# Credit Card Transaction Fraud

## Index

# Executive Summary

This project aims to develop a supervised machine-learning model that can detect and predict credit card transaction fraud. Credit card transaction fraud is a type of fraud where someone illegally uses another person's credit card information to make unauthorized purchases or cash advances. This can result in significant financial losses for both the cardholder and the card issuer and damage to the victim's credit score. The prevalence of credit card transaction fraud has increased with the growing use of online shopping and the widespread use of credit cards for payment transactions.

To achieve our goal, we followed a process consisting of data cleaning, variable creation, feature selection, and modelling. We examined the dataset and removed any inaccuracies, generated new and insightful features from the existing data, selected the most relevant features from the dataset, and built and tested multiple machine learning models, including Logistic Regression, Random Forest, Light GBM, and Neural Network.
This report aims to document the process of creating and completing a supervised machine-learning algorithm that can detect and prevent fraud in real-time.

- **Data Cleaning:** Examine the dataset and remove any inaccuracies

- **Variable Creation**: Generate new and insightful features from the existing data

- **Feature Selection**: Select the most relevant features from the dataset.

- **Modeling**: Build and test multiple machine learning models, to determine the most efficient and effective model for detecting and predicting fraud in real-time.

**Conclusion**: After testing different algorithms, including Logistic Regression, Random Forest, Light GBM, and Neural Network, we experimented with various hyperparameter combinations for each model and compared their performance based on **FDR at 3%**.

Our machine learning model has demonstrated that we are not overfitting and achieving good performance. **By rejecting the top 3% of credit card transaction applications, we are able to catch 55.63% of all fraud cases.** Furthermore, the dollar savings calculated at this cut-off point for the model are **estimated to be around 21 million dollars annually**, indicating that our model has the potential to prevent significant financial losses due to credit card transaction fraud.

# Description of the data

1. **Data Description**

   The dataset consists of 96,753 records and 10 distinct fields, documenting real card payments made between January 1, 2010, and December 31, 2010. Each record in this dataset captures essential transaction details such as card numbers, transaction dates, merchant numbers, descriptions, states, zip codes, payment amounts in US dollars, and fraud scores.

**Summary Table**

**Numerical Table**

| Field Name | % Populated | Min | Max | Mean | Stdev | %Zero |
|---|---|---|---|---|---|---|
| Date | 100.00 | 2010-01-01 | 2010-12-31 | NA | NA | 0.00 |
| Amount | 100.00 | 0.01 | 3102045.53 | 427.88 | 10006.140 | 0.00 |

*Table 1: Numerical Table*

**Categorical Table**

| Field Name | % Populated | Blank | Zeroes | #Unique Value | Most Common Value |
|---|---|---|---|---|---|
| Recnum | 100.00 | 0 | 0 | 96,753 | 1 |
| Cardnum | 100.00 | 0 | 0 | 1,645 | 5142148452.0 |
| Merchnum | 96.51 | 3375 | 231 | 13,091 | 930090121224 |
| Merch Description | 100.00 | 0 | 0 | 13,126 | GSA-FSS-ADV |
| Merch state | 98.76 | 1195 | 0 | 227 | TN |
| Merch Zip | 95.19 | 4656 | 0 | 4567 | 38118 |
| Transtype | 100.00 | 0 | 0 | 4 | P |
| Fraud | 100.00 | 0 | 0 | 2 | 0.0 |

*Table 2: Categorical Table*

**Visualization of Each Field:**

1. Field Name: Recnum
   Description: Recnum: Ordinal Unique positive integer for each record from 1 to 96753.

2. .Field Name: Date
   Description: Date: The “date” field indicates the date on which the individual has raised a request for an application. The Weekly use distribution across time.

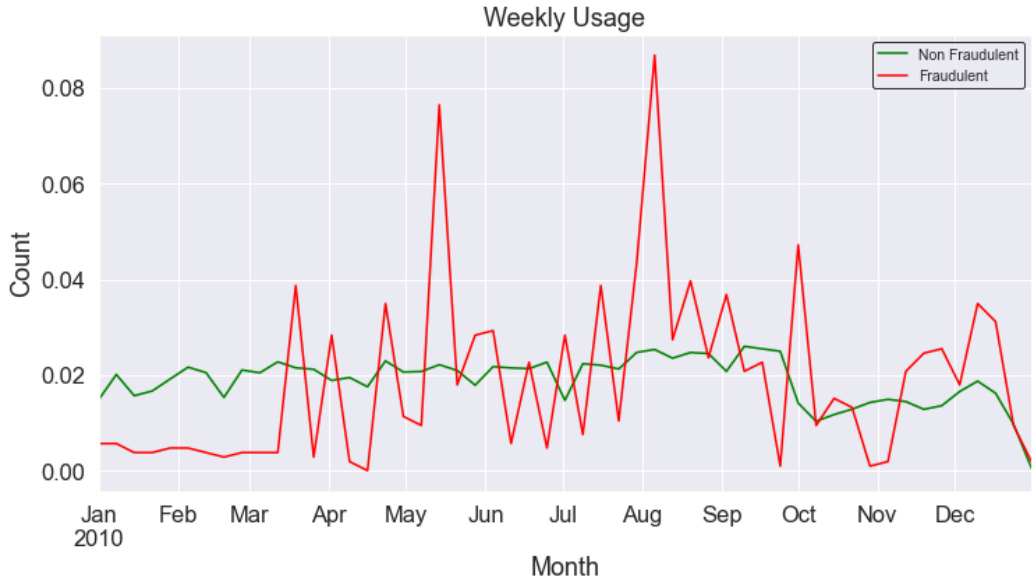



Fig 1. Weekly distribution of Transaction

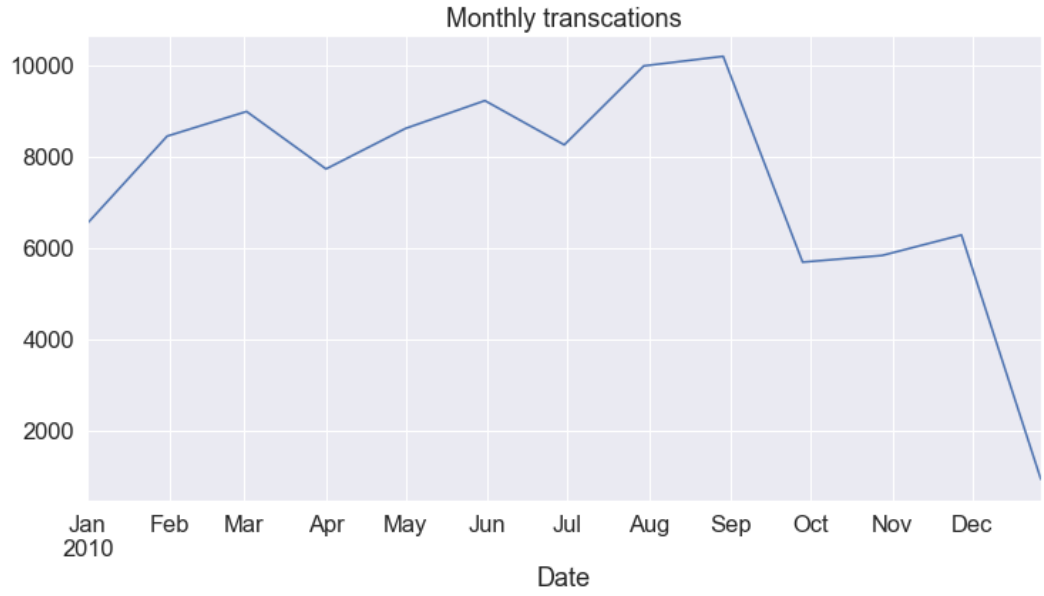



Fig 2. Monthly distribution of Transaction

3. Field Name: Cardnum
   Description: SSN: SSN refers to the Card number of the user. The distribution indicates the top 20 most common card users.
   The most common value of the 5142148452, the count is 1192.

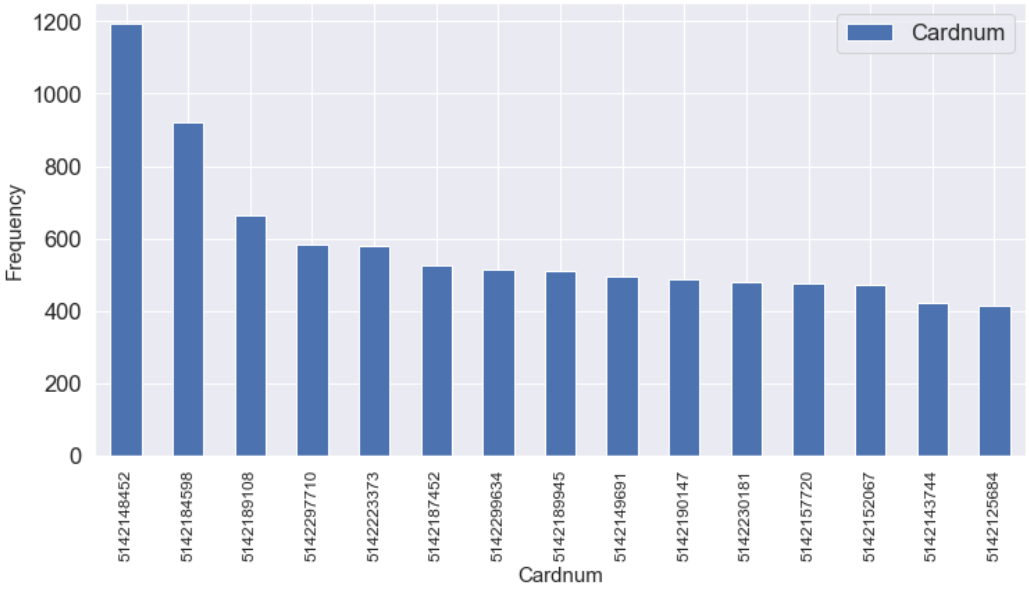



Fig 3. Bar plot displaying distribution of field 'Cardnum'

4. Field Name: Merchnum
   Description: Merchnum: The field "Merchnum" refers to the number of the Merchant. The distribution indicates the top 15 most common Merchant number used by the user.
   The most common value of the Merch num is 930090121224, the count is 9310.

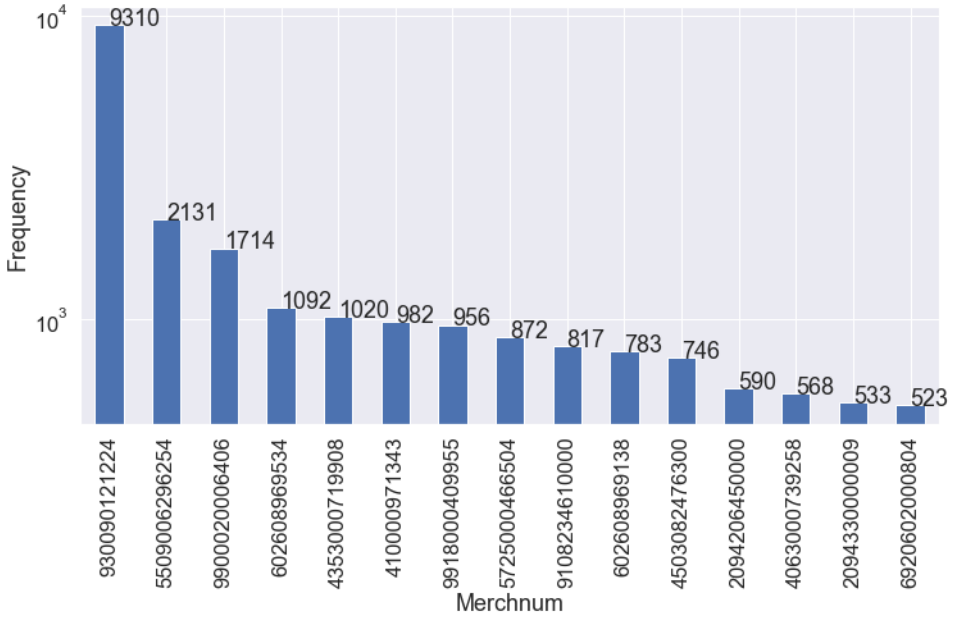



Fig 4. Bar plot displaying distribution of field 'Merchnum'

1. Field Name: Merch description
   Description: Merch description: The field “Merch description” refers to description of the merchant. The distribution indicates the top 15 most common Merchant used by the applicant.
   The most common value of the Merch Description is GSA-FSS-ADV, the count is 1688.

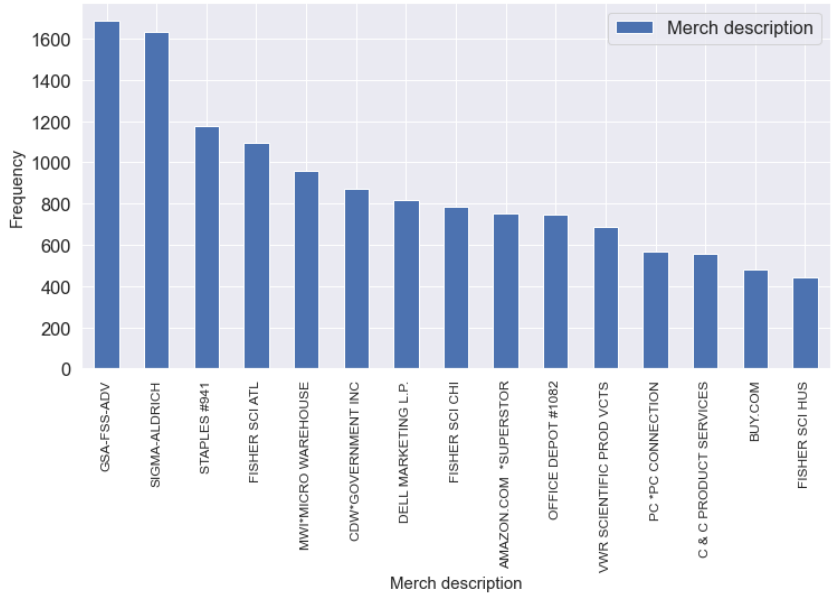



Fig 5. Bar plot displaying distribution of field ‘Merch description’

2. Field name: Merch state
   Description: Address: The field “Merch state” refers to the state of the merchant. The distribution indicates the top 15 most common merchant state used.
   The most common value of the Merch state is TN, the count is 12,35.

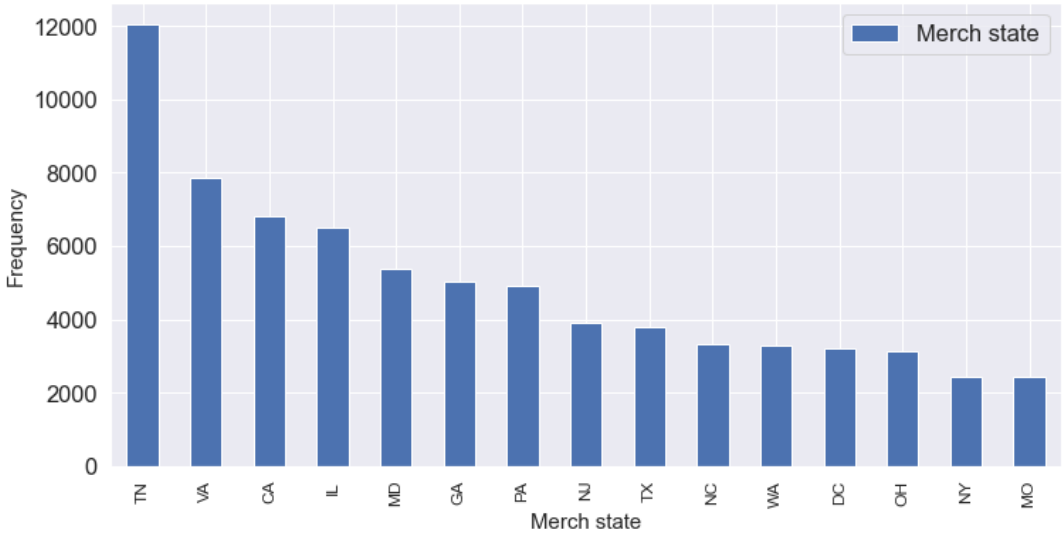



Fig 6. Bar plot displaying distribution of field ‘Merch state’

3. Field Name: Merch Zip
   Description: Merch Zip: The field “Merch Zip” refers to the Zip code of the user. The distribution indicates the top 15 most common Zip code used.
   The most common value of the Zip code is 38,118, the count is 11868.

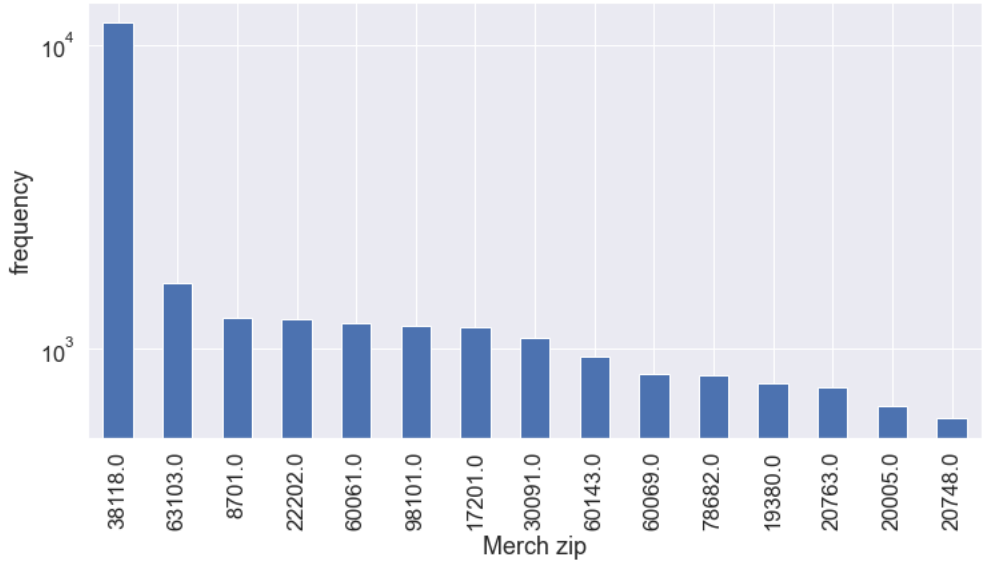



Fig 7. Bar plot displaying distribution of field ‘Merch zip’

4. Field Name: Transtype
   Description: Transtype: The field “Transtype” refers to the type of transaction. The distribution indicates the 4 types of transactions where P means Purchase, A stands for Approval, D stands for Debit and Y stands for Year-end.
   The most type of value of the transaction is P, the count is 96,398.

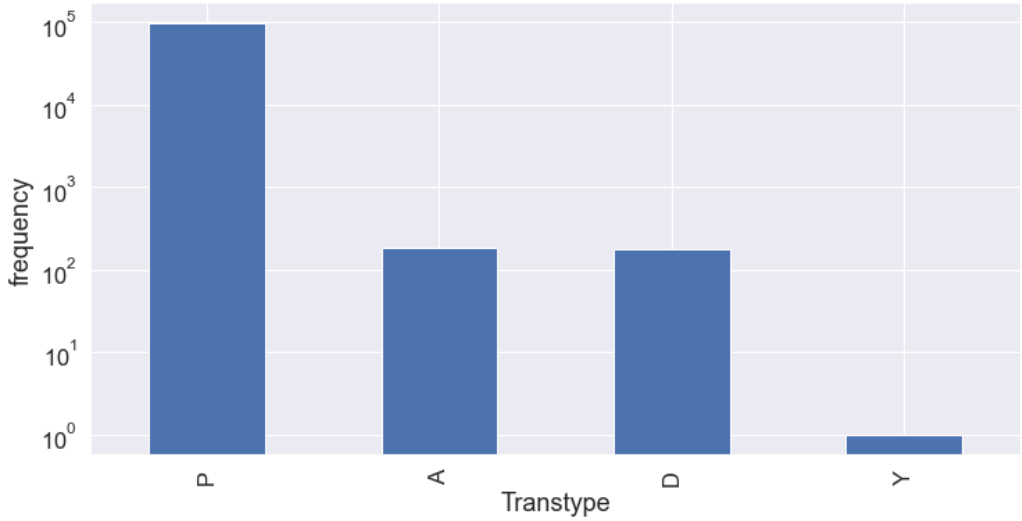



Fig 8. Bar plot displaying distribution of field ‘Transtype’

9.Field Name: Amount

Description: Amount: The field “Amount” refers to the Transaction Amount. The distribution indicates the top 10 most common transaction amount. The most common value of the Amount is 3.62, the count is 4,283.

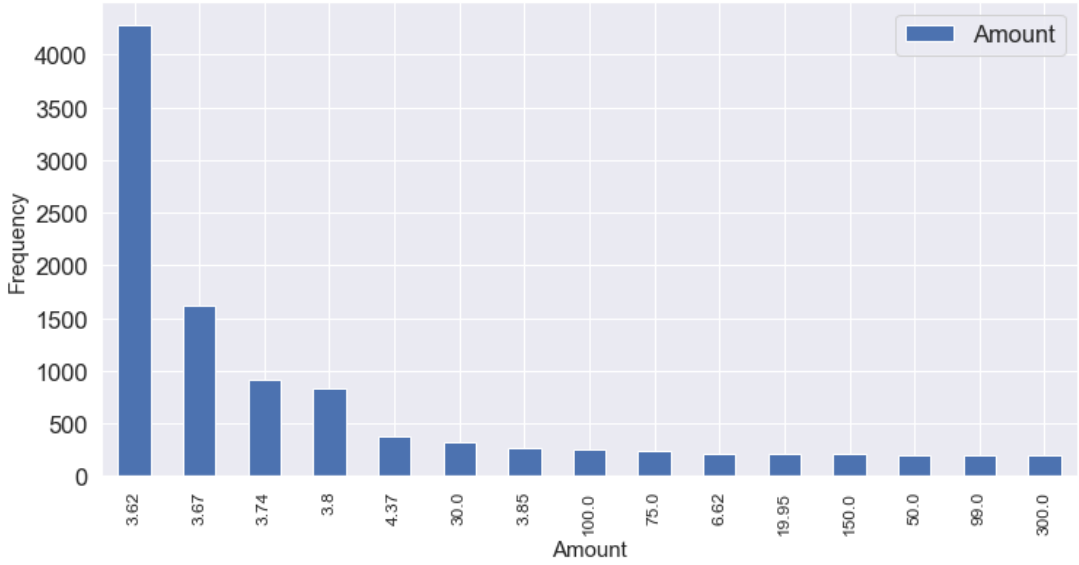



Fig 9. Bar plot displaying distribution of field ‘Amount’

10.Field Name: Fraud

Description: Fraud = 0 (no fraud label), Fraud=1 (fraud label)

The count of Fraud=0 (95,694) and Fraud =1 (1,059)


Fig 10. Bar plot displaying distribution of field ‘Fraud’

# Data Cleaning

After Exploratory data analysis, we found that data cleaning need to be performed before feature engineering, to remove outliers and fill missing values.

**Data Imputation Logic followed:**

**Transtype:** We are only keeping records that have "P" (Purchase) value for Transtype data field, because this transaction type is more relevant to our project for detecting Fraud. ( Result: **excluding 355** records that contained "A", "D", "D", "Y" entries in Transtype data field). Now our dataset consists of **96,398 records**.

**Amount:** The record with Recnum value of **'52715'** contained **Amount** value that was discovered with to be annotated in Mexican Peso, and its value was **3,102,045.53** . I excluded this this record because it can cause issues in our analysis. Now our dataset consists of **96,398 records**.

**Merch state:** This field had **1020 records** with a missing value, I started cleaning this field as this has a smaller number of missing value than the other two. ("Merch zip", Merchnum)

a) I used free database which has **33,790 US zip** codes provided by SimpleMaps.com available on https://simplemaps.com/data/us-zips **,** To form dictionary to map (zip code as key state_id as values) in zip_dict then mapped this dictionary to "Merch zip" data field to fill missing values in in corresponding "Merch state" field records , which reduced missing count to **980.**
b) Filter for non-null 'Merch zip' values then form merch_zip_dict which had merch zip as key and and most common "Merch state" values as value for each group after performing group by "Merch zip". Then mapped this to "Merch state field" and filled the missing value with most common mech state value after group by "Merch zip". As I was keeping not null "Merch

zip" as key in mask object so mapping original "Merch state" did not imposed problem. This step reduced count of missing values to **963 records.**

c) I filled the remaining missing values with the most common values after group by first by Cardnum then by Date, using unique combination of 'Cardnum' 'Date' field to fill the missing value with most common value. This step reduced count of missing values to **636 records**.
d) I filled the remaining missing values with the most common values after group by "Merch description". This step reduced count of missing values to **134 records**.
e) I filled the remaining missing values with the most common values after group by "Cardnum". This step reduced count of missing values to **31 records**.
f) Filled the remaining missing values with their corresponding "Recnum". Now our "Merch state" column is filled.

**Merch zip:** This field had **4300 records** with a missing value, used following logic to fill the missing value.

a) First selecting not null row of 'Merchnum then merging 'Merchnum', 'Merch state', and 'Merch zip' columns into a new Data Frame merch_not_null. Then, fills missing values in the 'Merch zip' column by grouping 'Merch zip' by unique combinations of 'Merchnum' and 'Merch state' and filling missing values with the mode of each group. This step reduced count of missing values to **2474 records**.
b) I filled the remaining missing values with the most common values after group by first by Cardnum then by Date, using unique combination of 'Cardnum' 'Date' field to fill the missing value with most common value. This step reduced count of missing values to **1628 records**.
c) I filled the remaining missing values with the most common values after group by first by "Merch Description" then by "Merch state", using unique combination of 'Merch Description' 'Merch state' field to fill the missing value with most common value. This step reduced count of missing values to **353 records.**
d) I filled the remaining missing values with the most common values after group by "Merch state". This step reduced count of missing values to **31 records**.
e) Filled the remaining missing values with their corresponding "Recnum". Now our "Merch zip" column is filled.

**Merchnum:** This field had **3198 records** with a missing value, and also this filed had 53 records with noise value as **“0”** as “Merchnum” , used following logic to fill the missing value.

a) First, I selected all the records in this column which had 0 “Merchnum” as 0 and replaced with **“NaN”.**
b) Now, total records which has missing values is **3251.**
c) I filled the remaining missing values with the most common values after group by first by ‘Merch description' then by 'Merch zip’ using their unique combination to fill the missing value with most common value. This step reduced count of missing values to **2177 records**.
d) Filled the remaining missing values with the most common values after group by first by ‘Merch description' then by 'Merch state’ using their unique combination to fill the missing value with most common value. This step reduced count of missing values to **2109 records.**
e) I filled the remaining missing values with the most common values after group by “Merch state”. This step reduced count of missing values to **48 records.**
f) Filled the remaining missing values with their corresponding **“Recnum”**. Now our “Merchum” column is filled.

# Variable Creation

It is the process of selecting, transforming, and creating variables from raw data to improve the performance of machine learning models. It involves steps like data cleaning, transformation, and selection of relevant features**.**

| Description of variables | #Variables |
|---|---|
| Original fields from the dataset excluding 'record' and 'fraudl' | 8 |
| Target Variable | 1 |
| Record Variable | 1 |
| New entities created: dow(day of the week),check_date, check_record | 3 |
| New entities combining/concatenating different original fields | 12 |

| | |
|---|---|
| **Velocity and Days since Variable**:<br>**Velocity Variable:** #records with same entity over last {0,1,7,14,30.60} days. Example: Cardnum_days_since<br><br>**Frequency and amount variable:** For various time periods (0, 1, 3, 7, 14, 30, and 60 days), counting the number of rows, calculating the mean, maximum, median, and sum of the 'Amount' | 687 |
| **Velocity change:** Speed at which the transaction behavior of each entity is changing over time, by looking at the relative difference in transaction count and total amount between two different time periods | 170 |
| **Vdratio:** The ratio of velocity change variables and the number of days since the last transaction for each entity. | 24 |
| **Nunique:** calculates the number of unique values in the field of the second entity for each value of the first entity | 109 |
| **Relative velocity:** Speed at which an entity is seen in the dataset for a particular transaction over a short period of time (0-1) days in relation to how often the same entity is seen over a longer period.<br>Example: A_total_amount_0_by_7 | 8 |
| **Variablilty:** Calculate the difference between the transaction amount and the check amount for each record | 211 |
| Benford variable(excluding them) | 16 |
| Total | 1233 |

# Feature selection

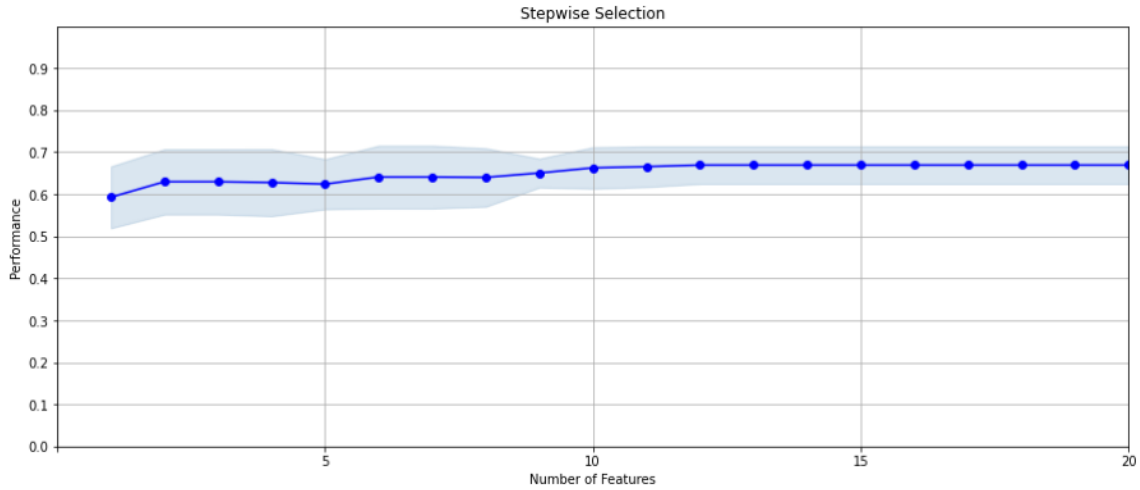
**Difference between forward and backward selection**

Forward selection starts with a (usually empty) set of variables and adds variables to it, until some stop- ping criterion is met. Similarly, backward selection starts with a (usually complete) set of variables and then excludes variables from that set, again, until some stopping criterion is met.

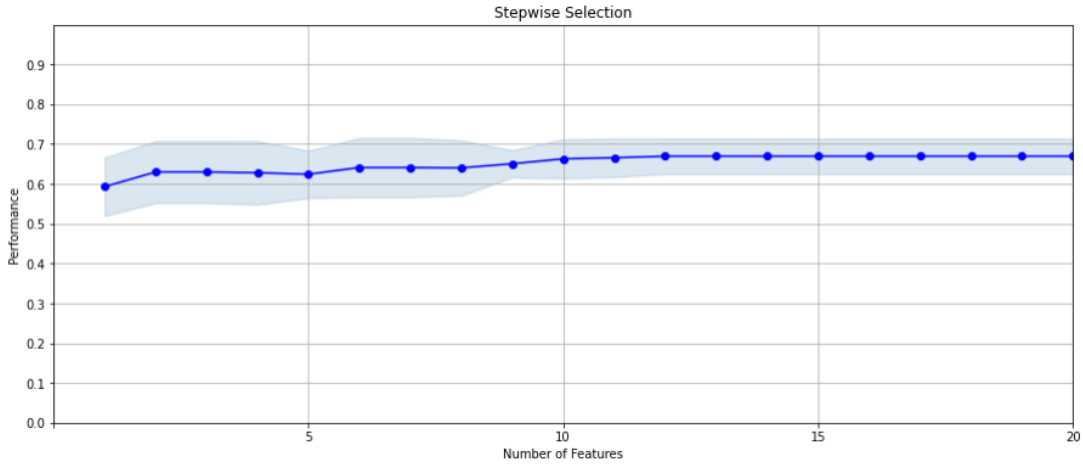
Tried different combinations of wrapper and filter combination using Forward and Backward Selection to identify the top 20 variables of the wrapper result.

Proving Stochastic and Non-Stochastic Nature of LGBM (Gradient Boosting) and Random
Forest (RF) using Forward Selection.
LGBM
First Attempt

Num_filter :- 150 , Num_wrapper :- 20




Since LGBM has no stochastic nature hence when we use this for wrapper model ; we will
always get the same result.
Second Attempt



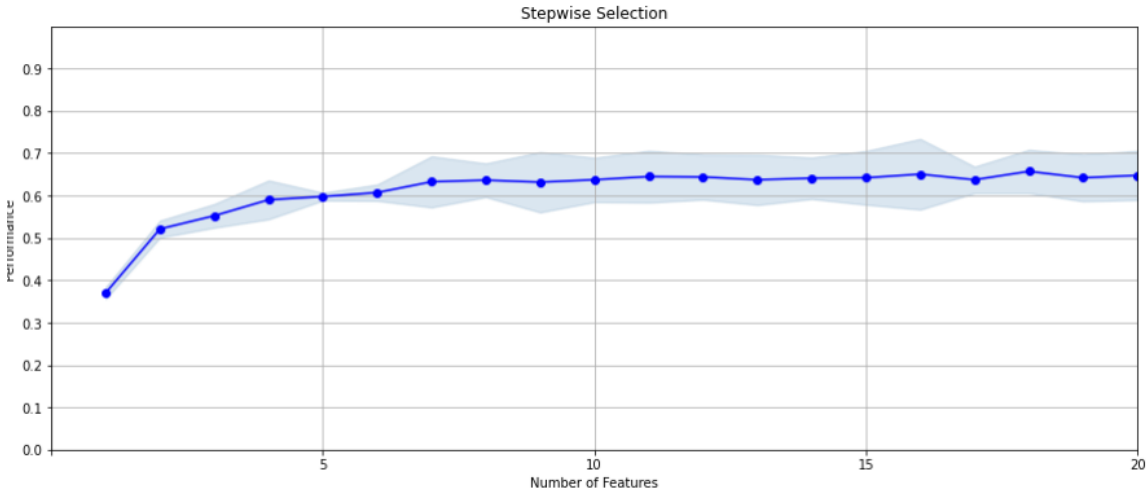
| First Attempt | |
|---|---|
| add variables in this order | variable name |
| 1.0 | card_merch_total_14 |
| 2.0 | Card_Merchnum_desc_max_60 |
| 3.0 | card_merch_total_30 |
| 4.0 | Merchnum_desc_total_7 |
| 5.0 | card_zip3_total_30 |
| 6.0 | Cardnum_avg_1 |
| 7.0 | Card_Merchnum_desc_total_30 |
| 8.0 | Card_Merchdesc_total_3 |
| 9.0 | card_zip3_max_30 |
| 10.0 | Card_Merchdesc_total_7 |
| 11.0 | Cardnum_max_0 |
| 12.0 | card_merch_max_60 |
| 13.0 | Card_Merchnum_desc_total_14 |
| 14.0 | Card_Merchdesc_total_30 |
| 15.0 | card_merch_max_30 |
| 16.0 | Card_Merchdesc_max_14 |
| 17.0 | Card_Merchdesc_max_30 |
| 18.0 | Card_Merchnum_desc_max_14 |
| 19.0 | card_merch_total_60 |
| 20.0 | Card_Merchnum_desc_max_30 |

| Second Attempt | |
|---|---|
| add variables in this order | variable name |
| 1.0 | card_merch_total_14 |
| 2.0 | Card_Merchnum_desc_max_60 |
| 3.0 | card_merch_total_30 |
| 4.0 | Merchnum_desc_total_7 |
| 5.0 | card_zip3_total_30 |
| 6.0 | Cardnum_avg_1 |
| 7.0 | Card_Merchnum_desc_total_30 |
| 8.0 | Card_Merchdesc_total_3 |
| 9.0 | card_zip3_max_30 |
| 10.0 | Card_Merchdesc_total_7 |
| 11.0 | Cardnum_max_0 |
| 12.0 | card_merch_max_60 |
| 13.0 | Card_Merchnum_desc_total_14 |
| 14.0 | Card_Merchdesc_total_30 |
| 15.0 | card_merch_max_30 |
| 16.0 | Card_Merchdesc_max_14 |
| 17.0 | Card_Merchdesc_max_30 |
| 18.0 | Card_Merchnum_desc_max_14 |
| 19.0 | card_merch_total_60 |
| 20.0 | Card_Merchnum_desc_max_30 |

Random Forest (RF)

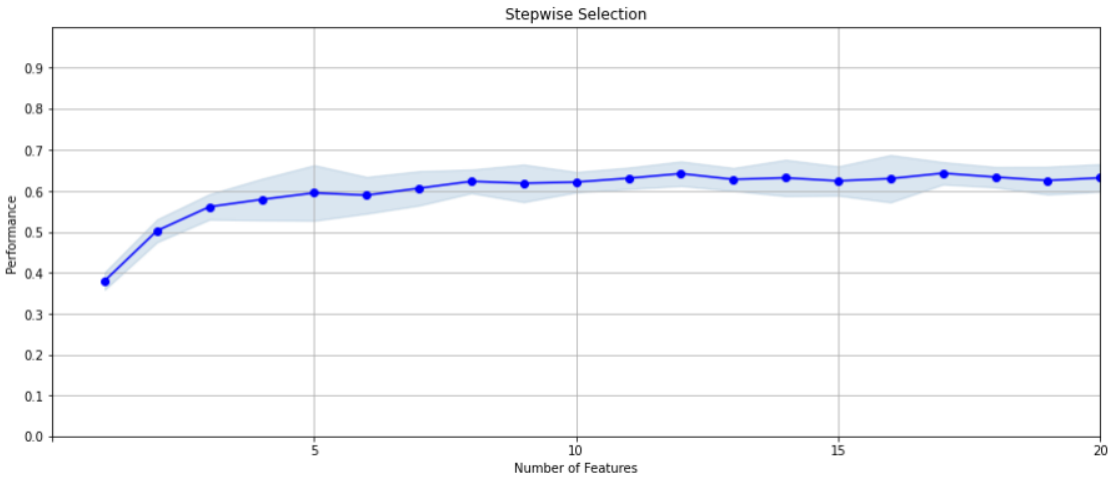
First Attempt

Num_filter = 150, Num_Wrapper = 20




Second Attempt

(Num_filter= 150, Num_wrapper= 20 )

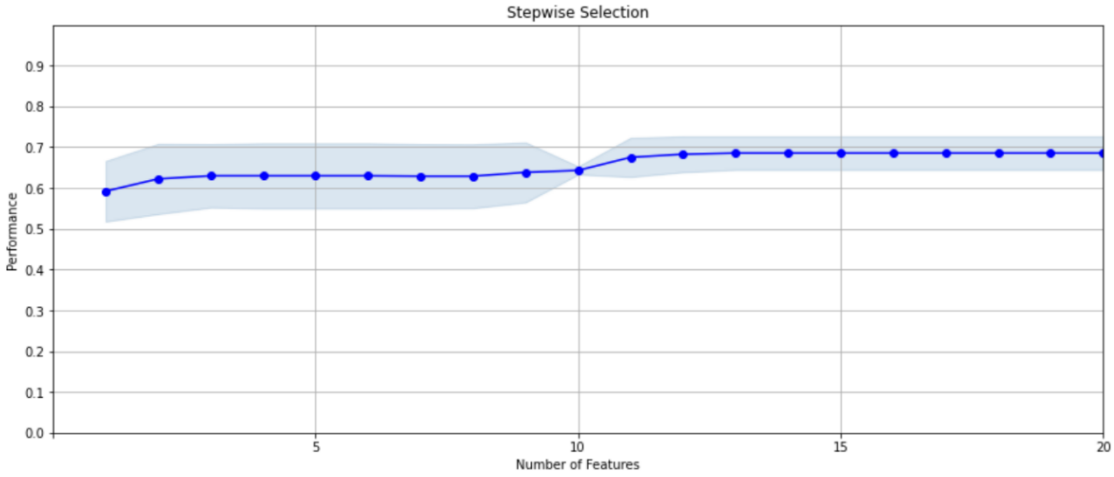



Since RF has a stochastic nature hence when we use this for wrapper model ; we will always
get different results.

| First Attempt | | Second attempt | |
|---|---|---|---|
| **add variables in this order** | **variable name** | **add variables in this order** | **variable name** |
| 1.0 | card_zip3_total_7 | 1.0 | card_zip3_total_3 |
| 2.0 | Card_Merchdesc_max_60 | 2.0 | Card_Merchdesc_max_30 |
| 3.0 | Card_Merchnum_desc_total_14 | 3.0 | Card_Merchdesc_total_30 |
| 4.0 | Merchnum_desc_total_1 | 4.0 | Merchnum_total_0 |
| 5.0 | Cardnum_total_3 | 5.0 | card_zip_total_30 |
| 6.0 | card_zip_avg_7 | 6.0 | card_merch_total_0 |
| 7.0 | zip3_actual/avg_60 | 7.0 | card_zip_max_60 |
| 8.0 | Card_Merchnum_desc_total_30 | 8.0 | Cardnum_total_14 |
| 9.0 | card_zip3_max_3 | 9.0 | card_zip3_max_14 |
| 10.0 | Cardnum_max_1 | 10.0 | card_zip3_avg_3 |
| 11.0 | Card_Merchdesc_avg_7 | 11.0 | Card_Merchdesc_avg_1 |
| 12.0 | card_merch_max_14 | 12.0 | card_merch_total_14 |
| 13.0 | Card_Merchnum_desc_max_60 | 13.0 | card_zip3_max_7 |
| 14.0 | card_merch_med_30 | 14.0 | Card_Merchdesc_total_0 |
| 15.0 | Card_Merchnum_desc_max_0 | 15.0 | Cardnum_avg_3 |
| 16.0 | Merchnum_desc_total_3 | 16.0 | card_merch_max_0 |
| 17.0 | card_zip_total_3 | 17.0 | Card_Merchdesc_max_3 |
| 18.0 | card_merch_total_60 | 18.0 | Card_Merchdesc_total_14 |
| 19.0 | Card_Merchnum_desc_max_3 | 19.0 | card_zip3_avg_0 |
| 20.0 | card_merch_max_30 | 20.0 | Card_Merchnum_desc_total_7 |

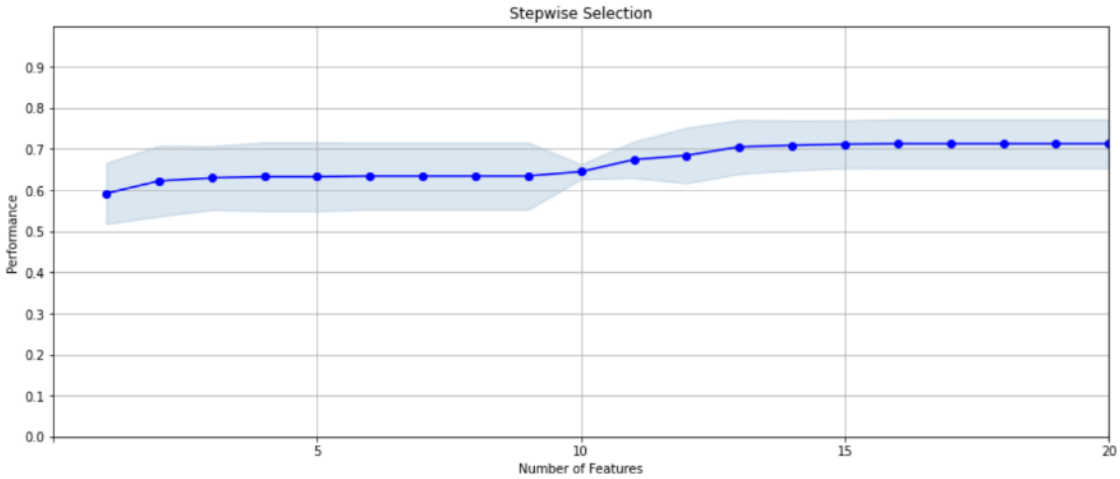
Other Trial Runs

LGBM(n_estimators=20,num_leaves=3,Num_filter :- 200,
num_wrapper :- 20 )

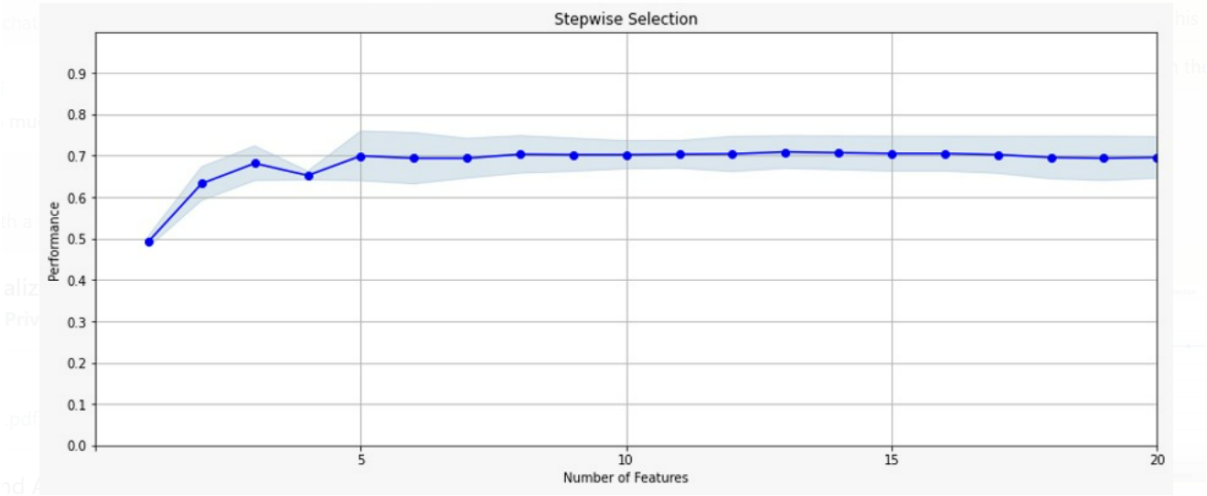



LGBM

Num_filter :- 350 num_wrapper :- 20




BACKWARD SELECTION
LGBM (Num_filter= 150, Num_wrapper= 20)




Backward Selection is slower and problematic
Decided to go with the wrapper result of LGBM Num_filter :- 350
num_wrapper :- 20;
since it has a performance rate of more than 0.70.

| wrapper or | variable | filter score |
|---|---|---|
| 1 | card_merch_state_total_14 | 0.6280605 |
| 2 | card_merch_state_max_60 | 0.5889724 |
| 3 | merchnum_desc_total_7 | 0.5138036 |
| 4 | merch_zip_med_7 | 0.3801237 |
| 5 | card_merch_total_14 | 0.6280395 |
| 6 | merchnum_desc_avg_7 | 0.4538652 |
| 7 | card_merch_max_60 | 0.5889409 |
| 8 | merchnum_desc_total_14 | 0.4909752 |
| 9 | merch_zip_avg_7 | 0.4539831 |
| 10 | card_state_max_14 | 0.6002801 |
| 11 | card_state_total_1 | 0.6075727 |
| 12 | merchnum_desc_avg_60 | 0.3537857 |
| 13 | merchnum_desc_total_1 | 0.5220379 |
| 14 | merchnum_desc_total_0 | 0.499112 |
| 15 | card_merch_state_total_30 | 0.6150971 |
| 16 | card_zip_avg_1 | 0.511248 |
| 17 | card_merch_total_30 | 0.6150552 |
| 18 | card_merch_desc_total_30 | 0.6052308 |
| 19 | card_merch_state_total_60 | 0.5983826 |
| 20 | card_merch_total_60 | 0.5983196 |

# Preliminary model exploration

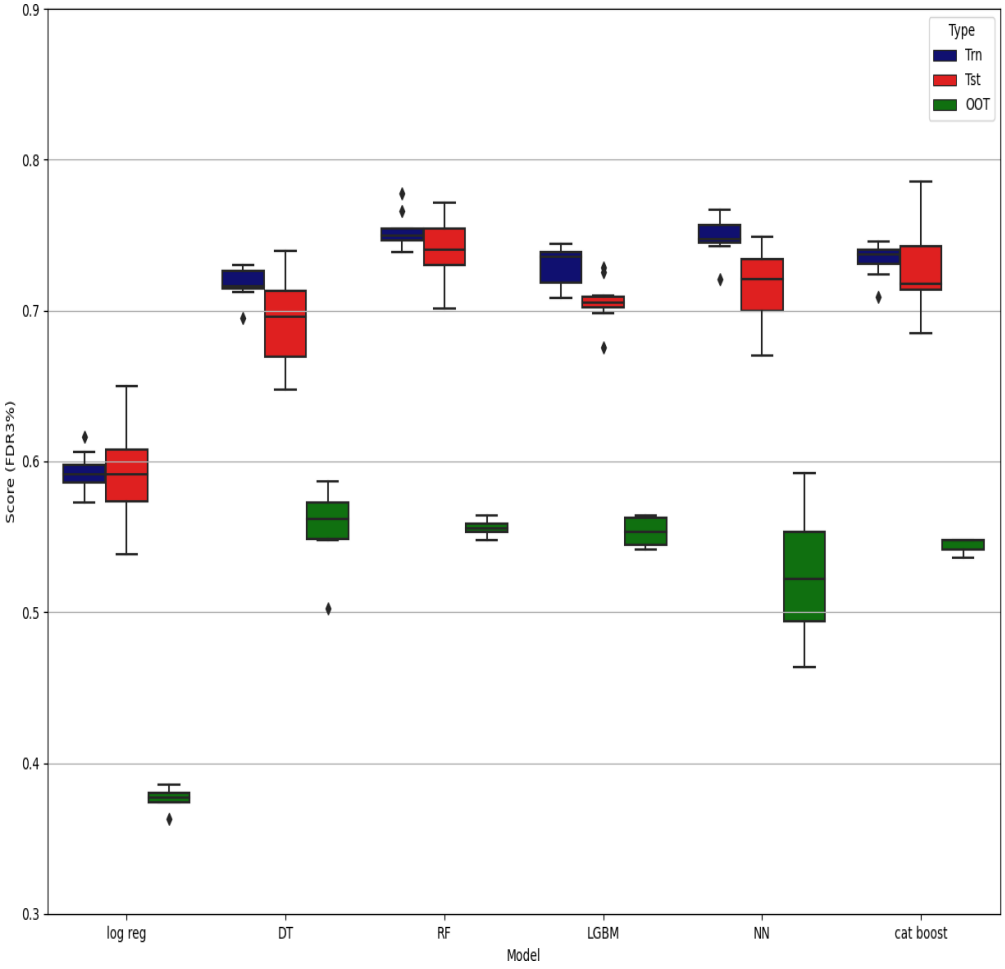
After testing different algorithms to fit the data, such as Logistic Regression, Decision Tree, Random Forest, Catboost, LightGBM, and Neural Network, experimented with various hyperparameters combinations for each model & compared their performance based on FDR at a 3% rejection rate.

**Random forest with parameters: max_depth=8, criterion = 'gini', n_estimators = 80, min_samples_split = 50, min_samples_leaf = 40, max_features = 8, was the best and final model.**

The FDR scores for this model on training, testing, and OOT data were 78.52%, 78.21%, and 55.41%, respectively. (Taking number of variables =10)

Final model shows that we are not overfitting and getting good performance and we can catch 55.63% of all the Fraud by rejecting the top 3% of the application.

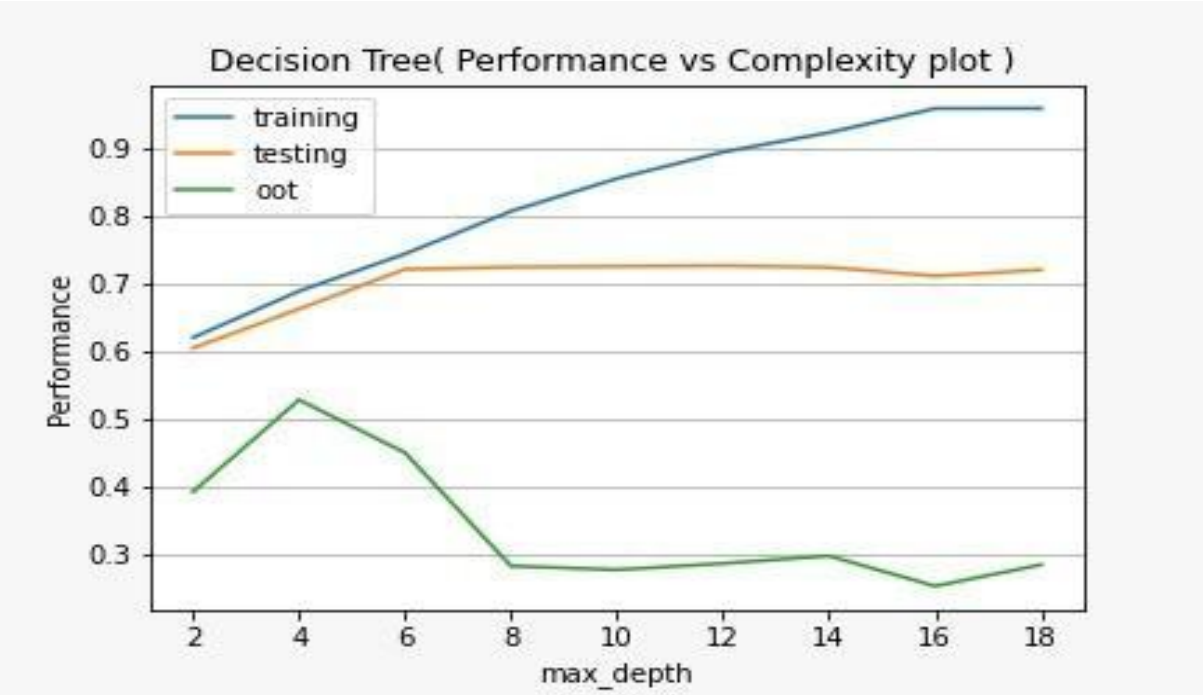
After comparing the results between logistics regression, boosted trees, random forest, and a neural network, we determined that Light Boosting Gradient Model performed the best. Random forest outperformed other models for both training, testing and out of time validation datasets with 78.52%, 78.21% and 55.41% respectively.
The chosen hyperparameters are:
No. of variables: 20
n_estimators: 80
max_depth: 8
min_child_split: 50
minsamples_leaf:40
max_features: 8
Performance Score (FDR @ 3%) of different models with best model parameter




Using different values of hypermeters forcing models to overfit:

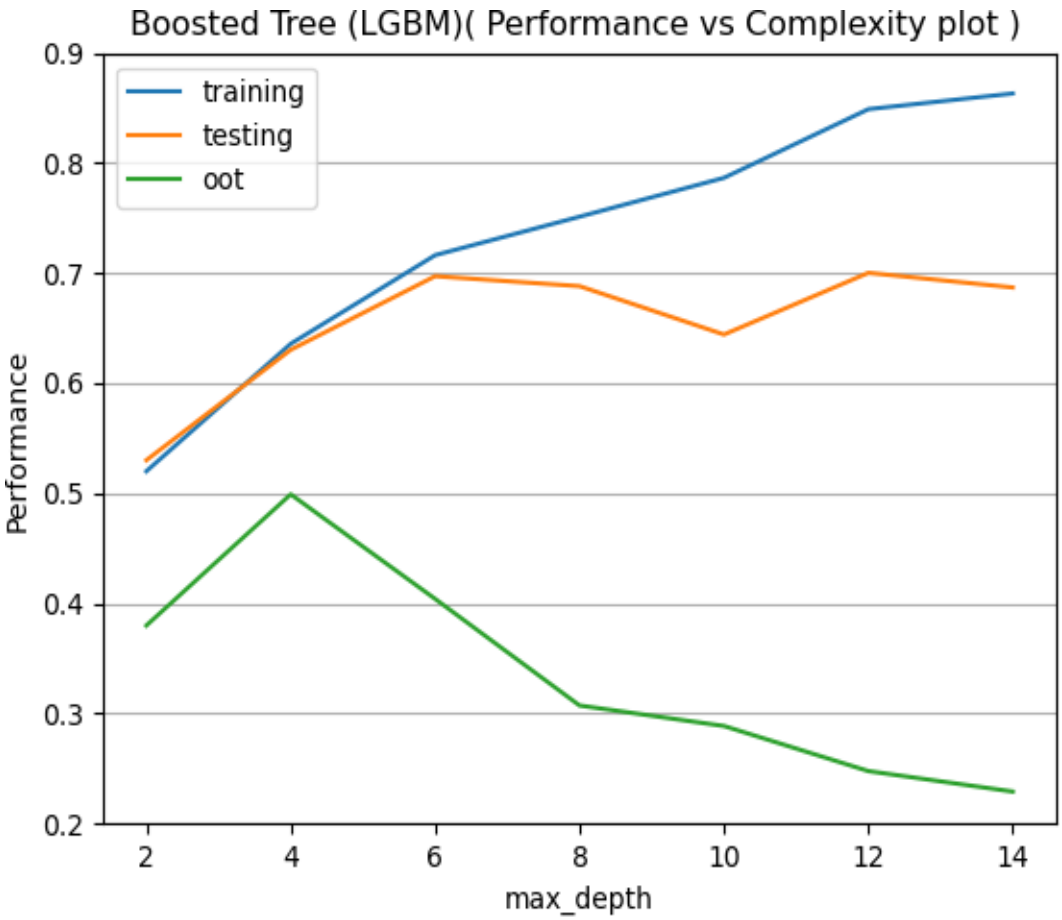
**Single Decision Tree:** In this we are increasing max_depth of DecisionTreeClassifier and making model to be more complex.( Increasing depth of Tree)
nitermax = 3, i in range (2, 20, 2)

model=DecisionTreeClassifier(max_depth=i,criterion='entropy',min_samples_leaf=5,min_samples_split=20)



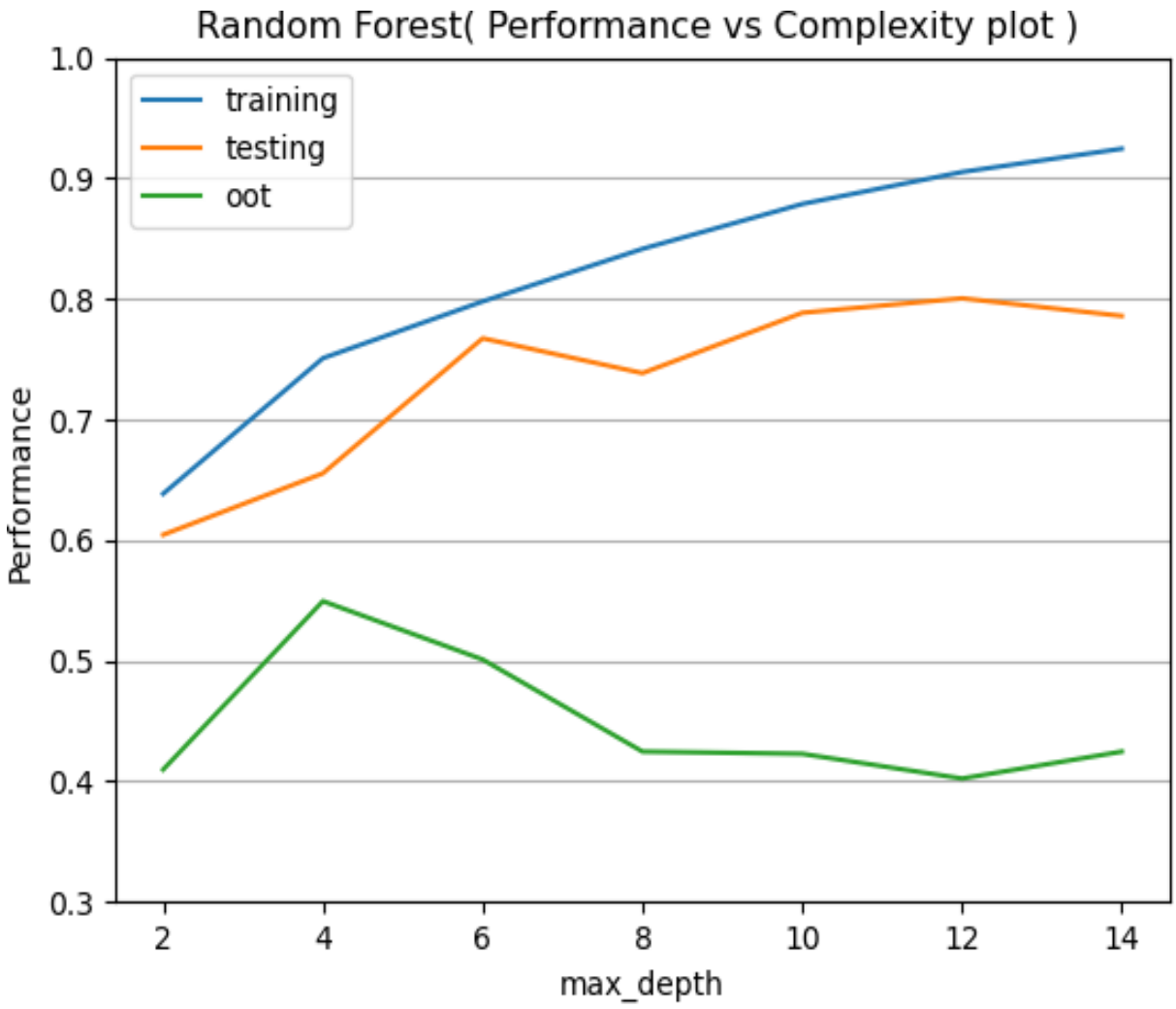
**Boosted Tree (LGBM):** With Each iteration we are increasing max_depth of model making it more complex and the interpretability of the model also decreases as the depth of the tree increases.

nitermax = 3, i in range (2, 16, 2)
model = lgb.LGBMClassifier(max_depth = i , min_child_samples = 10, n_estimators = 600, learning_rate = 0.00001, num_leaves = 7)



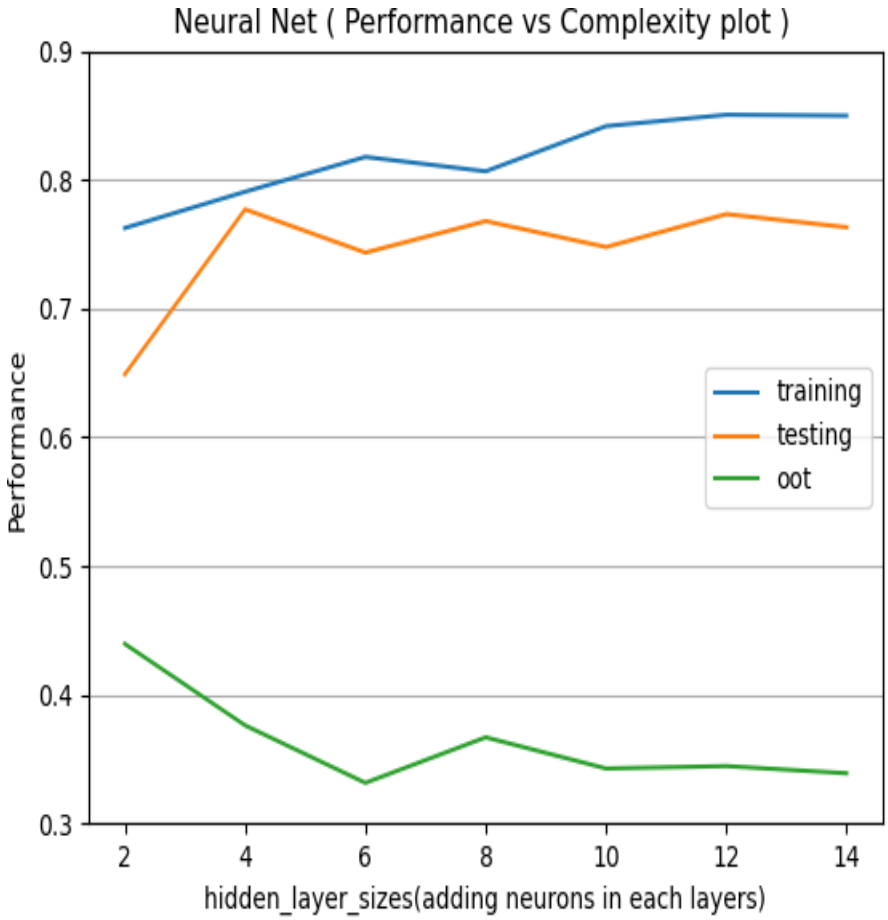
**Random Forest:** With Each Iteration , I am increasing max_depth which is casing model to overfit.

nitermax = 3, i in range(2,16,2)
model = RandomForestClassifier( max_depth = i, criterion = 'gini' ,n_estimators = 16, min_samples_split =  18, min_samples_leaf = 10, max_featurs = 4)



**Neural Net:** With each iteration we are adding increasing neruons in each of the layer, which cause model to be more complex and overfit.

nitermax = 3 , i in range(2,16,2)
model=MLPClassifier(activation = 'logistic', hidden_layer_sizes = (5*i , 5*i,), solver = 'adam', learning_rate = 'constant',learning_rate_init = 0.01, alpha = 0.00001,max_iter = 5000)



# Final model performance

The overall goal of the project is to develop a supervised machine-learning model that can be used to detect and predict fraud in credit card transaction. It aims to develop an efficient and effective statistical analysis model that can be applied in practice to predict fraud and identify fraudulent transactions.
After testing different algorithms to fit the data, such as Logistic Regression, Random Forest, XGBoost, Light GBM, and Neural Network experimented with various hyperparameters combinations for each model & compared their performance based on FDR at a 3% rejection rate.
**Final Model: Random Forest**

After comparing the results between logistics regression, boosted trees, random forest, and a neural network, we determined that Light Boosting Gradient Model performed the best. Random forest out performed other models for both training, testing and out of time validation datasets with 78.52%, 78.21% and 55.41% respectively.
The chosen hyperparameters are:
No. of variables: 20
n_estimators: 80
max_depth: 8
min_child_split: 50
minsamples_leaf: 40
max_features: 8

Summary Table has basically two categories: Bin statistics and cumulative statistics. The Fraud rate on the top of the Table is calculated by #Bads divided by #Goods and #Bads.
% Goods = Cum.Good/ Total Good

The fraud detection rate (%Bads) is calculated to be the number of true frauds in the bin, which are caught by the model, divided by the total number of true frauds exists in the entire dataset.

FDR reflects how many frauds can be caught by a model, with a fixed number of predicted positives.
FDR = Cum.Bad / Total Bad

KS is the maximum difference in cumulative fractions of goods and bads flagged at any possible cutoff.

KS= %Bad - % Good

False positive rate (FPR) – Percentage of total legitimate events that are incorrectly predicted as fraud.
FPR= Cum.Good/ Cum.Bad

FRD_TRN

| Bin Statistics | | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #Bins | #Records | #Goods | #Bads | % Goods | %Bads | Total #Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 590 | 224 | 366 | 37.966 | 62.034 | 590 | 224 | 366 | 0.3836 | 60 | 59.616 | 0.612 |
| 2 | 590 | 514 | 76 | 87.119 | 12.881 | 1180 | 738 | 442 | 1.2637 | 72.459 | 71.195 | 1.6697 |
| 3 | 590 | 562 | 28 | 95.254 | 4.7458 | 1770 | 1300 | 470 | 2.226 | 77.049 | 74.823 | 2.766 |
| 4 | 590 | 576 | 14 | 97.627 | 2.3729 | 2360 | 1876 | 484 | 3.2123 | 79.344 | 76.132 | 3.876 |
| 5 | 590 | 576 | 14 | 97.627 | 2.3729 | 2950 | 2452 | 498 | 4.1986 | 81.639 | 77.441 | 4.9237 |
| 6 | 591 | 581 | 10 | 98.308 | 1.692 | 3541 | 3033 | 508 | 5.1935 | 83.279 | 78.085 | 5.9705 |
| 7 | 590 | 580 | 10 | 98.305 | 1.6949 | 4131 | 3613 | 518 | 6.1866 | 84.918 | 78.731 | 6.9749 |
| 8 | 590 | 585 | 5 | 99.153 | 0.8475 | 4721 | 4198 | 523 | 7.1884 | 85.738 | 78.549 | 8.0268 |
| 9 | 590 | 583 | 7 | 98.814 | 1.1864 | 5311 | 4781 | 530 | 8.1866 | 86.885 | 78.699 | 9.0208 |
| 10 | 590 | 578 | 12 | 97.966 | 2.0339 | 5901 | 5359 | 542 | 9.1764 | 88.852 | 79.676 | 9.8875 |
| 11 | 590 | 587 | 3 | 99.492 | 0.5085 | 6491 | 5946 | 545 | 10.182 | 89.344 | 79.163 | 10.91 |
| 12 | 590 | 581 | 9 | 98.475 | 1.5254 | 7081 | 6527 | 554 | 11.176 | 90.82 | 79.643 | 11.782 |
| 13 | 590 | 585 | 5 | 99.153 | 0.8475 | 7671 | 7112 | 559 | 12.178 | 91.639 | 79.461 | 12.723 |
| 14 | 590 | 590 | 0 | 100 | 0 | 8261 | 7702 | 559 | 13.188 | 91.639 | 78.451 | 13.778 |
| 15 | 591 | 590 | 1 | 99.831 | 0.1692 | 8852 | 8292 | 560 | 14.199 | 91.803 | 77.605 | 14.807 |
| 16 | 590 | 589 | 1 | 99.831 | 0.1695 | 9442 | 8881 | 561 | 15.207 | 91.967 | 76.76 | 15.831 |
| 17 | 590 | 585 | 5 | 99.153 | 0.8475 | 10032 | 9466 | 566 | 16.209 | 92.787 | 76.578 | 16.724 |
| 18 | 590 | 590 | 0 | 100 | 0 | 10622 | 10056 | 566 | 17.219 | 92.787 | 75.568 | 17.767 |
| 19 | 590 | 590 | 0 | 100 | 0 | 11212 | 10646 | 566 | 18.229 | 92.787 | 74.557 | 18.809 |
| 20 | 590 | 590 | 0 | 100 | 0 | 11802 | 11236 | 566 | 19.24 | 92.787 | 73.547 | 19.852 |

FDR_TST

| Bin Statistics | | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #Bins | #Records | #Goods | #Bads | % Goods | %Bads | Total #Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 253 | 104 | 149 | 41.11 | 58.89 | 253 | 104 | 149 | 0.416 | 55.19 | 54.77 | 0.698 |
| 2 | 253 | 216 | 37 | 85.38 | 14.62 | 506 | 320 | 186 | 1.279 | 68.89 | 67.61 | 1.72 |
| 3 | 253 | 241 | 12 | 95.26 | 4.743 | 759 | 561 | 198 | 2.242 | 73.33 | 71.09 | 2.833 |
| 4 | 253 | 249 | 4 | 98.42 | 1.581 | 1012 | 810 | 202 | 3.237 | 74.81 | 71.58 | 4.01 |
| 5 | 252 | 246 | 6 | 97.62 | 2.381 | 1264 | 1056 | 208 | 4.221 | 77.04 | 72.82 | 5.077 |
| 6 | 253 | 248 | 5 | 98.02 | 1.976 | 1517 | 1304 | 213 | 5.212 | 78.89 | 73.68 | 6.122 |
| 7 | 253 | 249 | 4 | 98.42 | 1.581 | 1770 | 1553 | 217 | 6.207 | 80.37 | 74.16 | 7.157 |
| 8 | 253 | 249 | 4 | 98.42 | 1.581 | 2023 | 1802 | 221 | 7.202 | 81.85 | 74.65 | 8.154 |
| 9 | 253 | 249 | 4 | 98.42 | 1.581 | 2276 | 2051 | 225 | 8.197 | 83.33 | 75.14 | 9.116 |
| 10 | 253 | 249 | 4 | 98.42 | 1.581 | 2529 | 2300 | 229 | 9.193 | 84.81 | 75.62 | 10.04 |
| 11 | 253 | 252 | 1 | 99.6 | 0.395 | 2782 | 2552 | 230 | 10.2 | 85.19 | 74.99 | 11.1 |
| 12 | 253 | 252 | 1 | 99.6 | 0.395 | 3035 | 2804 | 231 | 11.21 | 85.56 | 74.35 | 12.14 |
| 13 | 253 | 251 | 2 | 99.21 | 0.791 | 3288 | 3055 | 233 | 12.21 | 86.3 | 74.09 | 13.11 |
| 14 | 253 | 251 | 2 | 99.21 | 0.791 | 3541 | 3306 | 235 | 13.21 | 87.04 | 73.82 | 14.07 |
| 15 | 253 | 252 | 1 | 99.6 | 0.395 | 3794 | 3558 | 236 | 14.22 | 87.41 | 73.19 | 15.08 |
| 16 | 252 | 251 | 1 | 99.6 | 0.397 | 4046 | 3809 | 237 | 15.22 | 87.78 | 72.55 | 16.07 |
| 17 | 253 | 253 | 0 | 100 | 0 | 4299 | 4062 | 237 | 16.24 | 87.78 | 71.54 | 17.14 |
| 18 | 253 | 253 | 0 | 100 | 0 | 4552 | 4315 | 237 | 17.25 | 87.78 | 70.53 | 18.21 |
| 19 | 253 | 251 | 2 | 99.21 | 0.791 | 4805 | 4566 | 239 | 18.25 | 88.52 | 70.27 | 19.1 |
| 20 | 253 | 253 | 0 | 100 | 0 | 5058 | 4819 | 239 | 19.26 | 88.52 | 69.26 | 20.16 |

FDR_OOT

| Bin Statistics | | | | | | Cumulative Statistics | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #Bins | #Records | #Goods | #Bads | % Goods | %Bads | Total #Records | Cumulative Goods | Cumulative Bads | % Goods | % Bads (FDR) | KS | FPR |
| 1 | 121 | 49 | 72 | 40.4959 | 59.5041 | 121 | 49 | 72 | 0.4111 | 40.2235 | 39.8123 | 0.6806 |
| 2 | 121 | 98 | 23 | 80.9917 | 19.0083 | 242 | 147 | 95 | 1.2334 | 53.0726 | 51.8392 | 1.5474 |
| 3 | 121 | 115 | 6 | 95.0413 | 4.9587 | 363 | 262 | 101 | 2.1984 | 56.4246 | 54.2262 | 2.5941 |
| 4 | 121 | 119 | 2 | 98.3471 | 1.6529 | 484 | 381 | 103 | 3.1968 | 57.5419 | 54.3451 | 3.699 |
| 5 | 121 | 118 | 3 | 97.5207 | 2.4793 | 605 | 499 | 106 | 4.1869 | 59.2179 | 55.0309 | 4.7075 |
| 6 | 121 | 115 | 6 | 95.0413 | 4.9587 | 726 | 614 | 112 | 5.1519 | 62.5698 | 57.418 | 5.4821 |
| 7 | 121 | 119 | 2 | 98.3471 | 1.6529 | 847 | 733 | 114 | 6.1504 | 63.6872 | 57.5368 | 6.4298 |
| 8 | 121 | 114 | 7 | 94.2149 | 5.7851 | 968 | 847 | 121 | 7.1069 | 67.5978 | 60.4909 | 7 |
| 9 | 121 | 118 | 3 | 97.5207 | 2.4793 | 1089 | 965 | 124 | 8.097 | 69.2737 | 61.1767 | 7.7823 |
| 10 | 121 | 120 | 1 | 99.1736 | 0.8264 | 1210 | 1085 | 125 | 9.1039 | 69.8324 | 60.7285 | 8.68 |
| 11 | 121 | 120 | 1 | 99.1736 | 0.8264 | 1331 | 1205 | 126 | 10.1108 | 70.3911 | 60.2803 | 9.5635 |
| 12 | 121 | 118 | 3 | 97.5207 | 2.4793 | 1452 | 1323 | 129 | 11.1009 | 72.067 | 60.9662 | 10.2558 |
| 13 | 121 | 120 | 1 | 99.1736 | 0.8264 | 1573 | 1443 | 130 | 12.1077 | 72.6257 | 60.518 | 11.1 |
| 14 | 121 | 120 | 1 | 99.1736 | 0.8264 | 1694 | 1563 | 131 | 13.1146 | 73.1844 | 60.0697 | 11.9313 |
| 15 | 121 | 119 | 2 | 98.3471 | 1.6529 | 1815 | 1682 | 133 | 14.1131 | 74.3017 | 60.1886 | 12.6466 |
| 16 | 121 | 120 | 1 | 99.1736 | 0.8264 | 1936 | 1802 | 134 | 15.12 | 74.8603 | 59.7403 | 13.4478 |
| 17 | 120 | 119 | 1 | 99.1667 | 0.8333 | 2056 | 1921 | 135 | 16.1185 | 75.419 | 59.3005 | 14.2296 |
| 18 | 121 | 120 | 1 | 99.1736 | 0.8264 | 2177 | 2041 | 136 | 17.1254 | 75.9777 | 58.8523 | 15.0074 |
| 19 | 121 | 120 | 1 | 99.1736 | 0.8264 | 2298 | 2161 | 137 | 18.1322 | 76.5363 | 58.4041 | 15.7737 |
| 20 | 121 | 120 | 1 | 99.1736 | 0.8264 | 2419 | 2281 | 138 | 19.1391 | 77.095 | 57.9559 | 16.529 |

## Financial curve and recommended cutoff

We have the score that ranks the transactions by a probability that event is a fraud. Evaluation is done based on rank ordering.

Business managers know the estimate that is the average loss that they make when a mistake of not catching a fraud transaction. Estimated penalty for a false positive (denying a good transaction thinking it's fraud) this will substantially lead to customer cancelling the account.
So the key number are:
- Estimate $ lost for every fraud transaction missed ($400).
- Estimate lost sales from every false positive, a good transaction that we denied (20).

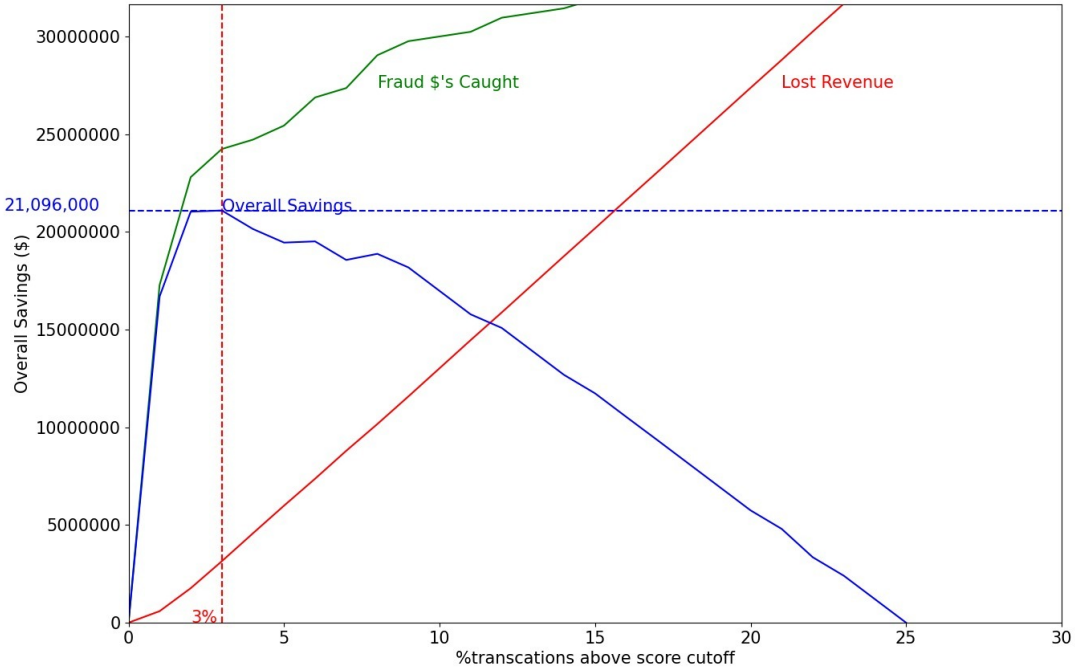
Overall saving is the difference between these values.

Score is used on the evaluation data to estimate the value the model will be and to set the cut off. Out of time data is used make fair assessment of the model performance after we implement it and to get score cutoff.

This plot is the based the 2 key numbers and different penetration percentiles for the overall savings.

Lost revenue is when a good transaction is denied.
Fraud caught is the benefit caught for every fraud caught.
Fraud caught is in the high percentile bin because it is sorted by fraud score.
After some time not a lot of fraud is caught.

Out of time is used because it's the most likely scenario and it is projected to an annualized number.




Overall saving = Fraud caught- lost revenue

**Fraud cutoff:**
Cut-off can't be kept at the steep part of the curve because behaviour can change in the future as the overall saving is very sensitive to the cut-off. So, we flat region of the overall saving curve. We must keep the cut-off far to the left as possible because of the denial for fewest number of transactions and still a decent amount of saving. Denying 5% is a lot so the cutoff is kept at 3%.

**Dollar Savings:**
To get the annual benefit we must annualize number of transactions (100,000).
Dollar savings is about a 21 million transactions

## Summary

**Step 1**: Exploratory data analysis and Data Cleaning: Exploratory data analysis was conducted to describe and visualize each of the 10 data fields. The data cleaning involved removing outliers upon knowledge group approval and addressing missing values by carefully crafted data imputation techniques.

**Step 2**: Feature Engineering: There were six different types of variables: amount variables, frequency variables, days-since variables, velocity change variables,vd ratio variables and variability variables were created along with two Benford's Law variables, and a day-of-the-week risk table variable.Target encoding and statistical smoothing was conducted on Day-of-week variable and Merchant state variables.

**Step 3**: Feature Selection: The data was divided into three sections for model development and analysis: training, testing, and out-of-time (OOT). Feature selection involved filtering based on Kolmogorov-Smirnov(KS) score and using the wrapper method. Top 20 best variables with each having a wrapper score more or less above 0.6.

**Step 4**: Model Selection: Having a logistic regression model as a baseline, several non-linear models with varying hyper parameters were explored. These non-linear models included Decision Tree, Random Forest, Gradient Boosting, and Neural Network. Random Forest delivered the best training, testing, and OOT data results along with the $ savings of 21,216,000 and the score cutoff at 3%.