DSO 562 Project 3
Credit Card Transaction Fraud

Advisor:
Prof. Stephen Coggeshall

Submitted By:
Chinmayi Bengaluru Prakash

May 05, 2022

CONTENTS

1. Executive Summary

As countries across the world move toward digital currency and cashless societies, credit card fraud is one of the biggest financial burdens for any economy. According to a report by Nilson[1], over 47% of Americans faced credit card fraud in the last 5 years, and the United States tops the fraud loss chart accounting for 36.8% of all fraud losses due to card transactions in 2018. In 2021, payment card fraud cost the global economy ~USD 33 billion and is expected to touch 40 billion by 2027.

This report describes in detail the processes used to evaluate credit card transactions to capture fraud, leveraging supervised machine learning techniques. The raw dataset consists of ~97K rows and 10 columns. The dataset ranges from January to December 2006. Last two months of data (November and December 2006) was set aside as validation data. The remaining 10 months of data was divided into train and test datasets in 70:30 ratio.

The fraud detection methodology involved exploring and visualizing each variable in the dataset followed by cleaning the raw data to handle missing and frivolous entries. As part of feature engineering, 1806 candidate variables were created by extracting features from the raw data. As part of feature selection, filtering was performed using KS (Kolmogorov-Smirnov) score as the evaluation metric. The top 80 variables were selected at the end of this step. Post which, boosted trees algorithm (LGBM Classifier) was used as the wrapping technique with forward stepwise selection to arrive at the top 20 variables. These variables were then used to feed into various machine learning models.

Logistic regression was considered as the baseline model. Following this, other modeling techniques including decision trees, random forest, gradient boosting, and neural networks were implemented. Finally, random forest model was chosen because it had the best performance on the out-of-time dataset with a fraud detection rate (FDR) of ~58.1% at 3% FDR and ~64.25% at 5% FDR. This means the model can detect more than half of the fraud applications given only the top 5% of data sorted by fraud algorithm score, yielding savings of ~$1.23M annually.

1. https://www.businesswire.com/news/home/20111121005121/en/U.S.-Leads-the-World-in-Credit-Card-Fraud-states-The-Nilson-Report

2. Description of the data

The data set contains information related to credit card purchases made in 2006 from a US government organization. It comprises 10 fields and 96,753 records of credit card transactions. There are 1,059 credit card transactions labeled as *fraud* which account for ~1.09% of all records.

2.1 Numerical Table

Field	Mean	Std	Max	Min	% Populated	% Zero
Date	-	-	2006-12-31	2006-01-01	100.00%	0.00%
Amount	$427.89	$10,006.14	$3,102,045.53	$0.01	100.00%	0.00%

Table 2.1: Numerical Table

2.2 Categorical Table

Field	# Unique Values	Most Common	Most Common % Total	% Populated
Recnum	96,753	All Unique	-	100.00%
Cardnum	1,645	5142148452	1.23%	100.00%
Merchnum	13,092	930090121224	9.62%	96.51%
Merch description	13,126	GSA-FSS-ADV	1.74%	100.00%
Merch state	228	TN	12.44%	100.00%
Merch zip	4,568	38118	12.27%	95.19%
Transtype	4	P	99.63%	100.00%
Fraud	2	0	98.91%	100.00%

Table 2.2: Categorical Table

2.3 Fields of Importance

In this section we have highlighted fields that we find especially important. The full data quality report with field descriptions and descriptive graphs can be found in the appendix.

2.3.1 Date

This field contains the dates for the credit card transactions recorded in the data set. Graph 2.3.1 shows the frequency of transactions by the week of the year. Figure 2.3.2 represents the frequency of transactions based on the month of the year. Finally figure 2.3.3 represents the proportion of

fraudulent (red) and legitimate transactions on a weekly basis. There appears to be a seasonal trend for fraudulent transactions which may need further investigation.

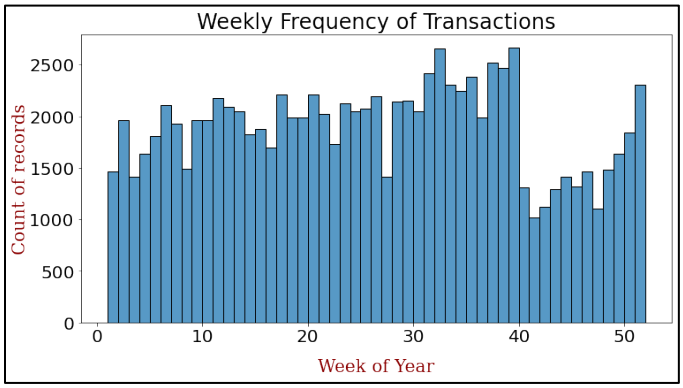


Figure 2.3.1.a: Date – Weekly Frequency

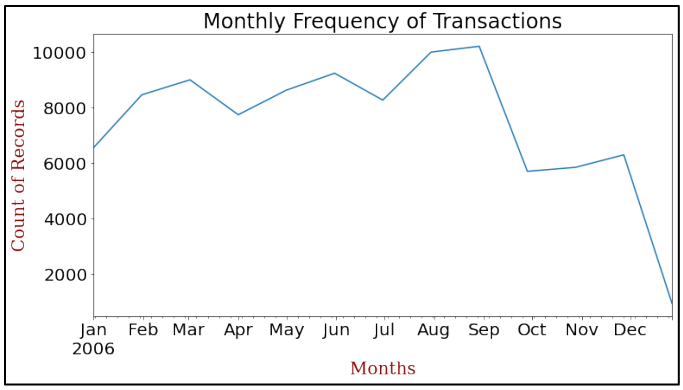


Figure 2.3.1.b: Date - Monthly Frequency

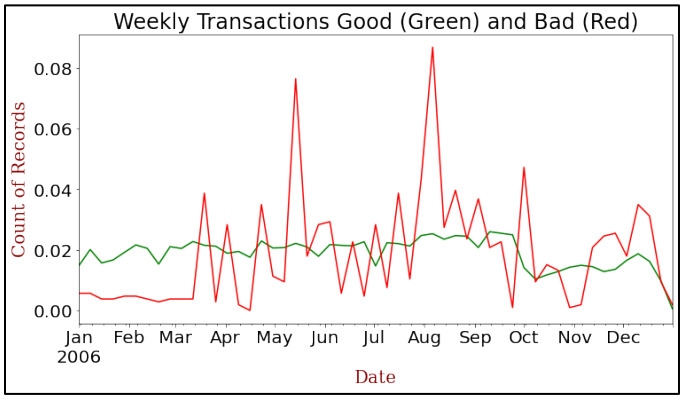


Figure 2.3.1.c: Date – Weekly Transactions Good and Bad

2.3.2 Merchnum

This field contains the merchant number associated with the credit card transactions recorded. The most common merchant number in the data set is ‘930090121224’ which accounts for approximately 9.62% of the total number of records.

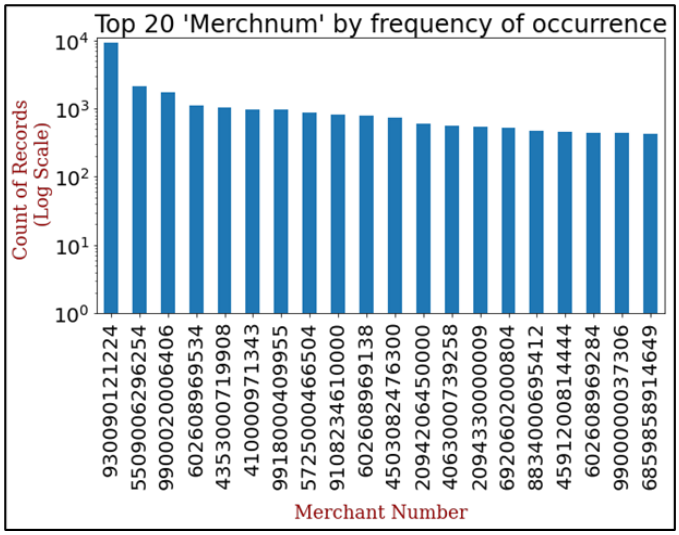


Figure 2.3.2.: Merchnum

2.3.3 Merch zip

This field contains the merchant zip associated with the credit card transactions recorded. The most common merchant zip in the data set is ‘38118’ which accounts for approximately 12.27% of the total number of records.

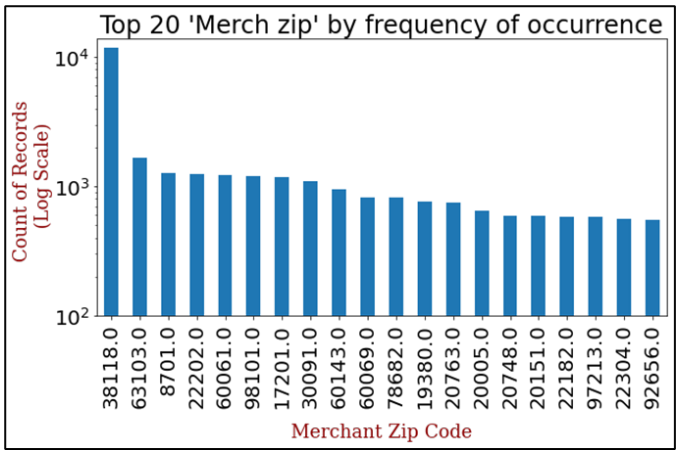


Figure 2.3.3: Merch zip

2.3.4 Amount

This field contains the dollar amount of the credit card transactions recorded. The most common dollar amount in the data set is '$3.62'. This field contained an outlier which was over $ 3M. This outlier needs to be removed before further analysis.

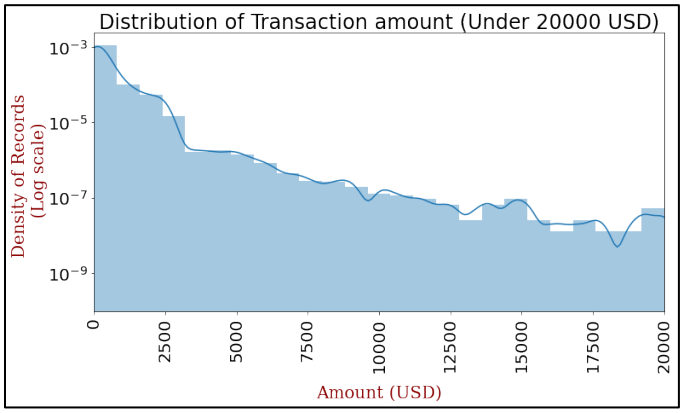


Figure 2.3.4: Amount

3. Data Cleaning

Prior to fitting any machine learning model to a dataset, it is necessary to clean the data by handling missing/null values, frivolous values, corrupted data, irrelevant and inaccurate data. These data points need to be replaced or modified or deleted before proceeding ahead.

3.1 Filtering and Outlier Removal

- For the column Transaction type (*Transtype*), we selected the transaction type as 'P', which stands for 'Purchase' transaction. This transaction type consists of ~99.63% of data.
- We also removed the outlier value for *Amount* column, with amounts more than 1 million.

3.2 Missing Value Imputations

Amongst all the columns, only three columns have missing values, namely *Merch State*, *Merch Zip* and *Merchnum*.

3.2.1 Merch State

Merch State had 1.24% values missing. The null values were mapped using *zip code*, *merchnum* and *merch description* columns, and remaining values were marked as 'UNKNOWN'.

3.2.2 Merch Zip

Merch Zip had 4.72% of its values missing. Here the null values were mapped using *merchnum* and *merch description*, and remaining values were marked as 'UNKNOWN'.

3.2.3 Merchnum

Lastly, *Merchnum* had 3.48% values missing. The null values were mapped using *merch description* column, and like *merch state* and *merch zip*, the remaining values were marked as 'UNKNOWN'.

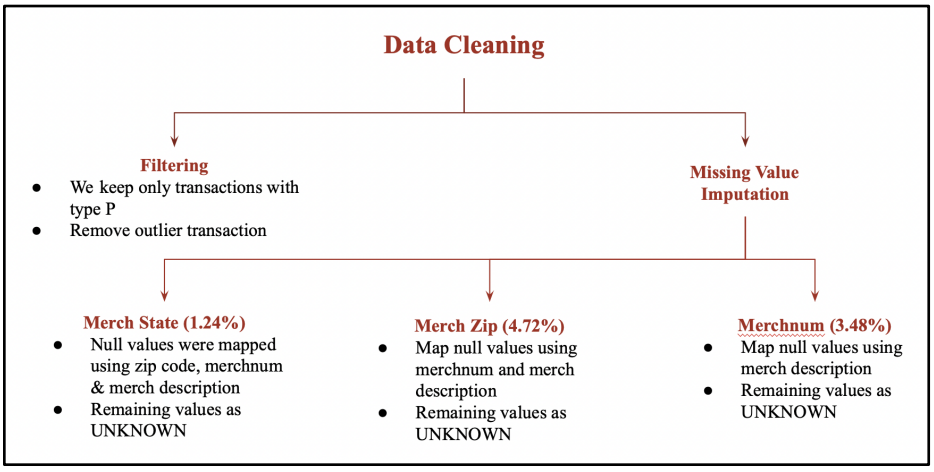


Figure 3.1: Data Cleaning Process

4. Candidate Variables

4.1 Feature Engineering

Feature engineering is the process of leveraging domain knowledge to extract new features from the raw data. It is the second most important step in the entire life cycle of model building after designing a solution approach.

Feature engineering consists of creation, transformation, extraction, and selection of features, that are most conducive to creating an accurate ML algorithm.

A feature is any measurable input that can be used in a predictive model. New features that are not present in the raw data are created using this machine learning technique. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

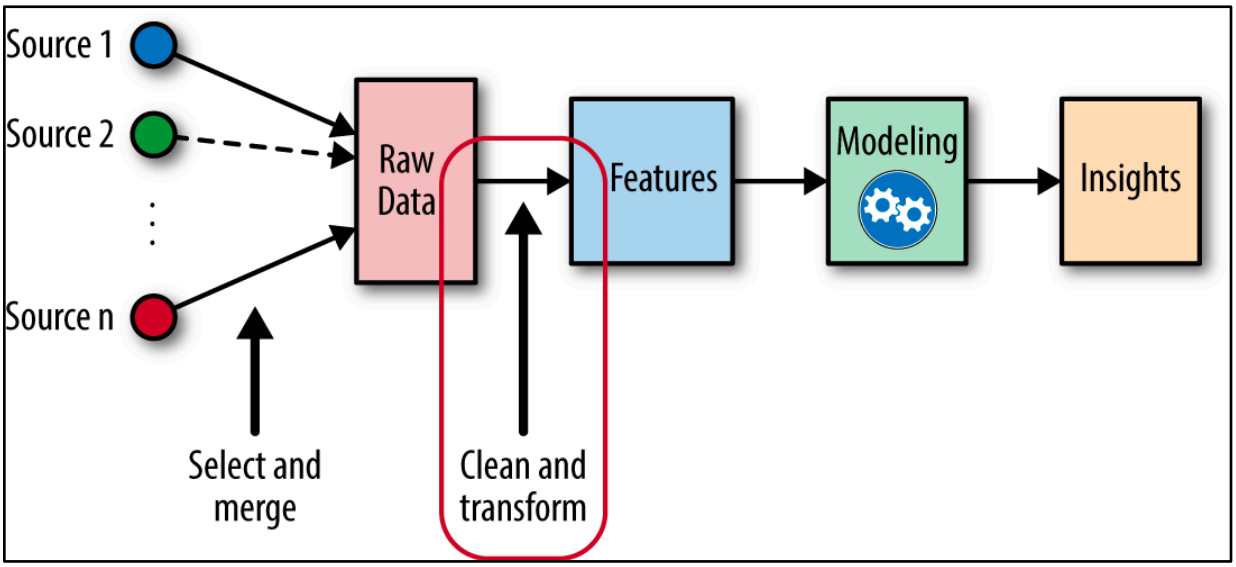


Fig 4.1: Feature Engineering in Machine Learning

In this project, we have executed the following steps as part of feature extraction:

- Target encoding
- Statistical smoothing
- Creation of features/variables
  - *Risk table* variables
  - *Benford's law* variables
  - *Days-since* variables
  - *Frequency* and *amount variables*
  - *Velocity related* variables
  - *Unique Entity* variables
  - *Acceleration* variables
  - *Variability* variables
  - *Interesting* variables

4.2 Target Encoding

Encoding categorical variables is a very important step in feature engineering. Generally, encoding of categorical variables is a procedure of replacing categorical variables with one or more numeric variables, so that the resulting data set may be used in the statistical and machine learning algorithms that expect numeric variables. There are many encoding techniques including dummy variable encoding, one-hot encoding, ordinal encoding, target encoding and Bayesian target encoding.

In target encoding, we compute the mean of the target variable for each possible category and encode that category with the target mean. Hence, each categorical field becomes a numeric variable. This technique works for both binary classification and regression. For multiclass classification a similar technique is applied, where we encode the categorical variable with (m-1) new variables, where m is the number of classes.

A good practice of target encoding is to build a table with values for each category. For instance, consider a categorical field, say xx, that has possible values aa, bb, cc, dd

- Calculate v_{aa} = <y>|xx=aa, v_{bb} = <y>|xx=bb, v_{cc} = <y>|xx=cc, v_{dd} = <y>|xx=dd
- Then encode xx as:

$$xx \Rightarrow xx_{new} = \begin{cases} v_{aa} & \text{when } xx=aa \\ v_{bb} & \text{when } xx=bb \\ v_{cc} & \text{when } xx=cc \\ v_{dd} & \text{when } xx=dd \end{cases}$$

Fig 4.2: Target encoding

In this project, we carry out target encoding (i.e., Risk Tables) as the cardinality of the variable (number of categories) is more than 2. This method ensures no dimensionality expansion, direct encoding of the prediction variable and the easiest for the model to compute the y = f(x) relationship. However, drawbacks of target encoding are losing interaction information and overfitting.

Overfitting in target encoding can be avoided ensuring the following:

- Training data to be used when calculating the table values
- Statistically sufficient sample is present in each category, say, at least several dozen records
- If there are not enough records in a category, then use expert knowledge to group categories, for example, combine A and B together into a single category
- Use a smoothing formula (described in section 1.3)
- If overfitting persists despite ensuring the above, then systematically remove any of these table variables and observe the result. This will identify any table variable that is overfitting.

In this project, we perform target encoding for 2 variables:

a. Day of week to create the *dow* variable

b. Merchant State to create the *merchstate* variable

Also, it is important to note that the train-test data is separated from the validation data (data with date above '2006-11-01') prior to target encoding. In summary,

- Train-Test dataset: Jan 1 – Oct 31, 2006
- Validation dataset: Nov 1 – Dec 31, 2006

4.3 Statistical Smoothing

A smoothing formula is used to smoothly transition a value from one number to another. It is used in target encoding to overcome the problem of overfitting. We have used the following logistic formula as our smoothing formula in our project.

$$\text{Value} = Y_{\text{low}} + \frac{Y_{\text{high}} - Y_{\text{low}}}{1 + e^{-(n - n_{mid})/c}}$$

where: Y_{low} is one number
Y_{high} is the other number
n_{mid} is the value of n where the smoothed value is halfway between Y_{low} and Y_{high}
c is a measure of how quickly it transitions
n is the smoothing counter (integer/continuous)

Here, n_{mid} and c are the transition parameters

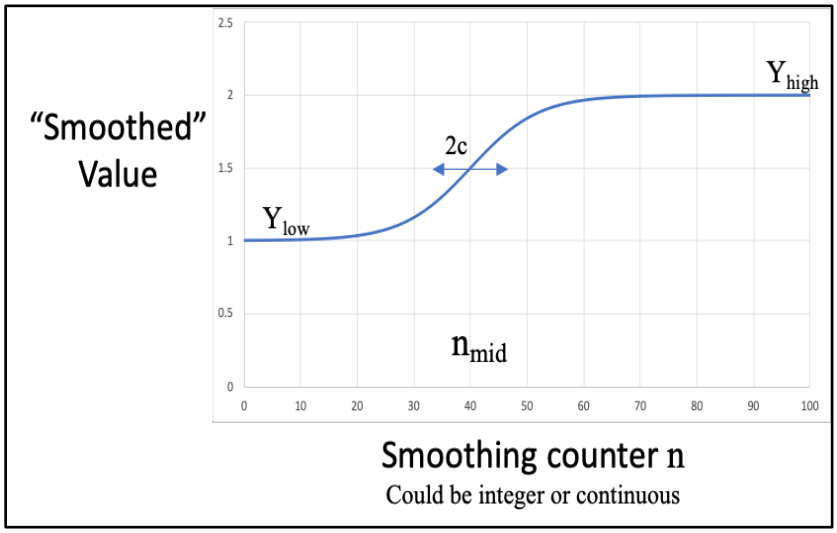


Fig 4.3: Statistical smoothing

In this project, we have taken $n_{mid} = 15$ and $c = 3$

- *dow* variable

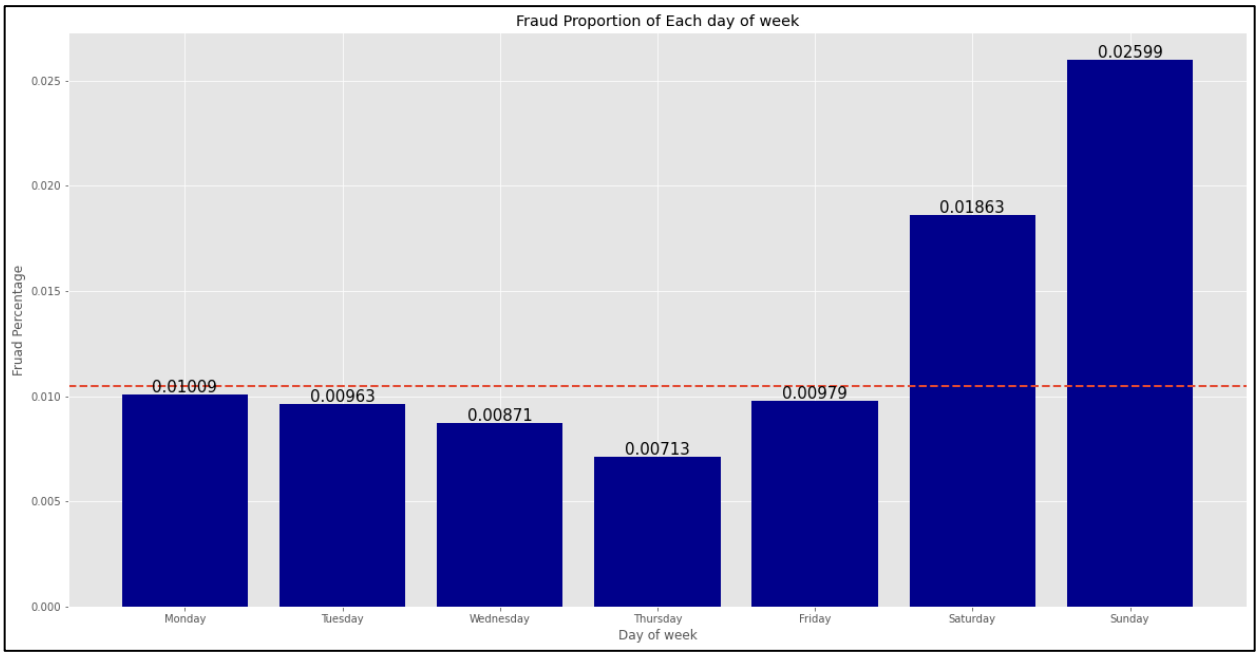


Fig 4.4: Statistical smoothing for dow

- *merch state* variable

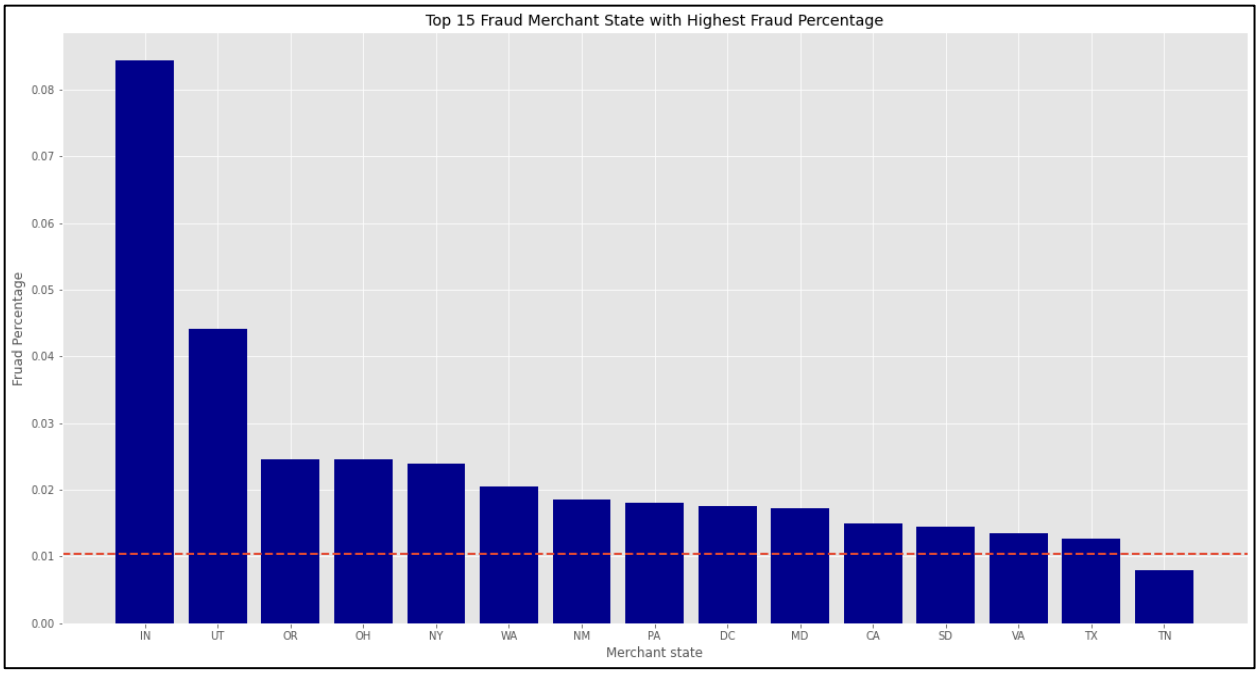


Fig 4.5: Statistical smoothing for merch state

4.4 Rationale behind creation of variables

An individual fraudster might have access to a lost/stolen card, the card information might have been stolen through skimmers at stores or merchants. These can be identified by evaluating common MO (Modus Operandi) the fraudster might use.

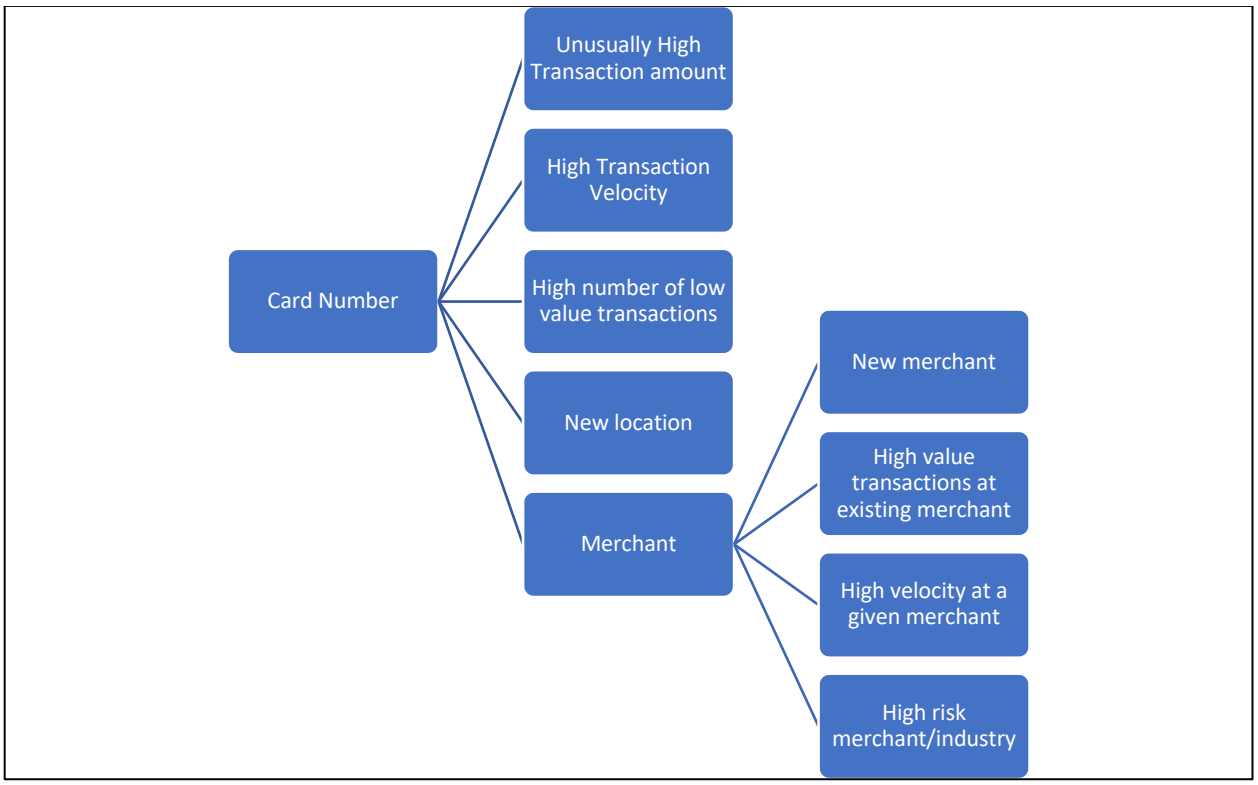


Fig 4.6: Rationale for variable creation

Also, merchant-customer collusion is a major concern for credit card companies providing buyer or seller protection for the transactions.

Thus, creating combination variables, creating stemmed variables, velocity variables, amount variables, and these in combination with other variables looking back over various time windows will help in identifying various fraud MO. Therefore, we have decided to create various types of candidate variables highlighted in section 4.5.

Later, we select the top 15 expert variables by feature selection techniques to feed them into the model.

4.5 Creation of combination groups:

We created new features like *card_merch* by combining *cardnum* and *merchnum*, *merch_zip3* by combining *merchnum* and the first 3 digits of the *zip*, and so on.

Here is a list of initial combination groups that we created out of combining entities which are further used to create variables such as velocity variables, acceleration variables, etc.,

Combination groups		
Card_merch	Zip3	Merchnum_zip
Card_zip	Card_zip3	Merchnum_zip3
Card_state	Merchnum_state	Card_merch_zip3

Table 4.1: Rationale for variable creation

4.6 Creation of variables:

For each entity or combination group, we create a set of following candidate/expert variables.

- *Risk table* variables
- *Benford's law* variables
- *Days-since* variables
- *Frequency* and *amount variables*
- *Velocity related* variables
- *Unique Entity* variables
- *Acceleration* variables
- *Variability* variables
- *Interesting* variables

4.6.1 Risk table variables

The idea behind the creation of these variables is to convert categorical variables into numeric to feed them into the model

4.6.2 Benford's law variables

Benford's Law is the nonintuitive fact that the first digit of many measurements is not uniformly distributed, and the digit '1' appears about 30% of the time. The idea behind the creation of these variables is to identify scenarios where this law is not followed.
For this project, we combine the probability of occurrence of first digit '1' or '2' and evaluate if it follows those dictated by Benford's law.

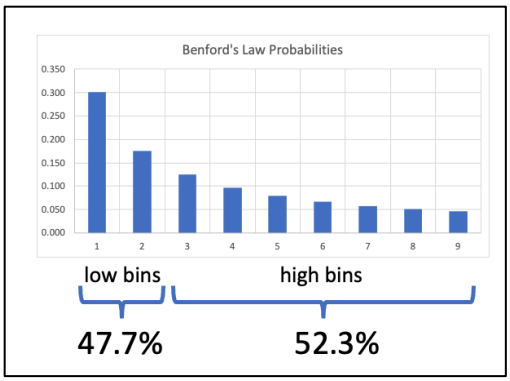


Fig 4.7: Benford's law probabilities

We measure the unusualness factor U* given by the formula:

$$U^* = 1 + \left(\frac{U - 1}{1 + \exp^{-t}} \right)$$

where:

$t = (n - n_{mid}) / c$
$U = \max(R, 1/R)$
$R = 1.096 * (n_{low}/n_{high})$
n_{low} = # first digits starting with 1 or 2
n_{high} = # first digits not starting with 1 or 2

4.6.3 Days-since variables

The idea behind the creation of days-since variables is to check how many days has it been since we last encountered each entity or combination group.

4.6.4 Frequency and amount variables

- **Frequency variables** are created to understand how many times we encountered each entity or combination group over the past 'n' days where n = 0, 1, 3, 7, 14, 30, 45, 60 & 90 days.
- **Amount variables** are created to measure statistical summary metrics such as *average, maximum, median, total* and ratios of these metrics *by* or *at* these entities over the past 'n' days where n = 0,1,3,7,14,30,45,60,90 days.

Time window n = 0 implies same day for the calculation of the said metric.

4.6.5 Velocity related variables

- **Relative velocity** variables are created to determine the ratio of short-term velocity to a longer-term averaged velocity.

$$\frac{\#\text{ applications with that } \textbf{\textit{entity/combination group}} \text{ seen in the recent past}}{\#\text{ applications with that } \textbf{\textit{same entity/combination group}} \text{ seen in the past } \{0, 1, 3, 7, 14, 30\} \text{ days}}$$

- **Velocity change** variables are created to measure the change in number of card transactions in the past 0 or 1 days over the average daily number of transactions in the past 7, 14, 30, 45, 60 & 90 days.
- **Velocity days-since** variables are created to measure the *velocity change variables* for an entity over the *days-since variables* for the same entity.

4.6.6 Unique Entity variables

Unique entity variables were created to calculate the occurrences of unique entities/combination groups for a particular entity/combination group.

4.6.7 Acceleration variables

The idea behind creating these variables is calculating the ratio of *velocity change variables* for given entities over the square of average daily number of card transactions with the same entities over the past 7, 14, 30, 45, 60 and 90 days

4.6.8 Variability variables

The variability variables help measure the variation (mean, max, median) in the difference in transaction amounts over the past 'n' days where n = 0, 1, 3, 7, 14 and 30 days for a given entity.

4.6.9 Interesting variables

- We created a variable *card_merch* merging card number and merchant number to identify unique occurrences of these combinations, and further use these in creating velocity, days-since and acceleration variables
- Also, we created a *zip3* variable by considering only the first three digits of the traditional 5-digit zip code variable. We also used this as an entity to further link it with other entities/combination groups and hence the above-mentioned variables.
- In addition, we have created a *card_merch_zip3* variable which includes the *zip3* variable, card number and merchant number to zoom in further on specific merchant-card transactions and investigate in our analysis to detect fraud.

4.6.10 Summary of variables

A total of ***1806 variables*** have been created for this project.
The following table provides a detailed description of variables, their description, number of variables created and approximate CPU time.

Name	Description of Variables	# Variables Created	Approximate CPU Time
RISK TABLE VARIABLES	**Risk Table Variables:** The following categorical variables are used to create the target encoded variables by taking the average of the target variable for each category. 1. Day of Week 2. Merch state Variables created are: 1. Low_risk (for day of week) 2. Merch_state_risk (for Merch state)	**2**	**0.96 s**
BENFORD'S LAW VARIABLES	**Benford's Law Variables:** The following two entities have been used to create Benford's Law variables: 1. Cardnum 2. Merchnum Variables created are: 3. U* for Cardnum 4. U* for Merchnum	**2**	**22.85 s**
CUSTOM VARIABLES	**Custom Variables:** These variables are formed by joining two or more variables together. The following entities have been created. 1. Card_merch 2. Card_zip 3. Card_state 4. Zip3 5. Card_zip3 6. Merchnum_state 7. Merchnum_zip 8. Merchnum_zip3 9. Card_merch_zip3	**8**	**0.3 s**

Name	Description of Variables	# Variables Created	Approximate CPU Time
	Note: Zip3 variable is created taking the first 3 digits of Merch zip variable.		
DAYS-SINCE VARIABLES	**Days-since Variables:** Current date minus date of most recent transaction with same entities. The entities are: 1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3	**13**	**123.57 s**

FREQUENCY AND AMOUNT VARIABLES	**Frequency Variables:**
	
	# Card transactions with these entities over the past 'n' days.
	n = 0,1,3,7,14,30,45,60,90 days
	
	The entities are:
	
	1. Cardnum
	2. Merchnum
	3. Merch description
	4. Merch state
	5. Merch zip
	6. card_merch
	7. card_zip
	8. card_state
	9. zip3
	10. card_zip3
		1053	**265.83 s**

Name	Description of Variables	# Variables Created	Approximate CPU Time
	11. merchnum_state 12. merchnum_zip 13. merchnum_zip3 **Amount Variables:** Average, Maximum, Median, Total, Actual/average, Actual/maximum, Actual/median, Actual/total amount by or at these entities over the past n days where n = 0,1,3,7,14,30,45,60,90 days. The entities are: 1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3		
VELOCITY CHANGE VARIABLES	**Velocity change variables (Ratio):** (# Card transactions with same entities over the past 0 or 1 day) **Divided by** (Average daily # Card transactions with same entities over the past 7, 14, 30, 45, 60 and 90 days) The entities are: 1. Cardnum 2. Merchnum	**156**	**0.56 s**

Name	Description of Variables	# Variables Created	Approximate CPU Time
	3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3		
VELOCITY DAYS-SINCE VARIABLES	**Velocity days-since variables (Ratio):** (Velocity change variables for an entity) **Divided by** (Days-since variables for the same entity) The entities are: 1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3	**26**	**0.15 s**
UNIQUE VARIABLES	**Unique variables:** # Unique entities for those particular entities: The entities are:	**156**	**5.55 s**

Name	Description of Variables	# Variables Created	Approximate CPU Time
	1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3		
ACCELERATION VARIABLES	**Acceleration variables:** (Velocity change variables for given entities) **Divided by** (Square of Average daily # Card transactions with same entities over the past 7, 14, 30, 45, 60 and 90 days) The entities are: 1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3	**156**	**1.24 s**
VARIABILITY VARIABLES	**Variability variables**	**234**	**256.31 s**

Name	Description of Variables	# Variables Created	Approximate CPU Time
	Variation (mean, max, median) in the difference in the transaction amounts looking over the past 'n' days where n = 0,1,3,7,14,30 days for a given entity. The entities are: 1. Cardnum 2. Merchnum 3. Merch description 4. Merch state 5. Merch zip 6. card_merch 7. card_zip 8. card_state 9. zip3 10. card_zip3 11. merchnum_state 12. merchnum_zip 13. merchnum_zip3		

Table 4.2: Variable Summary Table

5. Feature Selection

Feature selection is also known as variable selection, attribute selection or variable subset selection. It is the process of selecting a subset of relevant variables for use in model construction.

Feature selection is one of the core concepts in machine learning which hugely impacts the performance of the model. Having irrelevant variables can decrease the accuracy of the models and make the model learn based on these irrelevant variables.

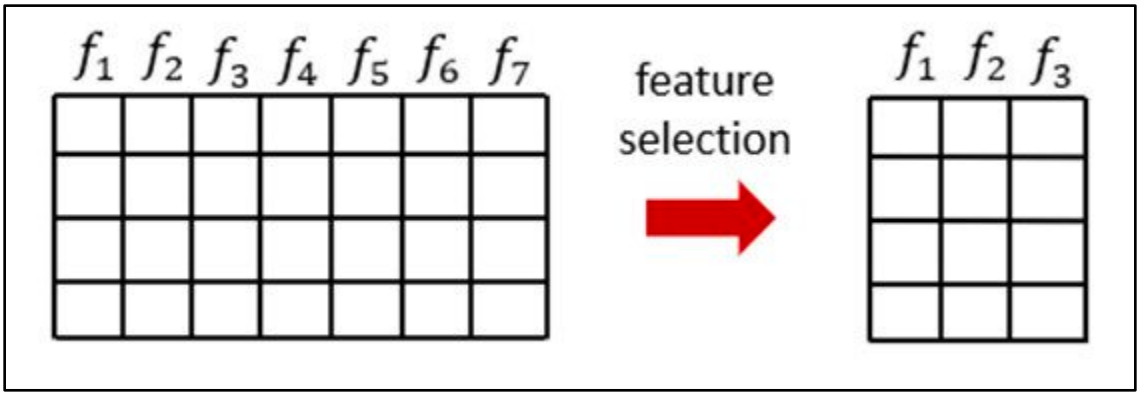


Figure 5.a: Illustration of feature selection on tabular data

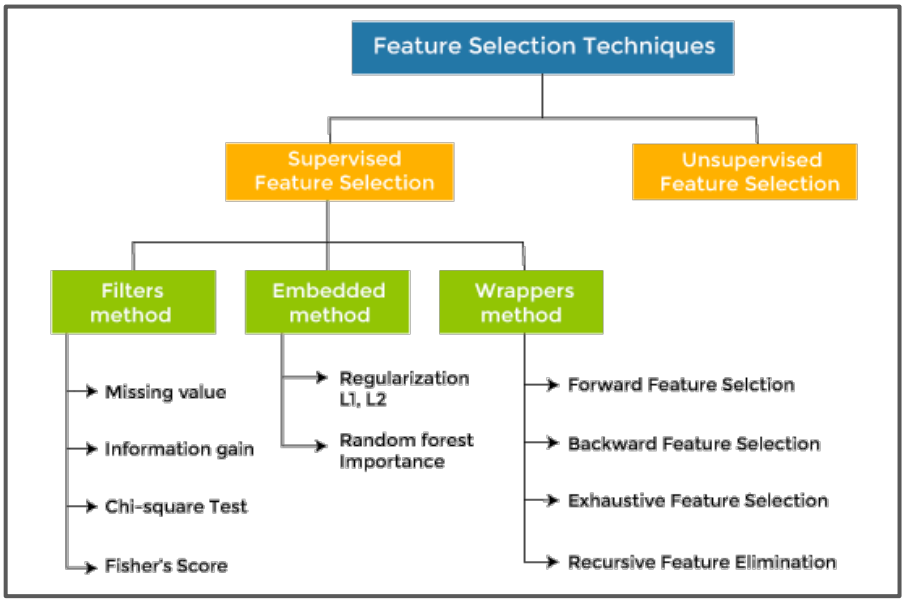


Figure 5.b: Illustration of feature selection techniques

The objectives of feature selection include:

- Avoiding the curse of dimensionality
- Enhanced generalization by reducing overfitting
- Shorter training times
- Simplification of models to make them easier to interpret by users

Dimensionality reduction is taken seriously due to the following reasons:

- Data is always sparse
- All points become outliers
- Need exponentially more data to see true nonlinearities rather than noise

After creating thousands of new variables during feature engineering, the variable selection process is performed for dimensionality reduction. In this project, supervised learning has been employed and the two-step approach has been adopted as shown below:

1) Filtering
2) Wrapping

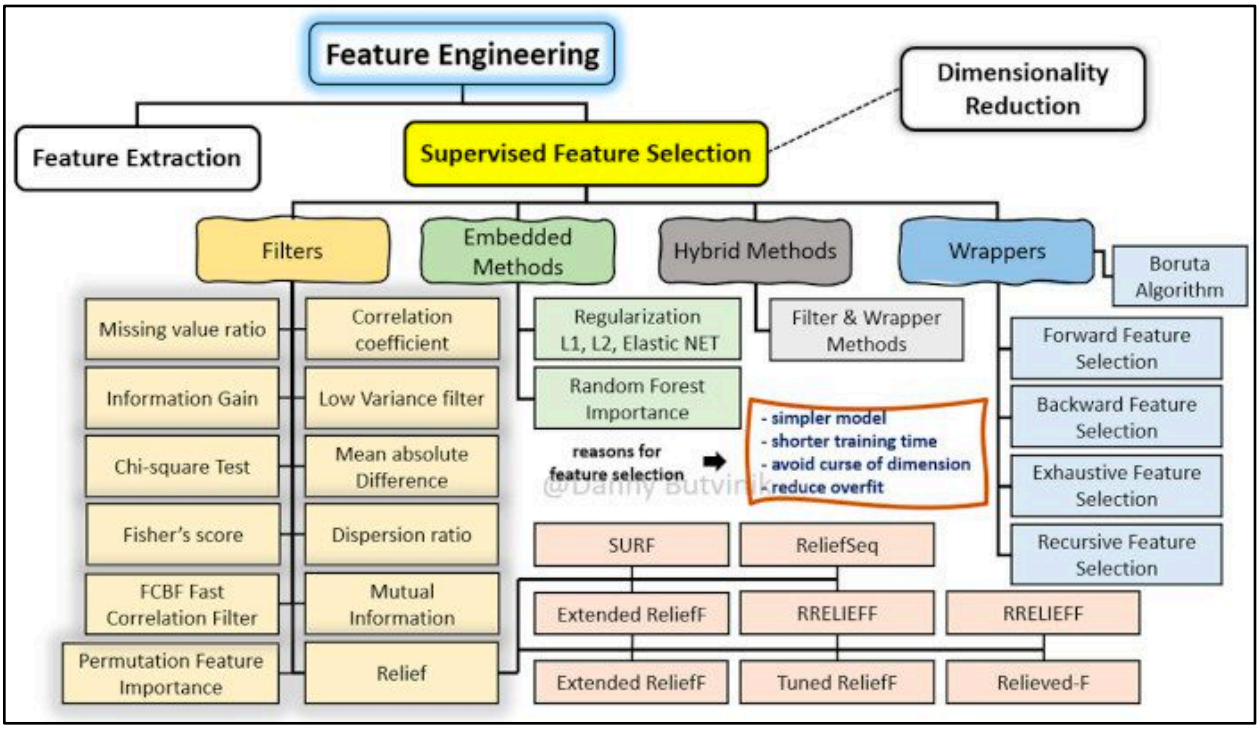


Figure 5.c: Feature selection techniques for Supervised Learning

5.1 Filtering

The selection of variables by filtering method is independent of any machine learning algorithm and takes less computational time. The variables are selected based on their statistical scores for their correlation with the target variable. Statistical tests are used to select those features that have the strongest relationship with the output variable. Depending on the type of features and response variables, different scores are used. Since our project is a binary classification, our target variable is binary which is discrete in nature.

Variable / Response	Continuous/Metric Ordering	Categorical
Continuous/Metric Ordering	Pearson's Coefficient or KS	LDA
Categorical	Anova	Chi-Squared

Table 5.1: Various filtering methods

Since our response variable is binary discrete, we have used the univariate KS (Kolmogorov-Smirnov) filtering method which is the most useful one for this scenario.

5.1.1 Univariate Kolmogorov-Smirnov (KS)

In this filtering method, for each candidate variable, goods and bads are plotted separately.
Goods: Normalized distribution of that variable for which the target variable is good i.e., Fraud label = 0

Bads: Normalized distribution of that variable for which the target variable is bad i.e., Fraud label = 1

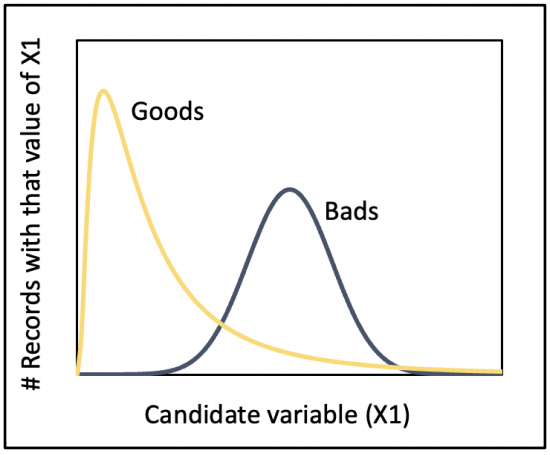


Figure 5.1.1: Normalized distributions of Goods and Bads

The more different the curves the better is the variable for separating, and thus the more important is the variable. KS is an efficient and simple measure of how separate these curves are. KS is given by the following formula:

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

In our project, we have calculated univariate KS for each of the candidate variables. The candidate variables with low KS scores are thus discarded.

Therefore, for the filtering process, the following steps are performed:

- Firstly, data of the last two months has been removed as the Out-Of-Time (OOT) data. Additionally, the records of the first two weeks were also removed as their variables were not well formed.
- Secondly, data was separated as goods and bads as described above.
- Thirdly, KS filtering was performed on all the 1806 candidate variables.
- Lastly, the top 80 variables with high KS scores and strong correlation with the target variable were selected.

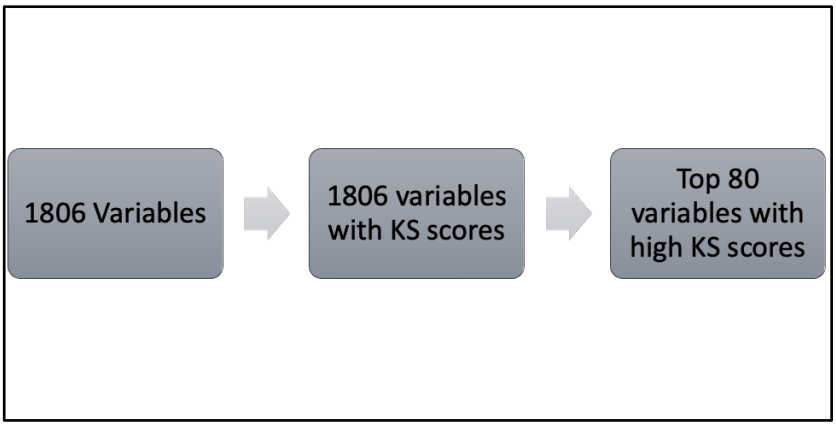


Figure 5.1.2: Method to choose top 80 variables

5.2 Wrapping

In the wrapper method, a subset of variables is taken, and a model is trained on them. Hence, it is called as a wrapper method as a model is "wrapped" around the process. Based on this iterative process, variables were added or removed to improve the model performance.

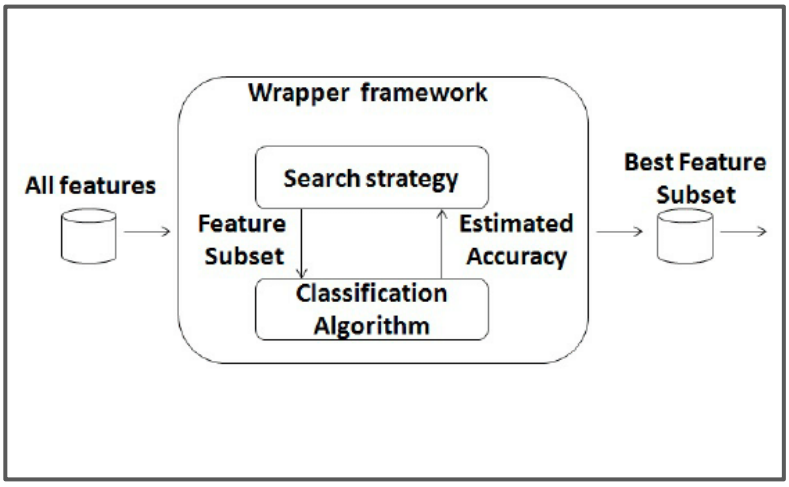


Figure 5.2: A wrapper framework

In general, the following stepwise selection methods can be used for wrapping:

- Forward Stepwise Selection
- Backward Stepwise Selection
- General Stepwise Selection

In our project, the forward feature selection method was used for wrapping. During the forward selection method, 1806 separate 1-D models have been built in the first step. The best variable is kept while adding the next best variable in each step thus, building all possible models.

This approach finds a good set of candidate variables while removing correlations. Therefore, at the end of wrapping, the top 20 variables were obtained in the order of multivariate importance.

5.2.1 Forward Feature Selection:

The process starts with the best performing feature and then only those features which increase the model performance are added.

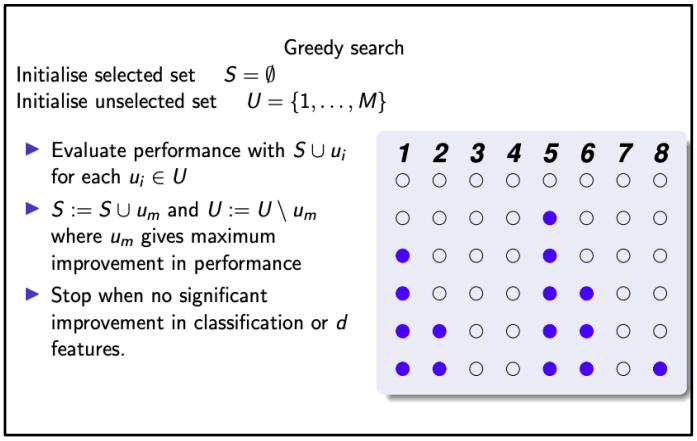


Figure 5.2.1.a: Forward feature selection procedure

The forward feature selection process involves the following steps:

- Train the model with a single variable, the one which gives the better result based on the evaluation metric.
- Select a second variable which in combination with the first gives the best performance.
- Continue the above steps.
- Stop when no significant improvement is observed, or the limit of required variables is reached.

In addition, the chart below indicates the performance achieved by the forward stepwise selection wrapper that was used for filtering the expert variables.

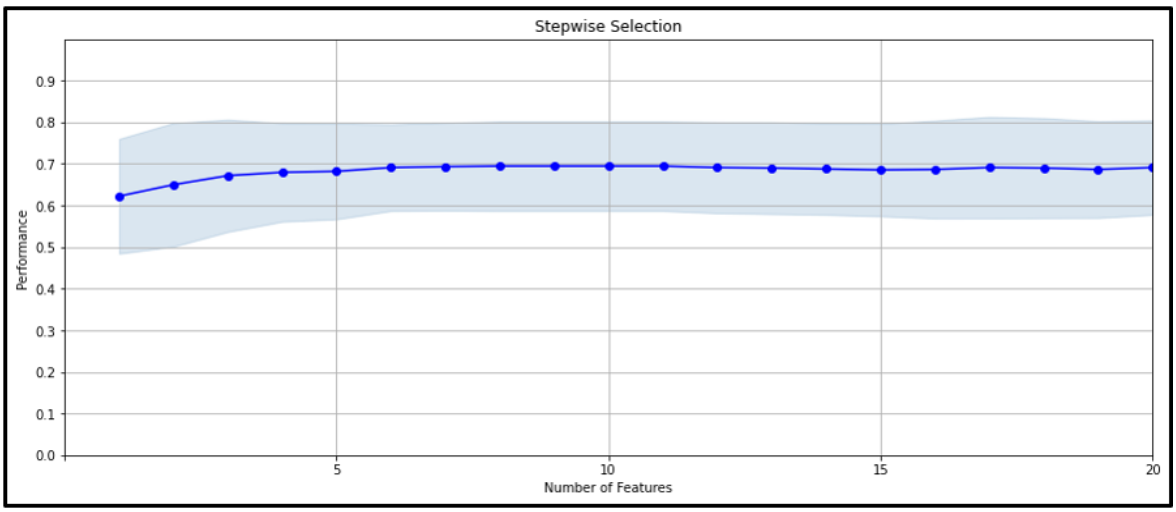


Figure 5.2.1.b: Wrapper Performance

To conclude, Boosted Trees (LGBM Classifier) was used as the machine learning algorithm for wrapping on the resultant variables from the filtering phase. The top 20 variables resulted from these processes are listed below along with their KS scores and Average scores post wrapping.

#	Variable	Average Score after Wrapping	KS Score
1	card_zip3_total_7	0.622	0.696
2	merchnum_state_total_1	0.650	0.604
3	Cardnum_total_3	0.672	0.602
4	card_merch_max_3	0.680	0.645
5	card_zip_total_30	0.682	0.656
6	card_zip_max_45	0.691	0.643
7	card_zip_total_0	0.694	0.610
8	card_merch_max_30	0.695	0.650
9	card_zip3_total_0	0.695	0.615
10	card_merch_total_0	0.695	0.611
11	card_state_total_0	0.695	0.611
12	card_state_total_30	0.691	0.636
13	card_state_total_14	0.690	0.669
14	card_zip_max_3	0.688	0.650
15	merchnum_zip3_total_1	0.686	0.602
16	card_state_total_7	0.687	0.670
17	card_zip3_total_30	0.691	0.661
18	card_state_total_45	0.690	0.607
19	card_state_max_3	0.687	0.648
20	card_state_max_1	0.691	0.626

Table 5.2.1.c: Top 20 Variables with their KS scores

Top 20 Variables after Forward Selection (Sorted by Importance)			
feature_idx	Index of Most Important Variables	Variable name	Order of Most important variables
(0,)	0	card_zip3_total_7	1
(0, 73)	73	merchnum_state_total_1	2
(0, 73, 78)	78	Cardnum_total_3	3
(0, 35, 73, 78)	35	card_merch_max_3	4
(0, 23, 35, 73, 78)	23	card_zip_total_30	5
(0, 23, 35, 38, 73, 78)	38	card_zip_max_45	6
(0, 23, 35, 38, 64, 73, 78)	64	card_zip_total_0	7
(0, 23, 28, 35, 38, 64, 73, 78)	28	card_merch_max_30	8
(0, 23, 28, 35, 38, 56, 64, 73, 78)	56	card_zip3_total_0	9
(0, 23, 28, 35, 38, 56, 59, 64, 73, 78)	59	card_merch_total_0	10
(0, 23, 28, 35, 38, 56, 59, 61, 64, 73, 78)	61	card_state_total_0	11
(0, 23, 28, 35, 38, 45, 56, 59, 61, 64, 73, 78)	45	card_state_total_30	12
(0, 11, 23, 28, 35, 38, 45, 56, 59, 61, 64, 73, 78)	11	card_state_total_14	13
(0, 11, 23, 27, 28, 35, 38, 45, 56, 59, 61, 64, 73, 78)	27	card_zip_max_3	14
(0, 11, 23, 27, 28, 35, 38, 45, 56, 59, 61, 64, 73, 77, 78)	77	merchnum_zip3_total_1	15
(0, 10, 11, 23, 27, 28, 35, 38, 45, 56, 59, 61, 64, 73, 77, 78)	10	card_state_total_7	16
(0, 10, 11, 14, 23, 27, 28, 35, 38, 45, 56, 59, 61, 64, 73, 77, 78)	14	card_zip3_total_30	17
(0, 10, 11, 14, 23, 27, 28, 35, 38, 45, 56, 59, 61, 64, 68, 73, 77, 78)	68	card_state_total_45	18
(0, 10, 11, 14, 23, 27, 28, 31, 35, 38, 45, 56, 59, 61, 64, 68, 73, 77, 78)	31	card_state_max_3	19
(0, 10, 11, 14, 23, 27, 28, 31, 35, 38, 45, 51, 56, 59, 61, 64, 68, 73, 77, 78)	51	card_state_max_1	20

Figure 5.2.4: Top 20 variables with their indices

In a nutshell, the image below represents the high-level summary of feature selection that was used in our project.

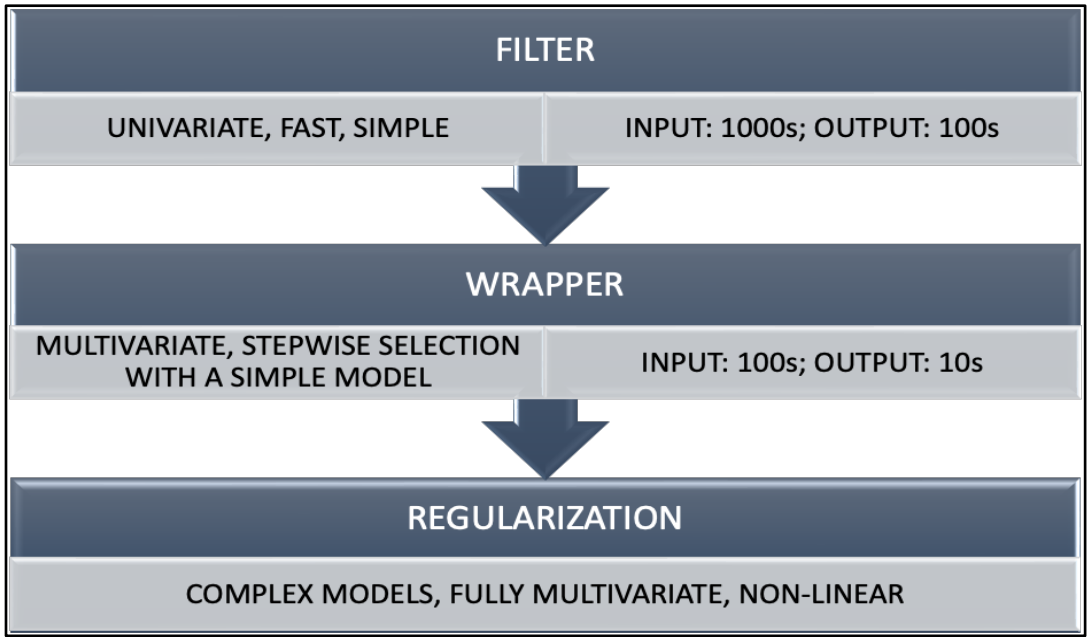


Figure 5.3: High level summary of feature selection

6. Model Algorithms & hyperparameter tuning

After getting the final variables we explore different machine learning algorithms to predict fraud. We have used the following algorithms:

1. Logistic Regression
2. Decision Tree
3. Random forest
4. Gradient boosting
5. Neural Network

This section gives a brief idea about each one of these algorithms and then the results we got after using them with different hyperparameters.

6.1 Logistic Regression

For this project we have used logistic regression as a baseline model since it's simple and linear. This method is used for classification and models the probability of response variable given independent variables using a logistic function. It can be summarized mathematically as follows:

$$P = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Where P is P (Y = 1 | X = x)

The logistic or sigmoid function ensures that the modeled probability lies between 0 and 1.

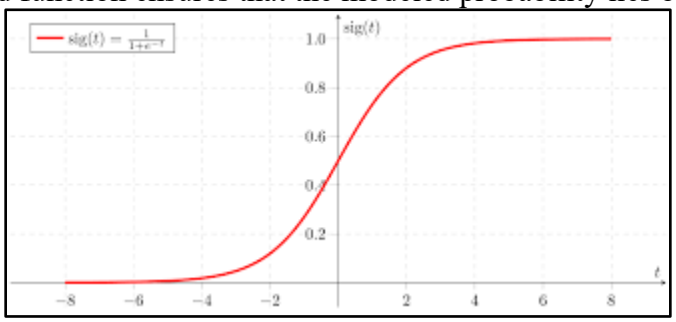


Figure 6.1: Sigmoid function

Following hyperparameters were explored for logistic regression:

- Penalty: to specify which regularization method is to be used. L1, L2 or elastic net
- C: Used to adjust the strength of regularization. Smaller values indicate stronger regularization.
- Solver: Algorithm to use for solving the optimization problem.

6.2 Decision Tree

This is a nonlinear classification method (could be used for regression problems as well) where the algorithm partitions the feature space till it achieves sufficient purity in each partition. In other words, it learns simple decision rules inferred from the data features to predict the value of Y. Decision trees are easy to interpret. Following picture shows the different components of the decision tree.

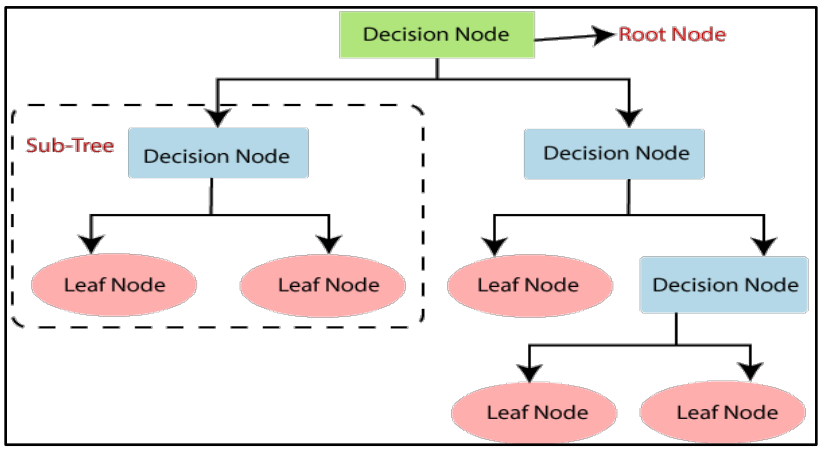


Figure 6.2: Illustration of decision tree

Decision trees can easily overfit the data hence some hyperparameters must be kept in mind to reduce overfitting.
Some common hyperparameters are:

- Criterion: Criterion to calculate impurity of nodes. Common values are gini and entropy. Gini impurity is the weighted average of the impurities of each node in a split done by a feature. Impurity of node is calculated as follows:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

  Entropy is defined as follows and measures the impurity of each node

$$H[X] = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

- Max_depth: Maximum number of levels in the trained tree
- Min_samples_split: Minimum number of samples to be present at the node for it to be considered for splitting
- min_samples _leaf: Minimum number of samples to be present in the leaf nodes if the split were to happen
- Splitter: strategy used to perform the split. Options are ‘best’ or best ‘random’ split.

6.3 Random Forest

This is an ensemble of decision trees used for classification or regression problems. It creates multiple decision trees. Each individual tree is created through some sort of randomness for example using only a randomly chosen subset of variables or records for each tree and/or for each split iteration within a tree. Results from all trees are combined using voting or average.

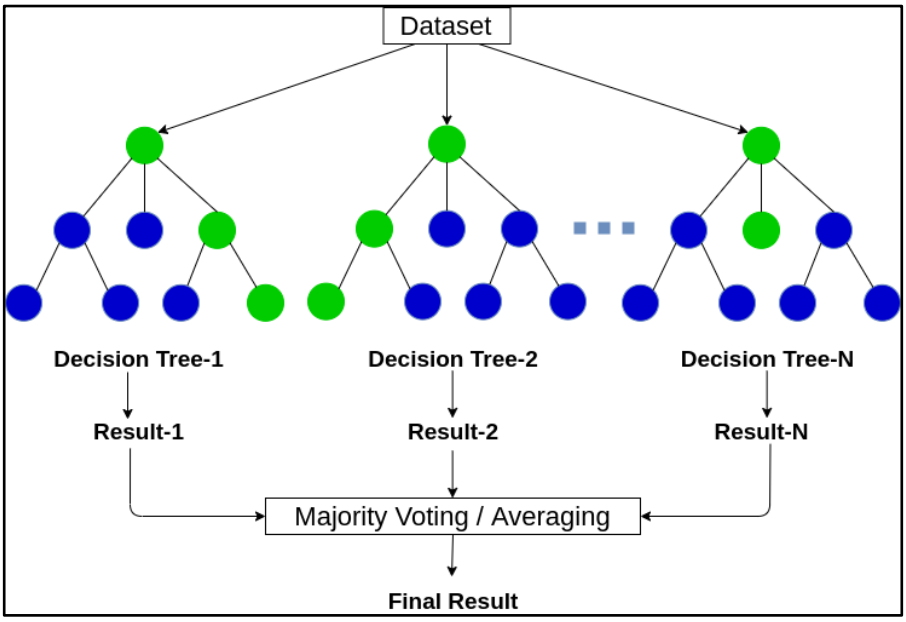


Figure 6.3: Illustration of Random Forest

Random forests improve over decision trees in terms of overfitting and stability.
In addition to decision tree hyperparameters, random forests have some more hyperparameters like:

- n_estimators: number of trees in the forest
- Bootstrap: whether bootstrap samples are used to build the trees.

6.4 Light Gradient Boosting

Boosted trees are another way of improving on the drawback of decision trees. These are different from random forests because construction of trees is sequential and not random. Light GBM is a fast, distributed, high-performance gradient boosting framework based on a decision tree algorithm. Unlike other boosting algorithms, light GBM grows leaf wise. It also uses less memory and is faster than other boosting algorithms.

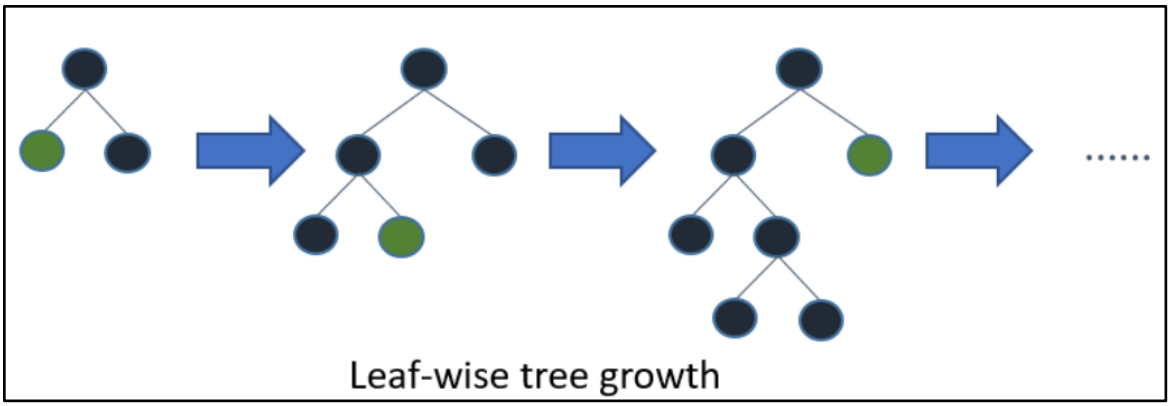


Figure 6.4: Illustration of LGB algorithm

Following hyperparameters have been explored in this project:

- n_estimators: number of boosted trees to fit
- num_leaves: maximum tree leaves for base learners
- learning_rate: boosting learning rate
- boosting_type: Gradient Boosting Decision Tree (GBDT) or Dropouts meet Multiple Additive Regression Trees (DART)
- max_depth: depth of each tree

6.5 Neural Network

Artificial neural networks (ANNs) are composed of layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network.
Otherwise, no data is passed along to the next layer of the network. This mechanism mimics how the human brain works.

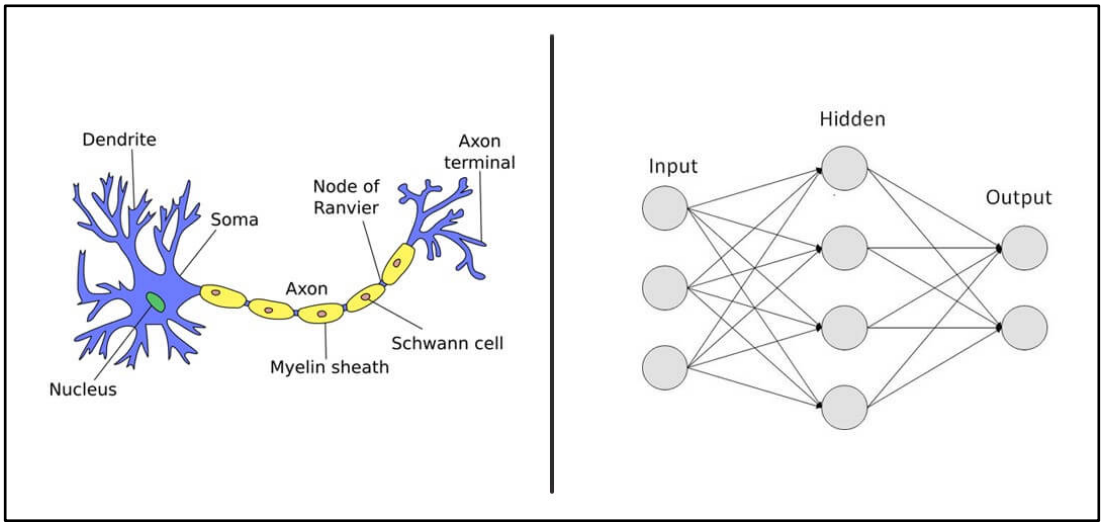


Figure 6.5: Basic components of neuron and perceptron

Some hyperparameters:

- N_hidden: number of hidden layers
- hidden_layer_sizes: number of neurons in hidden layers
- activation: Activation function for the hidden layer
- solver: solver for weight optimization. Sgd implies stochastic gradient descent and adam implies different version of stochastic gradient descent
- alpha: L2 (regularization) parameter
- learning_rate: kind of learning. 'constant' is a constant learning rate given by 'learning_rate_init'. 'invscaling' gradually decreases the learning rate at each time step 't'. 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing.
- learning_rate_init: The initial learning rate used
- momentum: Momentum for gradient descent update for sgd solver.

6.6 Hyperparameter Tuning

We ran 1000+ of models across the below algorithms, choosing combinations of hyperparameters, and obtained the average model performance for each model over 10 iterations.

In figure 6.6 we display a select number of models, their corresponding hyperparameters and performance results among the 1000+ models that were explored. We tried a range of hyperparameters that could potentially underfit, fit properly and overfit the data.

Sl No.	Model		Variables	Hyperparameters								Average FDR at 3%		
	Model Name	Iteration	#Total Variables	penalty			C			solver		Train	Test	OOT
1	Logistic Regression	1	15	None (default)			1.0 (default)			saga		68.42%	67.32%	31.79%
		2	15	l2			0.1			sag		68.45%	68.19%	32.23%
		3	15	l1			0.001			saga		62.04%	61.33%	43.24%
		4	15	l1			0.01			saga		62.73%	61.97%	42.30%
		5	15	l1			0.001			liblinear		63.47%	64.15%	41.73%
		6	15	l2			50			lbfgs		68.67%	67.45%	32.85%
		7	15	l2			0.001			newton-cg		66.28%	67.36%	32.79%
		8	15	l2			10			lbfgs		68.62%	67.43%	32.79%
		9	15	l2			1			newton-cg		68.23%	69.28%	32.68%
	Model Name	Iteration	# Total Variables	max_features	criterion	max_depth	min_samples_split		min_samples_leaf	splitter		Train	Test	OOT
2	Decision Trees	1	15	None (default)	gini (default)	None (default)	2 (default)		1 (default)	best		100.00%	62.73%	25.14%
		2	15	None	gini	3	10		10	best		56.31%	52.98%	30.73%
		3	15	None	gini	15	100		20	best		86.48%	77.57%	47.21%
		4	15	None	gini	50	500		50	random		69.69%	69.10%	44.25%
		5	15	None	entropy	100	1100		70	random		66.50%	66.11%	42.29%
		6	15	None	gini	15	1000		1	best		75.54%	72.82%	50.56%
		7	15	None	gini	100	500		2	random		73.20%	71.23%	50.00%
		8	15	None	gini	15	500		2	random		72.25%	71.30%	49.61%
		9	15	None	gini	100	500		1	random		72.54%	72.77%	49.11%
		10	15	None	gini	15	500		1	random		72.72%	70.34%	48.72%
		11	15	None	gini	100	500		5	random		72.25%	69.73%	48.44%
	Model Name	Iteration	# Total Variables	n_estimators	max_features	max_depth	min_samples_split		min_samples_leaf	bootstrap		Train	Test	OOT
3	Random Forest	1	15	100	5	None (default)	2 (default)		1(default)	TRUE		100.00%	85.22%	56.59%
		2	15	150	10	5	10		10	TRUE		74.53%	72.45%	52.68%
		3	15	50	15	8	100		20	TRUE		81.07%	76.74%	57.21%
		4	15	20	10	10	500		30	FALSE		80.29%	77.66%	57.49%
		5	15	10	5	10	500		20	FALSE		79.16%	76.21%	55.70%
		6	15	30	10	8	500		30	TRUE		76.98%	74.02%	55.20%
		7	15	20	10	8	500		30	FALSE		76.83%	74.07%	55.14%
		8	15	20	10	10	100		20	FALSE		76.00%	73.28%	54.86%
		9	15	30	5	8	100		20	FALSE		75.69%	74.79%	54.80%
		10	15	10	10	8	100		20	TRUE		76.71%	75.75%	54.75%
		11	15	10	3	20	1000		50	TRUE		76.87%	73.60%	46.82%
	Model Name	Iteration	# Total Variables	n_estimators	num_leaves	max_depth	learning rate	subsample	min_samples_leaf	min_samples_split		Train	Test	OOT
4	Boosted Trees	1	15	100 (default)	31 (default)	default	0.1 (default)	1.0 (default)	default	default		99.95%	84.86%	53.30%
		2	15	10	100	4	0.1	0.9	30	100		82.10%	76.06%	55.03%
		3	15	50	200	5	0.01	0.8	30	500		83.09%	79.58%	55.59%
		4	15	50	50	4	0.01	0.9	25	500		79.12%	76.24%	53.91%
		5	15	100	50	4	0.01	0.9	15	500		81.16%	78.64%	53.18%
		6	15	100	100	4	0.01	0.9	15	100		82.16%	79.12%	53.07%
		7	15	50	100	4	0.01	0.9	25	100		78.27%	75.52%	52.85%
		8	15	10	200	4	0.01	0.9	15	500		73.80%	72.88%	50.00%
		9	15	1000	50	8	0.02	0.6	50	1500		100.00%	83.45%	57.82%
		10	15	100	50	-1	0.01	0.9	25	100		92.96%	82.27%	59.55%
	Model Name	Iteration	# Total Variables	max_iter	hidden_layer_sizes	activation	solver	alpha	learning_rate	learning rate init	momentum	nesterovs_momentum	Train	Test	OOT
5	Neural Network	1	15	200 (default)	100 (default)	relu (default)	adam (default)	0.0001 (default)	N/A	0.001 (default)	N/A	N/A	82.80%	79.19%	52.46%
		2	15	100	50	relu	adam	0.0001	N/A	0.005	N/A	N/A	84.53%	78.98%	54.69%
		3	15	50	1	logistic	sgd	0.001	adaptive	0.02	0.9 (default)	TRUE (default)	69.10%	69.88%	35.92%
		4	15	50	10	relu	lbfgs	0.0001	N/A	0.0001	N/A	N/A	74.93%	73.52%	50.00%
		5	15	300	200	identity	lbfgs	0.01	N/A	0.01	N/A	N/A	68.55%	67.19%	32.07%
		6	15	200	200	tanh	adam	0.0001	N/A	0.001	N/A	N/A	81.84%	79.16%	55.53%
		7	15	100	100	tanh	adam	0.0001	N/A	0.001	N/A	N/A	80.46%	78.71%	55.14%
		8	15	200	200	tanh	adam	0.001	N/A	0.001	N/A	N/A	81.06%	77.82%	54.97%
		9	15	100	100	relu	lbfgs	0.001	N/A	0.001	N/A	N/A	80.51%	77.51%	54.75%
		10	15	100	200	relu	lbfgs	0.0001	N/A	0.001	N/A	N/A	80.97%	76.50%	55.81%
		11	15	500	400	logistic	sgd	0.001	adaptive	0.02	0.9 (default)	TRUE (default)	68.32%	68.06%	32.01%

Figure 6.6: Hyperparameter tuning

1. Logistic regression:
   All the hyperparameter combinations for logistic regression seem to underfit train data.

2. Decision Tree:
   Decision tree with default values seems to overfit the data whereas the second combination with max_depth of 3 and min_samples_split and min_samples_leaf of 10 underfits the data. Somewhere between these two extremes we find that the best performance is given by a tree with a min_sample_split of 1000.

3. Random Forest:
   Random forests with default values also overfits the data. The best set of hyperparameters are those obtained in the fourth iteration.

4. Light Gradient Boosting:
   LGB overfits the data in the ninth iteration since it has about 1000 trees. The best case with no overfitting is iteration 5 with 100 trees, 50 number of leaves, 0.01 learning rate, 15 min_samples_leaf and 500 min_samples_split.

5. Neural Networks: The lowest performance is 32% FDR using sgd solver, logistic activation function and 400 nodes in hidden layer. Third model is underfitting the data with only 1 hidden layer node and similar other parameters.
   Optimal hyperparameter set is iteration 10 with 55.81% FDR in validation set

7. Results

Based on our exploration of hyperparameters and algorithms, we have decided the following to be our final model:

- Random forest with 20 trees, 10 max_features, 10 max_depth, 100 min_samples_split, 20 min_samples_leaf and bootstrap as false.

Following are the top 5 features in our model. All of them are amount variables as discussed in section 4.6.4:

- *Cardnum_total_3*: The total amount spent using that card number in the last 3 days
- *Card_zip3_total_7*: The total amount within past 7 days spent using a card in an area represented by the first three digits of the zip code
- *Card_merch_max_3*: The maximum amount spent within last 3 days for each combination of card number and merchant number
- *Merchnum_zip3_total_1*: The total amount within last 1 day for each combination of merchant number and zip3
- *Card_zip_max_45*: The maximum amount spent using a card in a zip code within the last 45 days

Feature	Random Forest Feature Importance
Cardnum_total_3	0.43
Card_zip3_total_7	0.2899
Card_merch_max3	0.0585
Merchnum_zip3_total_1	0.0552
Card_zip_max_45	0.0528

Table 7.1: Top 5 Features

Feature importance is computed as the mean and standard deviation of accumulation of the impurity decrease within each tree. We used this metric to get top variables since our final model is a tree-based model.

The following tables show how the FDR increases as we increase the percentage of the top population in training, test and validation set:

Training set

Training	#Records			#Goods		#Bads			Fraud Rate		
	59008			58401		607			0.010286741		
	Bin Statistics					Cumulative Statistics					
Population Bin%	#Records	#Goods	#Bads	%Goods	%Bads	Total	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	590	250	340	42.37	57.63	590	250	340	0.43	56.01	55.59	0.74
2	590	491	99	83.22	16.78	1180	741	439	1.27	72.32	71.05	1.69
3	590	539	51	91.36	8.64	1770	1280	490	2.19	80.72	78.53	2.61
4	590	566	24	95.93	4.07	2360	1846	514	3.16	84.68	81.52	3.59
5	590	584	6	98.98	1.02	2950	2430	520	4.16	85.67	81.51	4.67
6	590	584	6	98.98	1.02	3540	3014	526	5.16	86.66	81.49	5.73
7	591	578	13	97.80	2.20	4131	3592	539	6.15	88.80	82.65	6.66
8	590	582	8	98.64	1.36	4721	4174	547	7.15	90.12	82.97	7.63
9	590	586	4	99.32	0.68	5311	4760	551	8.15	90.77	82.62	8.64
10	590	588	2	99.66	0.34	5901	5348	553	9.16	91.10	81.95	9.67
11	590	588	2	99.66	0.34	6491	5936	555	10.16	91.43	81.27	10.70
12	590	589	1	99.83	0.17	7081	6525	556	11.17	91.60	80.43	11.74
13	590	585	5	99.15	0.85	7671	7110	561	12.17	92.42	80.25	12.67
14	590	587	3	99.49	0.51	8261	7697	564	13.18	92.92	79.74	13.65
15	590	588	2	99.66	0.34	8851	8285	566	14.19	93.25	79.06	14.64
16	590	588	2	99.66	0.34	9441	8873	568	15.19	93.57	78.38	15.62
17	590	589	1	99.83	0.17	10031	9462	569	16.20	93.74	77.54	16.63
18	590	588	2	99.66	0.34	10621	10050	571	17.21	94.07	76.86	17.60
19	591	589	2	99.66	0.34	11212	10639	573	18.22	94.40	76.18	18.57
20	590	590	0	100.00	0.00	11802	11229	573	19.23	94.40	75.17	19.60

Figure 7.1: Table showing FDR across Training set population bins

Test Set

Test	#Records			#Goods		#Bads			Fraud Rate		
	25290			25017		273			0.010794781		
	Bin Statistics					Cumulative Statistics					
Population Bin%	#Records	#Goods	#Bads	%Goods	%Bads	Total	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	253	94	159	37.15	62.85	253	94	159	0.38	58.24	57.87	0.59
2	253	213	40	84.19	15.81	506	307	199	1.23	72.89	71.67	1.54
3	253	237	16	93.68	6.32	759	544	215	2.17	78.75	76.58	2.53
4	253	244	9	96.44	3.56	1012	788	224	3.15	82.05	78.90	3.52
5	252	246	6	97.62	2.38	1264	1034	230	4.13	84.25	80.12	4.50
6	253	248	5	98.02	1.98	1517	1282	235	5.12	86.08	80.96	5.46
7	253	249	4	98.42	1.58	1770	1531	239	6.12	87.55	81.43	6.41
8	253	246	7	97.23	2.77	2023	1777	246	7.10	90.11	83.01	7.22
9	253	251	2	99.21	0.79	2276	2028	248	8.11	90.84	82.74	8.18
10	253	253	0	100.00	0.00	2529	2281	248	9.12	90.84	81.72	9.20
11	253	253	0	100.00	0.00	2782	2534	248	10.13	90.84	80.71	10.22
12	253	252	1	99.60	0.40	3035	2786	249	11.14	91.21	80.07	11.19
13	253	252	1	99.60	0.40	3288	3038	250	12.14	91.58	79.43	12.15
14	253	250	3	98.81	1.19	3541	3288	253	13.14	92.67	79.53	13.00
15	253	253	0	100.00	0.00	3794	3541	253	14.15	92.67	78.52	14.00
16	252	251	1	99.60	0.40	4046	3792	254	15.16	93.04	77.88	14.93
17	253	252	1	99.60	0.40	4299	4044	255	16.17	93.41	77.24	15.86
18	253	252	1	99.60	0.40	4552	4296	256	17.17	93.77	76.60	16.78
19	253	253	0	100.00	0.00	4805	4549	256	18.18	93.77	75.59	17.77
20	253	253	0	100.00	0.00	5058	4802	256	19.19	93.77	74.58	18.76

Figure 7.2: Table showing FDR across Test set population bins

Validation Set

Validation	#Records			#Goods		#Bads			Fraud Rate		
	12099			11920		179			0.014794611		
	Bin Statistics					Cumulative Statistics					
Population Bin%	#Records	#Goods	#Bads	%Goods	%Bads	Total	Cumulative Goods	Cumulative Bads	%Goods	%Bads (FDR)	KS	FPR
1	121	56	65	46.28	53.72	121	56	65	0.47	36.31	35.84	0.86
2	121	93	28	76.86	23.14	242	149	93	1.25	51.96	50.71	1.60
3	121	110	11	90.91	9.09	363	259	104	2.17	58.10	55.93	2.49
4	121	117	4	96.69	3.31	484	376	108	3.15	60.34	57.18	3.48
5	121	114	7	94.21	5.79	605	490	115	4.11	64.25	60.14	4.26
6	121	121	0	100.00	0.00	726	611	115	5.13	64.25	59.12	5.31
7	121	118	3	97.52	2.48	847	729	118	6.12	65.92	59.81	6.18
8	121	115	6	95.04	4.96	968	844	124	7.08	69.27	62.19	6.81
9	121	112	9	92.56	7.44	1089	956	133	8.02	74.30	66.28	7.19
10	121	119	2	98.35	1.65	1210	1075	135	9.02	75.42	66.40	7.96
11	121	118	3	97.52	2.48	1331	1193	138	10.01	77.09	67.09	8.64
12	121	119	2	98.35	1.65	1452	1312	140	11.01	78.21	67.21	9.37
13	121	120	1	99.17	0.83	1573	1432	141	12.01	78.77	66.76	10.16
14	121	121	0	100.00	0.00	1694	1553	141	13.03	78.77	65.74	11.01
15	121	119	2	98.35	1.65	1815	1672	143	14.03	79.89	65.86	11.69
16	121	119	2	98.35	1.65	1936	1791	145	15.03	81.01	65.98	12.35
17	121	119	2	98.35	1.65	2057	1910	147	16.02	82.12	66.10	12.99
18	121	121	0	100.00	0.00	2178	2031	147	17.04	82.12	65.08	13.82
19	121	119	2	98.35	1.65	2299	2150	149	18.04	83.24	65.20	14.43
20	121	120	1	99.17	0.83	2420	2270	150	19.04	83.80	64.76	15.13

Figure 7.3: Table showing FDR across OOT set population bins

The above table represents the result of the chosen model trained on entire train and test data and then applied on out of time validation data.

Dynamics of Fraud Scores

While this fraud detection algorithm can save 1.2 million dollars, many times it is possible for it to miss early frauds on lost or stolen cards. Fraud scores take time to recognize that a flow of transactions is unusual.
This could be understood by a simple example when one card number is used once or twice initially and as activity increases, these transactions look more suspicious due to the various count variables that we created.

Following two charts give specific examples for a card number that had initially 0 fraud score and as and when the number of transactions increased, the fraud score increased.

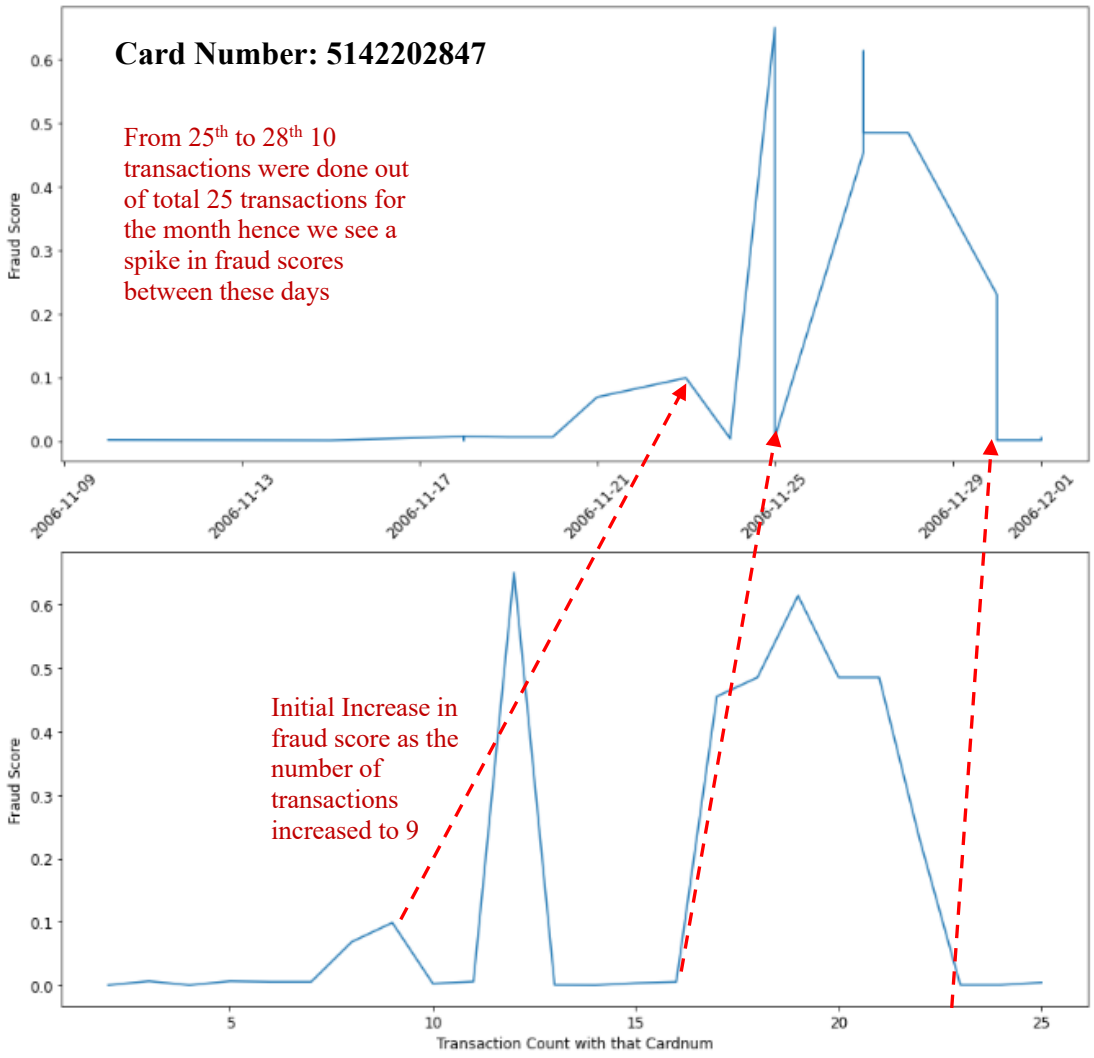


Figure 7.4 Fraud Score Dynamics for Card Number: 5142202847

The same can be seen for a specific merchant number '4353000719908'

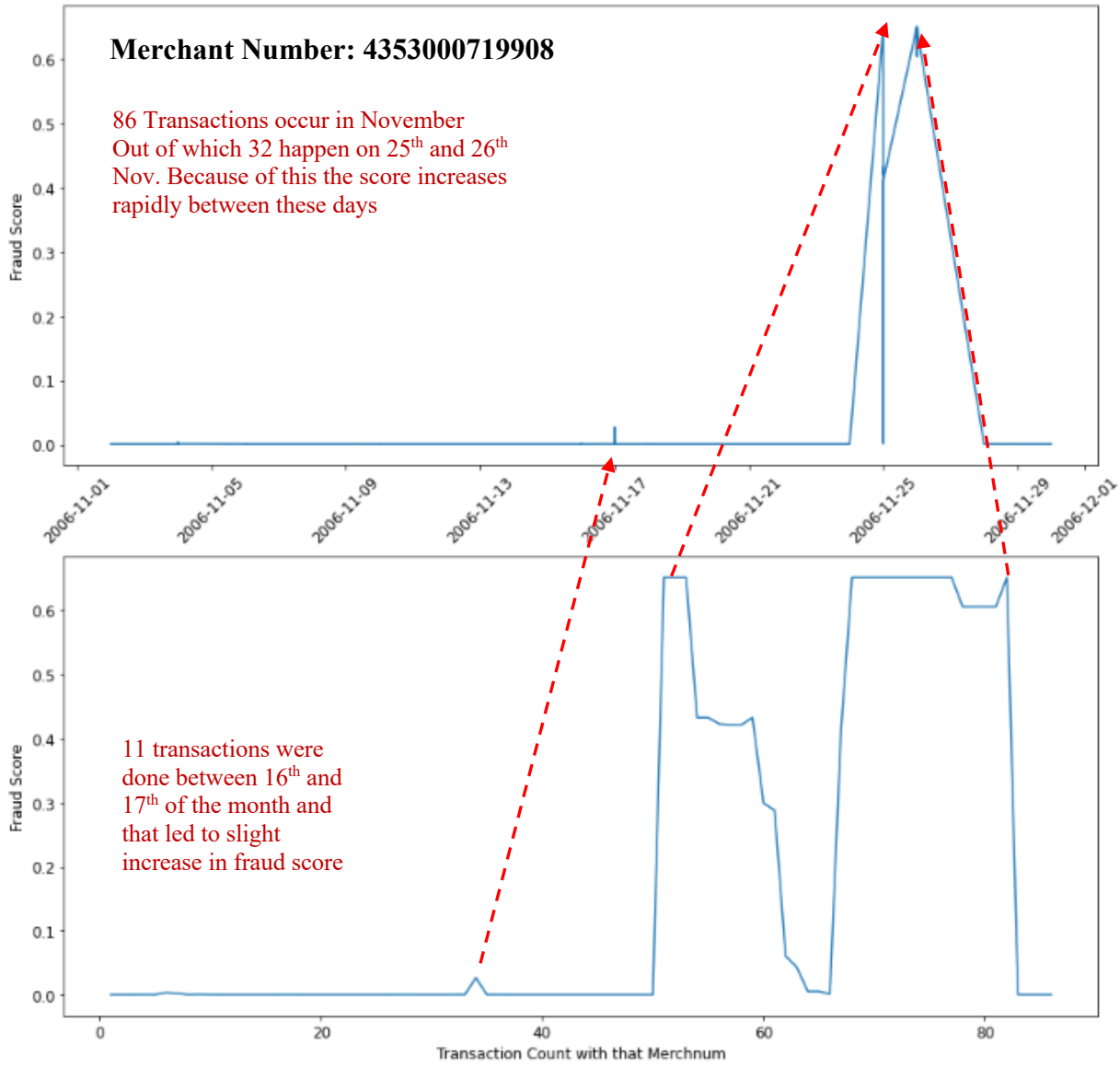


Figure 7.5 Fraud Score Dynamics for Merch Number: 4353000719908

Fraud Savings Chart

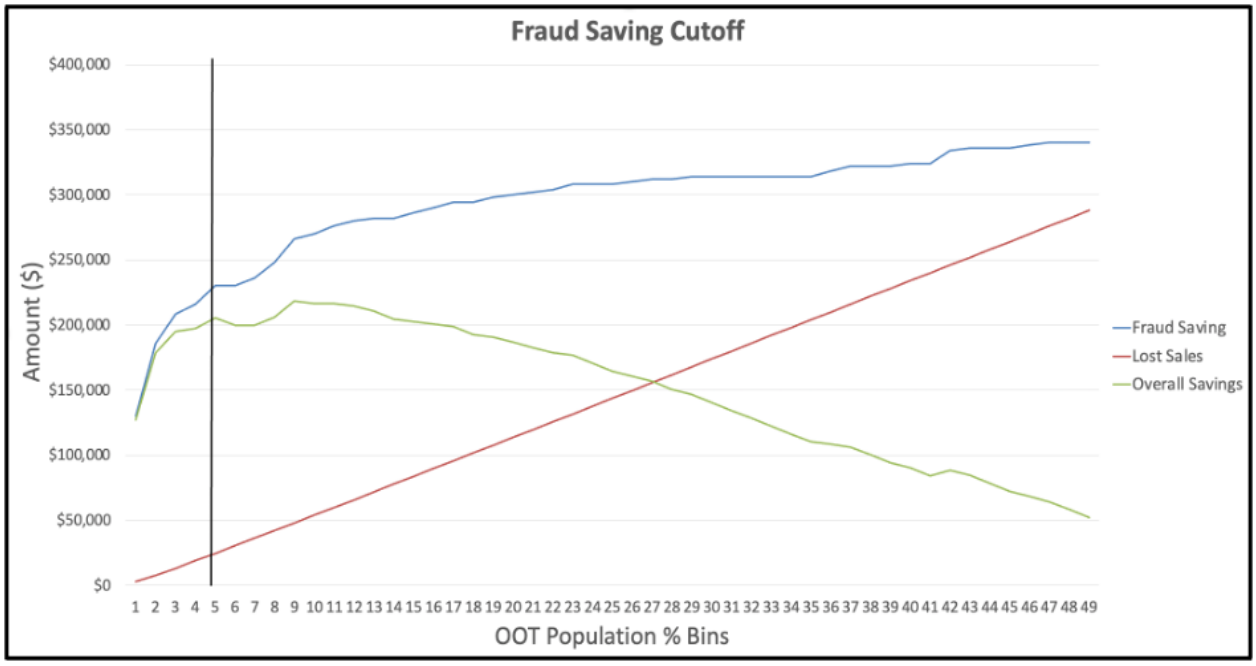


Figure 7.4: Savings Chart

The above chart depicts the amount we would be able to save using the final model. This chart shows savings for 2 months. Blue line corresponds to the amount saved for identifying fraudulent transactions and red line corresponds to amount lost due to rejecting applications. The green line represents the sum of the two and shows the overall savings.

If for each fraudulent transaction that we catch, we earn 2000 dollars and for every transaction rejected we lose 50 dollars then the maximum savings occur at 5% of the population.
Annually with this fraud detection algorithm, we would be able to save $1,233,000 if we reject only 5% of the transactions.

8. Conclusion

In this project, we went through the process of finding fraudulent credit card transactions leveraging machine learning techniques. We began by initially generating a Data Quality Report which outlines the overall distribution of data. Following which we cleaned the data, handled missing and frivolous values by imputation methodologies and dropped certain data points. Further, we performed exploratory data analysis to gain a solid understanding of the dataset.

We proceeded with feature engineering, creating 1806 candidate variables based on multiple combinations of the raw features. We created as many candidate variables as possible, as we believe that this is one of the most important steps which helps in identifying the right set of features which can generate great results even with linear models. Thereafter, we worked to reduce the dimensionality of our data by performing feature selection. We filtered for the top 80 variables using Kolmogorov-Smirnov (KS) score and used boosted tree technique for the wrapper with forward stepwise selection to arrive at the top 20 variables.

Following this, we ran 1000+ linear and non-linear models with various combinations of hyperparameters on our data. The modeling techniques included logistic regression as the base model, followed by decision trees, random forest, light GBM and neural networks. We chose random forest technique with selected hyperparameters as the final algorithm. Our model was able to achieve a Fraud Detection Rate (FDR) of 58.1% by declining top 3% of the population and 64.25% by declining top 5% of the population based on the fraud algorithm score, when tested on the out-of-time dataset.

Every project has its own limitations, but given more time and access to larger datasets, we would suggest the following improvements to our project:

- Interact with domain experts to create more quality candidate variables
- Reduce the imbalance in fraud labels by generating more labels for ‘fraud’ category using semi-supervised learning techniques such as co-training, Yarowsky algorithm, active learning, etc., and exploring techniques such as SMOTE
- Explore other model algorithms such as k-nearest neighbors, SVM
- Explore grid search CV and other boosting techniques to improve the model performance

Appendix

Card Transaction Data Quality Report

Section 1: High-level Description

The data set contains information related to credit card purchases made in 2006 from a U.S. government organization. The data set comprises ten fields and a total of 96753 records. There are 1059 fraudulent credit card transaction records in the data set which is approximately 1.09 % of the total number of transactions recorded.

Section 2: Summary Tables

Numerical Table

	Mean	Std	Max	Min	% Populated	% Zero
Date	-	-	2006-12-31	2006-01-01	100.00%	0.00%
Amount	$427.89	$10,006.14	$3,102,045.53	$0.01	100.00%	0.00%

Categorical Table

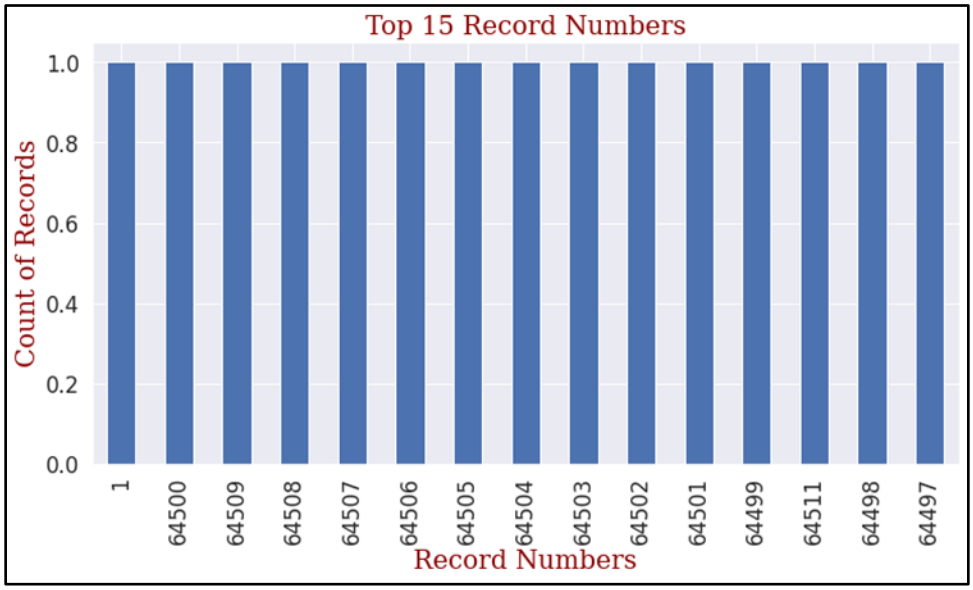
	# Unique Values	Most Common	Most Common % Total	% Populated
Recnum	96753	All Unique	-	100.00%
Cardnum	1645	5142148452	1.23%	100.00%
Merchnum	13092	930090121224	9.62%	96.51%
Merch description	13126	GSA-FSS-ADV	1.74%	100.00%
Merch state	228	TN	12.44%	100.00%
Merch zip	4568	38118	12.27%	95.19%
Transtype	4	P	99.63%	100.00%
Fraud	2	0	98.91%	100.00%

Section 3: Individual Fields

This section will give a brief description of the individual fields in the data set. The fields are listed in the same order they appear in the data set.

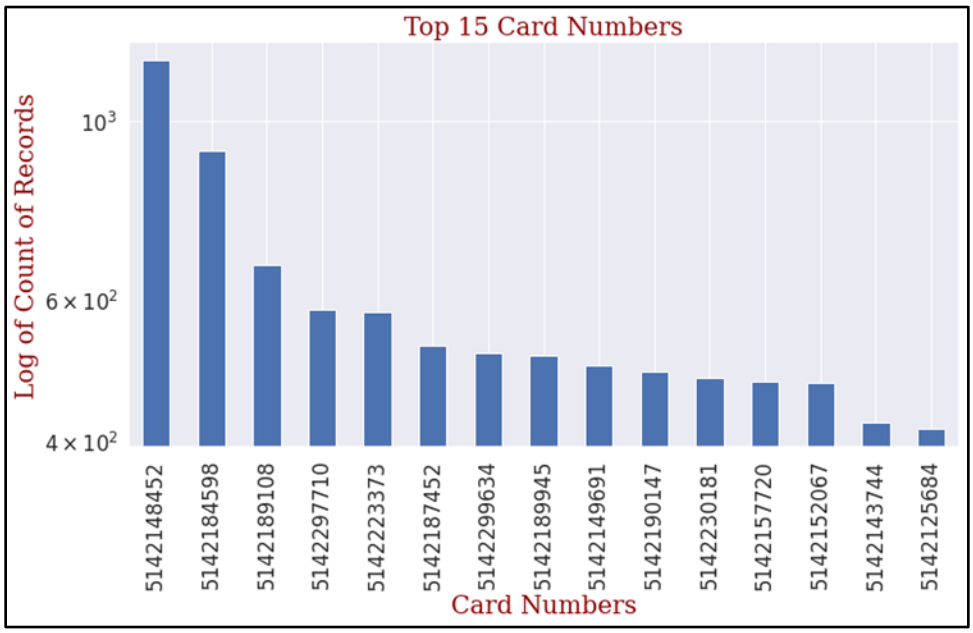
Recnum

This field contains the record numbers for each transaction included in the dataset. The record number is a unique identifier that is assigned to the record in the chronological order of its occurrence.



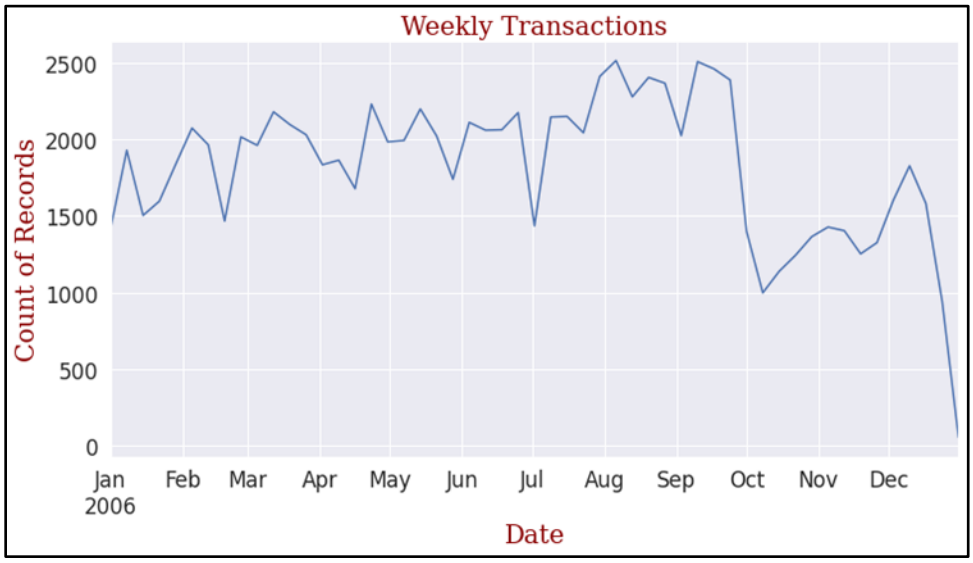
Cardnum

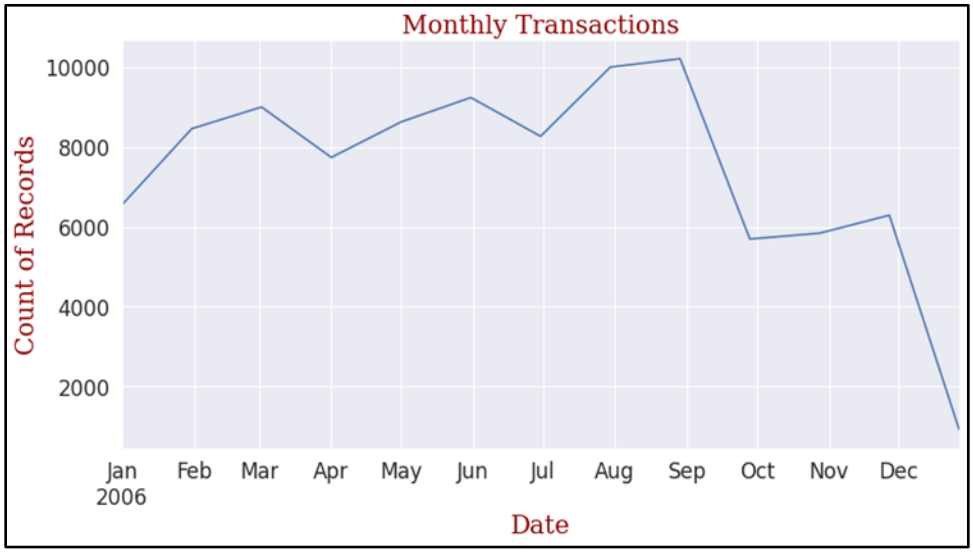
This field contains the card numbers associated with each record of the credit card transactions. The credit card number for the card with the most transactions is ‘5142148452‘ which accounts for approximately 1.23% of the total number of records.

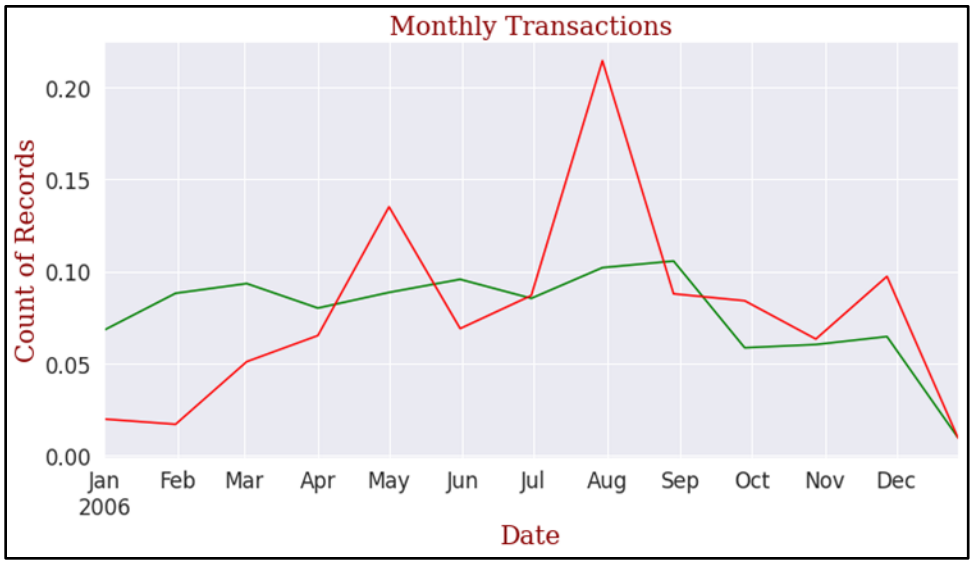


Date

This field contains the date of the recorded credit card transactions. The graphs below showcase the number of transactions on a weekly (7 days) and monthly(30 days) basis. The green color indicates good or legitimate credit card transactions while red indicates bad or fraudulent credit card transactions. There seems to be a seasonal trend for fraudulent transactions that may need to be further investigated.

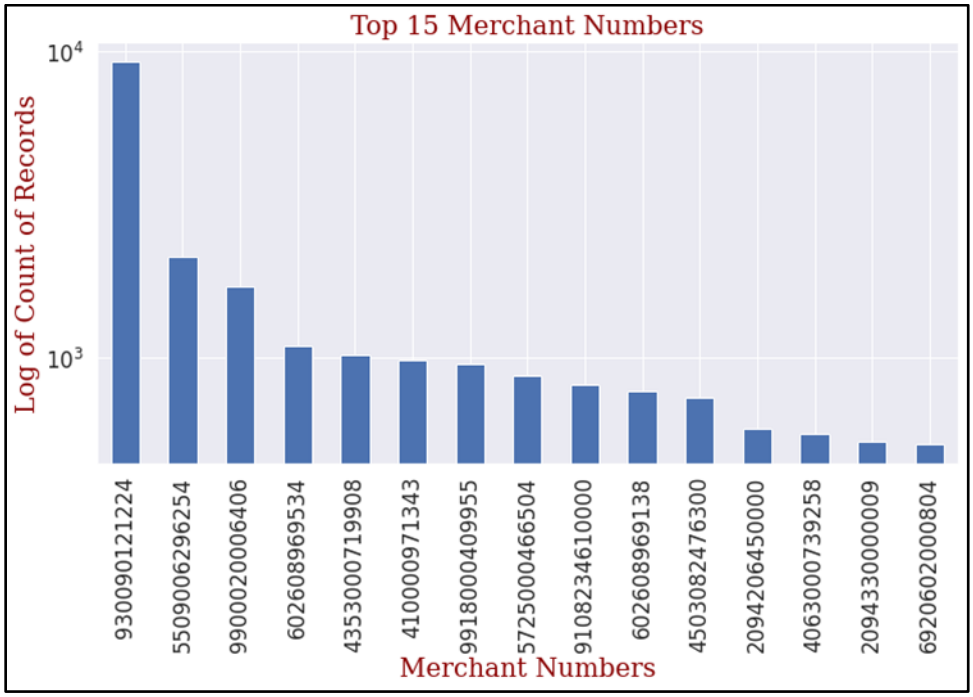






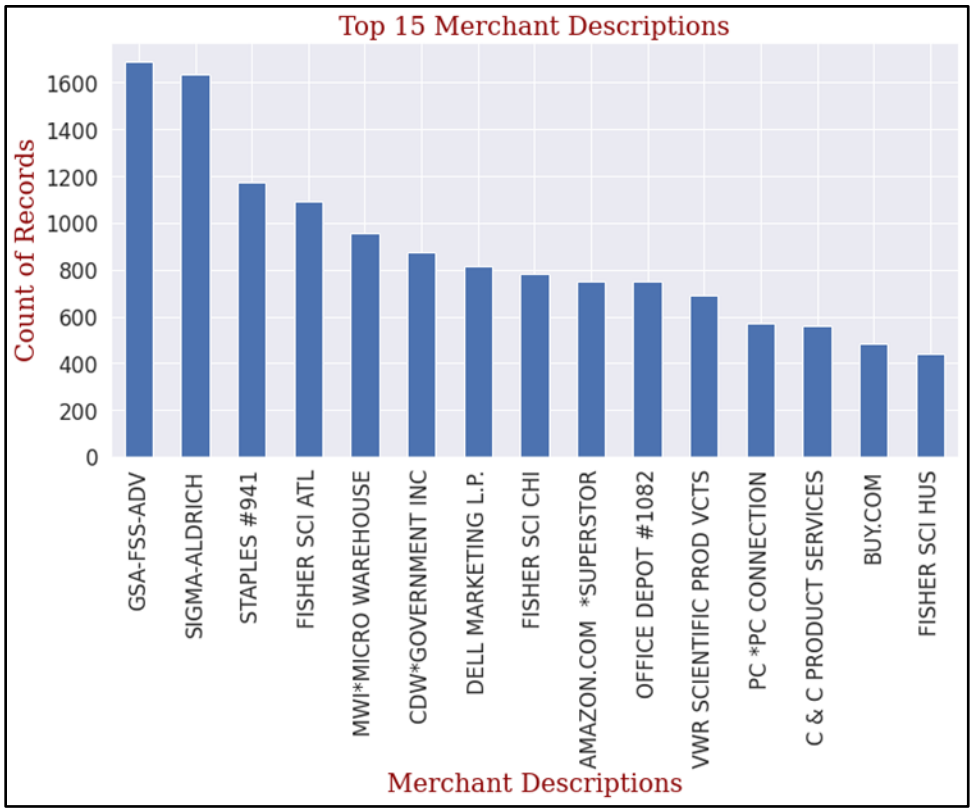
Merchnum

This field contains the merchant number associated with the credit card transactions recorded. The most common merchant number in the data set is '930090121224' which accounts for approximately 9.62% of the total number of records.



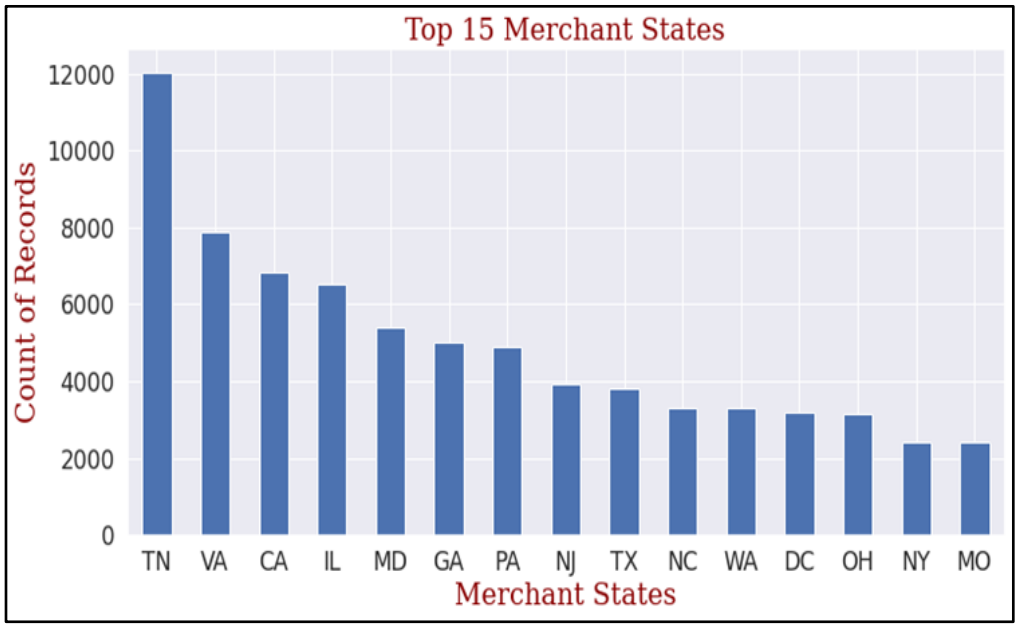
Merch description

This field contains the merchant description associated with the credit card transactions recorded. The most common merchant description in the data set is 'GSA-FSS-ADV ' which accounts for approximately 1.74% of the total number of records.



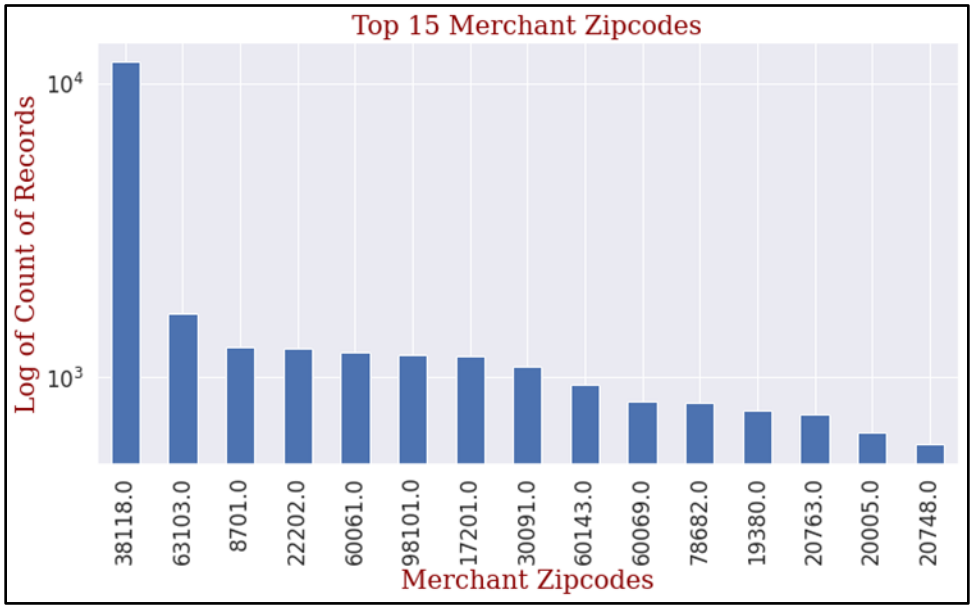
Merch state

This field contains the merchant state associated with the credit card transactions recorded. The most common merchant state in the data set is 'TN' which accounts for approximately 12.44% of the total number of records.



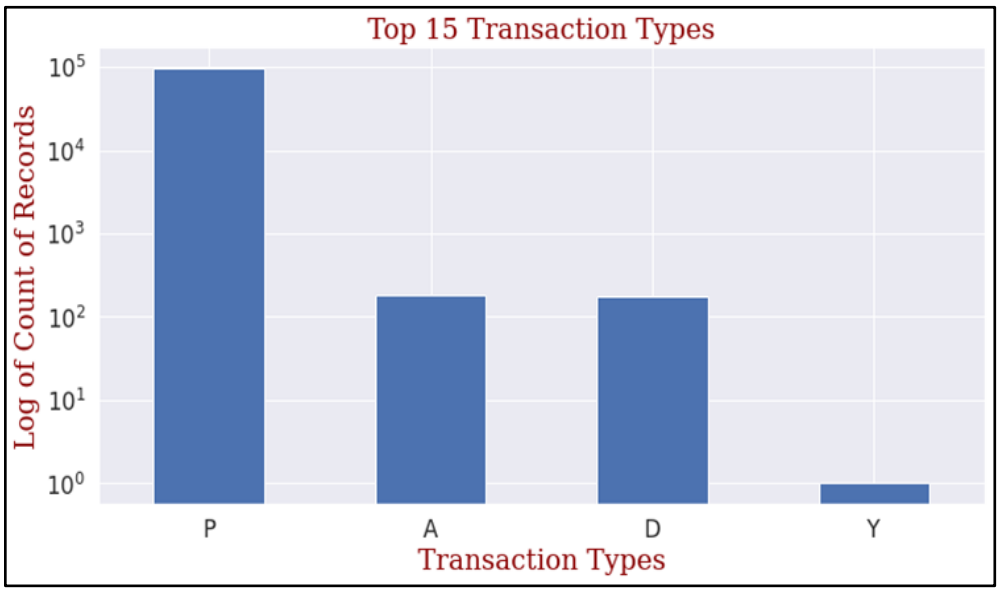
Merch zip

This field contains the merchant zip associated with the credit card transactions recorded. The most common merchant zip in the data set is '38118 ' which accounts for approximately 12.27% of the total number of records.



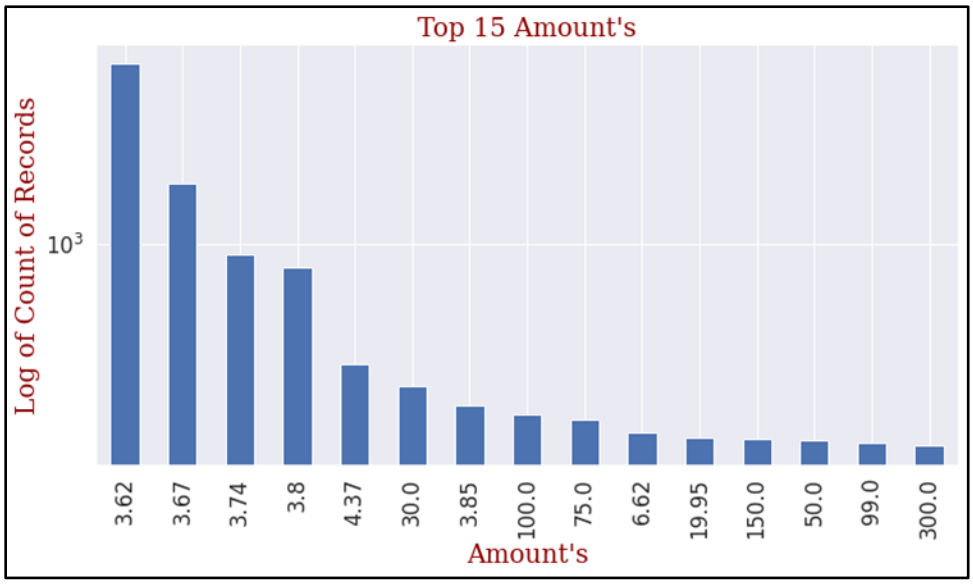
Transtype

This field contains the transaction type of the credit card transactions recorded. The most common transaction type in the data set is 'P' which accounts for approximately 99.63% of the total number of records.



Amount

This field contains the dollar amount of the credit card transactions recorded. The most common dollar amount in the data set is '$3.62'.



Fraud

This field contains the fraud associated with the credit card transactions recorded. '0' indicates that the transaction is legitimate and '1' indicates that the transaction is fraudulent. 1059 transactions are fraudulent which is approximately 1.09 % of the total number of transactions recorded.