*Final Report for ITI-105 Project*

# Credit Card Fraud Detection

by
Liew Chooi Chin

This report is organised into the following sections:

- ML formulation
- Data preparation and feature engineering
- Modelling and experiment
- Deployment
- References

## ML formulation

### Understand the Problem

With the proliferation of online shopping and data leaks, credit card (CC) numbers are easily being stolen and used illegally. Fraudulent CC transactions are divided into CP (card present) or CNP (card not present) scenarios.

### State the problem

The goal is to identify a transaction as fraud or non-fraud (legitimate).

### Clear use case for ML

With the enormous amount of payment that goes through the payment system, ML checking of transactions provides an effective way to flag suspicious transactions. The flagged transaction can then be acted upon by a human investigator. For example, the human investigator may be a customer service staff who will call the cardholder to ask if she indeed carries out the transaction.

## Framing an ML Problem

### Define the ideal outcome and the model's goal

Application: Detection of fraudulent CC transaction

Ideal outcome: Identify between legitimate transactions made by cardholders and fraudulent transactions by scammers.

Model's goal: Predict if a transaction is fraud or non-fraud.

### Identify the model's output

The output of the model should be Fraud (1) or Non-fraud (0).

Thus, we will cast our problem as a binary classification problem.

### Understand the problem's constraints

Our dataset was synthetically generated. The ML model learning from this dataset could not generalise the learning to the real world scenario.

### Define success metrics

Success metrics were defined as follows:

- Success: When a transaction is predicted as a fraud, the transaction is indeed a fraud.
- Failure: When a transaction is indeed a fraud, it is not predicted as a fraud.

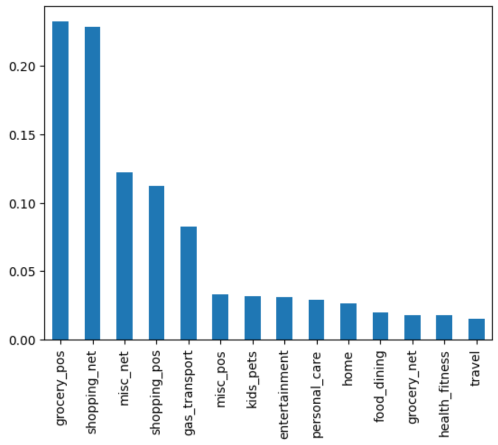
## Data preparation and feature engineering

The dataset used is a made-up dataset found on Kaggle. Columns in the dataset:

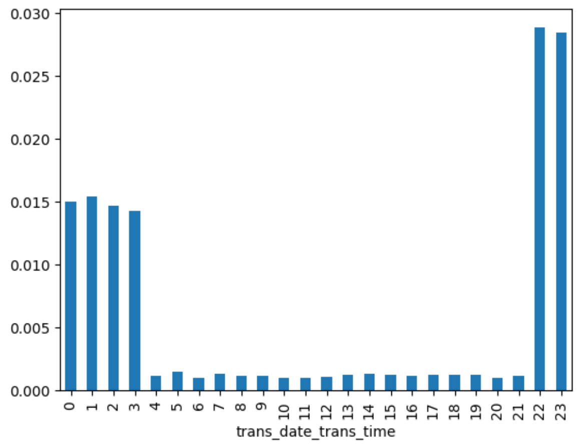
- index - Unique identifier for each row
- trans_date_trans_time - Transaction date time
- cc_num - Credit card number of customer
- merchant - Merchant name
- category - Category of merchant
- amt - Amount of transaction
- first - First name of credit card holder
- state - State of credit card holder
- zip - Zip of credit card holder
- lat - Latitude location of credit card holder
- long - Longitude location of credit card holder
- city_pop - Credit card holder's city population
- job - Job of credit card holder
- dob - Date of birth of credit card holder
- trans_num - Transaction number
- unix_time - UNIX time of transaction

- last - Last name of credit card holder
- gender - Gender of credit card holder
- street - Street address of credit card holder
- city - City of credit card holder

- merch_lat - Latitude location of merchant
- merch_long - Longitude location of merchant
- is_fraud - fraud flag <--- target class



Some categories of shops are more prone to fraudulent transactions.



When the frauds were grouped into hours in a day, we found that a high number of frauds occur between 22 to 03 hours at midnight.

**Features created for learning**:

**cat_code**: Categories of shopping or merchants. The categories were in text and were converted to ordinal encoding.

**is_midnight**: the hours were extracted from trans_date_trans_time. Hours between 22 to 03 were marked as 'is_midnight'.

**count_3d, count_7d, count_30d**: Rolling window periods of 3, 7 and 30 days. This was the number of average transactions of a customer. If the number of transactions was unusually high, the transactions were probably frauds.

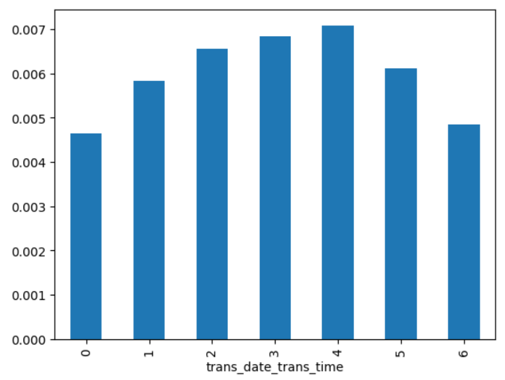
**amt_3d, amt_7d, amt_30d**: Rolling window periods of 3, 7 and 30 days. This was the average number of transactions of a customer. If the amount of transactions is unusually high, then the transactions were probably frauds.

**amt**: This was the actual transaction amount occuring at a point of sale.

**distance_3d, distance_7d, distance_30d**: Rolling window periods of 3, 7 and 30 days. This was the average distance between a customer and a point of sale. If the distance between a customer and a point of sale was unusually high from the average distance, the transactions were probably frauds.

**distance**: This was the actual distance between a customer and a point of sale when a transaction was carried out. When a distance is unusually far for a transaction, the transaction is likely to be suspicious.

**is_fraud**: The label of whether a transaction was fraud or legitimate.

| | |
|---|---|
|  There appears to be an equal amount of fraud happening any day of the week. This feature (day_of_week) is not particularly useful in prediction. | |

## Modelling and experiment

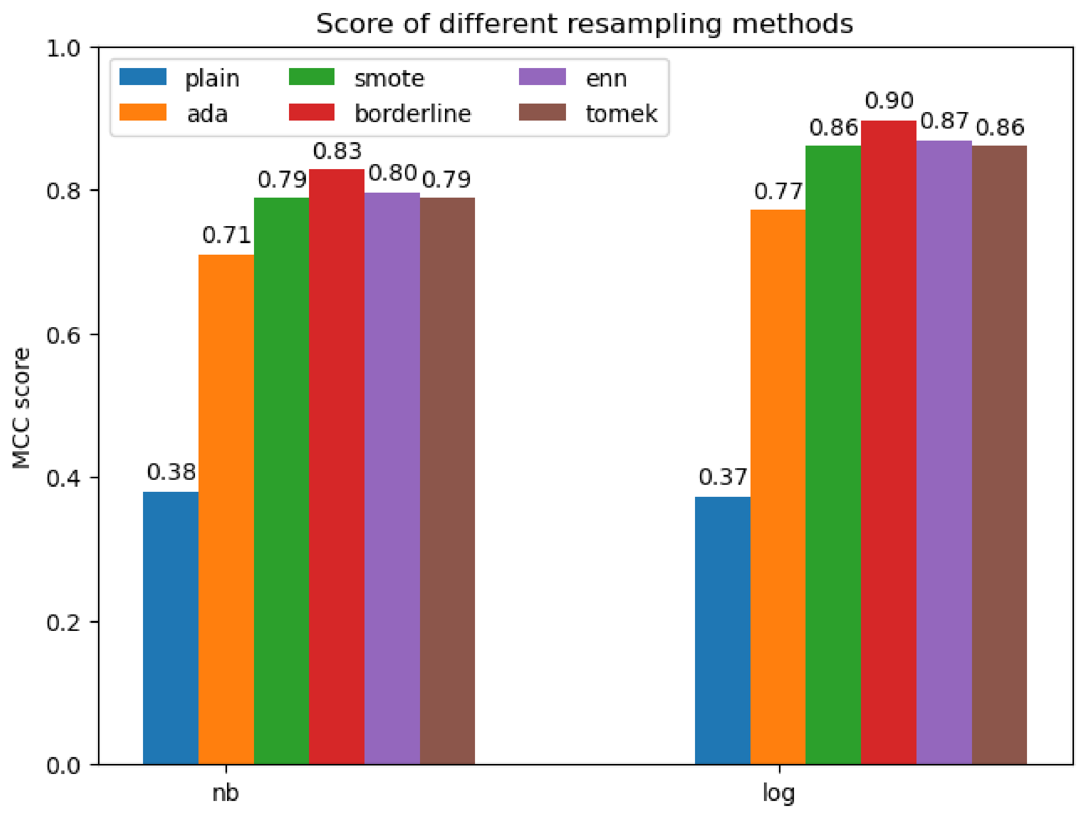
### Imbalanced dataset and resamplings

The number of samples for fraud and non-fraud is very imbalanced. There are a few frauds amid the majority of non-fraud (legitimate) transactions. There are resampling methods to rebalance the number of samples.

The resampling methods examined are:

- Over-sampling methods: AdaSync, SMOTE and Borderline,
- Combination of over and under sampling methods: SMOTE-ENN and SMOTE-Tomek

Gaussian NB and Logistic Regression were used as baseline algorithms to run cross validation with the resampled data. The plain original data was also tested alongside the resampled data.



Scores of different sampling methods: The chart shows the scores of different sampling methods. For both Gaussian NB and Logistic Regression, the ranking of performance scores were the same for each of the resampling methods. As expected, the score with the original, imbalanced data was the lowest. Among the various resampling methods, BorderlineSMOTE ranked the highest in scores. Therefore, BorderlineSMOTE was chosen as the resampling method to use for the experiment.

## Train and test set

The synthetic dataset came in two files: one for training, and one for testing.

Although the models were trained with resampled data, the test test should not come from the resampled data set.

The following steps were used to test the models:

- The original, imbalanced dataset was train_test_split.
- The test set was kept aside for validation.
- The train set was resampled using BorderlineSMOTE.
- Now, there are two train sets: one resampled, one unsampled.
- Both sets are used to train models.
- The resulting two models were tested against the original, unsampled test set. The purpose is for models that were trained with resampled data to make predictions with the “real” original, unsampled test data.

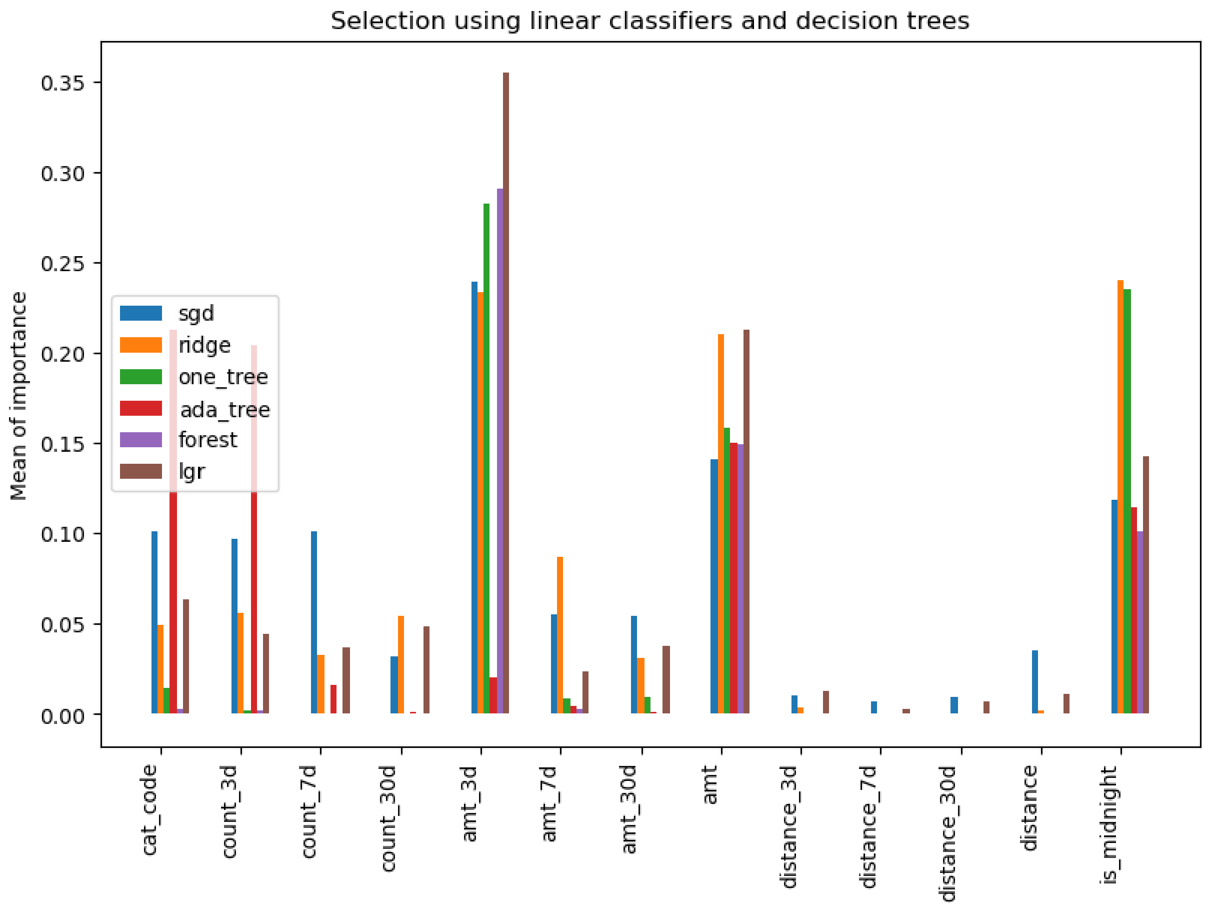
In short, one model was trained on the resampled data and another model was trained on the original, unsampled data. The purpose of training models on both resampled and unsampled data was to find out the performance of the model on real original, unsampled data.

The experiment found that sampled_model, i.e. models that were trained on resampled data was higher in recall rate, while org_model, models that were trained on original unsampled data was higher in precision rate.

## Experiments with features

Permutation importances is used to find the importance of features. Permutation importance is a way to find how much a feature is really contributing to predictions or it is due to random chance. The data rows in one column are shuffled, and then predictions errors (loss) are calculated. When a column is not present, and the more important this feature is, the more errors (loss) will occur in the predictions.

Using the sklearn permutation_importance, when the value is 0, the feature does not matter; while a negative value indicates the success of the feature is due to random chance. A high positive value indicates the importance of a feature.



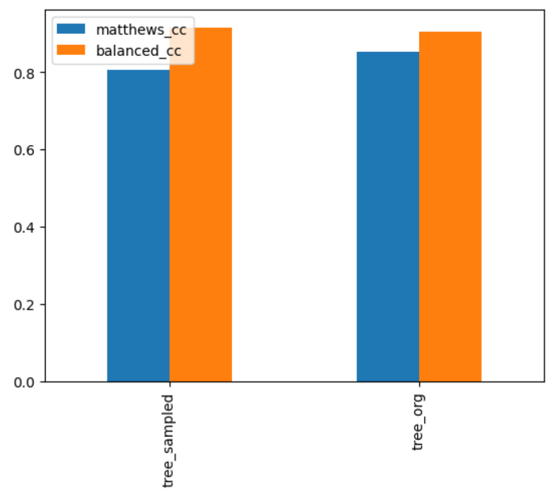
Permutation of importances from linear models and tree-based models. Final features for learning were selected based on their importances: cat_code, count_3d, count_7d, amt_3d, amt_7d, amt, distance and is_midnight.

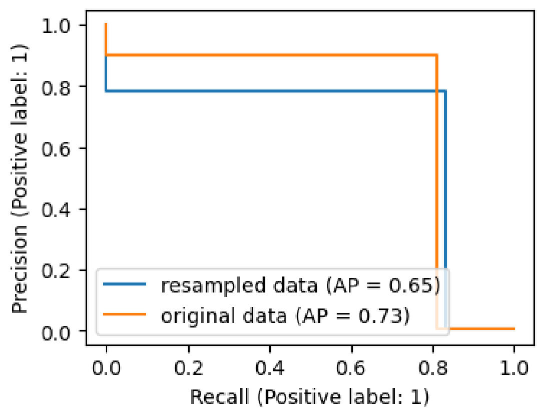
## Performance characteristics of individual models

Before deciding on the final estimator, the individual base models were examined for their performance and characteristics. Then, the individual base models would be input to a voting classifier to be the final estimator.

MCC and balanced accuracy scores were used for performance evaluation. According to sklearn, a balanced accuracy score is supposed to give a balanced accuracy on imbalance datasets. It is an average of recall. So, a high recall will result in high balanced accuracy.

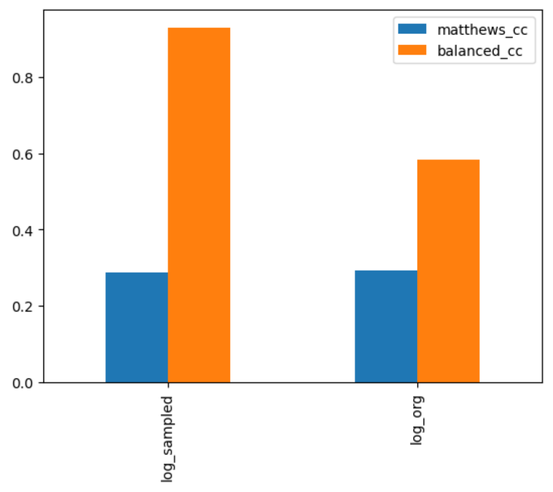
On the other hand, MCC is a correlation between predicted values and true values. So, the higher the predicted values correlate with the true values, the better the predictions. The experiment results showed that MCC was a more critical scorer than balanced accuracy, hence a better performance metric.

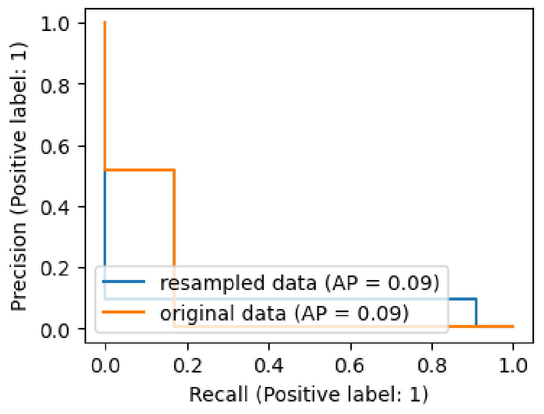




Decision trees trained with resampled and unsampled data and their corresponding MCC and balanced accuracy score.

Notice the recall of the two models on the orange bar. Both recall are more than 0.8 and hence the high balanced accuracy.





Now, compare the score with those from logistic regression. The MCC and balanced accuracy score for logistic regression showed large differences.

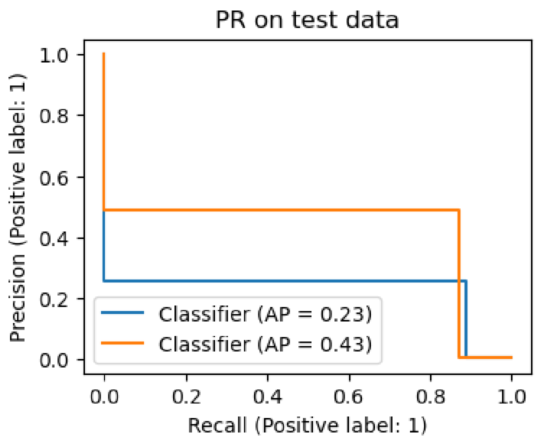
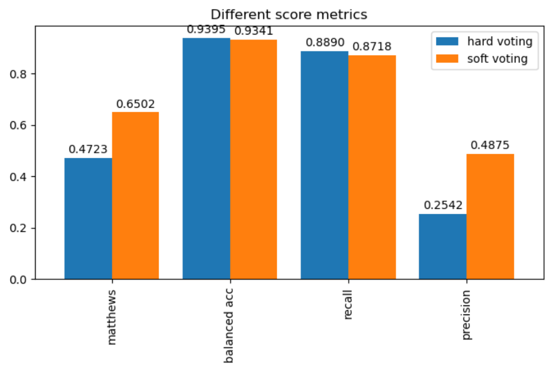
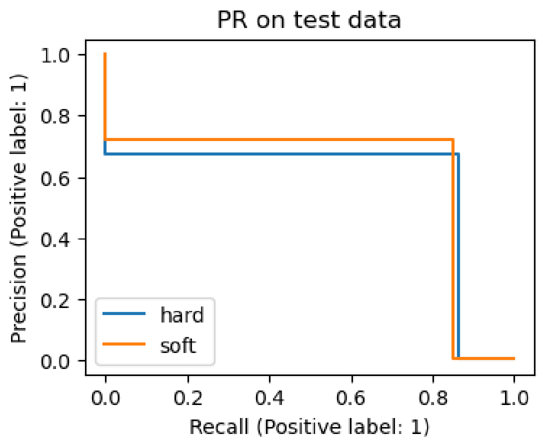
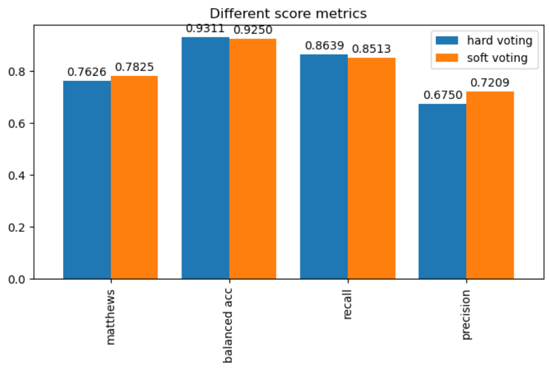
The interesting point to note was that a balanced accuracy score is an average of recall. The logistic regression model below illustrates this point well. The high recall rate (0.91) of the model trained with resampled data also had a high balanced accuracy rate. The model trained with original, unsampled data had a low recall rate; hence the low balanced accuracy.

The reason was that the model trained with original, unsampled data had a very low recall rate (0.17), but the precision was slightly higher (0.52). This higher precision made up for the MCC score.

| | |
|---|---|
| | Therefore, the MCC scores for the two models were about the same. |

## Final estimator: voting classifier

Soft and hard voting classifiers were tested for their performance.

| | |
|---|---|
| **First round**:<br>Ensemble of voting classifier with base models: Ada Boost, Decision Tree and Logistic Regression.<br>The precision was very low and it would produce many false alarms. So, this model is rejected. | **Second round**:<br>Ensemble of voting classifier with base models: Ada Boost, DecisionTree I (with best-split) and Decision Tree II (with random-split).<br>The precision improved and the recall was reasonable. This model was chosen as the **final** model. |
|   |   |

## Demonstration

The demonstrations are output as a log file comparing actual labels and prediction.

```
Output
plain data
           first      last      amt        category  Label  Prediction
365424    Marcia    Molina   762.14        misc_net      1           1
387697      Gina    Grimes   283.65     grocery_pos      1           1
389352      Gina    Grimes   666.71        misc_net      1           1
389373      Gina    Grimes   734.29        misc_net      1           1
403381     Sarah     Adams   291.54     grocery_pos      1           1
416892   Michael   Jackson   296.27     grocery_pos      1           1
425166     Jason   Johnson  1061.37    shopping_net      1           1
441426   Richard  Marshall   905.63        misc_net      1           1
447897     Karen  Sullivan   261.22            home      1           1
```

A short snippet of the data with its labels and predictions.

End of report