

Milestone Report

# Credit Card Fraud Detection

*by*

*Liew Chooi Chin*

## Introduction

The project proposal submitted previously had already detailed the background and literature review of the project. This milestone report will focus on experiment results and any other short planning regarding the project.

Each section lists the name of the team member who did the section.

## Organisation of report

This proposal comes in the following sections:

- Exploratory Data Analysis
- Experiment results
- Further plans

## Exploratory Data Analysis

The dataset used is a made-up dataset found on Kaggle. The url is <https://www.kaggle.com/datasets/kartik2112/fraud-detection?select=fraudTest.csv>

Columns in the dataset:

- index - Unique Identifier for each row
- trans\_date\_trans\_time - Transaction DateTime
- 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
- 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
- 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
- merch\_lat - Latitude Location of Merchant
- merch\_long - Longitude Location of Merchant
- is\_fraud - Fraud Flag <--- Target Class

## Exploring features

unix\_time and trans\_date\_trans\_time are the same information. To make use of the various time functionality in pandas, we convert the trans\_date\_trans\_time to datetime64[ns] type.

We found that each of the unique cc\_num corresponds to its own unique address. Dob (date of birth), and job is not useful in prediction, so we will not be using these features.

We also look at the merchant names and their categories of shops. The merchants' names are not useful for fraud prediction. But for the categories, we found that some

categories of shops are more prone to fraudulent transactions, so we keep the categories for features to be used in learning.

Distance between a customer and a point of sale is useful in predicting frauds. When a distance is unusually far for a transaction, the transaction is suspicious. We use the latitude and longitude of the customer and the merchant to calculate the distance between them.

We use the pandas rolling window to calculate the average number of transactions, the average amount, and the average distance between a customer and a point of sale. Average amount is useful as a guide to find an unusually high amount of fraudulent transactions. While the average number of transactions is useful as a guide to find high volume but low amount frauds.

When we grouped the frauds into hours in a day, we found that a high number of frauds occur between 22 to 03 hours at midnight. So, we include this feature for learning.

unix\_time and trans\_date\_trans\_time are the same information. To make use of the various time functionality in pandas, we convert the trans\_date\_trans\_time to datetime64[ns] type.

We found that each of the unique cc\_num corresponds to its own unique address. Dob (date of birth), gender and job is not useful in prediction, so we will not be using these features.

We also look at the merchant names and their categories of shops. The merchants' names are not useful for fraud prediction. But for the categories, we found that some categories of shops are more prone to fraudulent transactions, so we keep the categories for features to be used in learning.

Distance between a customer and a point of sale is useful in predicting frauds. When a distance is unusually far for a transaction, the transaction is suspicious. We use the latitude and longitude of the customer and the merchant to calculate the distance between them.

We use the pandas rolling window to calculate the average number of transactions, the average amount, and the average distance between a customer and a point of sale. Average amount is useful as a guide to find an unusually high amount of fraudulent transactions. While the average number of transactions is useful as a guide to find high volume but low amount frauds.

When we grouped the frauds into hours in a day, we found that a high number of frauds occur between 22 to 03 hours at midnight. So, we include this feature for learning.

## Summary of selected features

These features are explained here.

- **cat\_code**: Categories of shopping or merchants. We found that some categories have high chances to be used for fraudulent transactions.
- **is\_midnight**: We group the transactions by hours. From the hourly groups, we found that at 22 to 03 hours, the number of frauds is unusually high. Therefore, we keep this feature for model learning.
- **count\_3d, count\_7d, count\_30d**: Rolling window periods of 3, 7 and 30 days. This is the number of average transactions of a customer. If the number of transactions is unusually high, then we can suspect that the transactions are suspicious.
- **amt\_3d, amt\_7d, amt\_30d**: Rolling window periods of 3, 7 and 30 days. This is the average of transactions of a customer. If the amount of transactions is unusually high, then we can suspect that the transactions are suspicious.
- **amt**: This is the actual transaction amount occurring at a point of sale.
- **distance\_3d, distance\_7d, distance\_30d**: Rolling window periods of 3, 7 and 30 days. This is the average distance between a customer and a point of sale. If the distance between a customer and a point of sale is unusually high from the average distance, we can suspect that the transactions are suspicious.

- **distance**: This is the actual distance between a customer and a point of sale when a transaction is carried out.
- **is\_fraud**: The label of whether a transaction is fraud or legitimate.

## Organization of the datasets

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

In the preliminary experiment, the training data was train\_test\_split to get a “test” set to test the models. In the final round of choosing a final estimator, the provided test set was used.

## Problems encountered

When running the experiments, one big problem encountered was that the file size is rather large ( 100 to 180 MB). Also the training time for some algorithms took a long time to finish. Autologging by MLFlow added almost double the time for the experiment to run.

Facing problems like this, lightweight algorithms were chosen to run the experiments.

## Recall vs precision

In this fraud detection scenario, accuracy is not a good performance measure. It is because the number of non-fraud cases is over 99.42% percent and any prediction will have a 99.42% chance of being correct!

Considering the amount of money and harm caused to the victims, it is desirable that as many frauds be caught as possible. This would inevitably set off some false alarms. In deciding that priority would be capturing frauds, recall is prioritised before precision.

## Resampling methods

The dataset 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. It would be beneficial to examine the performance of the various methods before deciding on which one to use.

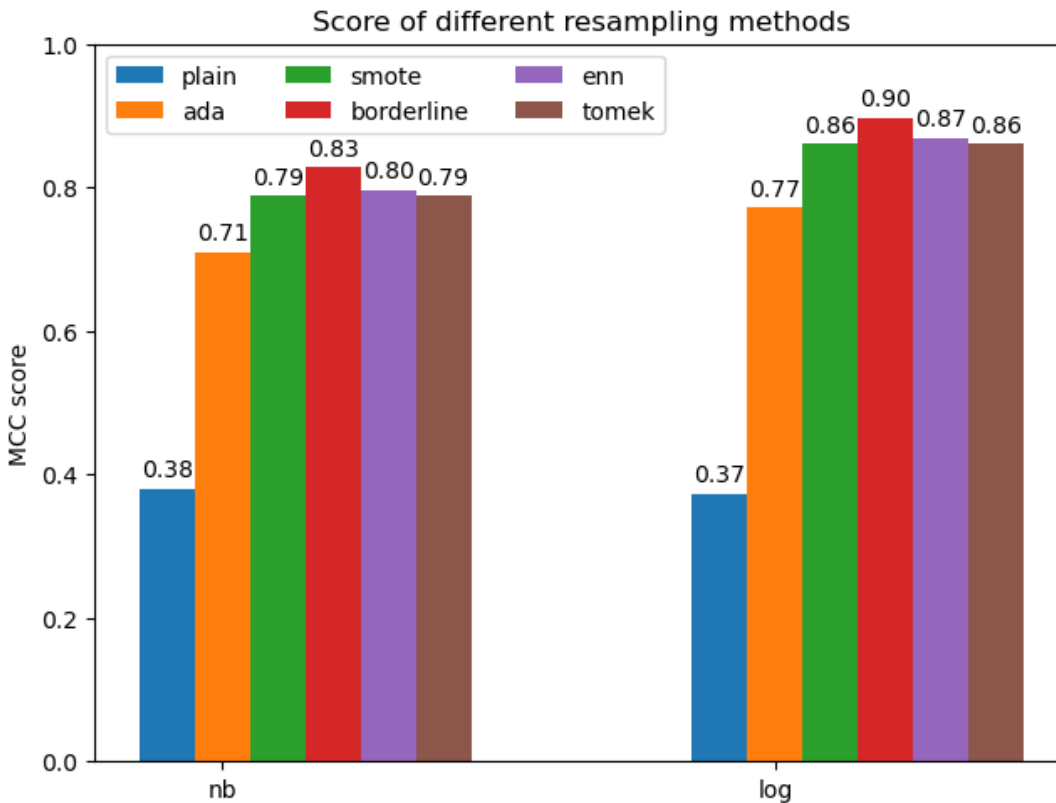
The resampling methods examined are:

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

After resampling, the data was doubled. The data size was very large and the processing time was long. So, lightweight algorithms were chosen to test each of the resampling methods. 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.

Apart from manageable running time, there was another reason for choosing lightweight algorithms. If simple and straightforward algorithms could do well in the resampled set, other more involved algorithms should be able to do as good, if not better than the baseline algorithms.

The chart below shows the scores of different sampling methods. For both Gaussian NB and Logistic Regression, the score rankings of the different resampling methods are the same. The lowest to highest ranking were from AdaSync, plain SMOTE, SMOTE-Tomek, SMOTE-ENN to BorderlineSMOTE. As expected, the score with the original, imbalanced data is low. Among the various resampling methods, BorderlineSMOTE ranked the highest in scores. BorderlineSMOTE was chosen as the resampling method to use for the experiment.



## Experiment with resampling data

Although the models were trained with resampled data, the test set should not come from the resampled data set. This was exactly the mistake that was made. During the experiment, the model predictions on the resampled “test” set is too good to be true. Thus, something was not right.

The following steps should be used instead:

- The original, imbalanced dataset was used to generate a train and test set using `train_test_split`.
- The test set was kept for validation.
- The train set was resampled using BorderlineSMOTE. This resampled train set was used to train a model.
- The train set that was unsampled was used to train another model.
- Both of the models used the same algorithm.

- Then, finally, the two models were tested against the test set. The test set was the original imbalanced and unsampled data. Now, the model that was trained with resampled data had to make predictions with the “real world” 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 testing models on both resampled and unsampled data was to find out which type of data to train the final estimator on.

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. This would be further discussed below.

## **Experiments with base models**


Base models were built to understand their characteristics before inputting them into an ensemble for learning.

## **Experiments with features**

Permutation importance 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 (*Permutation Importance* (n.d.)). The data rows in one column are shuffled, and then predictions errors (or loss) are calculated. The more important a 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.



Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24



Source: <https://www.kaggle.com/code/dansbecker/permutation-importance>

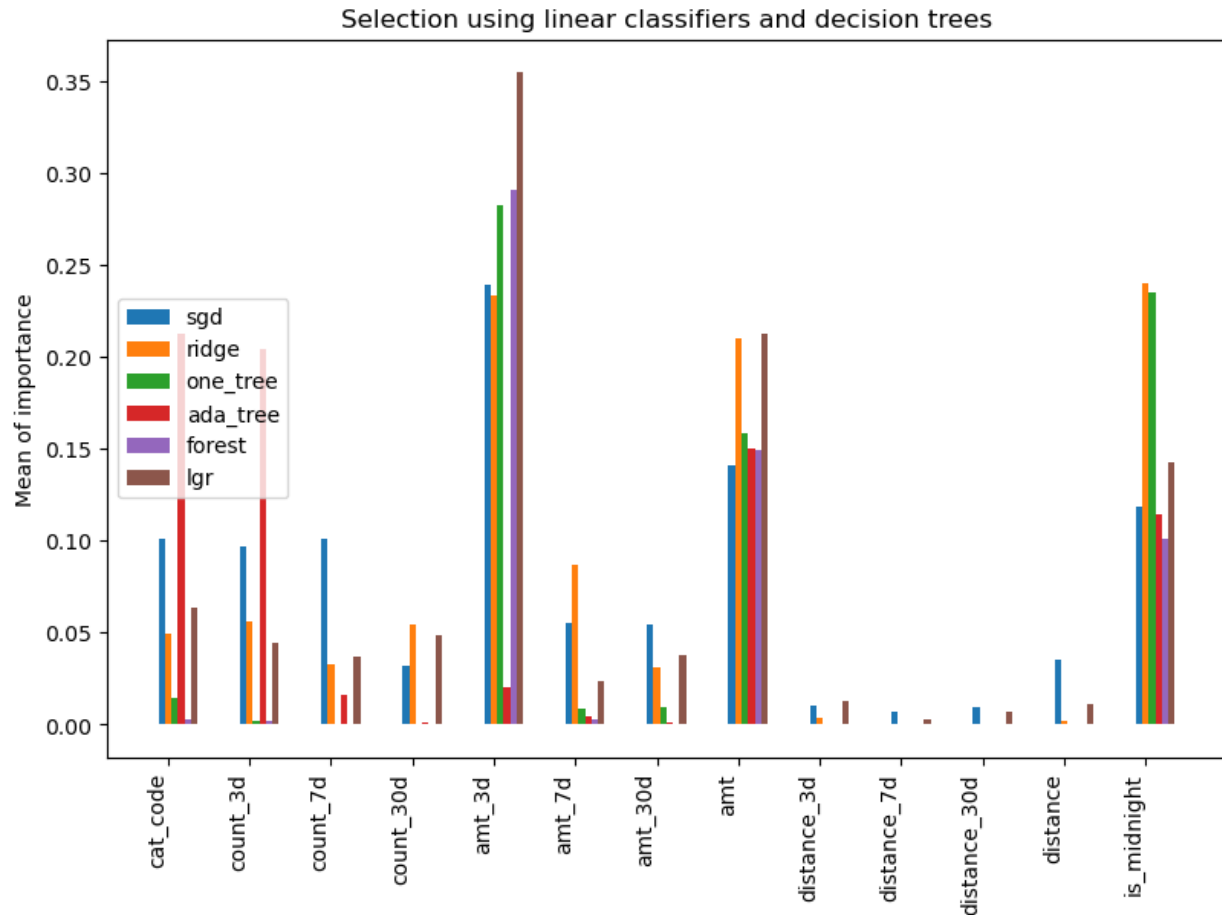
## Preliminary exploration of feature importances

To explore all the features that were selected, two types of algorithms were used to check for feature importances.

For linear models, Ridge, SGD Classifier and Logistic Regression were used. The coefficients of the linear model will indicate how important a feature is. Regularization parameters in linear models will force coefficients of less importance to be very small. The coefficients returned by the models were ranked accordingly.

For tree-based algorithms, a single decision tree, adaptive boosting and random forest were used. Tree-based algorithms have built-in `feature_importances_`. However, these feature results should not be used as is. In the sklearn documentation, tree-based impurity-based feature importances will give higher importance to features with high cardinality, i.e. features with many unique values. Sklearn provides a `permutation_importance` function that calculates every feature equally well.

The following chart shows the features importance by various models. The top three most important features are `amt_3d`, `amt` and `is_midnight`.



## Discussion of features importance

From the chart, the linear models and the tree-based algorithms unanimously ranked the same top three features: `amt_3d`, `amt`, and `is_midnight`. The `amt` and `amt_3d` could show the difference between a legitimate and a fraudulent transaction involving a large sum of money. The `is_midnight` was also consistent with the observation in the EDA that an unusually high number of fraudulent transactions occur during 11 to 03 hours in the midnight.

The `cat_code` of categories of shopping was consistent with the observation in the EDA that certain categories were more prone to be used for fraudulent transactions. The `count_3d` features contributed somewhat to the algorithms. During the EDA phase, the count (the number of transactions) was thought to be a good indicator of small amounts but high volume of transactions. The importance of the value was relatively

low, about the same as `cat_code`. One reason could be that this synthetic dataset did not feature many fraudulent transactions of this nature (small amount, high volume).

The most negligible feature turns out to be the `distance` feature. It was thought that an unusually huge distance of a transaction could indicate stolen CC numbers being used overseas. However, this was not the case with the dataset. Perhaps such fraud patterns were not featured in the synthetic generation of the transaction data. Another reason could be that `is_midnight` is already an indicator of the card numbers being used in another time zone.

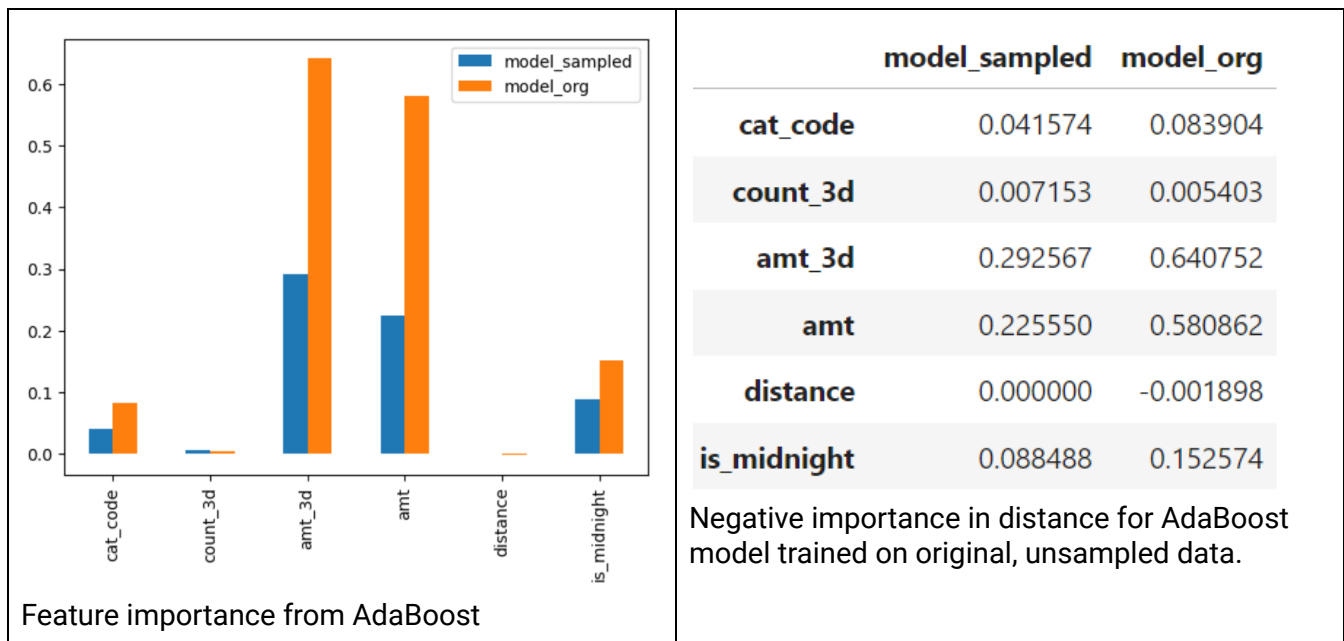
From the experiment above, the following features are marked for further experiment. Although the distance was not important, it was included anyway. Features to be considered were `cat_code`, `count_3d`, `count_7d`, `amt_3d`, `amt_7d`, `amt`, `distance` and `is_midnight`. Before confirming the final candidates of features, another experiment was conducted to make further confirmation.

**Further feature inspection with Adaptive Boosting**

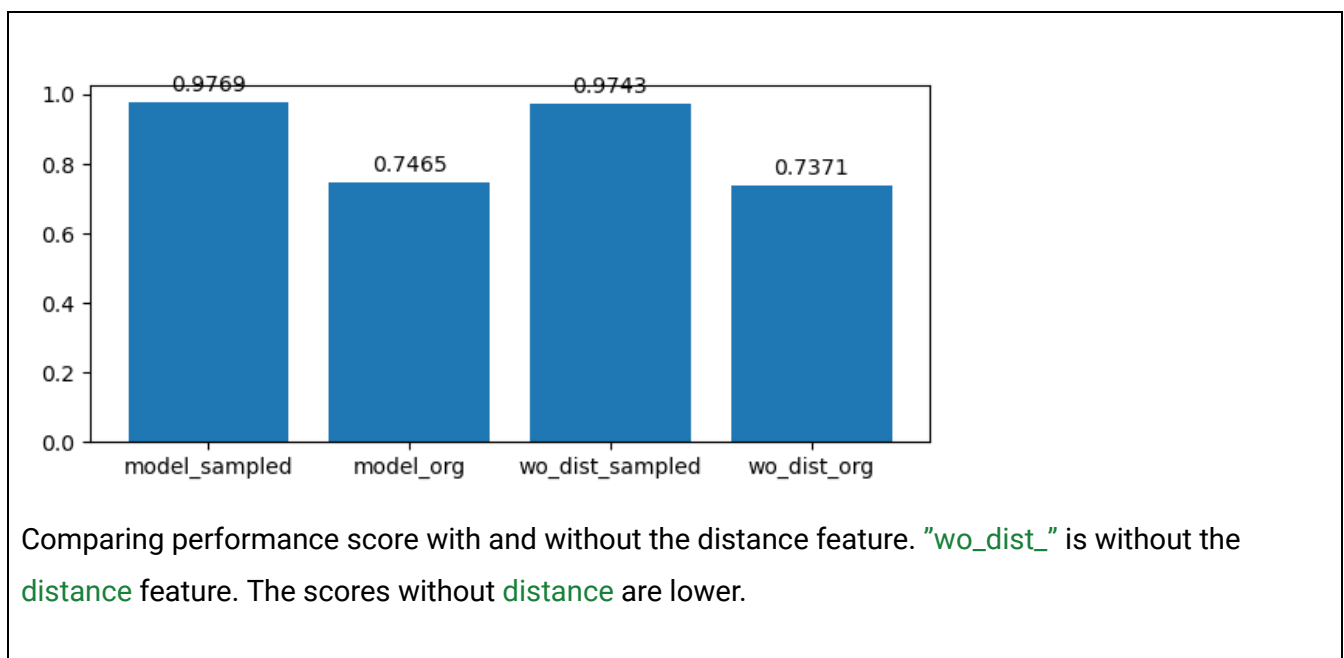
In this next experiment, features are examined again. `amt_3d` and `amt_7d` show a high correlation. This collinearity will affect algorithms, so `amt_7d` is dropped from the features. The correlation between `amt_3d` and `amt_7d` is also high. To avoid the problem of collinearity, `amt_7d` is also dropped from the features.

<div>Linear correlation of amt_3d and amt_7d</div> <div><div>amt_3damt_7d</div><div>amt_3d1.0000000.720595</div><div>amt_7d0.7205951.000000</div></div> <div>Linear correlation of amt_3d and amt_7d is high.</div>	<div>Linear correlation of amt and amt_3d</div> <div><div>amtamt_3d</div><div>amt1.0000000.393281</div><div>amt_3d0.3932811.000000</div></div> <div>The linear correlation between amt and amt_3d is low.</div>
<div>Linear correlation of count_3d and count_7d</div> <div><div>count_3dcount_7d</div><div>count_3d1.0000000.865255</div><div>count_7d0.8652551.000000</div></div> <div>Linear correlation of count_3d and count_7d is high.</div>	

After dropping the highly correlated features, permutation importance was calculated. The **distance** feature was causing negative importance(-0.001534) in AdaBoost.

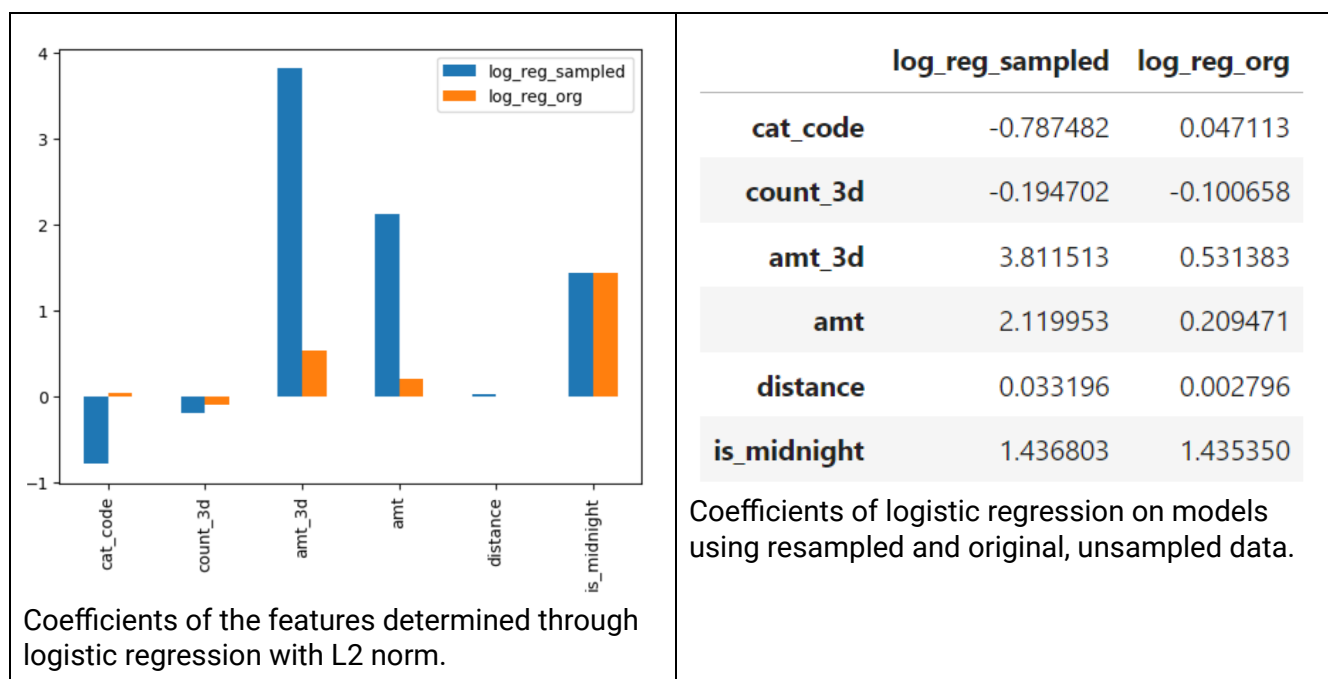


The **distance** feature distance was removed and the experiment was run again. However, the cross validation score decreased slightly, in other words, the performance of the model suffered slightly. Therefore, the **feature** distance was kept in the list of final features to be used.



## Coefficients and feature importances

Logistic regression provides another way to examine feature inspection through coefficients. Linear models like logistic regression are regularized by L1 or L2 norms and features of less importance will have low coefficients. 'distance' remains the smallest feature of importance. The top three most important features are still `amt_3d`, `amt` and `is_midnight`.

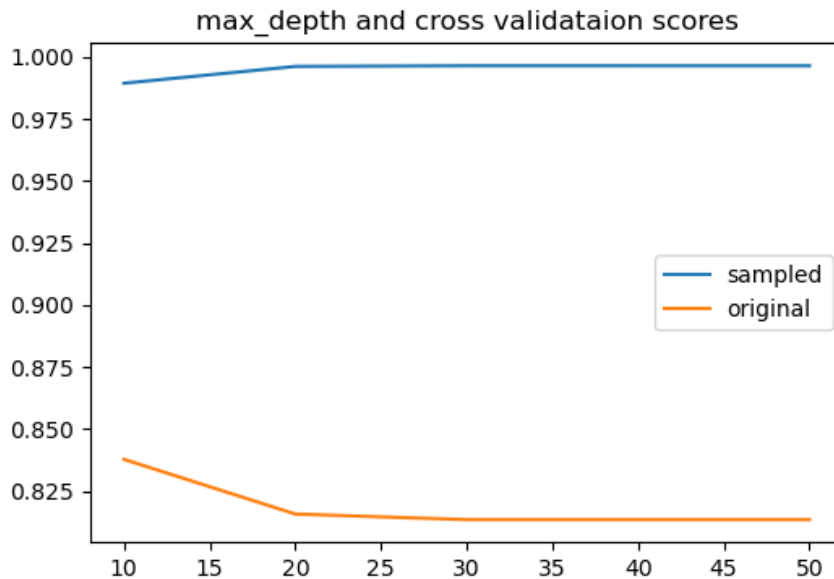


In summary, the final features were `cat_code`, `count_3d`, `amt_3d`, `amt`, `distance` and `is_midnight`.

## Max\_depth of decision trees

For decision trees, the `max_depth` of a tree decides how deep a tree grows to make the best predictions. Is the deeper the better? A grid search was performed for a `max_depth` from 10 to 50. In the chart below, the `max_depth` for the tree trained on original, unsampled data gave the best score at `max_depth`=10, while the tree trained on the resampled\_data at `max_depth`=30. One reason for the deeper depth was that the tree trained on resampled data might have too many options to consider, hence a deeper

tree was needed to make a decision. This experiment dispels the belief that a deeper tree is a better tree.



Grid search from 10 to 50 for decision tree and the corresponding cross validation score. Notice that the y-axis starts from 0.800 and hence the large distance between the scores of the two trees.

## Performance evaluation

### Preliminary findings of performance results

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.

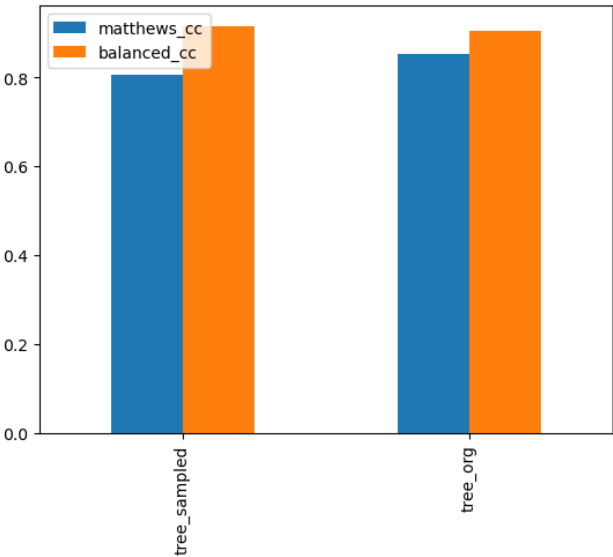
### Adaptive boosting

Adaptive boosting had been explored above.

Decision tree

Models were trained on resampled data and original, unsampled data. Then models were tested on a test set . 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. 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. In fact, MCC is widely used in bioinformatics as an evaluation metric for imbalanced datasets (Chicco & Jurman, 2020). The experiment results showed that MCC was more critical than balanced accuracy, hence a better performance metric.

Two decision tree models were scored with MCC and balanced accuracy. Balanced accuracy gave a more optimistic scoring than MCC. Overall, the MCC scores were quite high for both of the trees trained with resampled or original data.



Decision trees trained with resampled and unsampled data and their corresponding MCC and balanced accuracy score. Both scores are quite good. Notice the recall of the two models

	precision	recall
non_fraud	1.00	1.00
fraud	0.78	0.83

Decision tree with resampled data. Notice the accuracy. Models trained with resampled data usually have higher recall than models trained with unsampled data.

on the right pane. Both recall are more than 0.8 and hence the high balanced accuracy.

	precision	recall
non_fraud	1.00	1.00
fraud	0.90	0.81

Decision tree with original, unsampled data.  
Notice the higher precision.

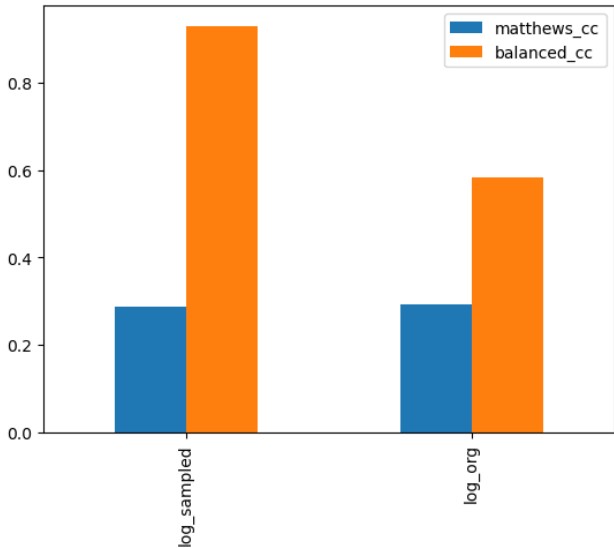
### Logistic regression

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.

But, what about the MCC score? The two models seemed to do equally well on this.

The reason was that the model trained with original, unsampled data had a very low recall rate (0.17), but the precision was not bad (0.52). This higher precision made up the MCC score. Therefore, the MCC scores for the two models were about the same.



 <p>The balanced accuracy score and the recall rate of logistic regression models.</p>	<table><thead><tr><th></th><th>precision</th><th>recall</th></tr></thead><tbody><tr><td>non_fraud</td><td>1.00</td><td>0.95</td></tr><tr><td>fraud</td><td>0.10</td><td>0.91</td></tr></tbody></table> <p>Notice the high recall of the model trained with resampled data, hence the high balanced accuracy score.</p>		precision	recall	non_fraud	1.00	0.95	fraud	0.10	0.91									
	precision	recall																	
non_fraud	1.00	0.95																	
fraud	0.10	0.91																	
	<table><thead><tr><th></th><th>precision</th><th>recall</th></tr></thead><tbody><tr><td>non_fraud</td><td>1.00</td><td>1.00</td></tr><tr><td>fraud</td><td>0.52</td><td>0.17</td></tr></tbody></table> <p>Notice the low recall of the model trained with unsampled data, hence the low balanced accuracy score. But, it has surprisingly better precision than the resampled model above.</p>		precision	recall	non_fraud	1.00	1.00	fraud	0.52	0.17									
	precision	recall																	
non_fraud	1.00	1.00																	
fraud	0.52	0.17																	
<p>Confusion matrix with resampled data In count form</p> <table><thead><tr><th></th><th>non-fraud</th><th>fraud</th></tr></thead><tbody><tr><td>non-fraud</td><td>245126</td><td>12708</td></tr><tr><td>fraud</td><td>136</td><td>1365</td></tr></tbody></table> <p>This classification report shows that among the (136+1365=1501) fraud cases, 1365/1501=0.91 proportions of fraud cases are identified as fraud. A fairly good recall.</p>		non-fraud	fraud	non-fraud	245126	12708	fraud	136	1365	<p>Confusion matrix with original</p> <table><thead><tr><th></th><th>non-fraud</th><th>fraud</th></tr></thead><tbody><tr><td>non-fraud</td><td>257599</td><td>235</td></tr><tr><td>fraud</td><td>1248</td><td>253</td></tr></tbody></table> <p>For original, unsampled data, among the 1501 fraud cases only 253 cases are identified. Thus, recall is 253/1501=0.17</p>		non-fraud	fraud	non-fraud	257599	235	fraud	1248	253
	non-fraud	fraud																	
non-fraud	245126	12708																	
fraud	136	1365																	
	non-fraud	fraud																	
non-fraud	257599	235																	
fraud	1248	253																	

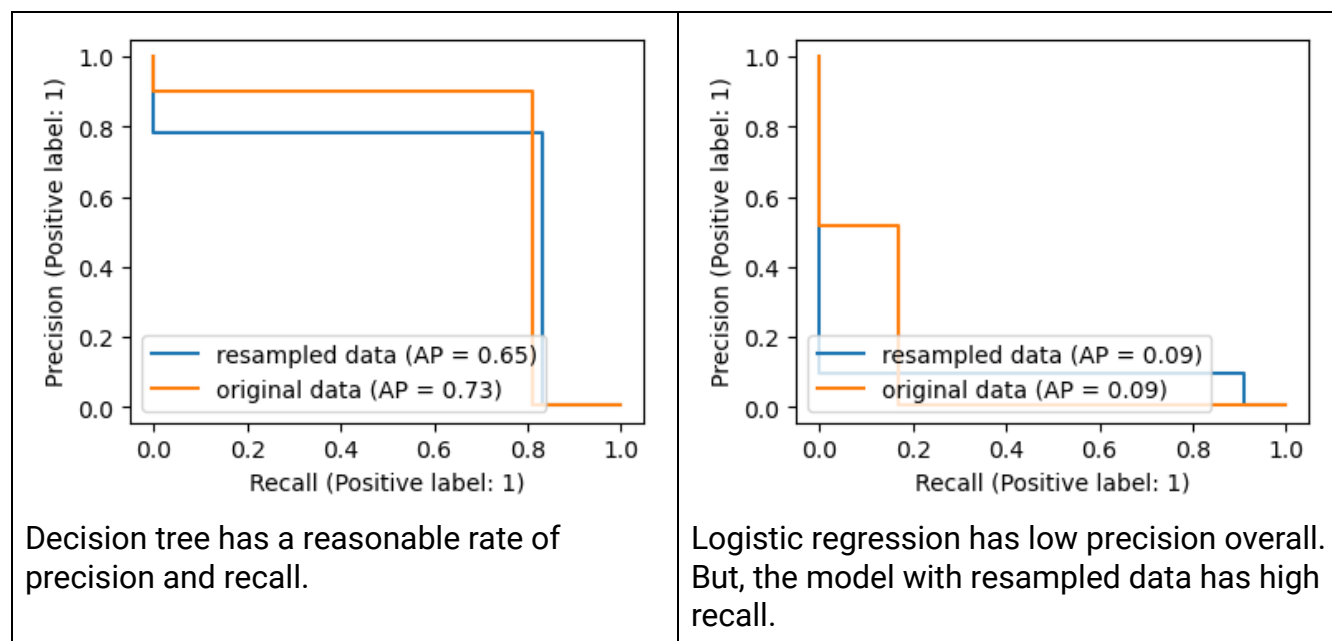
## Precision and recall

From the experiments, it appeared that models trained on resampled data had higher recall rate, while models trained on original, unsampled data had higher precision rate. The reason could be that with resampled data, there were many instances to “learn” from by the model, so it could identify many unseen samples with correct classes. But, with unsampled data, the model had to work hard to “learn” from the very small samples

that it had. It learnt very specific patterns to identify unseen samples, hence higher precision.

-	precision		recall	-	precision		recall
non_fraud	1.00		1.00	non_fraud	1.00		1.00
fraud	0.78		0.83	fraud	0.90		0.81
Tree with resampled data with higher recall but lower precision.				Tree with original, unsampled data with lower recall but higher precision.			

In the charts below, the precision rate of the decision tree and the logistic regression is very different.



## Choosing the final estimator

### Round 1: Not successful because of very low precision

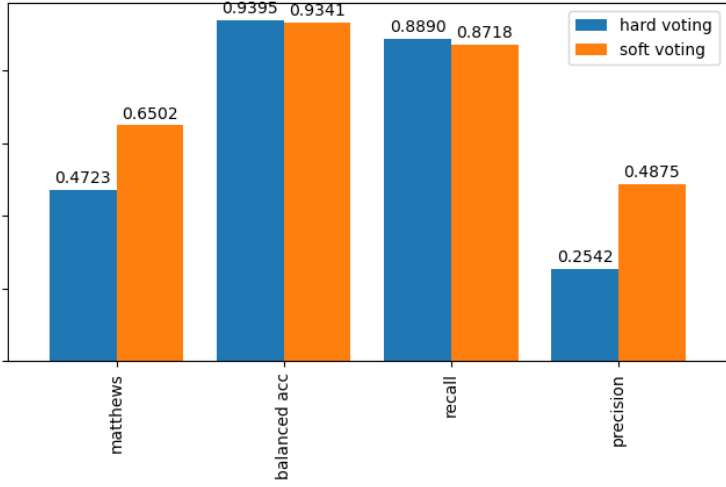
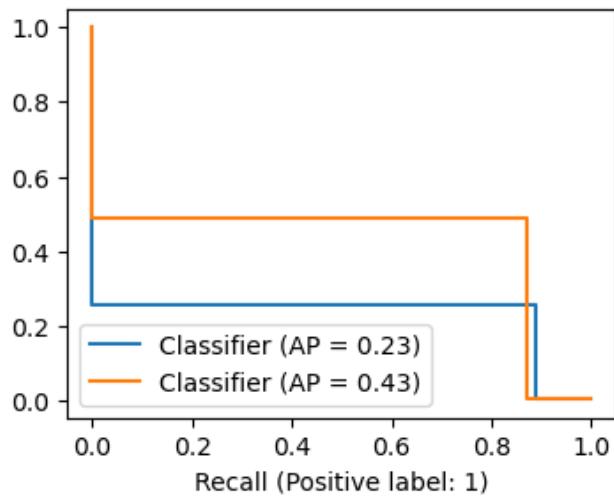
From the experiment above, it was found that models trained on resampled data were higher in recall, while models trained on unsampled data were higher in precision. For fraud detection, considering the harm that it would cause the card owner, detecting as many frauds as possible is important. Thus, recall is more important. As a tradeoff, the precision would be lower.

From the experiments above, the characteristics of models trained with resampled data had higher recall. Thus, for the final estimator, it would be trained on resampled data for better recall.

A voting classifier would be used. Odd number of voters were chosen to prevent a tie in hard voting. A base estimator made up of the three classifiers experimented would be used: AdaBoost, DecisionTree and LogisticRegression. The voting classifier was tested on hard and soft voting.

In the first round of determining the final estimator, the result was not satisfactory. The precision was way too low. This was likely to be caused by the LogisticRegression base estimator. In the previous experiment, LogisticRegression was found to have low precision. That was the reason the voting classifier consisting of LogisticRegression performed poorly on precision. Although LogisticRegression contributed to very good recall (0.94 for hard voting), the very low precision (0.25 of hard voting) was not acceptable.

LogisticRegression was removed from the voting classifier and the voting classifier was tested with AdaBoost and DecisionTree.

<p>Different score metrics</p>  <table><thead><tr><th>Metric</th><th>hard voting</th><th>soft voting</th></tr></thead><tbody><tr><td>matthews</td><td>0.4723</td><td>0.6502</td></tr><tr><td>balanced acc</td><td>0.9395</td><td>0.9341</td></tr><tr><td>recall</td><td>0.8890</td><td>0.8718</td></tr><tr><td>precision</td><td>0.2542</td><td>0.4875</td></tr></tbody></table>	Metric	hard voting	soft voting	matthews	0.4723	0.6502	balanced acc	0.9395	0.9341	recall	0.8890	0.8718	precision	0.2542	0.4875	<p>Ensemble of voting classifier with AdaBoost, DecisionTree and LogisticRegression.</p> <p>The recall is high, but the precision is very low.</p> <p>Balanced accuracy gives high scores compared to MCC. Thus, balanced accuracy alone will neglect the effect of precision. MCC is a better scorer that takes into account recall and precision.</p>
Metric	hard voting	soft voting														
matthews	0.4723	0.6502														
balanced acc	0.9395	0.9341														
recall	0.8890	0.8718														
precision	0.2542	0.4875														
<p>PR on test data</p>  <table><thead><tr><th>Classifier</th><th>AP</th></tr></thead><tbody><tr><td>Classifier (blue)</td><td>0.23</td></tr><tr><td>Classifier (orange)</td><td>0.43</td></tr></tbody></table>	Classifier	AP	Classifier (blue)	0.23	Classifier (orange)	0.43	<p>Ensemble of voting classifier with AdaBoost, DecisionTree and LogisticRegression.</p> <p>A very low precision but high recall for both the soft and hard voting classifiers.</p>									
Classifier	AP															
Classifier (blue)	0.23															
Classifier (orange)	0.43															

## Round 2: With acceptable results

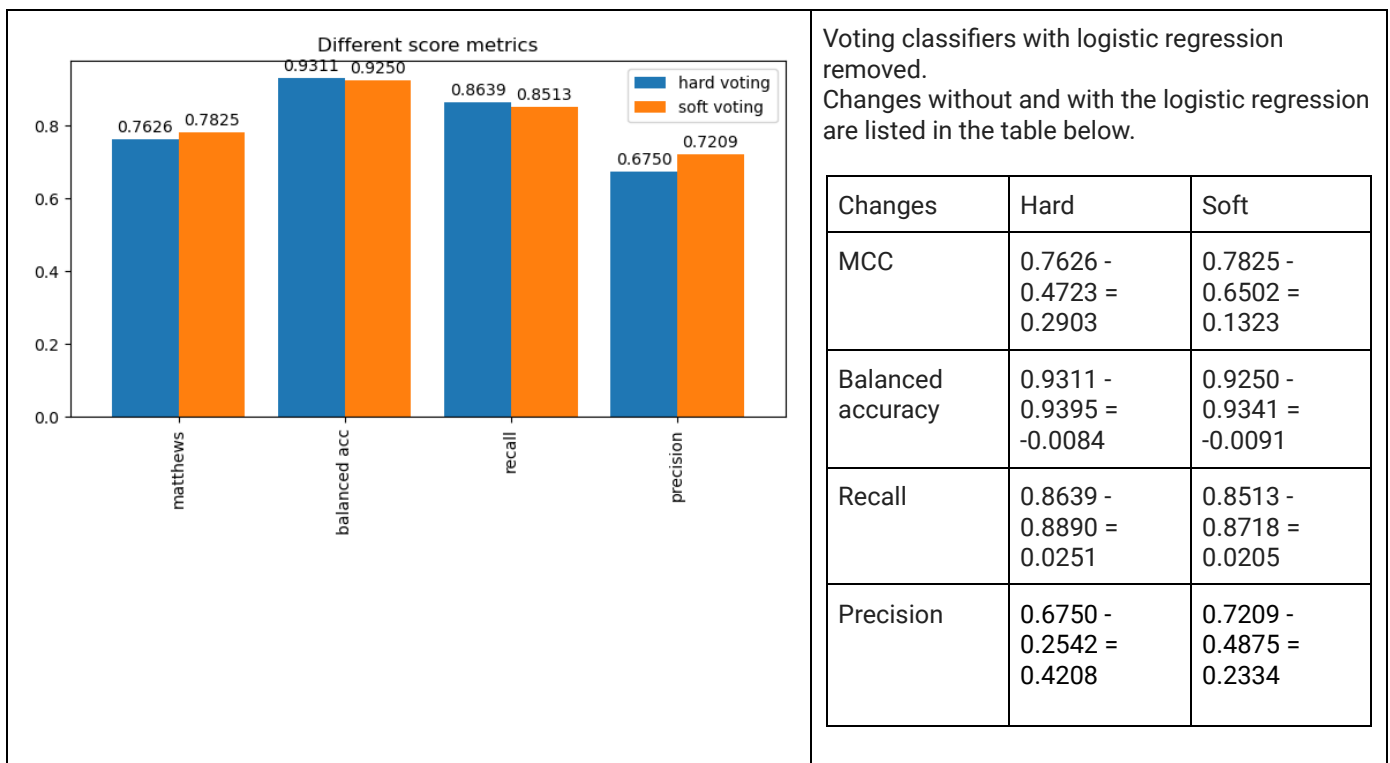
With LogisticRegression removed, the precision increased by a lot: 0.4208 for the hard voting classifier, and 0.2334 for the soft voting classifier.

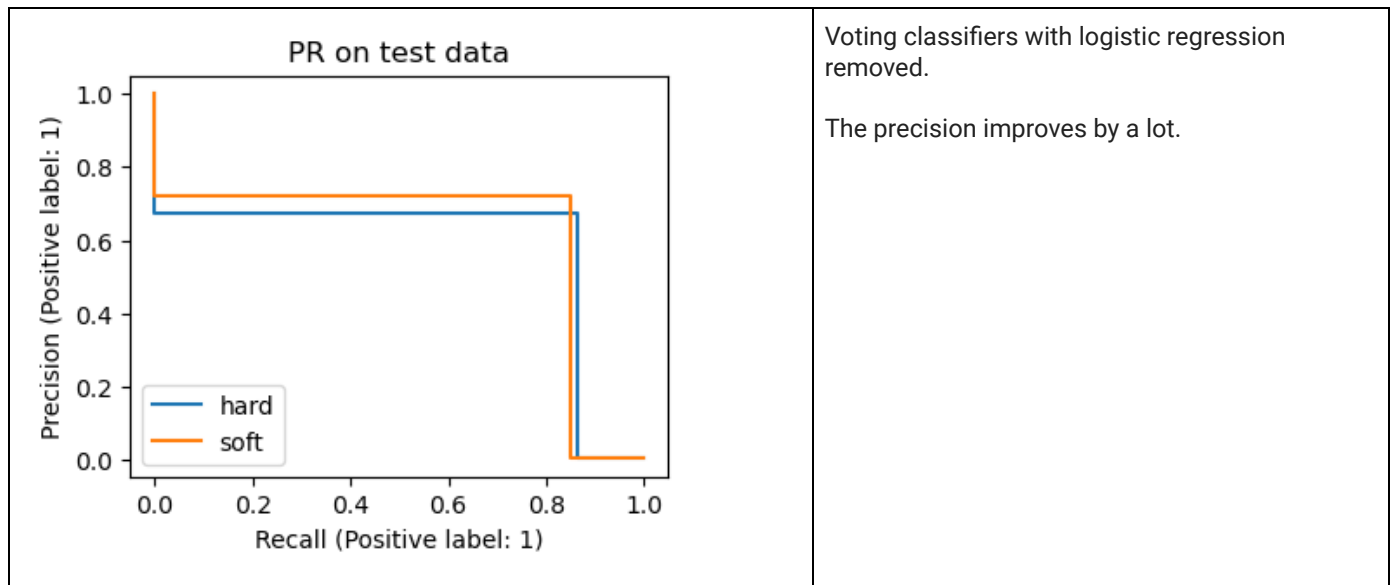
The recall dropped 0.0251 for the hard voting classifier, 0.0205 for the soft voting classifier. This showed that LogisticRegression was useful in getting higher recall. However, when it was included in the voting classifiers, the precision was very poor.,

The soft classifier had a smaller drop in recall, and its MCC was higher at 0.7825 compared to the hard voting classifier at 0.7626. Thus, the soft voting classifier would be chosen as the final estimator.

Final estimator: Soft voting classifier with base estimators (AdaBoost, and two different decision trees).

The scores metrics of the final soft voting classifier tested on the provided test file was shown in the chart below.





## Further plans

The further plan is to develop a demo and a presentation of the project.

---

## References

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1).

*Payment card number*. (2023, June 14). Wikipedia.

[https://en.wikipedia.org/wiki/Payment\\_card\\_number](https://en.wikipedia.org/wiki/Payment_card_number)

*Permutation Importance*. (n.d.). Kaggle.com.

<https://www.kaggle.com/code/dansbecker/permutation-importance>

---

# Appendix

Notebooks on the source codes.