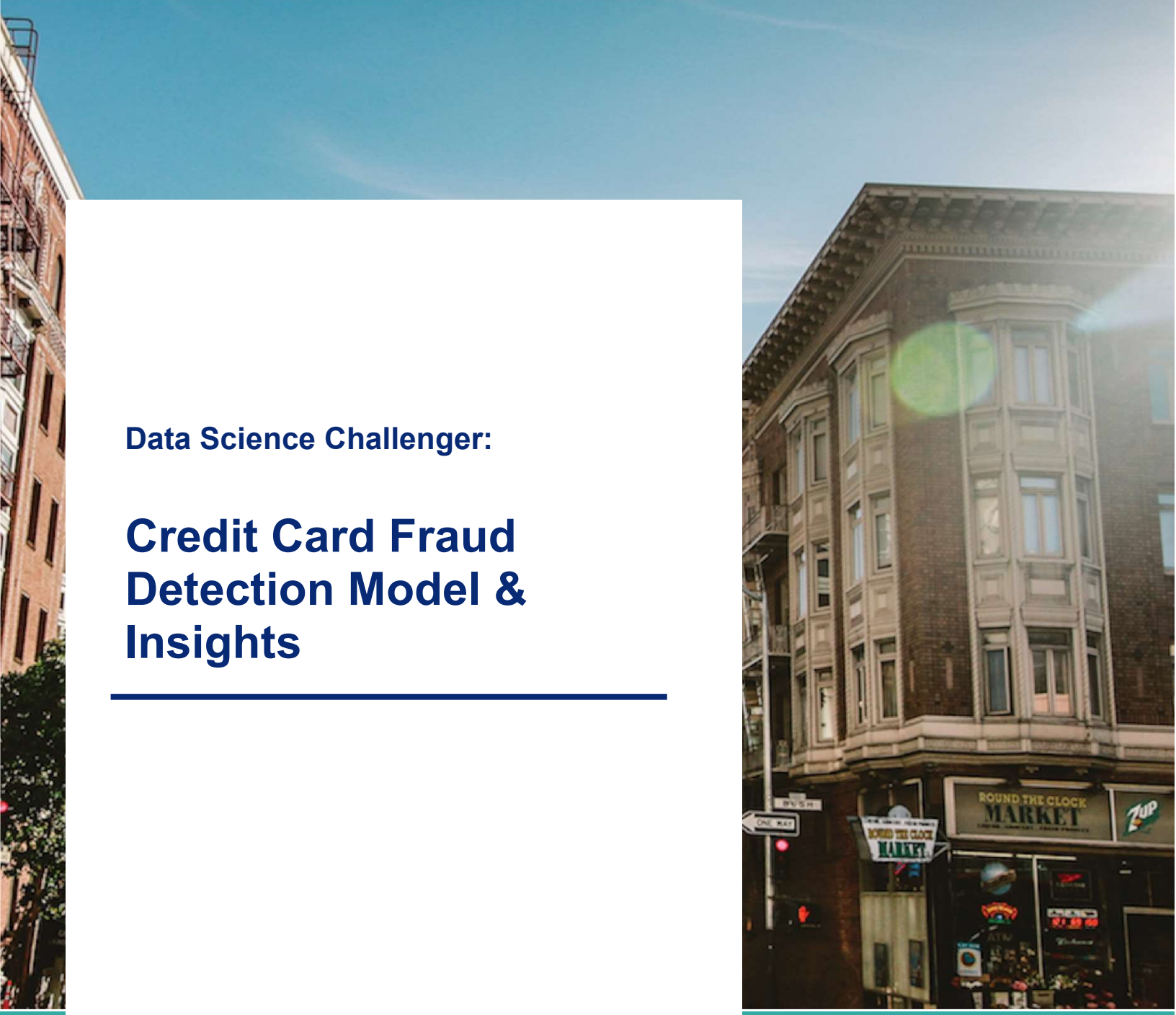


Data Science Challenger:

Credit Card Fraud Detection Model & Insights

29Th OCT-1st NOV 2020

Gaurav Singh

Candidate ID: C1708930

"The world is one Big data problem"

: Andrew McAfee

I appreciate Capital one for giving me the opportunity to work through the data science challenger, which I found super interesting. Should you have any further questions, comments and feedback on this document, please feel free to reach out to me at:

gaurav11235@gmail.com
+1-919-867-9649
Candidate ID: C1708930

Table of Contents

Background

As part of data science challenger, a sample of credit card transaction data was provided along with 4 questions to solve in about a week of time. This report provides the description of my solution along with some additional insights, modeling considerations and areas of improvement. The findings in this report are based on outputs that I produced in JupytorLab notebook using python, is being delivered along with this document.

Following 4 questions were asked as part of data science challenger:

Question 1: Load

- Programmatically download and load into your favorite analytical tool the transactions data. This data, which is in line delimited JSON format, can be found here (link provided in the email)
- Please describe the structure of the data. Number of records and fields in each record?
- Please provide some additional basic summary statistics for each field. Be sure to include a count of null, minimum, maximum, and unique values where appropriate.

Question 2: Plot

- Plot a histogram of the processed amounts of each transaction, the transactionAmount column.
- Report any structure you find and any hypotheses you have about that structure.

Question 3: Data Wrangling - Duplicate Transactions

- You will notice a number of what look like duplicated transactions in the data set. One type of duplicated transaction is a reversed transaction, where a purchase is followed by a reversal. Another example is a multi-swipe, where a vendor accidentally charges a customer's card multiple times within a short time span.
- Can you programmatically identify reversed and multi-swipe transactions?
- What total number of transactions and total dollar amount do you estimate for the reversed transactions? For the multi-swipe transactions? (please consider the first transaction to be "normal" and exclude it from the number of transaction and dollar amount counts)
- Did you find anything interesting about either kind of transaction?

Question 4: Model

- Fraud is a problem for any bank. Fraud can take many forms, whether it is someone stealing a single credit card, to large batches of stolen credit card numbers being used on the web, or even a mass compromise of credit card numbers stolen from a merchant via tools like credit card skimming devices.
- Each of the transactions in the dataset has a field called isFraud. Please build a predictive model to determine whether a given transaction will be fraudulent or not. Use as much of the data as you like (or all of it).

- Provide an estimate of performance using an appropriate sample and show your work.
- Please explain your methodology (modeling algorithm/method used and why, what features/data you found useful, what questions you have, and what you would do next with more time)

The solution of these questions is provided in subsequent sections. There may be some additional information reported in each section which were not categorically asked to be produced but the modeling process necessitated those insights and helped moving forward.

Section 1: Load data and Produce Basic Summary

The data for this exercise was provided in github. The python program in jupyter notebook shows the program that programmatically read the data into pandas.
As I reviewed the data, I made following observations:

- The data had 786,363 rows and 29 columns.
- All the date columns were 'object' type. All other columns were numeric (float or integer) or boolean. As part of data cleaning process, all the date columns were converted to datetime format and all the spaces were converted to 'Nan' and its report was generated provided in subsequent section.
- It was customer-credit card-transaction level data with start date as 1st Jan'2016 and end date as 30th Dec'2016. The transactions were made by 5000 customers and each customer had one card (both customer id and account number matched exactly).
- All the customers in the data had at least 36 months to card expiry.
- There was no new account opened during the 1-year period for which transactions were provided. Essentially, it implied that the data was sampled to contain transactions of only those customers who already had their accounts opened prior to start date 1st Jan 2016.
- Some customer had minimum available money in negative, indicating that at times, credit limit was breached.
- There were about 6 columns that had only 1 unique value while top value was blank space, indicating that all those 6 columns were blank. As part of data cleaning process, those 6 columns were removed:
    I. echoBuffer
    II. merchantCity
    III. merchantState
    IV. merchantZip
    V. posonPremises
    VI. recurringAuthInd

The table 1 shows the count of missing and its percentage for each column. As mentioned earlier, the 6 columns that had all the values missing are shown as having 100% missing in the report below:

Table 1: Missing value report

No	column_name	Data_type	Missing_count	Missing_percent
0	accountNumber	int64	0	0
1	customerId	int64	0	0
2	creditLimit	int64	0	0
3	availableMoney	float64	0	0
4	transactionDateTime	datetime64[ns]	0	0
5	transactionAmount	float64	0	0
6	merchantName	object	0	0
7	acqCountry	object	4562	0.58
8	merchantCountryCode	object	724	0.092
9	posEntryMode	object	4054	0.516
10	posConditionCode	object	409	0.052
11	merchantCategoryCode	object	0	0
12	currentExpDate	int64	0	0
13	accountOpenDate	datetime64[ns]	0	0
14	dateOfLastAddressChange	datetime64[ns]	0	0
15	cardCVV	int64	0	0
16	enteredCVV	int64	0	0
17	cardLast4Digits	int64	0	0
18	transactionType	object	698	0.089
19	echoBuffer	float64	786363	100
20	currentBalance	float64	0	0
21	merchantCity	float64	786363	100
22	merchantState	float64	786363	100
23	merchantZip	float64	786363	100
24	cardPresent	int32	0	0
25	posOnPremises	float64	786363	100
26	recurringAuthInd	float64	786363	100
27	expirationDateKeyInMatch	int32	0	0
28	isFraud	int32	0	0

As requested in the 1st question, some basic summarization was produced on all the columns of the raw table. The summary of all the columns is provided in table 2,3,4,5 below:

Table 2: Summarization of numeric columns

	count	mean	std	min	25%	50%	75%	max
creditLimit	786363	10760.00	11640.00	250.00	5000.00	7500.00	15000.00	50000.00
availableMoney	786363	6251.00	8881.00	-1006.00	1077.00	3185.00	7500.00	50000.00
transactionAmount	786363	137.00	147.70	0.00	33.65	87.90	191.50	2012.00
cardCVV	786363	544.50	261.50	100.00	310.00	535.00	785.00	998.00
enteredCVV	786363	544.20	261.60	0.00	310.00	535.00	785.00	998.00
cardLast4Digits	786363	4757.00	2997.00	0.00	2178.00	4733.00	7338.00	9998.00
currentBalance	786363	4509.00	6457.00	0.00	689.90	2452.00	5291.00	47500.00

(excludes account number and customerid)

Table 3: Summarization of object columns

	count	unique	top	freq
merchantName	786363	2490	Uber	25613
acqCountry	786363	5	US	774709
merchantCountryCode	786363	5	US	778511
posEntryMode	786363	6	5	315035
posConditionCode	786363	4	1	628787
merchantCategoryCode	786363	19	online_retail	202156
transactionType	786363	4	PURCHASE	745193
echoBuffer	786363	1		786363
merchantCity	786363	1		786363
merchantState	786363	1		786363
merchantZip	786363	1		786363
posOnPremises	786363	1		786363
recurringAuthInd	786363	1		786363

Table 4: Summarization of Boolean columns

	count	unique	top	freq
cardPresent	786363	2	FALSE	433495
expirationDateKeyInMatch	786363	2	FALSE	785320
isFraud	786363	2	FALSE	773946

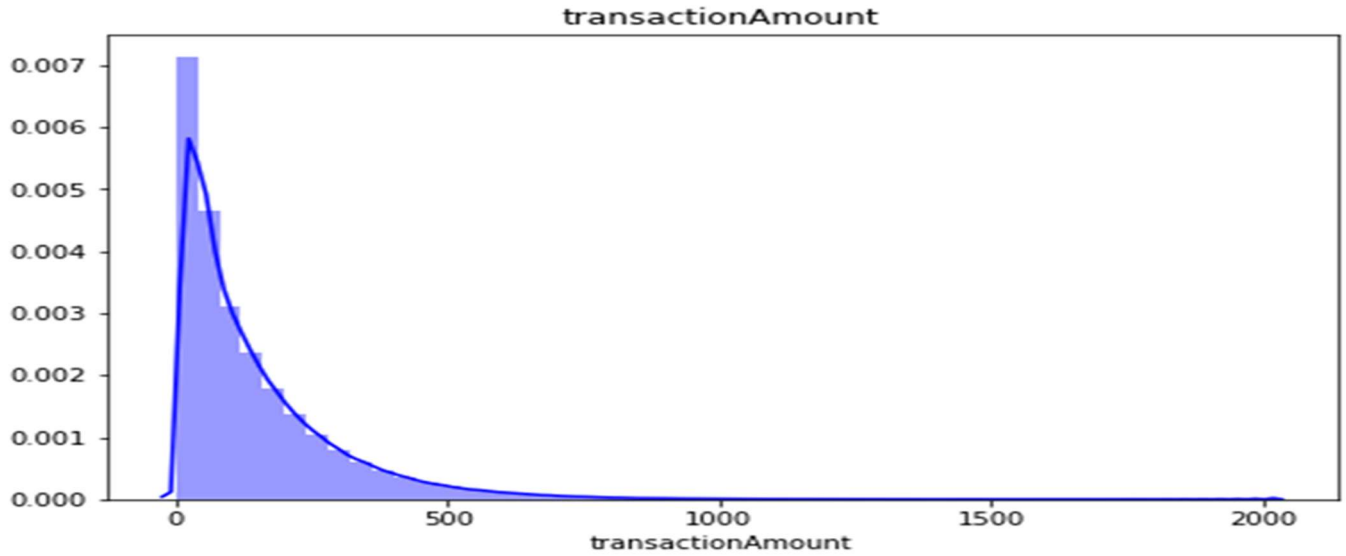
Table 5: Summarization of Date columns

	count	unique	top	first	last
transactionDateTime	786363	776637	5/28/2016 14:24	1/1/2016 0:01	12/30/2016 23:59
accountOpenDate	786363	1820	6/21/2014 0:00	8/22/1989 0:00	12/31/2015 0:00
dateOfLastAddressChange	786363	2184	3/15/2016 0:00	8/22/1989 0:00	12/30/2016 0:00

Section 2: Plot histogram of transaction Amount

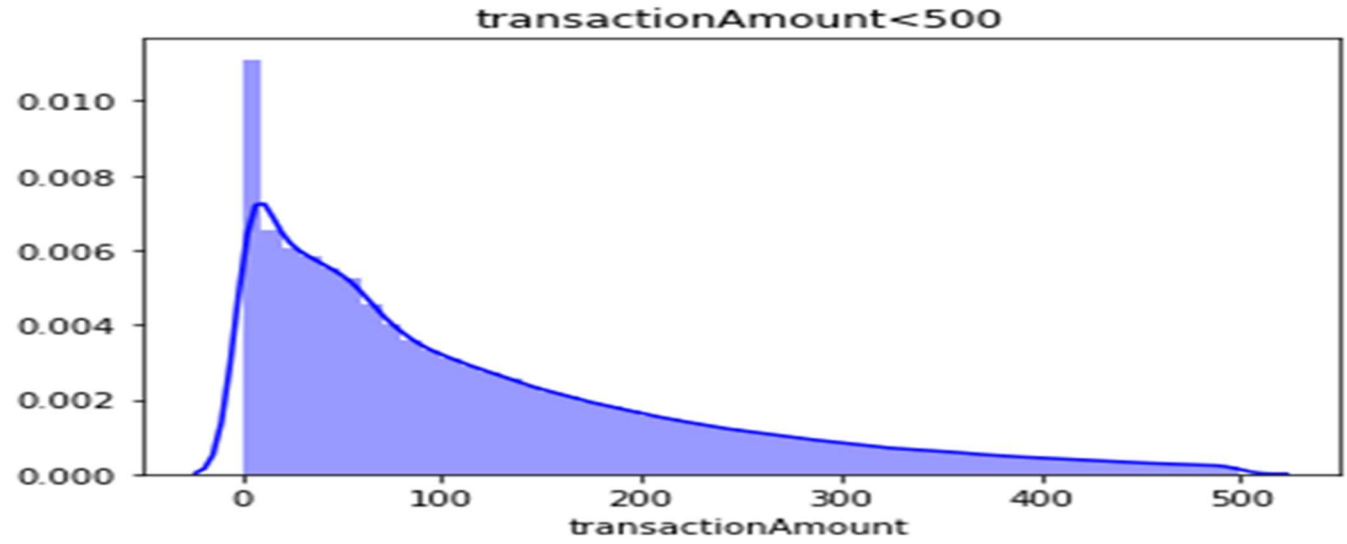
I plotted histogram for transaction amount to better understand its distribution. The chart 1 shows the plot. As shown in the plot, majority of transactions have occurred under $500 and the distribution is right skewed.

Chart 1: Transaction Amount histogram



To propose a hypothesis, I created another chart for transaction<$500. Looks like majority of transactions are occurring under $50. That may imply that transactions could be towards subscriptions, gas station, restaurant, or online purchases.

Chart 2: Transaction Amount <500



Section 3: Identify Multi-Swipe & Reversal transactions

Multi-Swipe and reversal transactions are the duplicate transactions that are done by same customer, same merchant and of same amount. The goal of this section was to programmatically identify which transactions are 'reversal' and 'multi-swipe' while keeping the first transaction as is.
Since, customers also pay towards subscriptions or sometimes can go to same merchant and spend same amount, all those transactions may look like duplicate transactions. My goal was to separate out 'reversal' and 'multi-swipe' from genuine transactions. I created a dataset that had duplicates when looking at accountNumber, merchant name and transactionAmt as key. The next step was to define each type of transactions and produce some statistics. After performing some data exploration, I concluded to go with following definitions:

1. Reversal: Duplicate transactions that occur within 29 days after its first occurrence and are not tagged as 'subscription' or 'online subscription' by merchant category code.
2. Multi-Swipe Transaction: Subset of 'reversal' but it occurs within an hour when card is present (another analysis was done regardless of card present)
3. The rest of the duplicate transactions were treated as genuine transactions

Based on above definition, I was able to produce following statistics in Jupyter notebook:

Count of Multi-Swipe Transaction (when card is present) is= 6127
Sum dollars of Multi-Swipe Transaction (when card is present) is= 904664.63 Dollars

Count of Multi-Swipe Transaction (regardless of card's presence) is= 13426
Sum dollars of Multi-Swipe Transaction (regardless of card's presence) is= 1934160.37 Dollars

Count of Reversal Transaction (occured<=29 days and non-subscription based)) is= 43892
Sum dollars of Reversal Transaction (occured<=29 days and non-subscription based)) is= 4329931.91 Dollars

Section 4: Fraud detection Modeling and results

As a critical part of the modeling, I spent quite a bit of time to better understand each feature, perform feature engineering and ensure that I have a good model data to build a model. In view of that, I would cover the description of the modeling process and results in following 3 sub-sections:

1. - Data exploration and Features engineering
    - Date Features
    - Numeric and Boolean Features
    - String/Object features
2. - Create train and test samples
    - Modeling consideration for sample selection
    - Treatment for imbalance data
3. - Build models and compare model performance
        - Light GBM
        - Neural Network
        - XGBoost
        - Random Forest
        - Logistic Regression
        - Decision Tree
        - CatBoost

Section 4.1: Data exploration and Features engineering

Based on initial exploration, here are some findings were observed:

- The data contains 1-year worth of transactions ranging from 1st Jan 2016 to 30th Dec 2016 with 5000 customers (using 5000 cards) making 786,363 transactions.
- Essentially, each customer owns only one card.
- All the accounts in the data are pre-existing accounts (opened before transaction data's start date), essentially implying that the provided data reflects a sample of 5000 accounts that were opened prior to start date of transaction data, and their transactions were pulled from 1st January 2016 to 30th December 2016.
- The fraud rate is 1.579%. The data is imbalanced in terms of target rate, we will cover it in more detail once we reach to the step of sample creation for model and validation

An important point to discuss here is review of the data. Overall, there are three types of information available in the data:

1. Information about card (like current balance, available money, credit limit, acq country)
2. Information about Merchant (like merchant category code, name, merchant country etc)
3. Information about transactions (like transaction date, amount, type etc)

Out of above 3 information, the third one gives us opportunity to create sequential features. Essentially, since, transactions are time ordered, the information from previous transactions can be used to predict fraud/not fraud of subsequent transaction. I created those features for the model and will be discussing them in more detail in subsequent section.

In the next section, I will share findings of my data exploration and feature engineering for the columns based on date features followed by other numeric and categorical columns. I will also be sharing the features that showed the power to differentiate between fraud and non-fraud based on one to one analysis. Essentially, we would like to see if there are certain sections of the feature that has higher fraud rate than the others. During this process, I learnt more about the data and built new derived features.

In next section, we will see how I created derived features based on date columns, followed by numeric and object type columns. For exact logic and code, please refer to notebook shared alongwith this document.

Section 4.1.1: Date Columns

As part of Feature engineering, I created some additional features to enrich the data. All the date columns can't be used for modelling directly, so I will be creating derived features off them. So far, we have following date columns:

- transactionDateTime
- dateOfLastAddressChange
- accountOpenDate
- currentExpDate

Derived Feature	Source Column	Derivation Logic
TimeSinceLastTransaction	transactionDateTime	difference between consecutive transaction in terms of days, hours, seconds and minutes
TimeOfTransaction	transactionDateTime	hour, weekday of transaction date time
daysSinceLastAddressChanged	dateOfLastAddressChange	difference between transaction date and last address changed
ageAccountDays	accountOpenDate	difference between transaction date and account open date
monthsToCardExpiry	currentExpDate	difference between transaction date and card expiry date

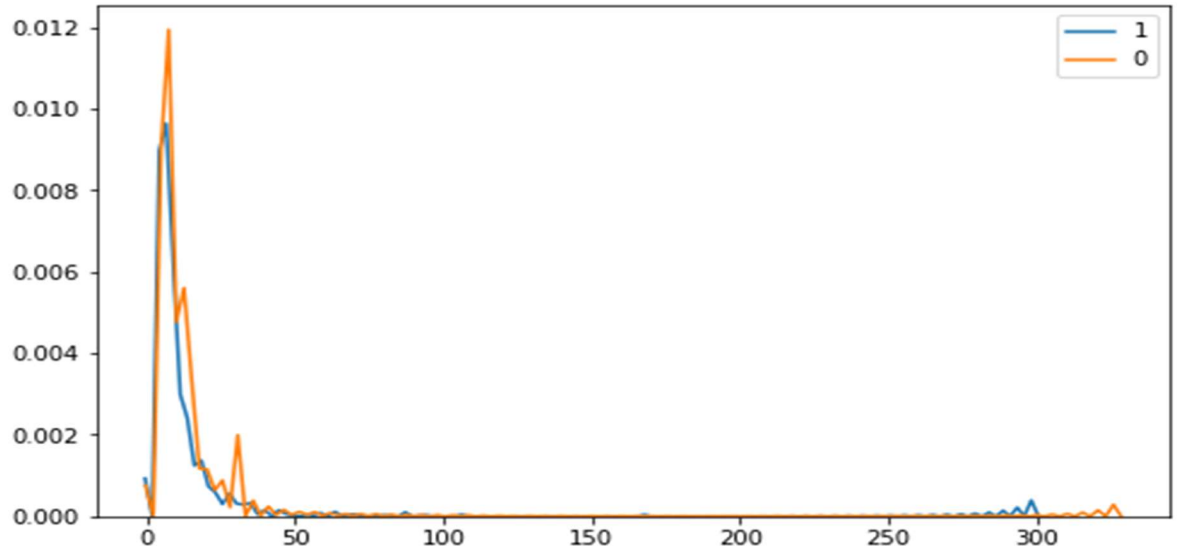
Exploration:

Days Since Last Transaction

I could not find much distributional difference of 'Days Since Last Transaction' for frauds and non-frauds indicating that this column may not be a great predictor alone.

Min DaysSinceLastTrasanction is=0
Max DaysSinceLastTransaction is= 328.16037037037034

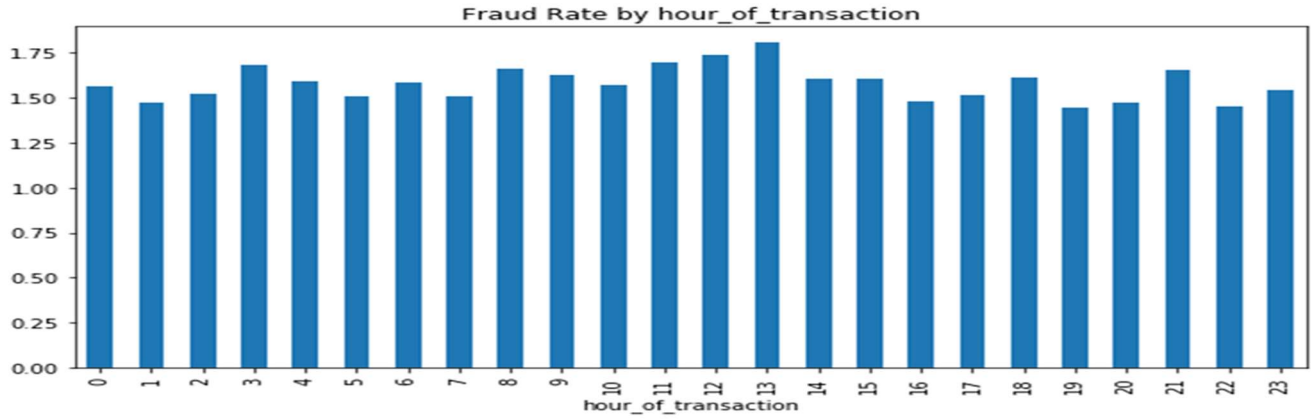
Chart 3: Days since last transaction with Fraud and non-fraud



Hour of Transaction:

The fraud rate did not vary much across different hours except there was a peak of fraud rate of 1.75% at 13th hour when compared to 1.5% fraud rate of overall population. Not sure, how good indicator it would be. I ended up creating dummy variable for each of those categories and included in the model to see if it can help predict in combination of any other feature.

Chart 4: Hour of the transaction with Fraud rate

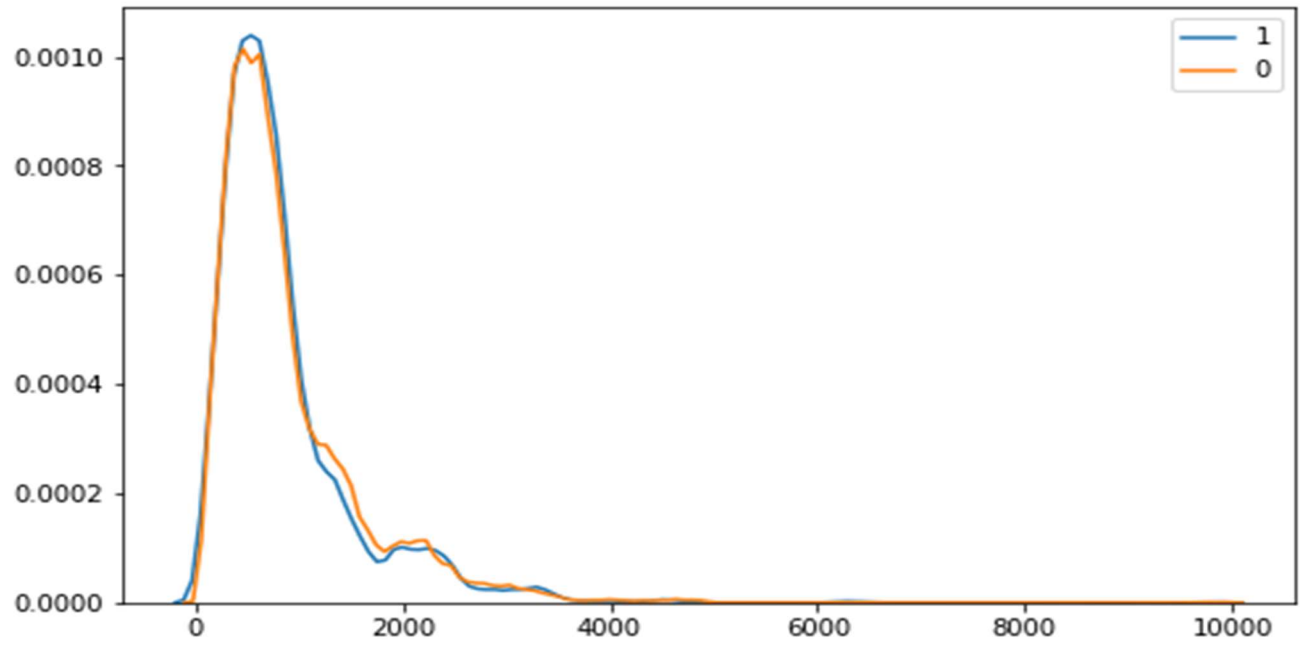


Age of the Account:

Max Age is= 9990 days, Min Age is= 1 days

The fraud and non-fraud distribution most remain overlapping by age of the account with the exception of fraud rate peaking at around 700 days.

Chart 5: Age of the account plotted with Fraud and non-Frauds

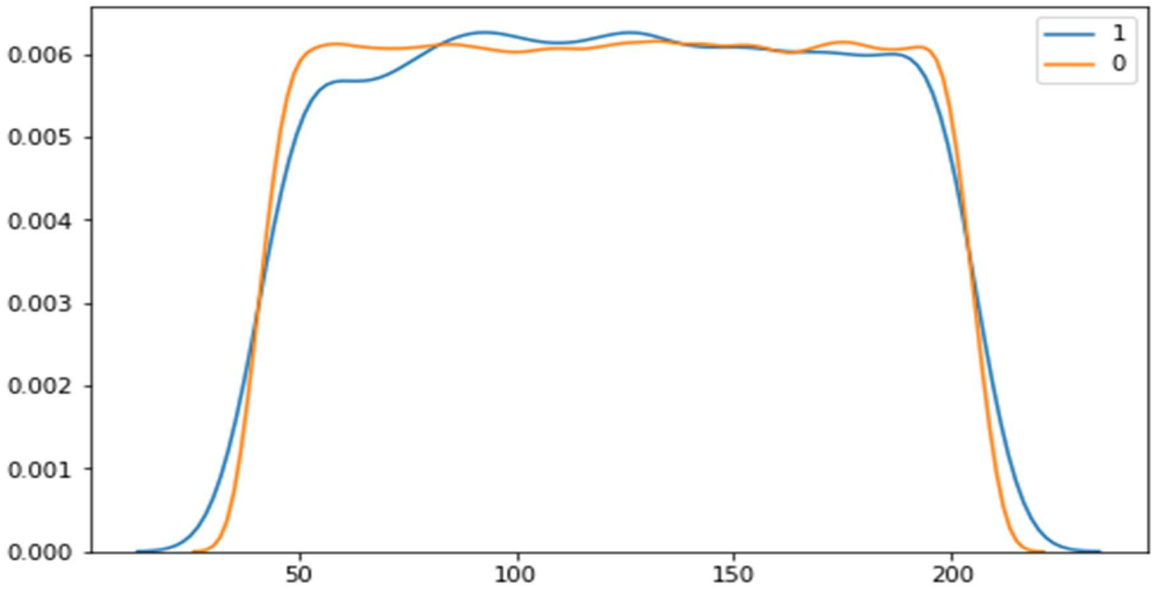


Months to card Expiry:

Max Months_to_Card_Expiry is= 211 Min Months_to_Card_Expiry is= 36

The fraud and non-fraud distribution most remain overlapping by months to card expiry with the exception of some instances where fraud rate was observed to be higher towards the lower and higher end of distribution.

Chart 6: Months to card Expiry with Fraud and non-Frauds

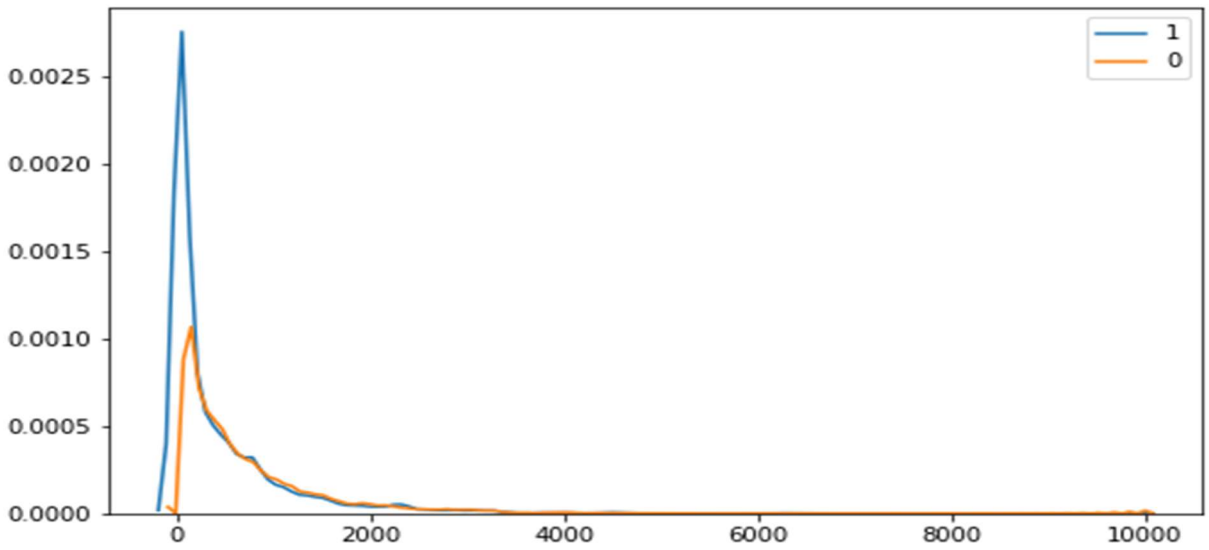


Days Since Address Changed:

Max daysSinceAddressChanged is= 9990 Min daysSinceAddressChanged is= 0

Days since last address changed shows strong differentiating power between frauds and non-frauds. If somebody has recently changed their address, they are at higher risk of fraud transaction from their account.

Chart 7: Days since address changed with Fraud and non-Frauds



Section 4.1.2: Numeric Columns

A number of derived columns were created using existing columns in the data. The goal was to create sequential variables which will use information from past, since transactions are sequential and a deviation from past behavior may indicate fraud transaction. In view of that, I spent quite a bit of tim e in creating these derived columns. I would specifically highlight some here:

- Transaction velocity (Count of transactions in last 1,3,7 days as rolling window)
- Transaction Momentum (avg transaction amount in last 1,3,7 days as rolling window)
- Ratio of transaction velocity with avg transaction velocity in past
- Ratio of transaction momentum with avg transaction momentum in past
- Ratio of avg transaction amount in a MC wrt avg transaction amount by total population in the same MC
- Ratio of avg transaction amount in a MC wrt to customer's own avg transaction amount in the same MC
- Count of fraud in past

All of these features are rolled up for every customer upto each transaction date to better assess seq uential behavior. The table below has more details about them:

Derived column	Source Column(s)	Description
utilization	Creditlimit, currentBalance	ratio of current balance wrt credit limit
cvv_match	cardCVV, enteredCVV	flag if cardCVV and enteredCVV matched
ratioAmtMC_Customer_Total	transaction amount, merchant category code, transactiondatetim	ratio of transaction amount made by customer in given merchant category with total avg spend in the same merchant category
ratioAmtMC_Customer	transaction amount, merchant category code	ratio of transaction amount made by customer in given merchant category with avg spend by same customer in past in the same merchant category
countTransaction_L*x*D	transactiondatetime	transaction velocity based on # transaction in past 'x' days
avgTransactionAmt_L*x*D	transactionamount, transactiondatetime	transaction momentum based on avg transaction amount in past 'x' days
ratioCountTransaction_L*x*D	transactiondatetime	ratio of transaction velocity with avg transaction velocity rolled up at each period
ratioTransactionAmt_L*x*D	transactionamount, transactiondatetime	ratio of transaction momentum with avg transaction momentum rolled up at each period
countPastFraud	isFraud	count of past frauds (excluding the one from current period)
cumsumCountTransaction	transactiondatetime	count of total transactions rolled up at period

In the next section, I will provide exploration of some of these features.

Transaction Amount

Max transactionAmount is= 2011.54 Min transactionAmount is= 0.0

From the chart 8, The transaction amount has fatter tail for frauds than non-frauds indicating the tran saction amount is usually higher for fraud transaction. Another observation is made from chart 9 that transaction amount that were fraud appeared to swell for transaction type=Missing when compared with non-fraud transactions.

Chart 8: Transaction amount with Fraud and non-Frauds

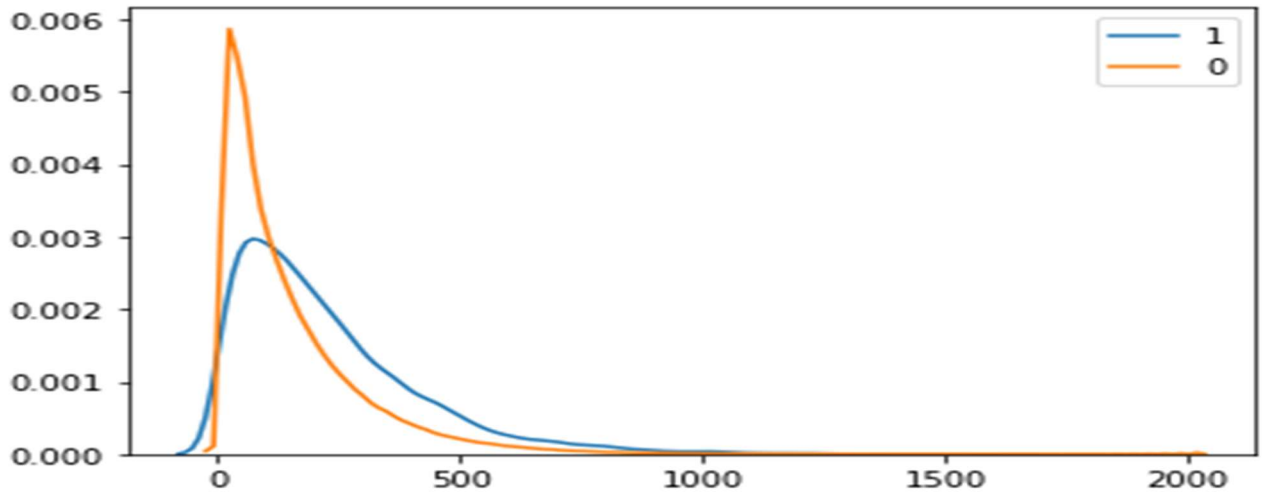
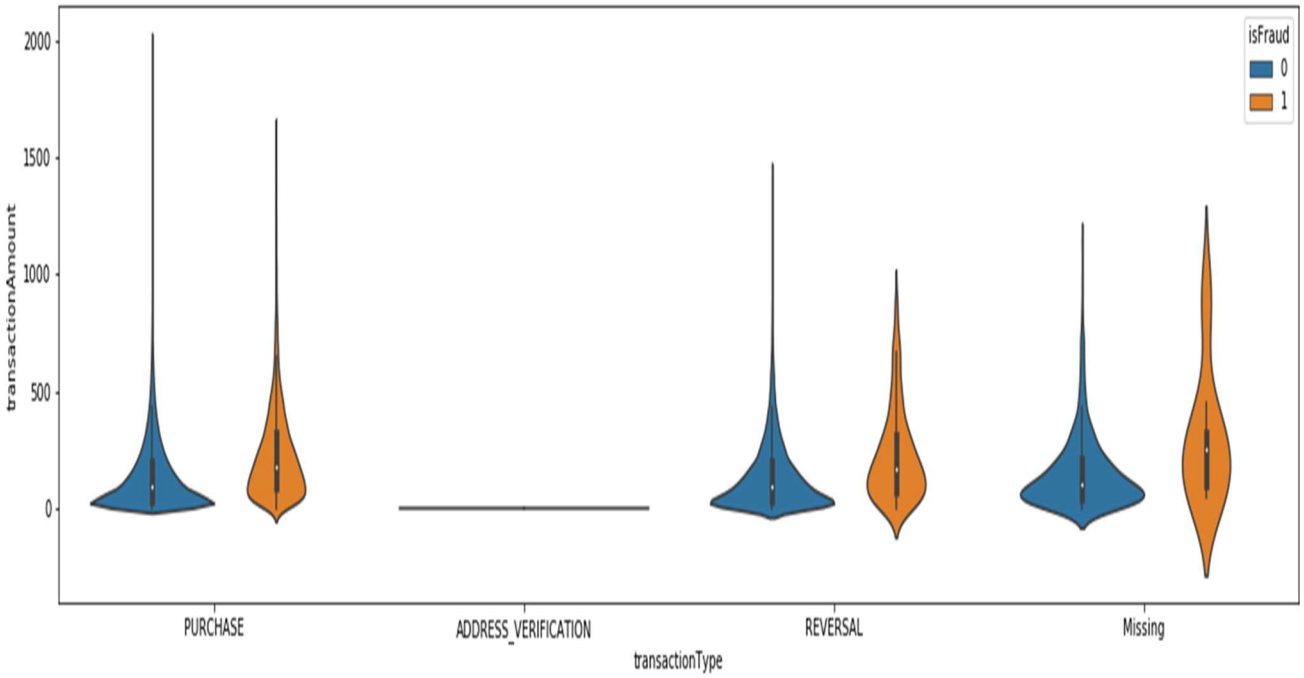


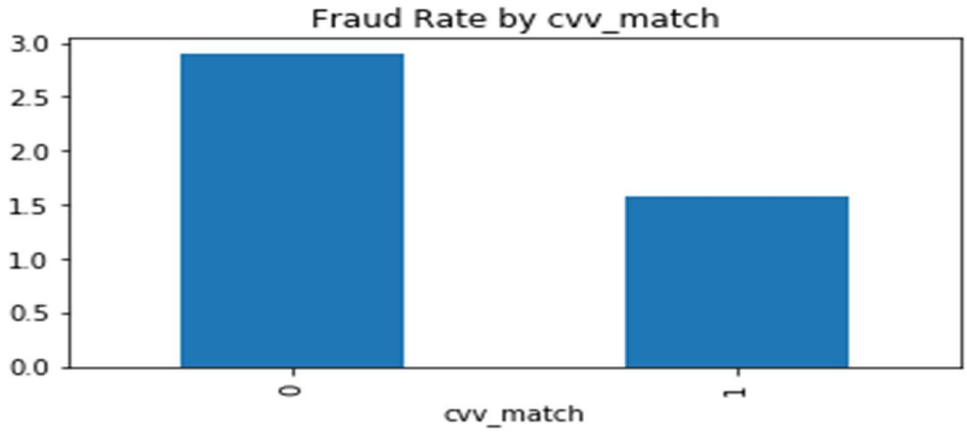
Chart 9: Transaction amount with Fraud and non-Frauds for each transaction type



CVV match

CVV match indicator was created to see if cvv matched with entered CVV. From the chart 10, the frau d rate is observed to be nearly double when cvv didn't match.

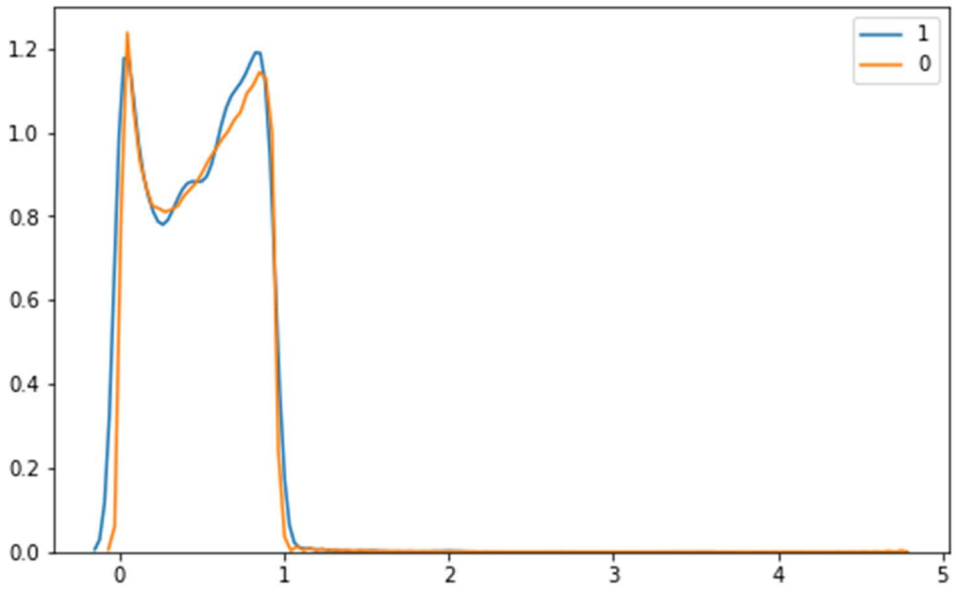
Chart 10: Fraud Rate by CVV match



Utilization

The utilization feature shows that fraud transactions are distributed more in higher utilization bucket when compared with non-frauds. That is natural to expect as fraudsters would likely make the best of use of credit limit when they can.

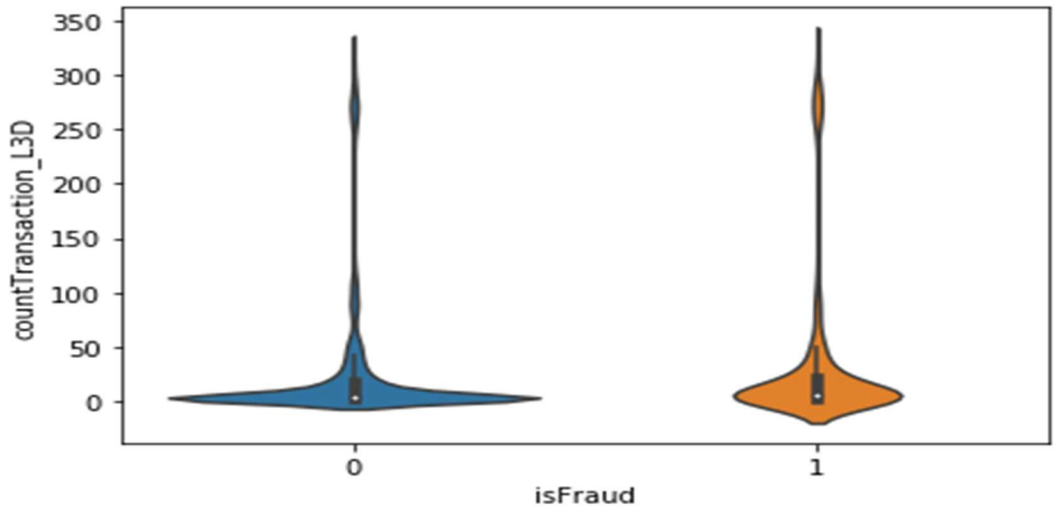
Chart 11: Distribution of utilization by fraud and non-fraud



Transaction Velocity

The transaction velocity over 3 days (count of transaction in last 3 days) shows that fraudsters have higher transaction velocity than non-fraudster.

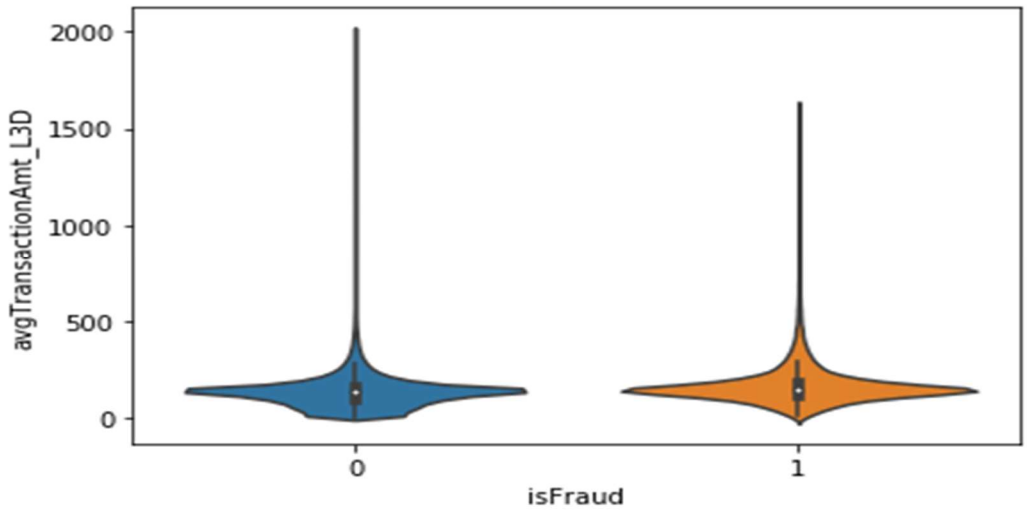
Chart 12: Distribution of transaction velocity by fraud and non-fraud



Transaction Momentum

The transaction momentum over 3 days (avg transaction amount in last 3 days) shows that fraudsters and non-fraudsters have similar transaction momentum. That surprises me a little bit, but I would let model make the best use of these features.

Chart 13: Distribution of transaction momentum by fraud and non-fraud



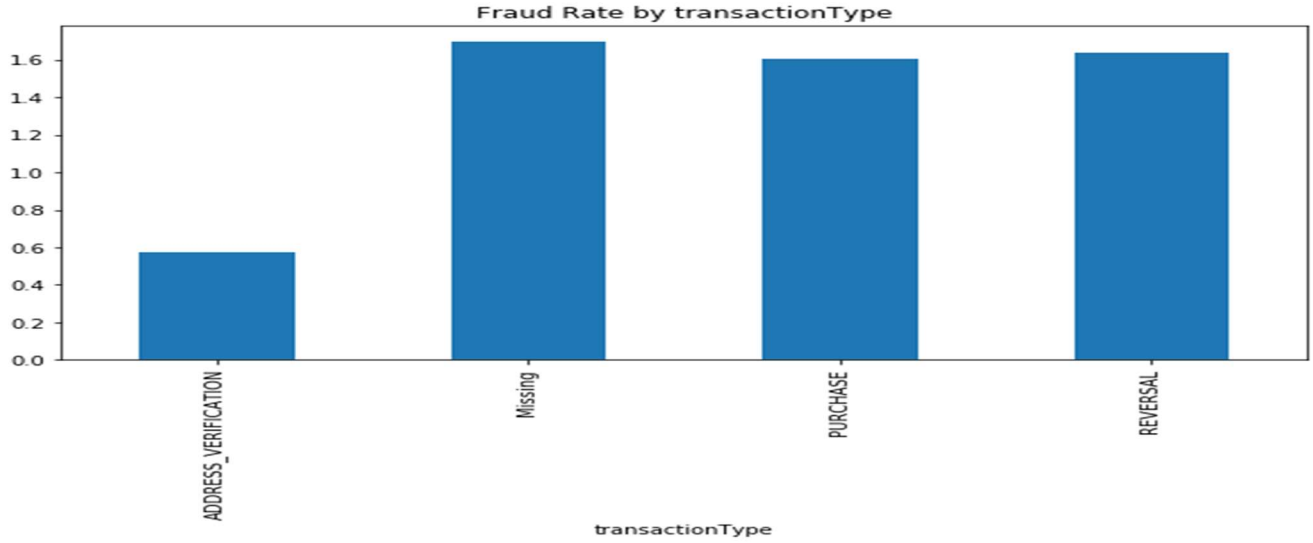
Section 4.1.2: Object Columns

In this section, we will review all the object/string columns and see if there is any opportunity to re-bin or recode any feature.

Transaction Type

The transaction type feature shows that except for address verification cases, all other cases have higher chances for fraud specially when transactiontype =missing.

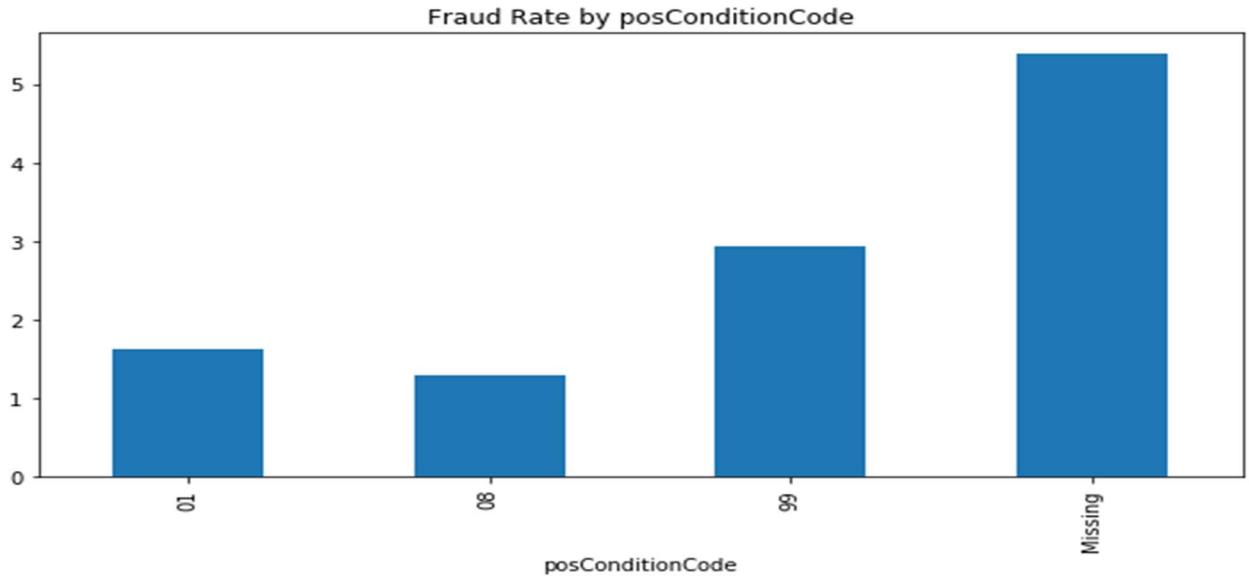
Chart 14: Fraud Rate by transaction type



PosConditionCode

As observed from the graph, when point of sale condition code is missing, then fraud rate is way higher, around 3 times of avg fraud rate

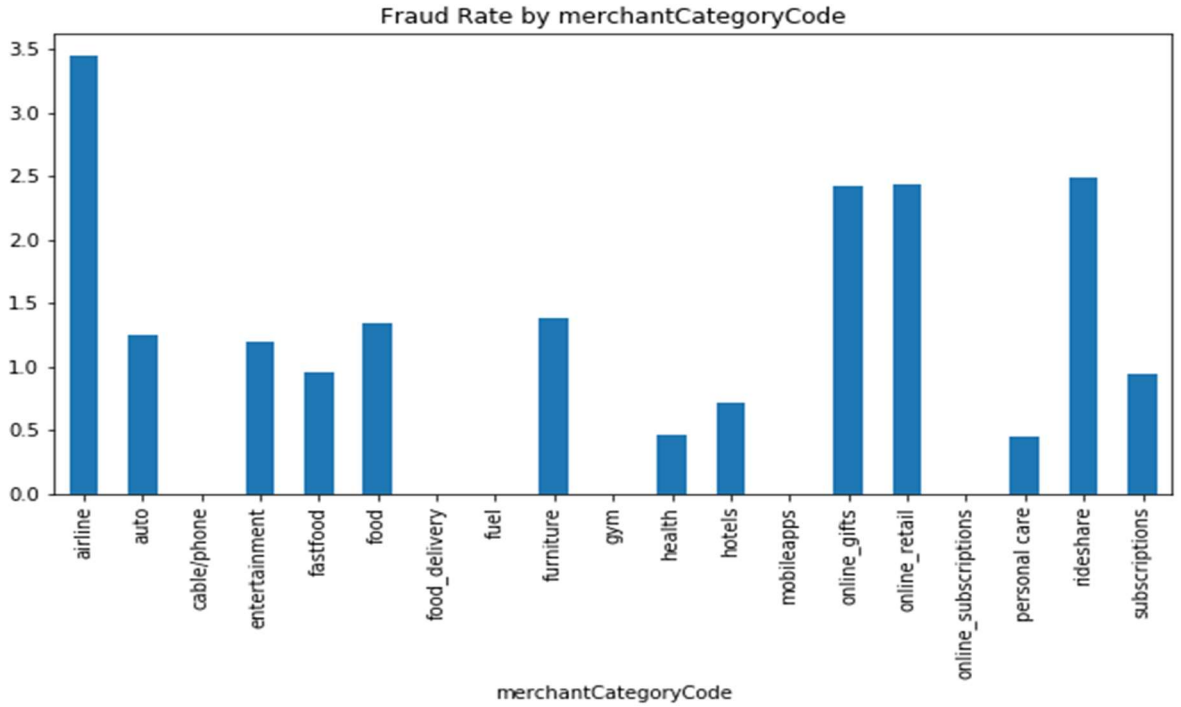
Chart 15: Fraud Rate by posconditioncode



Merchant Category Code

Merchant category code shows that airline has almost 2.5 times higher fraud rate than average fraud rate. There are some categories which do not have any frauds. Ideally, all the categories must be re-binned using more data to reflect long term risk code.

Chart 16: Fraud Rate by posconditioncode

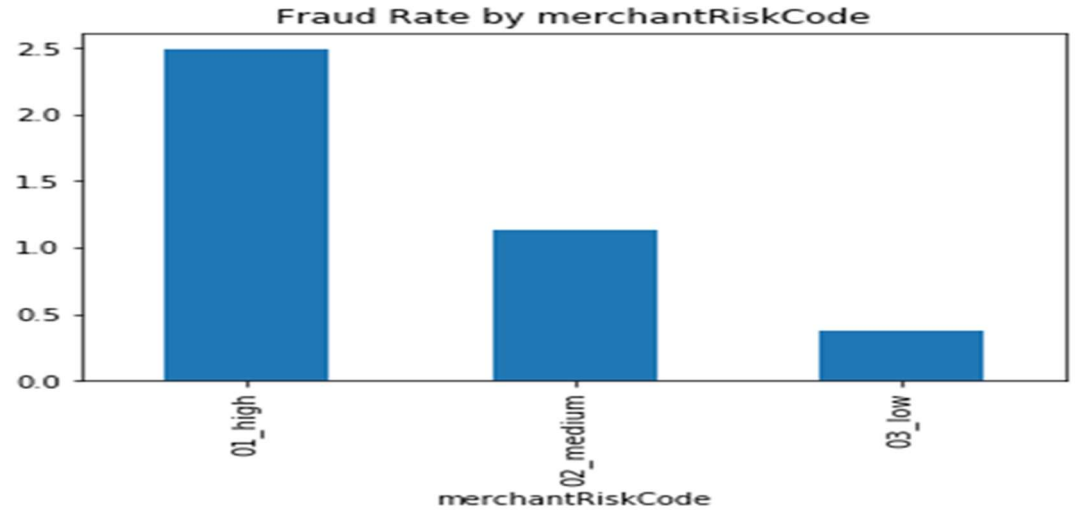


Merchant Risk Category Code

Merchant category code is re-binned based on following logic:

```
if x in ['airline','online_gifts','online_retail','rideshare']: return '01_high'
   elif x in ['auto','entertainment','fastfood','rideshare','food','subscriptions']:return '02_medium'
   else: return '03_low'
```

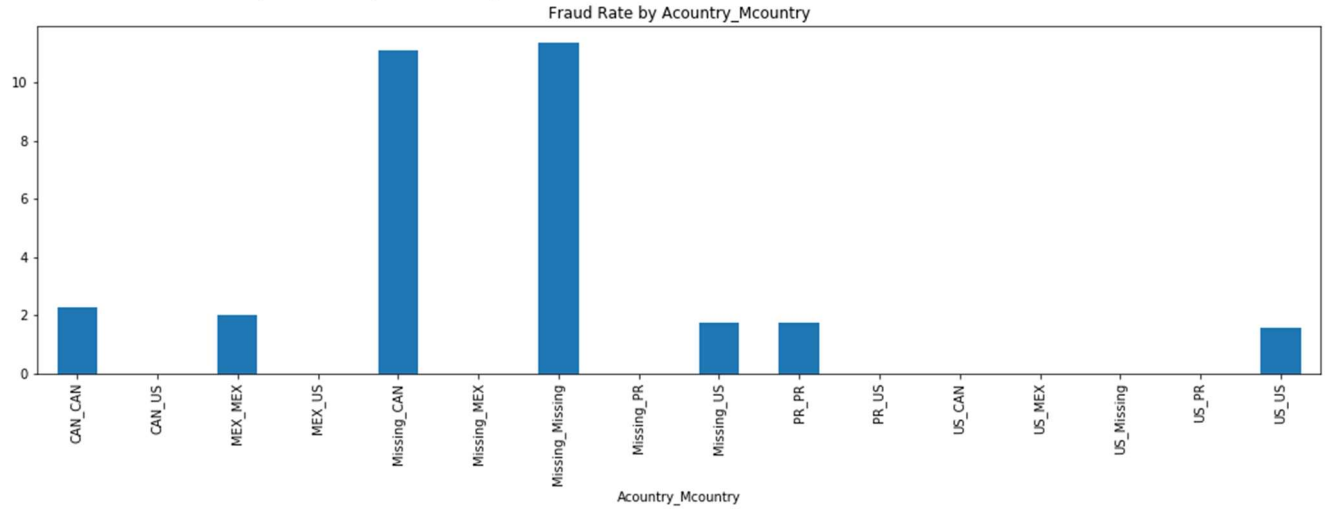
Chart 17: Fraud Rate by posconditioncode



Acountry_MCountry

Merchant country and aq country is used to create a composite feature that shows fraud rate for each of their combinations. When both are missing, the fraud rate is over 11%, over 7 times of avg fraud rate of population.

Chart 18: Fraud Rate by Acountry_Mcountry



MerchantName

There were 2490 categories of merchant names, so I first reviewed the top 10 categories by fraud. As shown in the table below, many of these merchants appear multiple times since they have an order number starting with '#' sign. I cleaned it up to create another column M_name_clean that had fewer level.

Chart 19: Fraud Rate by M name before clean up

merchantName	count	sum	mean
In-N-Out #949129	89	10	0.112
American Airlines	3139	295	0.094
In-N-Out #863086	96	9	0.094
In-N-Out #463194	96	8	0.083
In-N-Out #422833	110	9	0.082
In-N-Out #567597	89	7	0.079
Walgreens #475572	28	2	0.071
Fresh Flowers	8712	616	0.071
In-N-Out #17755	102	7	0.069
In-N-Out #899468	106	7	0.066

Chart 20: Fraud Rate by M name after clean up

M_name_clean	count	sum	mean
American Airlines	3139	295	0.094
Fresh Flowers	8712	616	0.071
ethanallen.com	490	24	0.049
Convenient Auto Services	1058	45	0.043
Rove Concepts	472	20	0.042
In-N-Out	6130	256	0.042
Dinosaur Restaurant	836	34	0.041
ebay.com	16905	639	0.038
Best Pub	823	28	0.034
NY BBQ	774	26	0.034

Section 4.2: Create train and test samples

There were few considerations to make before I could go ahead and create training and test samples. The first consideration is about how much data must be used for creating train and test sample.

Section 4.2.1: Sample model data for loop variables:

I have created several loop variables that compares current transaction with customer's past behavior. The goal is to see if the current transaction varies significantly from past and if that could help identify a fraud. However, since transaction data starts from 1st Jan 2016, the loop variables in initial few months can-not fully capture the true past behavior and hence, deviation from past will not be reliable. Therefore, it can potentially mislead the learning of the model. So, here is one way to handle it:

Build a model with all the variables including loop variables but sample the model data to keep only those customers who have made at least, let's say 30 transactions. This approach would essentially shift the start date to a point where we assume that we are able to capture customer's matured transaction behavior. In this case, the start date can be different for each customer since different customer can reach 30 transactions at different point of time. The downside of this model would be its inability to score relatively new accounts since model data will be based on customers with at least 30 transactions. Therefore, for young accounts or for customer with few transactions, another model can be built that would exclude 'loop' variables.

Section 4.2.2: Imbalanced Data

- Imbalanced data: The fraud rate in the current sample is about 1.5% which indicates imbalanced data. Typically, it is hard to build a good model in imbalance data unless there are some features that have extremely good classification power. There are some techniques that are deployed to handle imbalance:

1. Undersampling: This will remove some non-frauds to increase the fraud rate
2. Oversampling: This will synthetically repeat some frauds in the model data to increase fraud rate
3. SMOTE and ADASYN: Synthetic Minority Oversampling Technique will synthetically add additional observations that looks like fraud using K-means clustering. ADASYN improves SMOTE by adding a little random component to the synthetically created to make it less collinear with sample from where it was created. This would also increase fraud rate.

As such, this adjustment to fraud rate in modelling sample does not impact the risk ranking/ordering but it does impact posterior probabilities which must be adjusted to reflect the 'true' prior distribution. If ultimate goal is to see expected dollar loss prevention through fraud model as: (Probability of fraud) * (transaction amount) then, the interpretation can be misleading if posterior probabilities are not adjusted with 'true' prior distribution.

- Logistic regression model requires more in-depth handling of variables before sending for model. There are also some statistical assumptions built in and sometimes, the MLE may not converge due to high collinearity. Multicollinearity can be handled through regularization (Lasso and Ridge) but they produce biased estimates. It still helps to perform a variable selection through correlation chart or variable clustering to ensure that model variables can be interpreted correctly, and their signs are not counter intuitive.

In this assignment, I could not dedicate enough time to remove collinear variable through high VIF or correlation chart to build a good logistic regression model. That was also partly because I wanted to make sure that all the models use the same training and test sample for fair comparison of their performance. I also standardized all the columns before building the models on them.

Further, I built up a training (70%) and test (30%) split of the data after keeping the data only from the point when customer had at least made 30 transactions. Further, another set of modeling sample was created by applying oversampling technique ADASYN.

ADASYN technique works in a similar fashion as SMOTE. This technique synthetically creates more frauds by using K-means clustering approach to create observations that looks similar to fraud and would tag them fraud. In addition to that, it adds a small random component to each of these synthetically created frauds so that they don't completely match with frauds in terms of similarity. Further, as per part of oversampling, I only oversampled training data and not test data.

In the next section, we will see how our classification models performed.

Section 4.3: Build Models and Model Performance

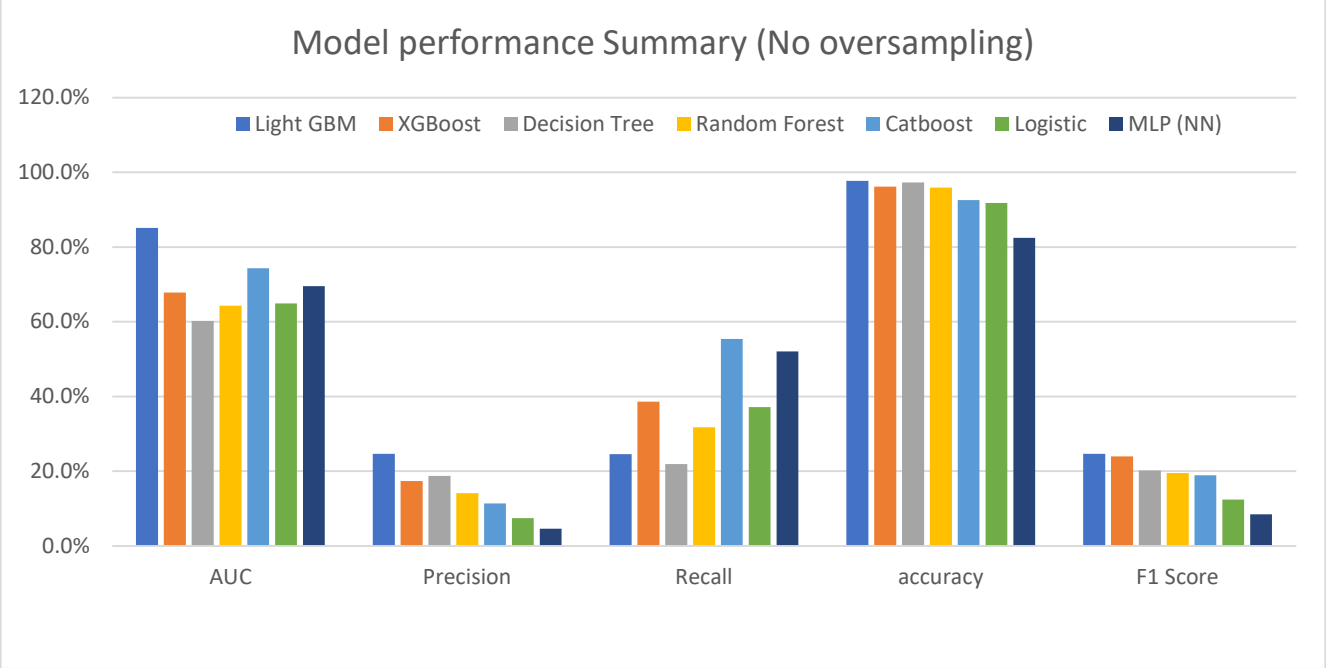
Section 4.3.1: Comparison model's performance

As part of modeling, seven models were built on training and tested on validation sample. The model's performance summary on validation sample is provided below. The models are sorted by their F1 Score, so the model appearing higher in the order performed better in terms of F1 score.

	Model Performance Statistics (No oversampling)				
Model Name	AUC	Precision	Recall	accuracy	F1 Score
Light GBM	85.1%	24.7%	24.6%	97.7%	24.6%
XGBoost	67.8%	17.4%	38.6%	96.2%	23.9%
Decision Tree	60.2%	18.7%	21.9%	97.3%	20.2%
Random Forest	64.3%	14.1%	31.7%	95.9%	19.5%
Catboost	74.3%	11.4%	55.4%	92.6%	18.9%
Logistic	64.9%	7.5%	37.2%	91.8%	12.4%
Neural Network	69.5%	4.6%	52.0%	82.4%	8.5%

As seen above, all the models appear to perform well in terms of accuracy, but don't appear to do well when it comes to precision. Precision measures the % of true frauds among predicted frauds. However, given the fraud rate of around 1.6% (after sampling) , all the model gives some lift over random. For instance, Light GBM has precision of nearly 25% implying that Light GBM would capture 25% frauds among predicted fraud while a random sample of same size would only have 1.6%, therefore giving a lift of nearly 15 times. So, an observation is 15 times more likely to be fraud when a sample of certain size is drawn from light GBM sorted by probability to fraud. The chart below provides visual summary of the same set of statistics.

Chart 20: Model performance Statistics

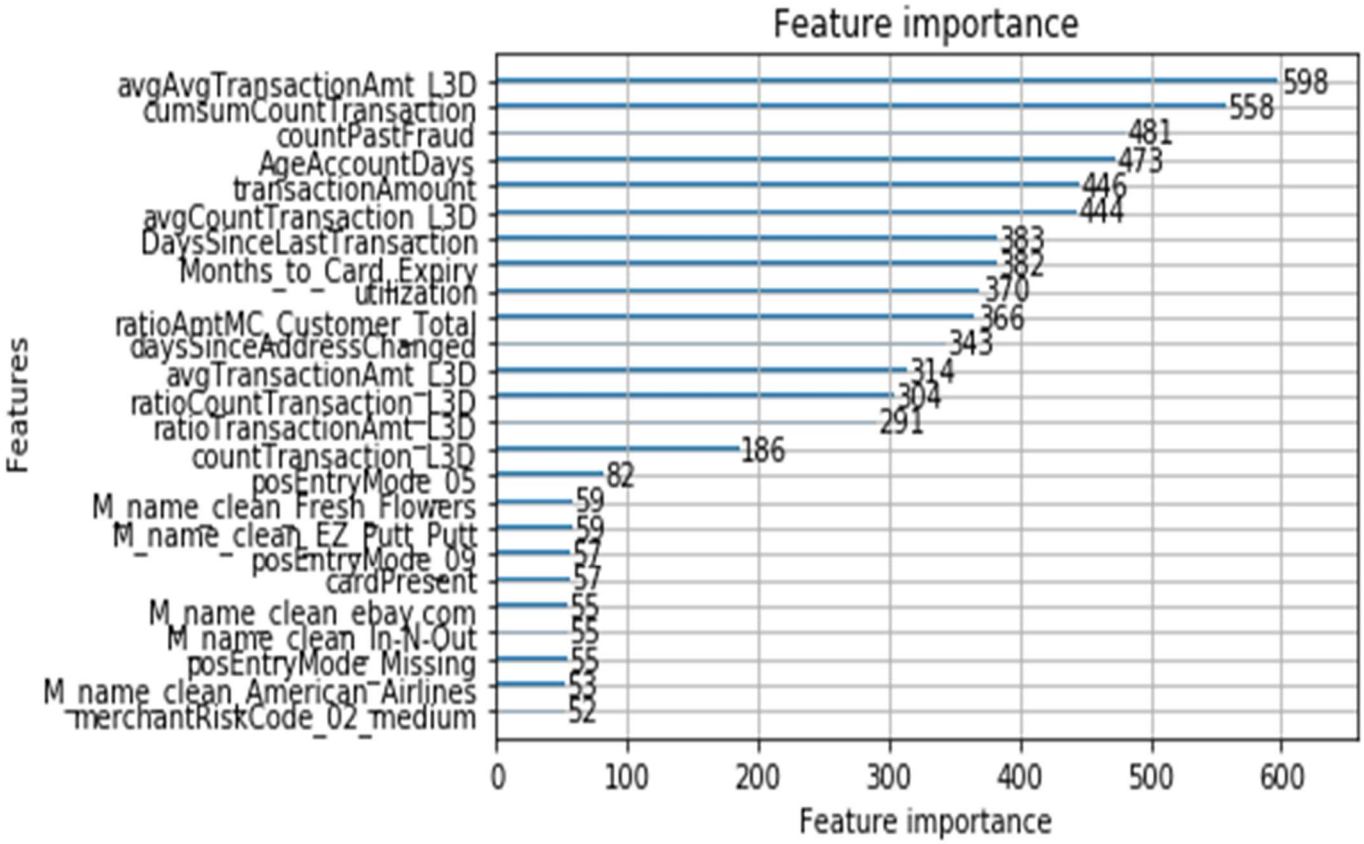


Section 4.3.2: Light GBM model performance

As Light GBM performed the best in all the statistics (except on recall where Catboost did better), I would cover it in a little bit more detail here. All other model outcomes are provided in the Jupytor notebook sent alongwith the this document. Light GBM is boosting algorithm that grows horizontally (leaves wise) instead of vertically (level wise). It is suitable for big size data and imbalanced data. As hyper parameter, following parameters were passed:

'objective': 'binary',
'metric': 'auc',
'is_unbalance': 'true',
'boosting': 'gbdt',
'num_leaves': 31,
'feature_fraction': 0.75,
'bagging_fraction': 0.5,
'bagging_freq': 20,
'learning_rate': 0.05,

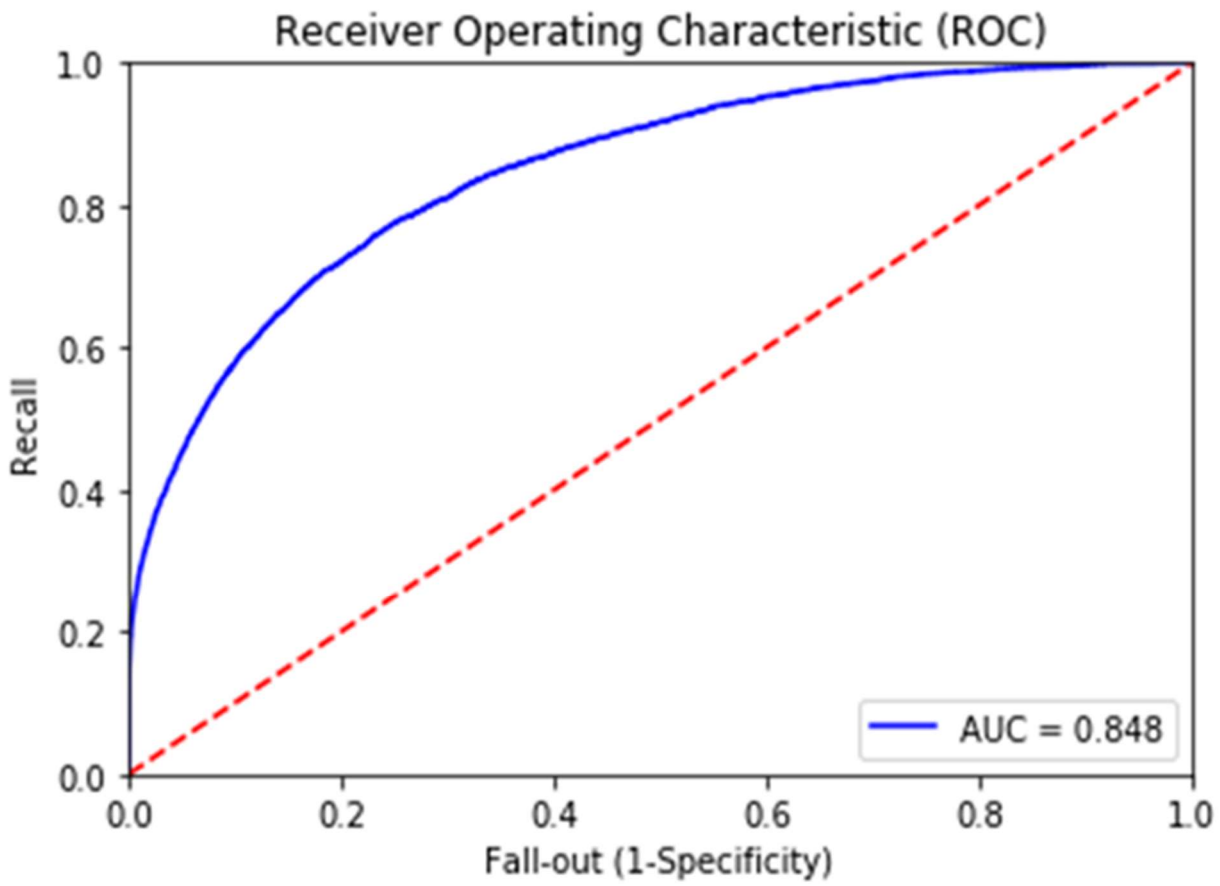
The model was tested with additional Feature fraction (0.80, 0.50), bagging fraction (0.75) and learning rate (0.01) but there was not much difference found in terms of model's performance. The feature importance plot resulted into the chart below:



Listing some of those features here again:

1. Avg Transaction momentum over last 3 days
2. Transaction velocity over last 3 days
3. Utilization
4. Count of past Frauds
5. Ratio of customer's expense in given MC with avg expense in the same MC by entire population
6. Transaction Amount
7. Certain Merchants (American Airlines, Fresh Flowers, ebay.com, In-n-Out)
8. Post Entry code=missing
9. Days since Address Changed

Chart 21: ROC curve for light GBM



Section 4.3.3: Model Performance on oversampled data:

Model performance didn't appear to improve in test sample when it was built using oversampled training data. I need to take a closer look at model's result to better investigate but overall, none of the models show any improvement in performance in validation sample when built using oversampled data. One thing to note here is how ADASYN creates oversamples as default. The default option creates as many 1s and 0s (50% Fraud Rate)and that can also make it hard for model to train. I guess, I should have attempted under sampling as an alternative to oversampling that reduces non-frauds to increase fraud rate.

	AUC	
Model Name	full sample	oversample
Light GBM	85.1%	84.8%
Decision Tree	60.2%	57.7%
XGBoost	67.8%	60.0%
Catboost	74.30%	71.1%
MLP (NN)	69.5%	50.0%
Random Forest	64.3%	67.6%

Section 5: Discussion on Future Improvement

Throughout the modeling process, several assumptions and modelling choices were made. However, since this assignment was to be completed in 7 days, not all assumptions and choices were fully tested. In fact, I could only perform one iteration of model run during this period. Further, I did not have access to more data or ability to contact business and data team to discuss some of the findings, as such, the models remain incomplete without their inputs. Due to that, there exists scope of improvement in the modelling process. However, in real life project, I would do following:

Section 5.1: Additional Time

- - With additional time, I would perform few more iterations of model runs using different set of features since some features were dropped from the training since their similar features were available (Ex: transaction velocity over 3 days was kept while dropped for 1 day and 7 days).
- - I would spend more time on tuning models by optimizing hyper parameter space through grid search. I would also run k-cross validation for training the models.
- - I would spend more time in building a good logistic regression model as my baseline model. Logistic regression model requires features to be carefully selected through variable clustering, and correlation chart.
- - I would build a model without using loop variables first and then, introduce loop variables to see incremental improvement in performance.
- I would like to build ensemble model after fine tuning all the underlying models.
- I would like to perform under sampling and see if model's performance improves.
- I would like to create additional hold out sample for testing of model's performance.
- I would try building model with under sampling and compare the model performance.

Section 5.2: Additional Data

- - If I had access to more data, I would see if there is a seasonal impact to fraud behavior.

- - I would also like to see the long-term fraud rates in each of the categorical columns since our sample may not reflect that.
- - I would prefer to combine the global data of Capital One to see the list of all merchant's and merchant categories to arrive at more exhaustive list of merchants. Since, people travel all the time globally, having a global list of merchants really helps.
- - I could make use of demography of the customer and see if the distance from home as a factor for fraud.
- - With merchant's address/zip code, I could create distance between two merchants when consecutive transactions occur in quick succession at two different merchants.
- - With more data, I could create true lifetime aggregates for each customer starting from account open date.

Section 5.3: Additional discussion with data team, modelers and Business

- - In my opinion, any modeling work should be collaborative process where several, if not all, modeling choices must be made after several discussions with data team, business and other reviewers in the group.
- - I would like to hear from data team if count_past_fraud can be used for modelling. Even though it is past fraud in the data, will this column reflect the most recent fraud at the time of scoring? Essentially, if there is a lag in reporting a transaction as fraud by customer, then model will not be able to make use of this feature at the time of scoring since this would not be updated at the time. If this feature gets updated fairly quickly, I would create additional features like time since last fraud etc.
- - I would like to work with business team to make a decision on what number of transactions constitute a satured behavior of customer.
- - I would like to consolidate the definition of reversed transaction with business and improve the logic that captures that.