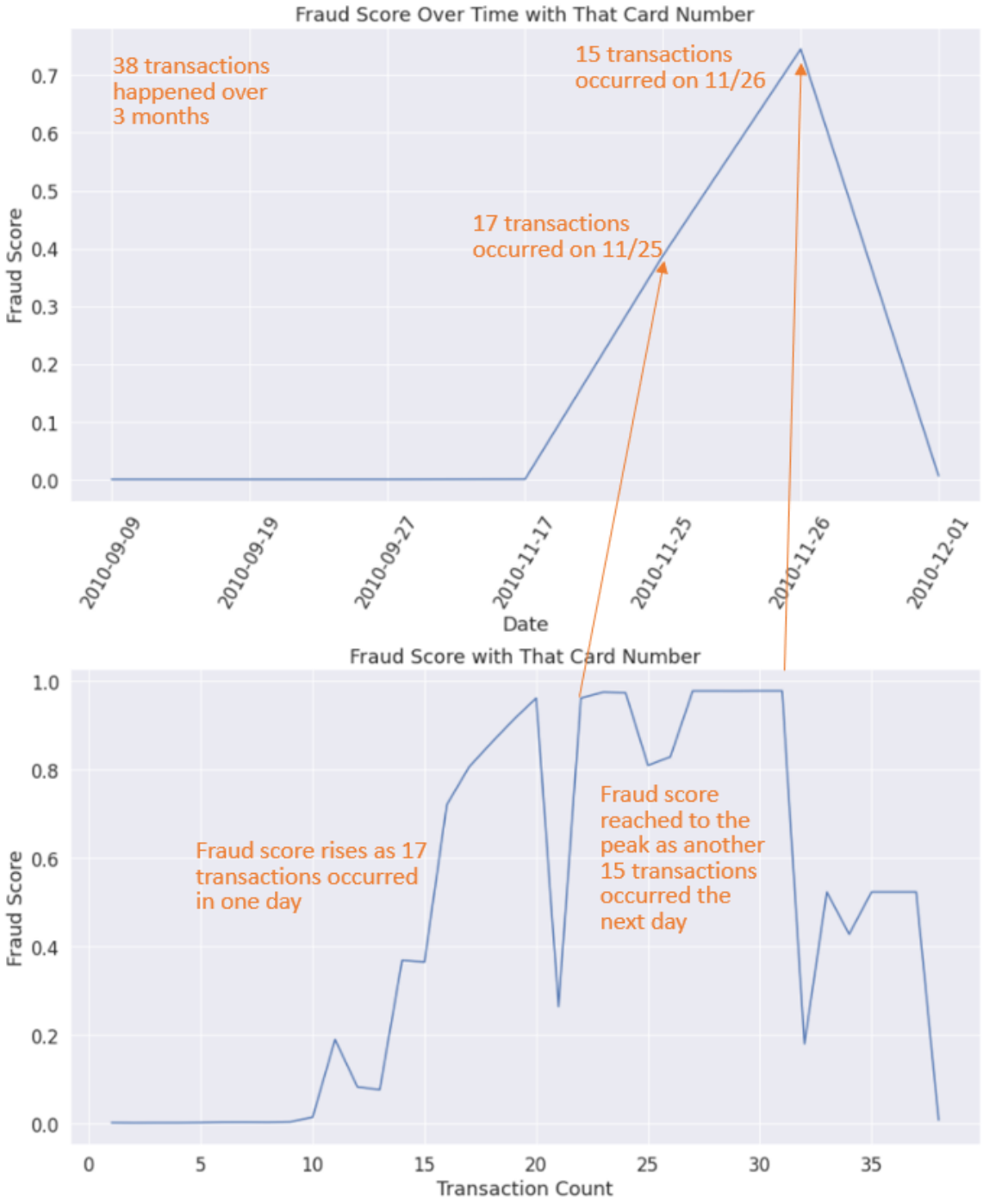
5.3 Fraud Scores Example

Finally, we included two fraud score examples to show how the fraud score evolved when the number of transactions increased over time. We chose one card number (#5142235211) and one merchant number (#6005030600003) from the OOT data, and then observed the fraud activities from a plot that shows the fraud score over time and another plot that shows the fraud score by the cumulative number of transactions.

Card Number: 5142235211

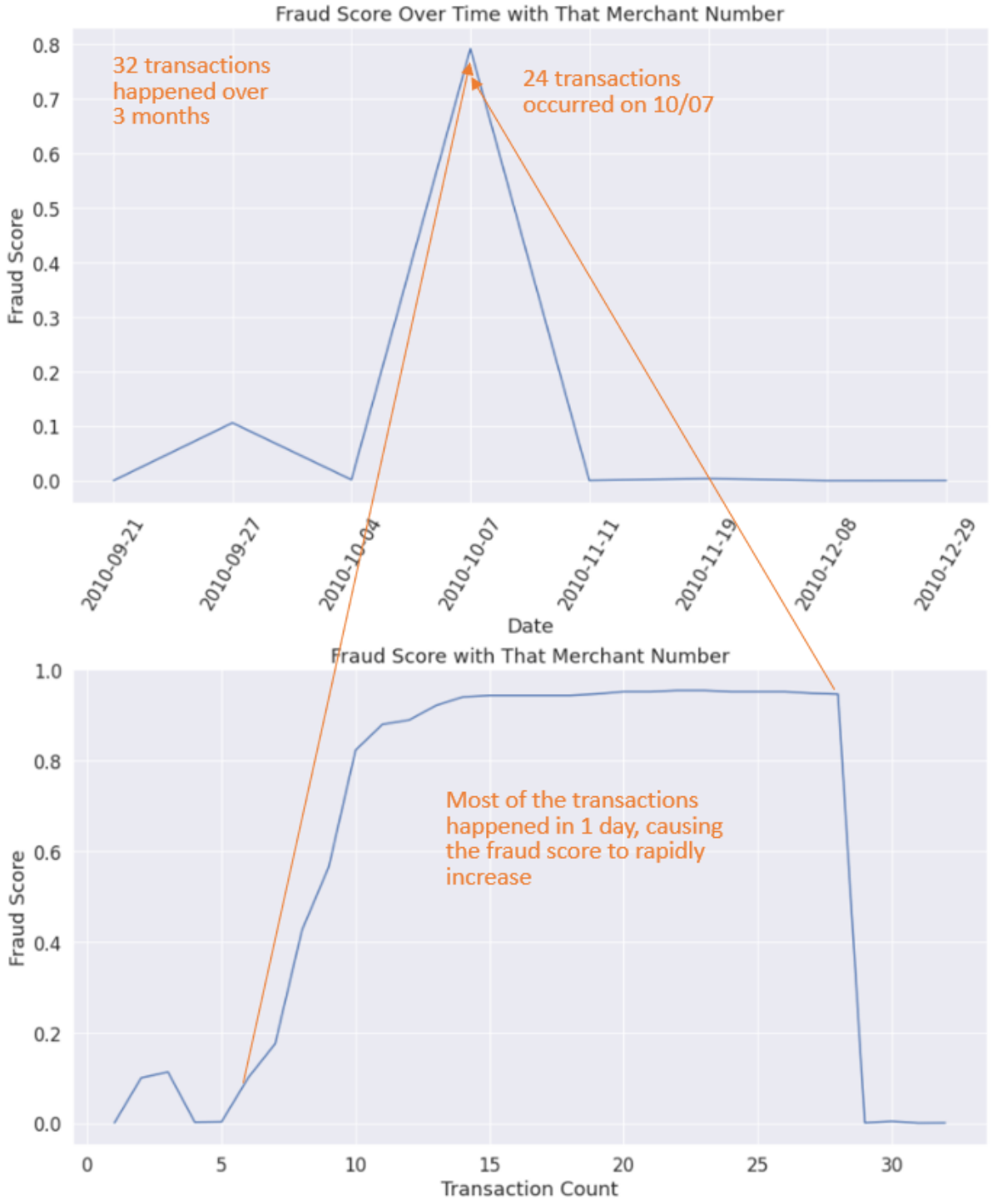
This card number had a total of 38 transactions in about 3 months. The fraud score remained low in September and October but then started to rise as 17 transactions occurred on 11/25. Another 15 transactions happened on the next day, and the fraud score reached the peak above 0.7 on 11/26.

Please see the plot on the next page.


Fraud Score Over Time with That Card Number
38 transactions
happened over
3 months
15 transactions
occurred on 11/26
17 transactions
occurred on 11/25
Fraud Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2010-09-09
2010-09-19
2010-09-27
2010-11-17
2010-11-25
2010-11-26
2010-12-01
Date
Fraud Score with That Card Number
Fraud score rises as 17
transactions occurred
in one day
Fraud score
reached to the
peak as another
15 transactions
occurred the
next day
Fraud Score
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
35
Transaction Count

Merchant Number: 6005030600003

This merchant number had a total of 32 transactions in about 3 months. In September the fraud score remained low with only 5 transactions occurring. However, 24 transactions happened on 10/07, causing the fraud score to rapidly increase, and it exceeded 0.8 when there are more than 10 transactions. Afterwards, the fraud score dropped as the activity became normal as usual.



Conclusion

In this project, a supervised machine learning model was developed to detect credit card fraud. In the first step of the process, EDA (Exploratory Data Analysis) was performed on the real-world dataset that contained 96,753 transactions. Then, outliers were excluded and missing values were imputed based on the findings from EDA. In the following step, over 550 candidate variables were created, which were then reduced to top 30 variables using filter and wrapper methods and based on KS (Kolmogorov-Smirnov) and FDR (Fraud Detection Rate) at 3%. The final variables were then used to train multiple models that were built using multiple classification algorithms including logistic regression, Neural Network, Random Forest and Gradient Boosted Trees. Each model was trained with a different combination of parameters. Among all the models tested, Gradient Boosted Tree using XGBoost (*learning_rate=0.1, max_depth=6, min_child_weight=5*) achieved FDR scores of 93.46%, 88.65% and 61.97% on train, test and OOT data respectively and was selected as our final, best model.

The model was developed through a thorough process, but there are areas that can be improved in a further development:

1. A larger dataset may be used to build a more reliable, generalized model. The dataset used in the project contained less than 100,000 records, which were then divided into three subsets: train, test and OOT. The size of the train set may have been relatively small compared to that used for fraud detection models in the real world.
2. More candidate variables may be created based on the domain expertise in the credit card industry.
3. Additional algorithms such as KNN (K-nearest neighbor) and LSTM (Long short-term memory) may be used to determine the model that best fits the data.
4. Different methods such downsampling, weighting and SMOTE may be tried to account for the imbalance of good (non-fraud) and bad (fraud).
5. More combinations of hyperparameters may be tried to find the optimal set for each algorithm used.

Appendix

A. Data Quality Report

Field 1:
Name: Recnum
Description: Unique ID for each record in the dataset

Field 2:
Name: Cardnum
Description: Card number

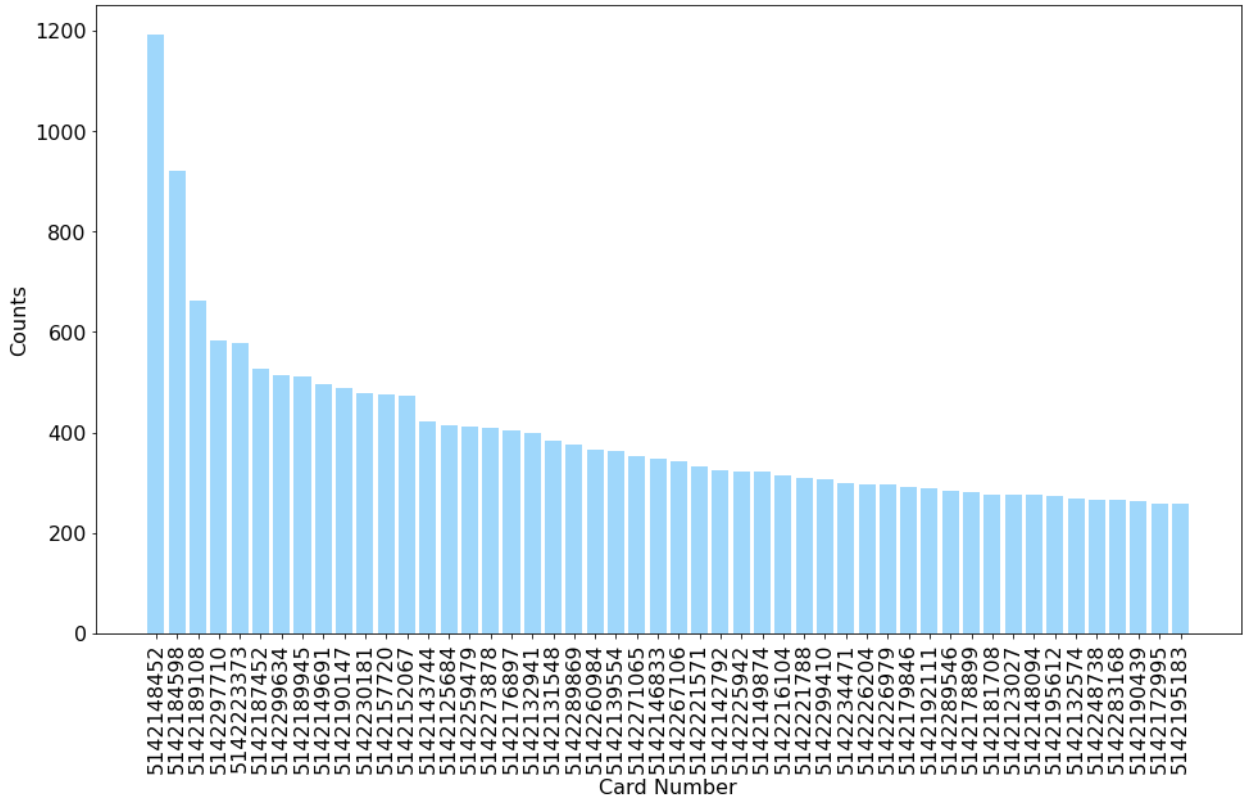


Figure A.1 Frequency distribution of card number (top 50)

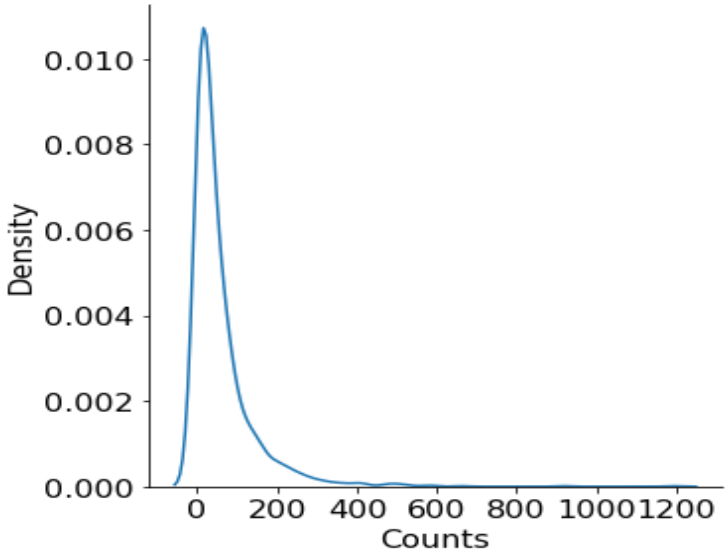


Figure A.2 Density plot of transaction frequency by card number

Field 3:

Name: Date

Description: Date of the transaction

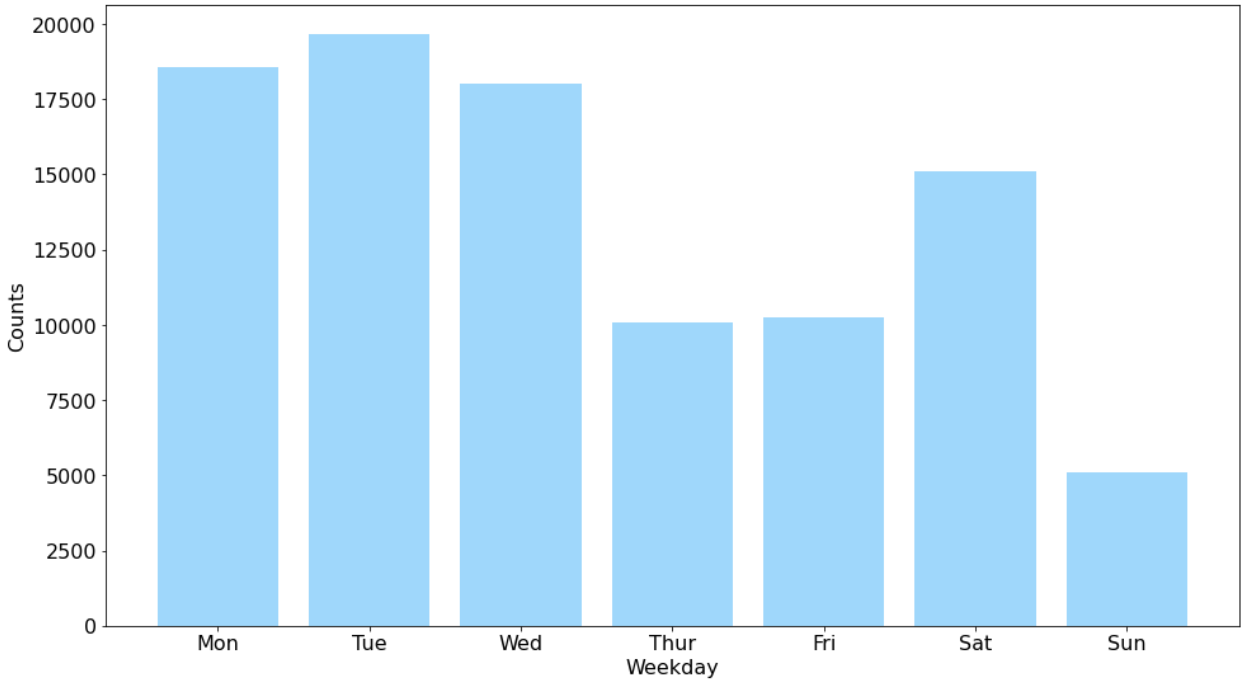


Figure A3 - 1 Frequency distribution of transactions on weekdays

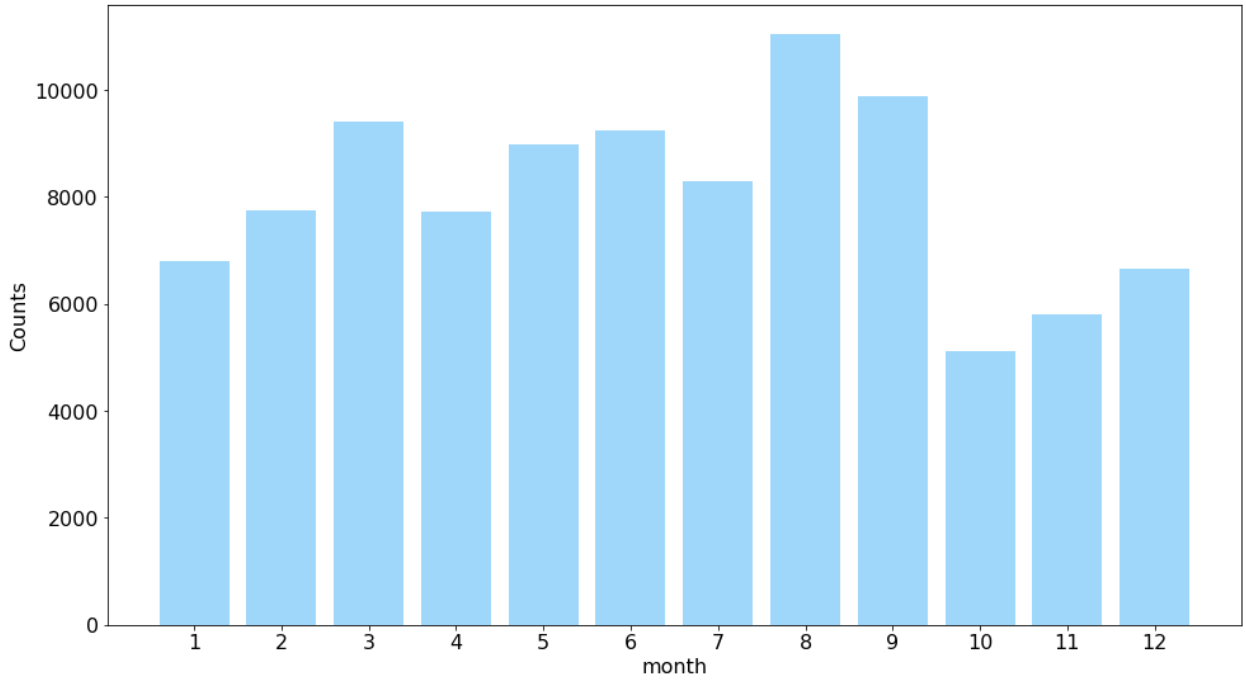


Figure A3 - 2 Frequency distribution of transactions in each month

Field 4:

Name: Merchnum

Description: Identification number of merchandise

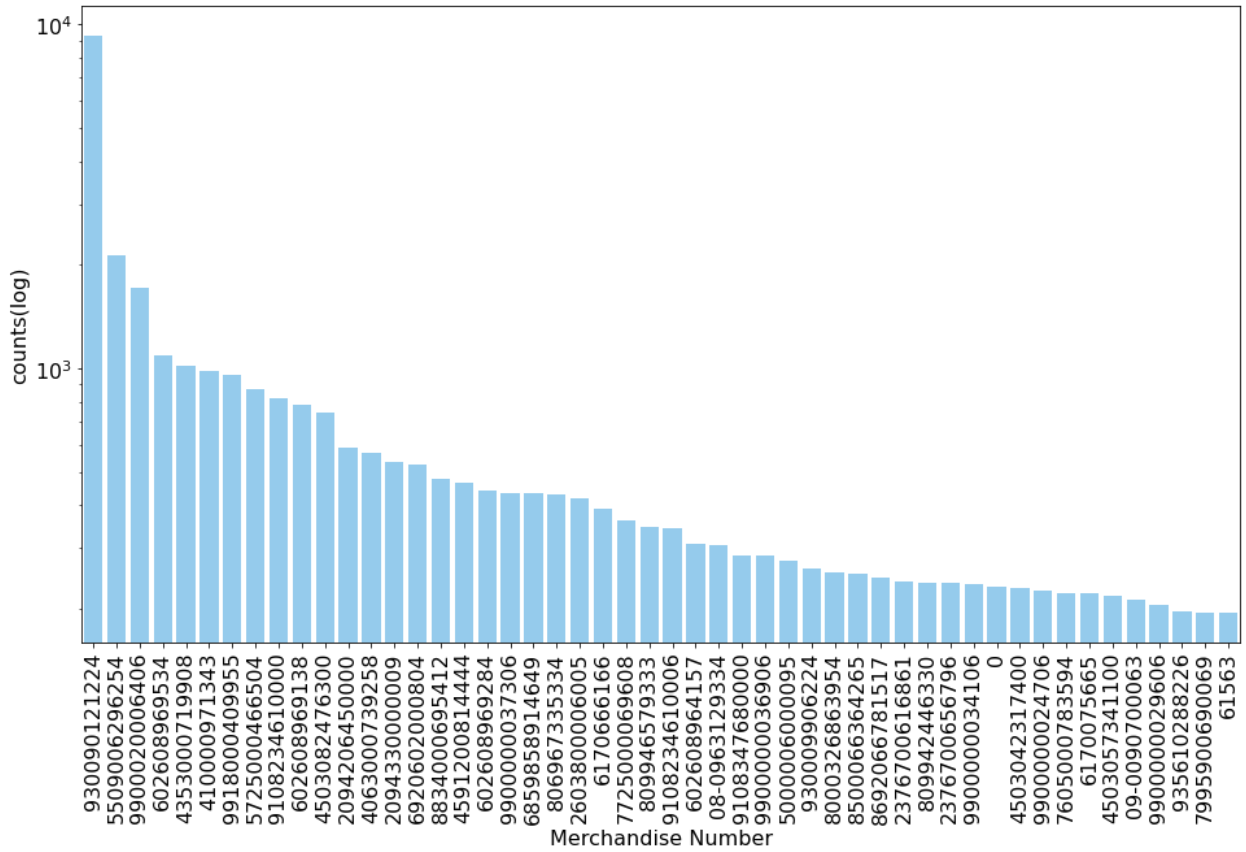


Figure A.4 Frequency distribution of merchandise number (top 50)

Field 5:

Name: Merch description

Description: Brief description of merchandise

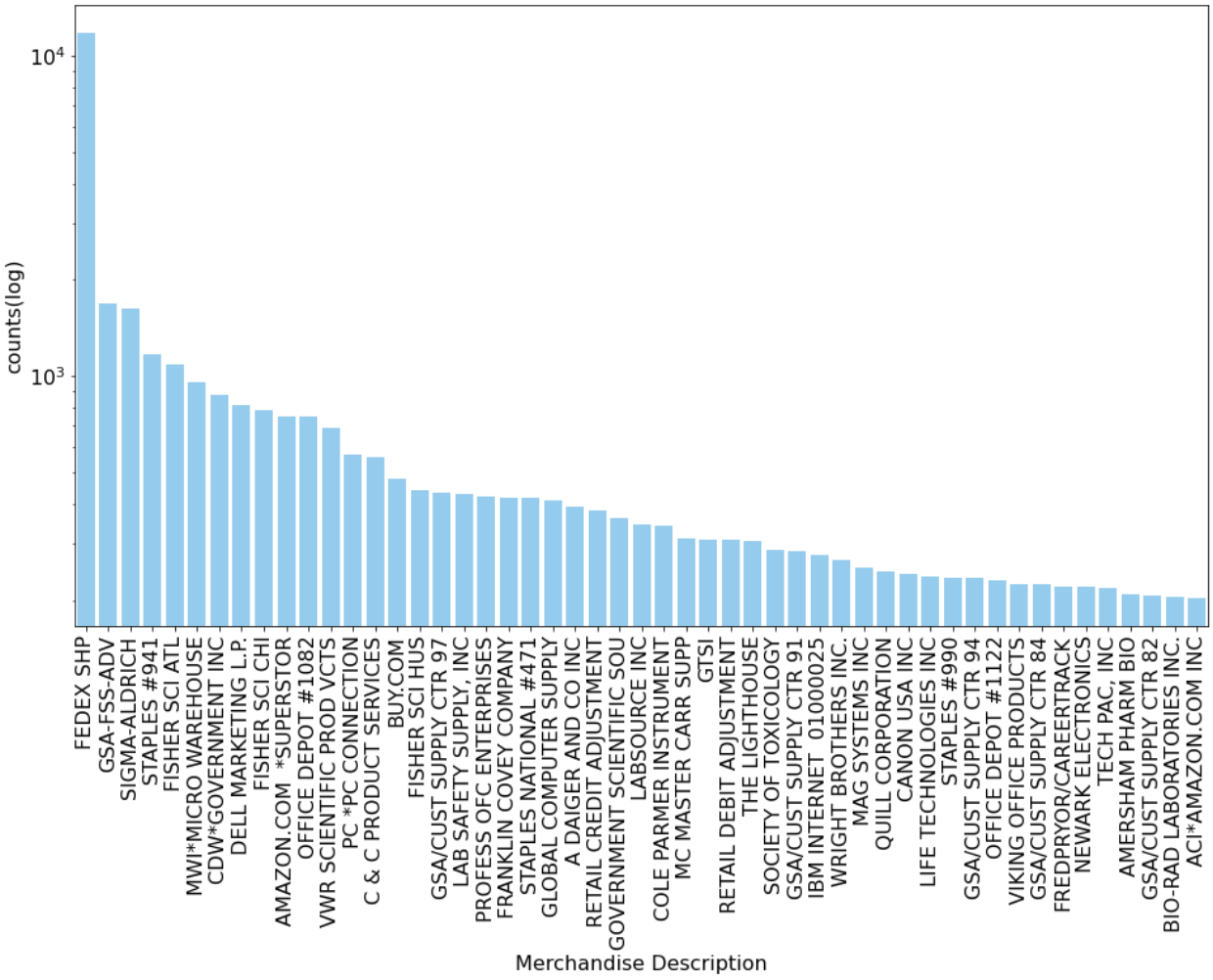


Figure A.5 Frequency distribution of merchants with more than 200 transactions

Field 6:

Name: Merch state

Description: The state where merchandise happened.

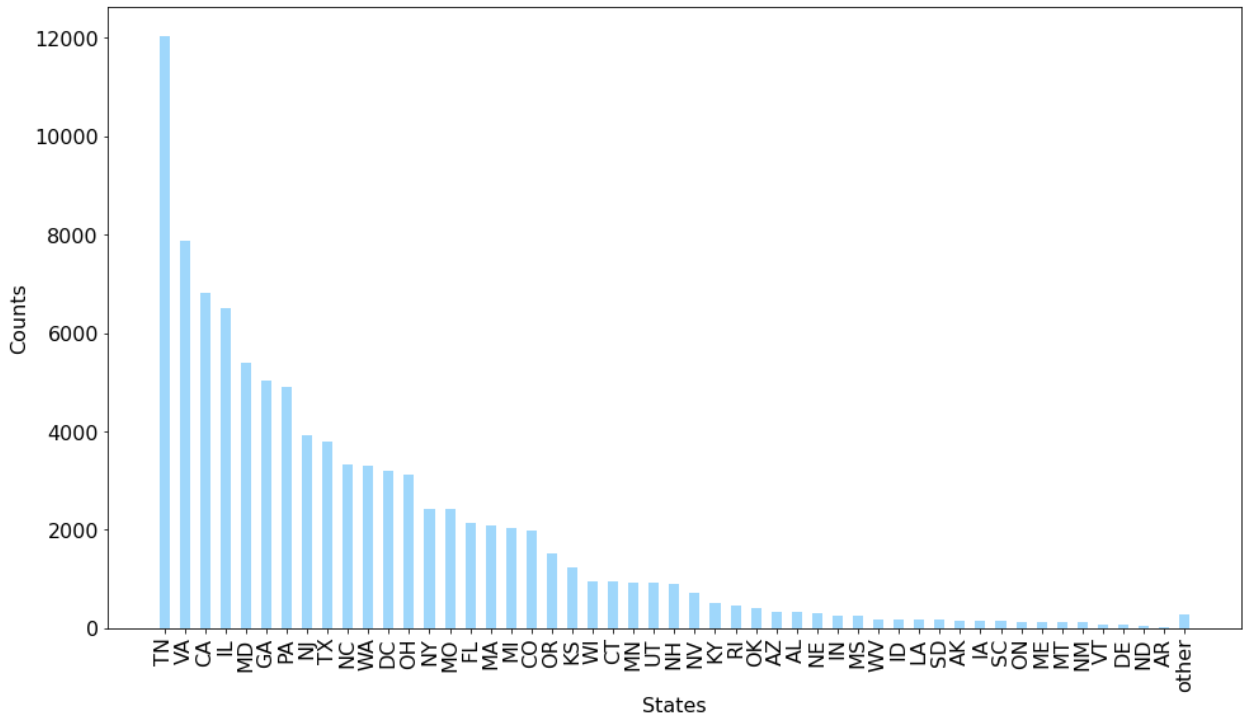


Figure A-6 Frequency distribution of merchants in different states

Field 7:

Name: Merch zip

Description: Zip code of the place where merchandise happened.

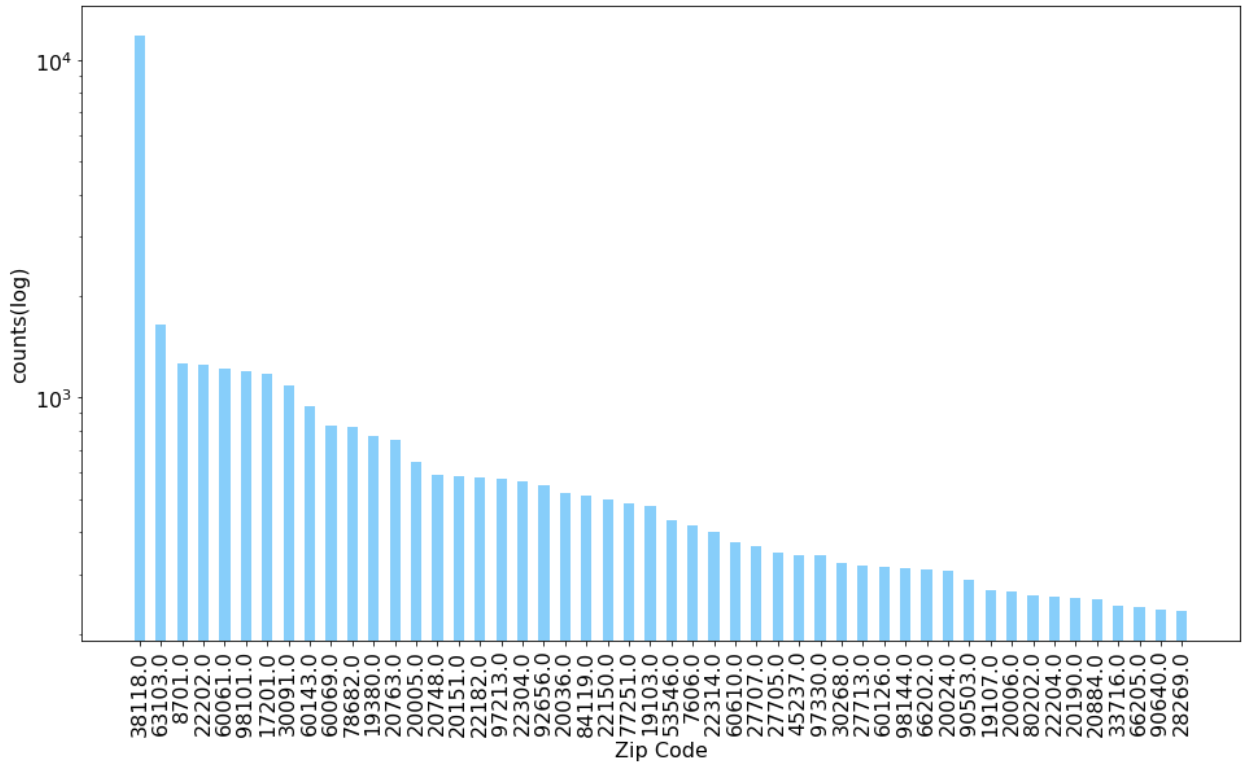


Figure A.7 Frequency distribution by zip code (top 50)

Field 8:

Name: Transaction type

Description: The type of the transaction.

Type: Categorical

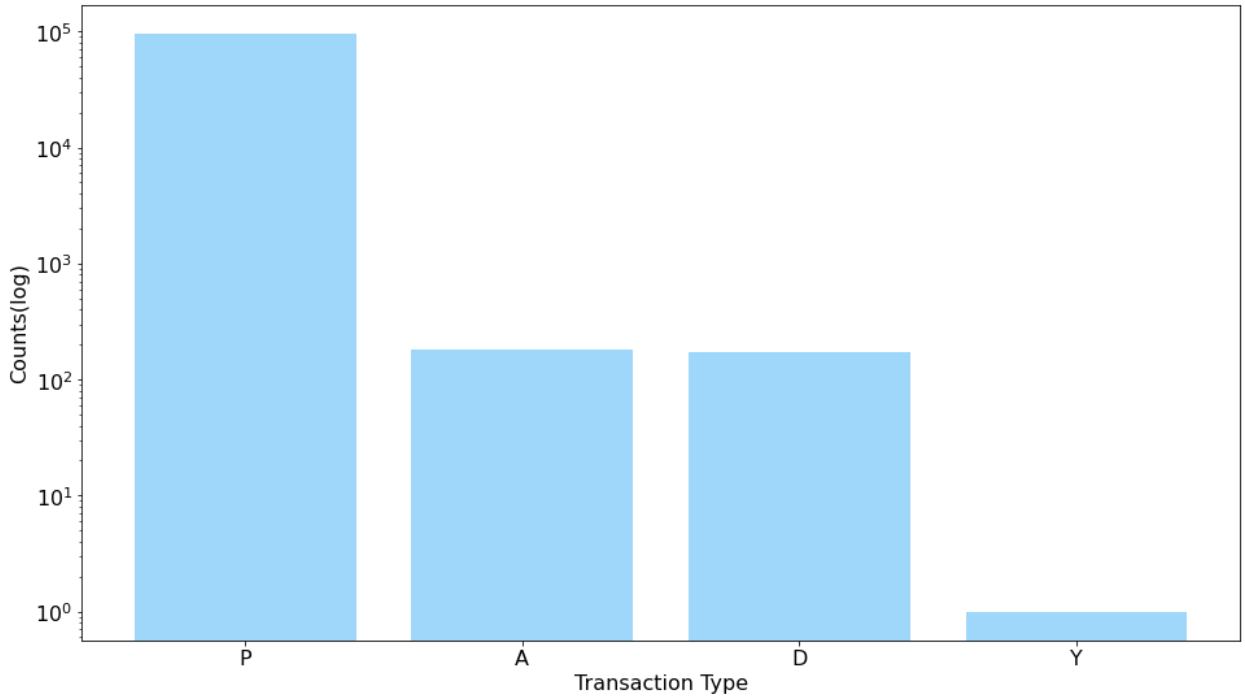


Figure A.8 Frequency distribution of merchants types

Field 9:

Name: Amount

Description: The monetary amount of the transaction in US dollar.

Type: Numerical

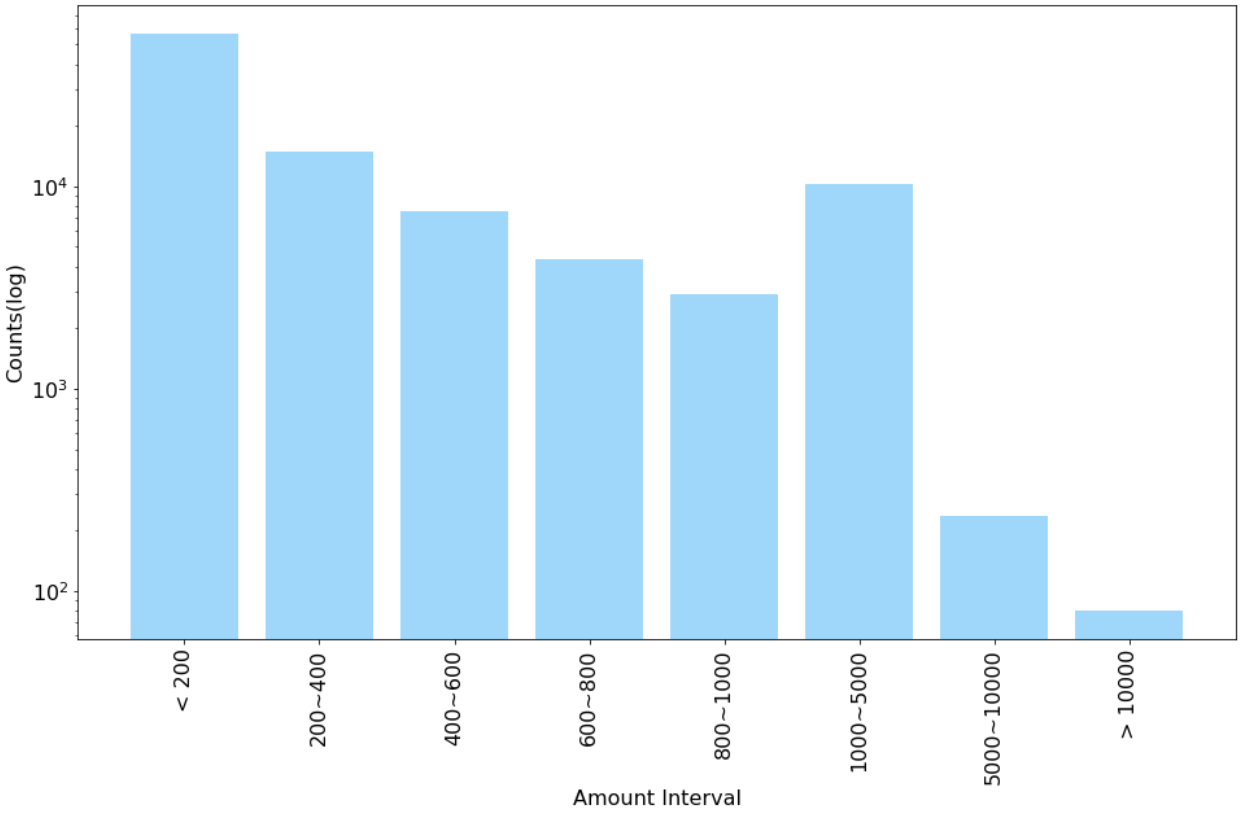


Figure A.9 Frequency distribution of transactions amount

Field 10:

Name: Fraud

Description: Whether a transaction is fraudulent. 1 means fraud, and 0 means a normal transaction.

Type: Categorical

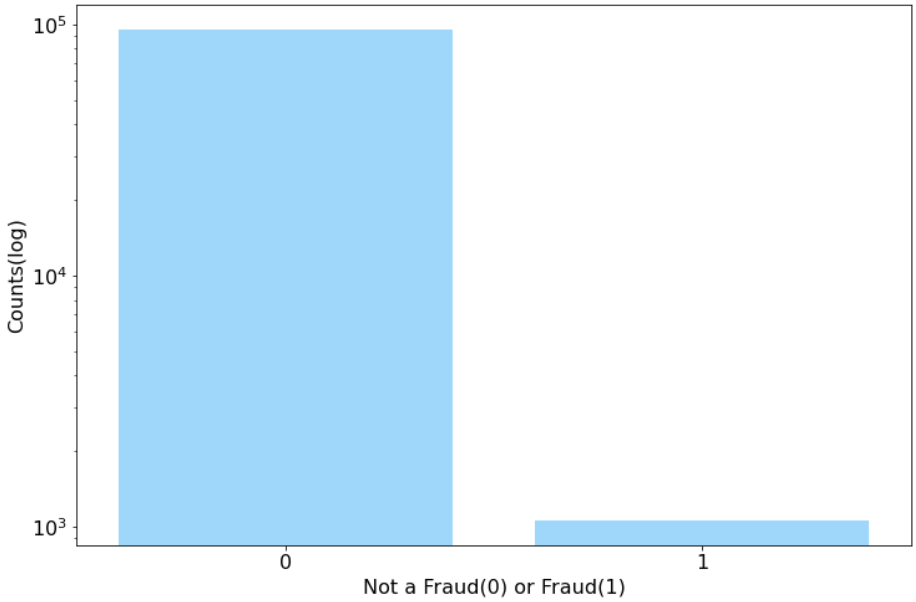


Figure 1.3.6 Frequency distribution of fraud and non-fraud transaction

B. Top 80 Variables (After Filtering)

Variable	KS	FDR @3%	KS Rank	FDR Rank	Average Rank
card_merch_total_7	0.667	0.622	563	561	562
card_desc_total_14	0.655	0.632	557	564	560.5
card_merch_total_14	0.658	0.627	558	562.5	560.25
card_zip_total_7	0.664	0.617	562	558	560
card_desc_total_7	0.649	0.627	555	562.5	558.75
card_state_total_3	0.658	0.617	559	558	558.5
card_merch_total_3	0.659	0.604	560	555	557.5
card_state_total_7	0.661	0.591	561	554	557.5
card_zip_total_14	0.649	0.617	554	558	556
card_zip_total_3	0.653	0.612	556	556	556
card_state_total_14	0.670	0.532	564	545.5	554.75
card_desc_total_3	0.634	0.619	546	560	553
card_desc_total_30	0.648	0.574	552	552	552
card_state_total_1	0.639	0.577	549	553	551
card_zip_total_30	0.648	0.565	551	549	550
card_merch_total_30	0.643	0.565	550	549	549.5
card_merch_total_1	0.635	0.565	547	549	548
card_zip_total_1	0.631	0.560	543	547	545
card_desc_total_1	0.609	0.568	531	551	541
card_state_max_7	0.628	0.502	541	535	538
card_desc_max_14	0.637	0.482	548	526.5	537.25
card_desc_max_30	0.633	0.481	545	524.5	534.75
card_zip_max_30	0.622	0.491	536	531.5	533.75
card_merch_max_14	0.630	0.481	542	524.5	533.25
card_state_max_14	0.617	0.495	533	533	533
card_state_total_30	0.648	0.462	553	513	533
card_state_total_0	0.583	0.531	522	544	533
card_zip_max_14	0.626	0.482	539	526.5	532.75
card_merch_total_0	0.581	0.525	521	542.5	531.75
card_merch_max_30	0.627	0.479	540	523	531.5
card_desc_max_7	0.632	0.469	544	514.5	529.25
card_state_max_3	0.621	0.478	535	522	528.5
Cardnum_total_3	0.576	0.518	517	539	528
card_state_max_30	0.589	0.489	525	530	527.5
card_merch_max_7	0.624	0.471	538	516	527
card_desc_total_0	0.572	0.525	510	542.5	526.25
card_zip_total_0	0.575	0.517	514	538	526
Cardnum_total_7	0.580	0.491	520	531.5	525.75
card_zip_max_7	0.623	0.469	537	514.5	525.75

card_desc_max_3	0.619	0.456	534	509	521.5
card_merch_max_3	0.613	0.456	532	509	520.5
card_zip_max_3	0.609	0.456	530	509	519.5
merch_desc_total_1	0.576	0.476	518	519.5	518.75
merch_state_total_1	0.575	0.473	513	517	515
card_state_max_1	0.597	0.440	529	498	513.5
card_desc_max_1	0.596	0.432	528	492.5	510.25
card_merch_max_1	0.589	0.432	526	492.5	509.25
merch_desc_max_3	0.569	0.456	507	509	508
merch_desc_total_0	0.548	0.532	470	545.5	507.75
merch_desc_total_3	0.594	0.412	527	488	507.5
card_zip_max_1	0.586	0.430	523	491	507
merch_desc_max_0	0.576	0.439	516	497	506.5
merch_desc_max_1	0.571	0.443	509	502.5	505.75
merch_state_total_3	0.586	0.409	524	487	505.5
merch_state_total_0	0.547	0.522	467	540.5	503.75
merch_state_max_0	0.573	0.437	512	495	503.5
merch_zip_total_1	0.564	0.455	497	505	501
merch_zip_total_0	0.542	0.508	460	536.5	498.25
Merchnum_max_0	0.565	0.437	499	495	497
merch_state_max_3	0.557	0.456	485	509	497
merch_state_max_1	0.563	0.442	492	500	496
Cardnum_total_1	0.542	0.496	458	534	496
card_desc_max_0	0.568	0.399	505	483	494
card_state_max_0	0.566	0.403	502	485	493.5
merch_zip_max_0	0.563	0.437	491	495	493
merch_zip_total_3	0.569	0.390	506	480	493
Cardnum_total_0	0.537	0.508	448	536.5	492.25
merch_zip_max_3	0.550	0.456	475	509	492
merch_zip_max_1	0.555	0.442	484	500	492
card_merch_max_0	0.564	0.399	493	483	488
Merchnum_total_0	0.536	0.488	446	529	487.5
Merchnum_max_1	0.550	0.442	473	500	486.5
merch_state_total_7	0.566	0.344	503	469	486
Merchnum_total_3	0.566	0.348	501	470	485.5
card_zip_max_0	0.561	0.399	487	483	485
Cardnum_max_7	0.527	0.483	439	528	483.5
Merchnum_max_3	0.543	0.449	461	504	482.5
Merchnum_total_1	0.558	0.380	486	478	482
merch_desc_max_7	0.532	0.476	440	519.5	479.75
card_zip_avg_30	0.575	0.272	515	442.5	478.75