Sherly Hartono

Capstone 1:
Credit card fraudulence prediction

Content:

1. Introduction

Credit card fraud by definition is the fraudulent use of a credit card through the theft of the cardholder's personal detail. With the rise of e-commerce, the problem of fraudulent use of credit cards has become more acute.

Given previous transaction datas, can we predict whether or not a new transaction is a fraudulent one ?

Losses to fraud incurred by payment card issuers worldwide reached $27.85 billion in 2018 and are projected to rise to $40.63 bullion in 10 years. Answering the above problem can reduce these damages faced by merchants and credit card issuers. Our goal is to correctly classify the minority class of fraudulent transactions.

	accountNumber	customerId	creditLimit	availableMoney	transactionDateTime	transactionAmount	merchantName	acqCountry	posEntryMode	...
0	737265056	737265056	5000.0	5000.0	2016-08-13T14:27:32	98.55	Uber	US	02	...
1	737265056	737265056	5000.0	5000.0	2016-10-11T05:05:54	74.51	AMC #191138	US	09	...
2	737265056	737265056	5000.0	5000.0	2016-11-08T09:18:39	7.47	Play Store	US	09	...
3	737265056	737265056	5000.0	5000.0	2016-12-10T02:14:50	7.47	Play Store	US	09	...
4	830329091	830329091	5000.0	5000.0	2016-03-24T21:04:46	71.18	Tim Hortons #947751	US	02	...

5 rows × 29 columns

Fig 1 Original dataset of 731560 rows X 29 columns

2. Data Wrangling

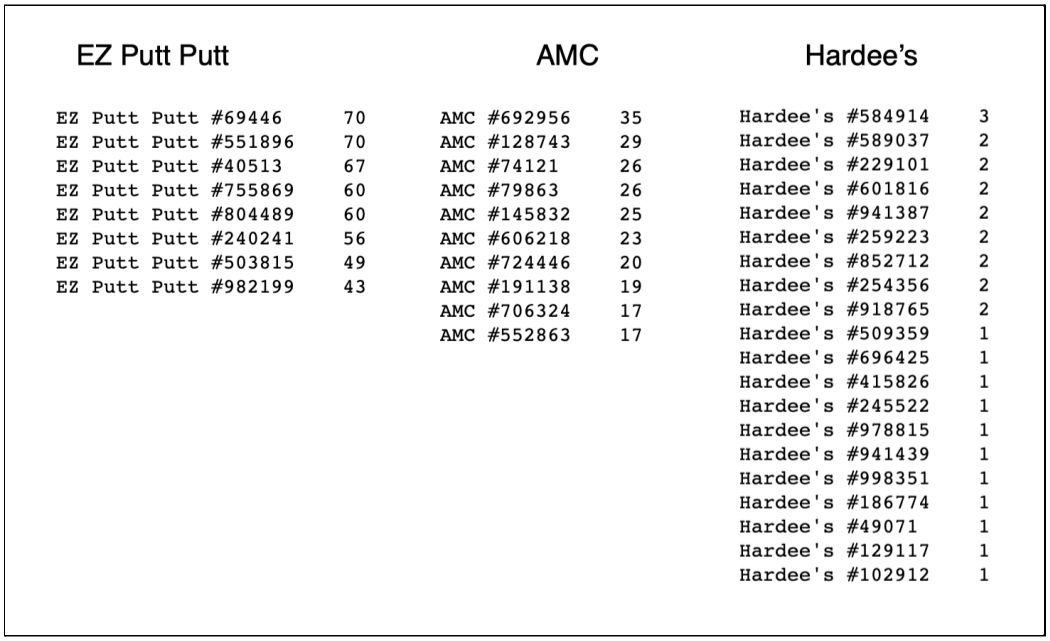
2.1 Missing Values and Data Type problems

I removed all columns with empty strings as we cannot really replace it with any values. I replaced all missing strings with Nan to see if there are any other columns with empty strings. I found 5 columns with Nan values, namely acqCountry, merchantCountryCode, posEntryMode, posConditionCode, and transactionType. For the first two, I simply replaced them with the mode as over 98% of the columns have the mode value as its entry. For the remaining, although not ideal I simply remove the rows. Given more time, I would use a classifier to predict what are the predicted values for all of these columns. There were no major data type problems other than changing datetime data type from object to datetime object.

2.2 Duplicate Datas

I removed unnecessary columns such as customerID as it has the same cardinality as accountNumber. 'enteredCVV' and 'cardCVV' can also be removed and create a new column of binary values called CVVMatch as this is essentially the only information we want from those two columns.

Some merchants have duplicate entries of different branch ID (fig 2.1). It is better to combine them under one name as there is no great variance between each branch.



EZ Putt Putt		AMC		Hardee's	
EZ Putt Putt #69446	70	AMC #692956	35	Hardee's #584914	3
EZ Putt Putt #551896	70	AMC #128743	29	Hardee's #589037	2
EZ Putt Putt #40513	67	AMC #74121	26	Hardee's #229101	2
EZ Putt Putt #755869	60	AMC #79863	26	Hardee's #601816	2
EZ Putt Putt #804489	60	AMC #145832	25	Hardee's #941387	2
EZ Putt Putt #240241	56	AMC #606218	23	Hardee's #259223	2
EZ Putt Putt #503815	49	AMC #724446	20	Hardee's #852712	2
EZ Putt Putt #982199	43	AMC #191138	19	Hardee's #254356	2
		AMC #706324	17	Hardee's #918765	2
		AMC #552863	17	Hardee's #509359	1
				Hardee's #696425	1
				Hardee's #415826	1
				Hardee's #245522	1
				Hardee's #978815	1
				Hardee's #941439	1
				Hardee's #998351	1
				Hardee's #186774	1
				Hardee's #49071	1
				Hardee's #129117	1
				Hardee's #102912	1

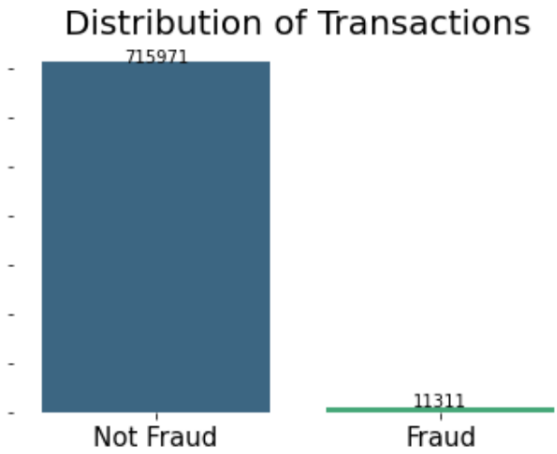
Fig 2.1 There is no much difference between the transactions from the most popular to the least popular branch.

2.3 New Features

Many of the most popular merchants have '.com' attached to its name. Adding a new feature that tells us whether or not a merchant have '.com' on its name may improve our model performance. So I added a new feature 'containsCom'. It is curious that 'cardLast4Digits' has some entries with digits less than 4. So I also created a new feature 'lengthOfLast4Digits'.

3. Exploratory Data Analysis

3.1 Distribution



Looking at the target features, we can see that we have a very imbalanced data. Only 1.5% of the data belong to our target class. This is a problem because many machine learning models are designed to maximize overall accuracy which is not the best metric to use for imbalance data. If we simply predict all transactions are not fraud, we can easily get an accuracy score of over 98% and miss the whole point of trying to identify the fraudulent case. Let's keep this in mind for when we do our training and modelling.

3.2 Measuring spread

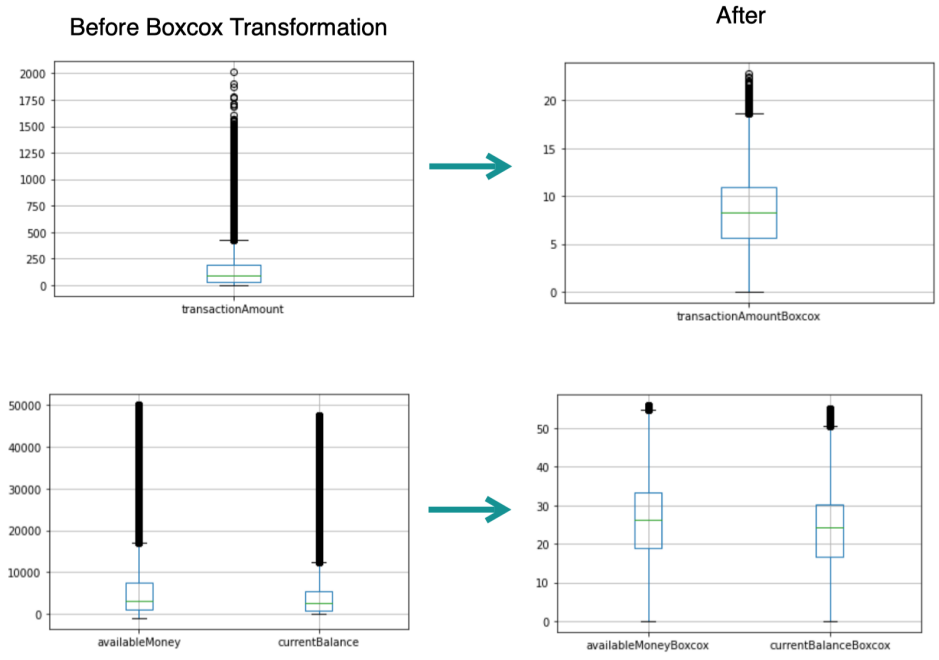


Figure 3.2 Box plot of continuous variable before box cox transformation (left) and after transformation (right).

If we look at the boxplot of our numerical datas, we have many outliers (left hand side of figure 3.2). We can see that this is what we called collective outliers. That is, there are subsets of data points that deviate

significantly from the entire data set, but the values of the individual data points are not themselves anomalous. For example in the transactionAmount class, we have 23,545 outliers that fall above $500. We should treat these outliers as it increases error variance and reduces the power of statistical tests. Most machine learning algorithms do not work well in the presence of outliers. So it is desirable to detect and remove outliers. Box cox transformation gives better results than log transformation. We can see that the resulting data has much less outliers and it has been normalized.

3.3 Datetime trends

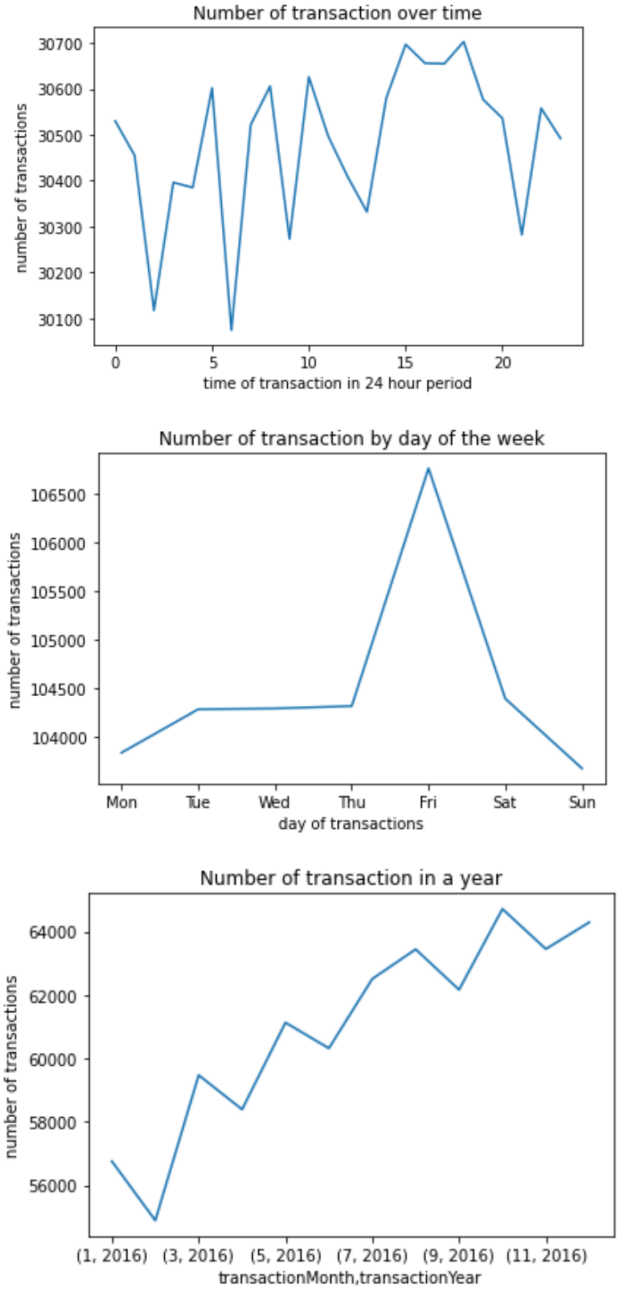


Fig 3.3 From top to bottom: First figure looks at the trend over a 24 hour period. Second figure tracks the trend on days of the week and the third figure looks at the trend throughout the whole year.

I split the time series data into day of the week, time of day and month and year to better identify if the data is stationary (if there is any observable trend). The 24 hour data looks stationary whereas the data for the day in a week and for a year are not stationary. There seems to be a peak in the number of transactions on Friday. We can also observe a sharp increase in the number of transactions in a year from 56,000 in January to 64,000 by the end of the year. I will add these features to the dataset as it could increase to the predictive power of the model.

3.4 Correlation Matrix

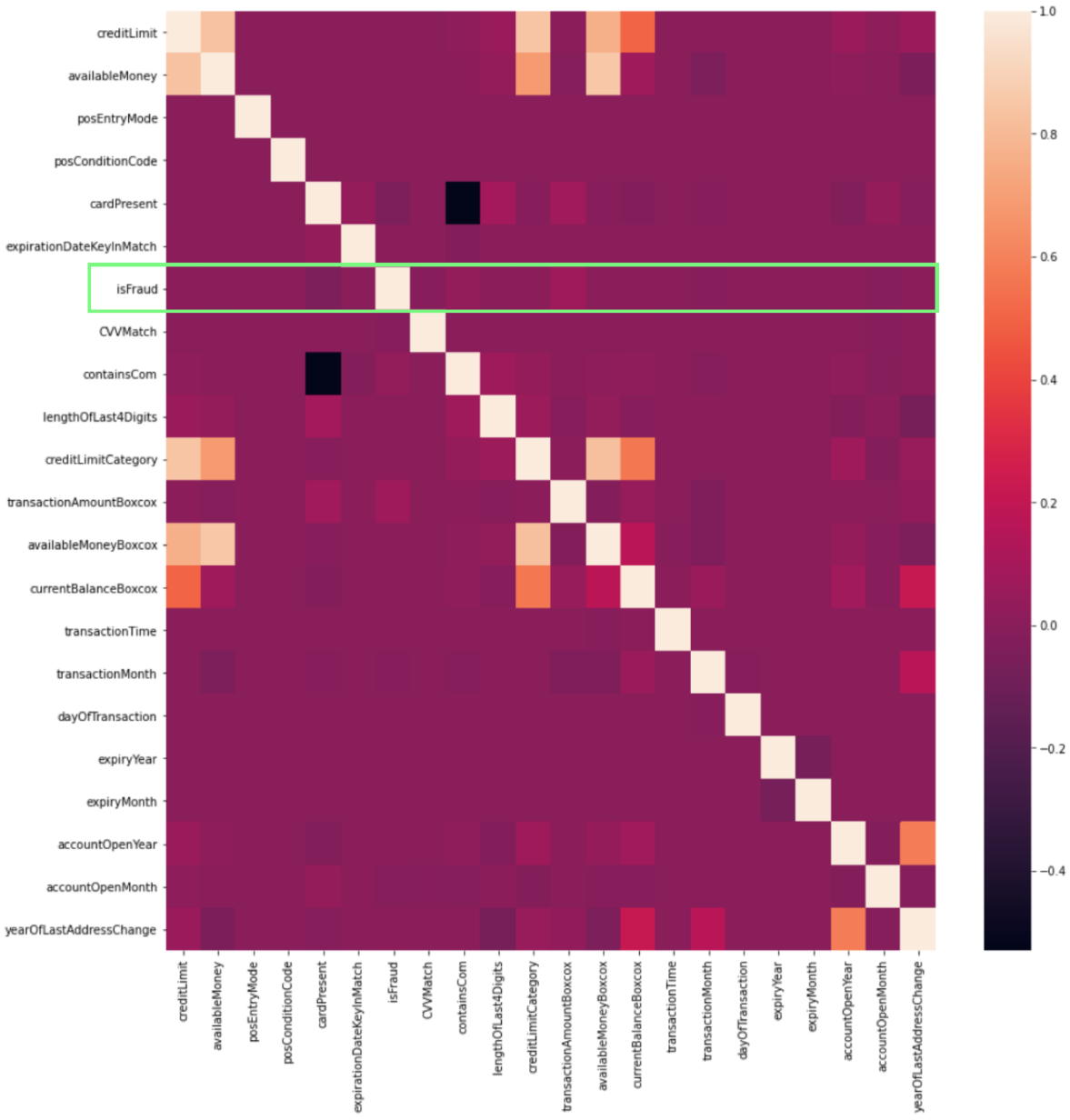
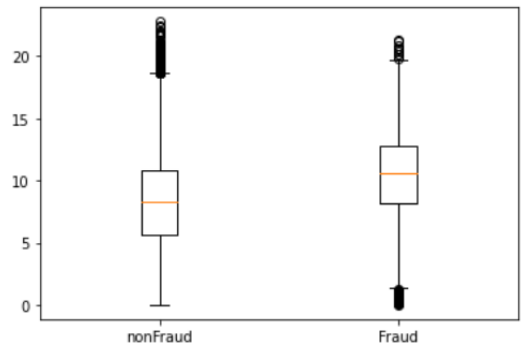


Fig 3.4 Correlation matrix of the dataset

Looking at the isFraud row that I framed in green in figure 3.4, we can observe positive correlation between fraudulent transaction and 'transactionAmountBoxCox' and 'containsCom', the feature that we just added! We can also see a negative correlation with 'cardPresent'. This makes sense as it implies that transactions that are fraud are usually done online when the credit card is not physically present. Also higher amounts of transactions are likely to be fraudulent. We can test this hypothesis using T-Test.

3.5 Hypothesis Test

We can see from the figure below that the mean transaction amount is higher in fraudulent transactions than that of non-fraud. Let's check if this is not just due by chance using two sample T-test.



The mean differences in transactions amount is $\mu\_fraud - \mu\_notFraud$ = 225.38−135.57 = 51.43 dollars

H_0: Mean of transaction amount in fraud and non fraudulent are the same

H_1: Mean of transaction amount in fraud and non fraudulent are NOT the same

First, get the variance of both fraud and non fraud transaction:
Variance of not fraud = 11.97
Variance of fraud = 13.73

We can use the T-test with unequal variance, which is a two-sample location test which tests the hypothesis that two populations have equal means. We now calculate the standard deviation which is equal to 3.58. Plug this in to get the t-statistics then the p value.

$$\text{t-stat} = \frac{(\bar{x}_A - \bar{x}_B) - (\mu_A - \mu_B)}{\sqrt{\frac{s_A^2}{n_A} + \frac{s_B^2}{n_B}}}$$

The result of P-value is less than 0.05. Thus we can reject the null hypothesis. Therefore we can conclude that there is a significant difference between the transaction amount of fraud and non fraudulent transaction.

4. Training and Evaluation

1. Encoding and Variance Threshold

Most of our datas are of ordinal types, that is, they are categorical and they cannot be ranked. We need to convert them as ML models require input and output variables to be numeric. We will use one-hot encoding to map our labels to a binary vector.

I also use a variance threshold to remove features with variation of 0. This will help in reducing dimension after doing encoding. A feature that doesn't vary much generally has very little predictive power so we can safely remove this.

I tested 3 different machine learning algorithms:

- Logistic Regression
- Random Forest
- XG Boost

2.1 Modelling: Logistic Regression

Keeping in mind that our data is imbalanced, we need to have a method in facing this problem. I tried 3 methods. The first is setting the class_weight parameter as 'balanced' which replicates the smaller class until we have as many samples as the larger ones. I used cross validation of 5 fold and checked the resultant roc_auc score.

Using the Logistic Regression model, I achieved the best score using undersampling techniques (roc_auc of 0.74). Below is the confusion matrix we get by using undersampling method vs setting class_weight parameter to 'balanced'. We reduced the number of false negatives from 197 to 137 by using undersampled data and increased the true positives from 488 to 548. This comes at the cost of reducing the number of true negatives (transactions are correctly identified as non-fraudulent and increasing false positives (transactions are not fraudulent but identified as fraud). Hower identifying the positives correctly are much more crucial for us.

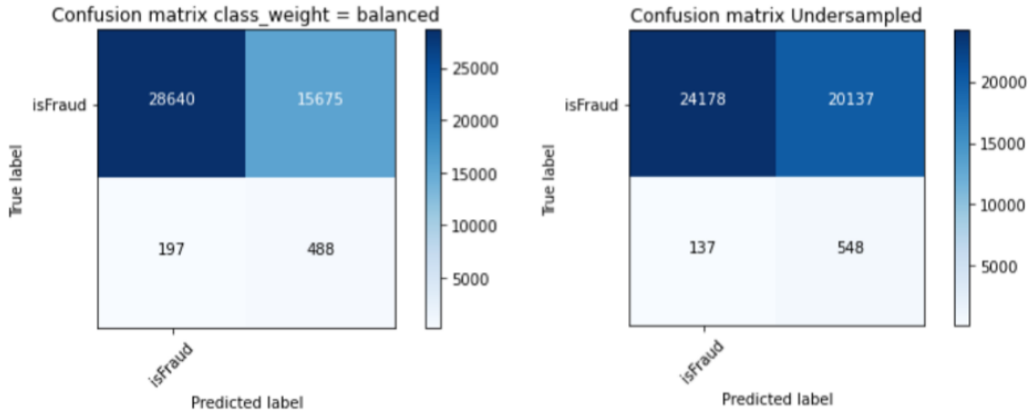


Fig 4.2.1 Confusion matrix using Logistic Regression with parameter class_weight set to balanced (left), and Logistic Regression using undersampled data.

2.2 Modelling: Random Forest

For the random forest model, we have many more hyperparameters to tune compared to Logistic Regression. So using RandomizedSearchCV would be a faster option as it only selects random combinations of the hyperparameter to train the model and score. Here are the parameters we will need to tune:

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

We get a much higher roc_auc score of 0.8 using this model.

2.3 Modelling: XG Boost

XGboost makes use of a gradient descent algorithm which is the reason that it is called Gradient Boosting. The whole idea is to correct the previous mistake done by the model, learn from it and its next step improves the performance. The previous results are rectified and performance is enhanced. There are a lot of hyperparameters to be tuned here. So again we will use RandomizerSearchCV for optimization. Our roc_auc at 0.79 is roughly the same as RandomForest.

3. Best Model

The ROC graph below shows that our RandomForest model performs the best. ROC curve is an evaluation metric for binary classification problems. It is a probability curve that plots the true positive rate against false positive rate at various threshold values. The area under the curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

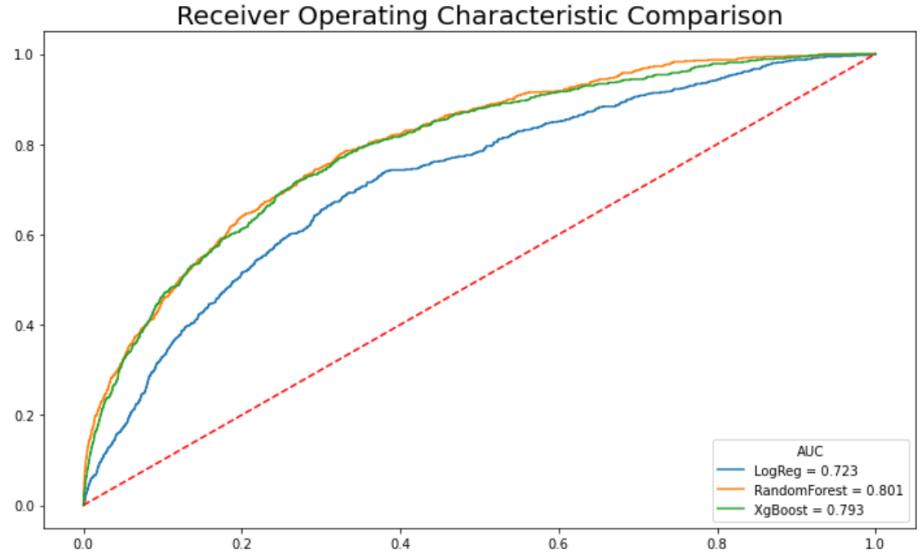


Fig 4.3 Comparison of ROC AUC curve of the three models

4. Feature Importance

The top important features agree with our correlation matrix we did in EDA. Higher amounts of transactions did give us a clue on whether or not a transaction is fraudulent. Its importance is more than double than other features. The next 2 strongest features represent similar information, account's current balance is strongly correlated to available money as shown in our correlation matrix earlier. Surprisingly, whether or not credit is present does not really help in our prediction. Datetime variables like accountOpenMonth, transactionTime and expiryYear have more predictive power.

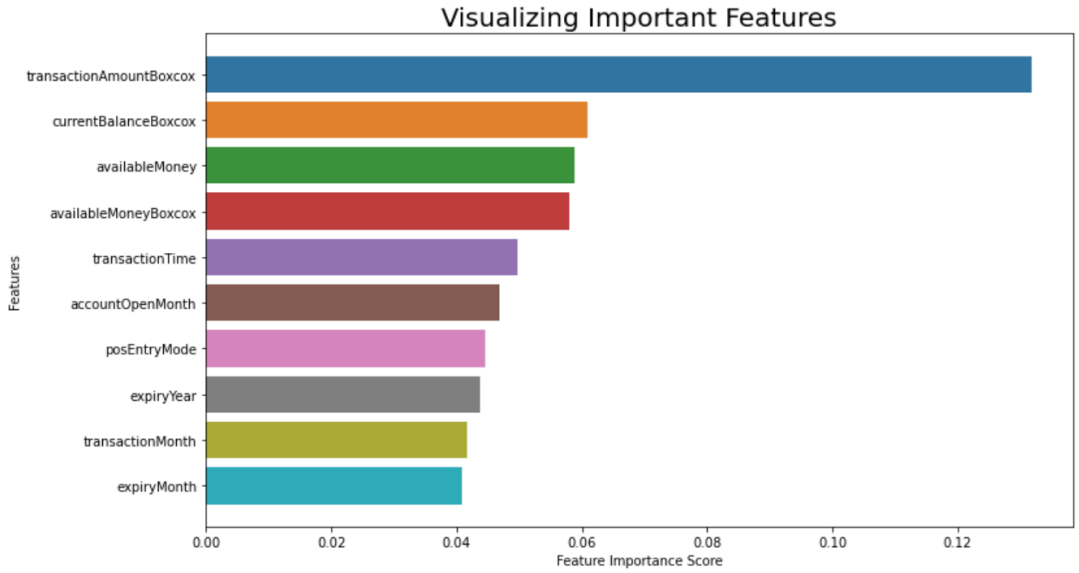


Fig 4.4 top ten important features in RandomForest model

5. Learning curve

Would more data be useful? We're often led to believe more data is always good, but gathering data invariably has a cost associated with it. Assess this trade off by seeing how performance varies with differing data set sizes. I used sklearn learning_curve to investigate this. As we can see from the graph, we can see that we will get a higher score with more data. The training and test score also seem to converge at the end suggesting our current best model, Random forest, is still underfitting. I cut down the data by more than three quarters as I am running many fits for each of my models that it takes more than 6 hours. With better CPU speed, I would be able to use all of the data provided

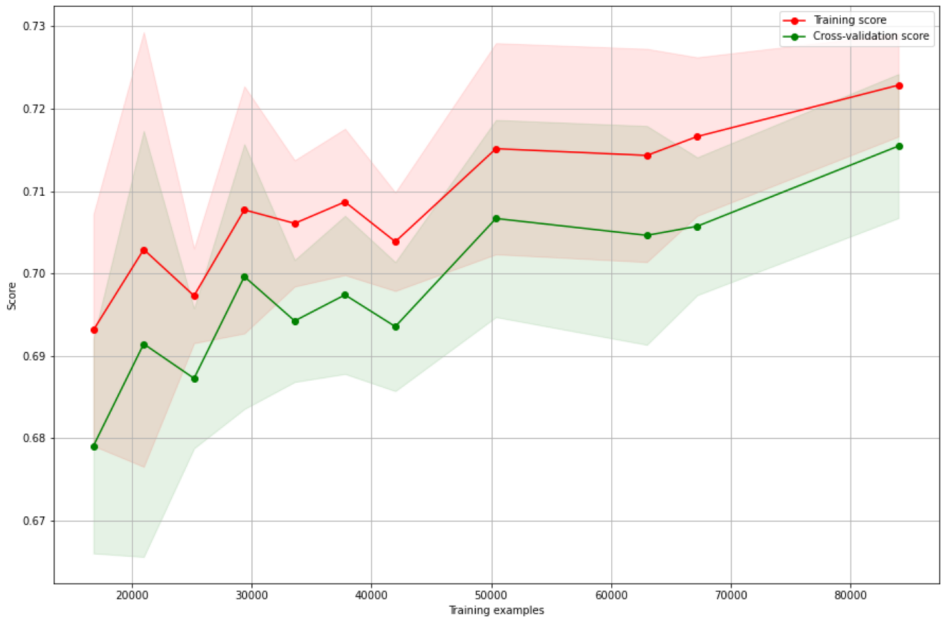


Fig 4.5 Learning curve for RandomForest model

5. Future work

One of the biggest problems that I encountered is the high dimensionality of data which significantly increases the processing time and forces me to only use less than half of the data entries. I could use target encoding instead of one-hot encoding to tackle this problem. In order for this to work however, I should ensure that I only implement this on features that do not have rare values in the data. PCA will not work here as it tries to minimise squared deviation which does not make sense when we have binary variables.

Another way to reduce the time taken for processing is to use Bayesian Optimization instead of stopping at RandomSearch in our hyperparameter tuning. Bayesian Optimization generates smaller validation set errors and runs faster than random search. RandomSearch is completely uninformed by past evaluations, and as a result, often spends a significant amount of time evaluating 'bad' parameters. Bayesian approaches, in contrast, keep track of past evaluation results which they use to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function.

For the cleaning segment, I would like to try using classification models to predict the missing values instead of only using mode. Using mode is a simple and common method of data imputation but it changes the statistical nature of the data. Not only can it skew our histograms. It also underestimates the variance in our data. Another thing that I could have done is to first explore the missing values to uncover whether or not the data is MNAR (Missing Not At Random) in which case I should not use mode imputation. For example, 'posConditionCode' may have a very strong correlation with the 'cardPresent' variable.