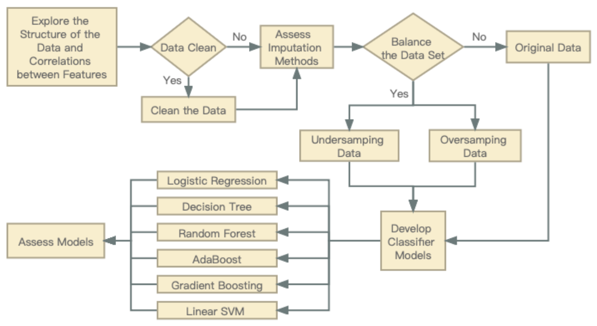
Client Report

Introduction

Credit card fraud detection is an important tool for banks to reduce their losses. The main purpose of this project is to predict credit card fraud by developing a classifier model. In this project, we have developed a model to predict credit card fraud by exploring the structure of the data and correlations between features, assess imputation methods to fully understand the relationships between the data required in this model and clean it up as necessary. Finally for the data set, keeping the extreme class imbalance in mind, six models, Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boost, and Linear Support Vector Machine (SVM), are built. Their results are analysed under three different data processing models. The results were evaluated by four criteria, Precision, Recall, the F1-score and lastly the area under the precision-recall curve (AUPRC). The AdaBoost and Gradient Boosting models were found to have good results in the case of oversampling the data. Finally, we make recommendations for improving customer data collection strategies to improve credit card fraud detection.

0 Methodology



0.1 *Method*

The broad strokes of this project are detailed in the flowchart pictures above. This report will separate this process into three key sections: Data Inspection, Imputation Assessment and Modelling.

0.2 *Data Source*

The dataset has been collected and analysed during a research collaboration of Wordline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection.

1 Data Inspection

1.1 *Initial Feature Drops & Data Structure*

Due to the nature of the beast with this data set, many of the features to be inspected and tested upon had already undergone treatment in the form of a PCA transformation - informally a reduction in size and complexity of given features while retaining their most useful information. The important thing to note here for inspection here meant features were anonymised for privacy, likely due to them denoting bank names etc. In addition, this transformation already scales these features.

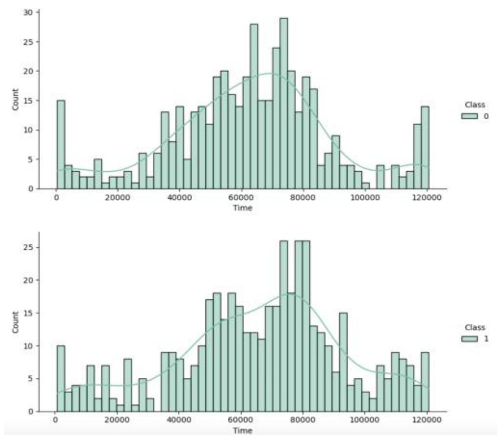
The data itself contained no duplicates or null entries and the 'id' column was initially dropped.

1.2 *Gaging the extent of the problem*

On average, each fraudulent transaction had a value of $108.39, over $50 above genuine ones on average. It is clear that when these transactions occur, they are more costly and extreme.

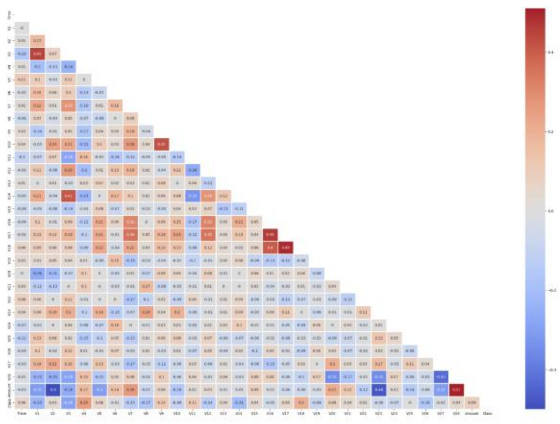
1.3 *Under Sampling & Pre-Scaling Analysis*

Given the data features, over 200,000 genuine transactions versus 469 fraudulent transactions, for the analysis it is useful to under-sample the genuine cases to better reveal correlations with 'Class' - necessary despite the data loss.



Before scaling the ‘Time’ feature, it was plotted against genuine and fraudulent transactions respectively, revealing some inclination that fraudulent cases occur at times when genuine ones do. Note that it does not refer to time of day, but rather the number of seconds between a particular transaction and the first transaction in the dataset

1.4 *Correlation Investigation*

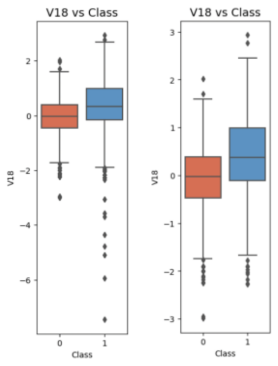


The correlation matrix pictured above illustrates some degree of collinearity between features, but nothing to an extent worth dropping anything over.

Looking more into correlations with the ‘Class’ variable revealed ten features to correlate very poorly (below plus or minus 0.05) so were dropped to save on computation expense for modelling.

1.4 *Outlier Decisions*

By producing box plots, which were separated by ‘Class’, and histograms for each of the remaining features, it was revealed there were a reasonably large number of outliers for each feature. Such as not to remove too much further data, given under-scaling, it was decided to remove outliers in only the most strongly correlating features with and against Class. Furthermore, only most extremal outliers were removed, setting the cut off at three times the interquartile range above and below the upper and lower 25th percentiles. Again, these were done based on the ‘Class’ category - removing outliers amongst fraudulent and non-fraudulent transactions separately.



The box plots to the left illustrate an example of this outlier removal strategy (leftmost is before, rightmost after)
- the difference is especially seen amongst fraudulent transactions (Class = 1).

2 Imputation Assessment

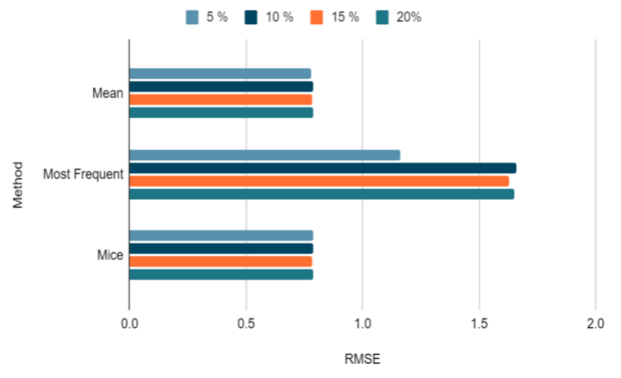
2.1 *Methods and Rationale*

The training data does not actually have any missing values, but to evaluate the interpolation method, we replicated the training data and removed a small number of randomly selected data values. The missing values are then estimated and the results compared to the original data.

To remove this data the numpy feature mask is used, a mask consisting of a list of boolean values, and when applied to the data the boolean values can be used to extract the corresponding data. In this project, the mask is used to remove data from a copy of the training data, the missing values can then be imputed and the results checked using the mask values.

To test the different imputing methods a varying percent of removed values was used which can be seen in the summary graph below

The three imputation methods were the mean, the most frequent, and mice. The mean and the most frequent are performed using the simple imputer. To measure the new values against the original values the Root Mean Square Error (RMSE) value is computed. This gives a good metric to best evaluate the performance of the imputation methods.



2.1 *Results*

As expected the most frequent method performs the worst resulting in high RMSE values. The mean and mice methods perform the best across all the values and this is visualised in the graph above. It is important to note that the percentage of values removed had only a small effect on the performance, this is due to a low variation in the data.

3 Modelling

3.1 *Models used and choice of data*

We used original data set, oversampling and undersampling methods to fit the models.

The models under consideration are: Logistic Regression, Decision Trees, Random Forests, AdaBoost, Gradient Boosting and lastly Linear Support Vector Machine (SVM).

Since the extreme class imbalance present in the data set, and the imbalance in the data set is likely to cause the classification algorithm to be less accurate, the data set needs to be balanced in some way. It was decided to use the original data set, oversampling and undersampling methods to fit the models.

For model evaluation we used Precision, Recall, F1-score and AUPRC metrics. The area under the precision recall curve (AUPRC) is a far better measure than accuracy. Because we can get over 99% accuracy by predicting "false" all the time. For the first method ROC curves are given when appropriate, and then only for the Gradient Boosting models further along (refer to Jupyter notebook).

3.2 *Using Original Data Set*

This is with no adjustments made other than the removal of the 'id' feature and appropriate scaling.

Model	Class	Precision	Recall	F1	AUPRC
Logistic Regression	0/1	1.00/0.01	0.85/0.65	0.89/0.01	0.33
Decision Tree	0/1	1.00/1.00	1.00/0.12	1.00/0.21	0.56
Random Forest	0/1	1.00/0.87	1.00/0.12	1.00/0.2	0.49
AdaBoost	0/1	1.00/0.52	1.00/0.13	0.90/0.21	0.33
Gradient Boosting	0/1	1.00/0.52	1.00/0.13	1.00/0.21	0.33
Linear SVM	0/1	1.00/0.5	1.00/0.02	1.00/0.03	0.26

Table 1: Model Performance Statistics using Original Data

Overall, we find that whilst predictions for class 0 are fairly good (maybe even overfitting), the same is not true for class 1 (as indicated by the red markings). The AUPRC (Area under Precision-Recall curve) is low for most of the models fitted. Varying hyperparameters yielded negligible improvements, and so, we continued on to the next method: oversampling. Also, note the colour markings, red indicate issues, green indicate good metric values.

3.2 *Oversampling*

This technique ensures both classes have the same number of observations and hence they are balanced (actually equal to each other)

Model	Class	Precision	Recall	F1	AUPRC
Logistic Regression	0/1	0.71 / 0.76	0.78 / 0.69	0.75 / 0.72	0.80
Decision Tree	0/1	0.62 / 0.83	0.91 / 0.45	0.74 / 0.58	0.78
Random Forest	0/1	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	~ 1.00
AdaBoost	0/1	0.78 / 0.81	0.82 / 0.76	0.80 / 0.79	0.85
Gradient Boosting	0/1	0.83 / 0.86	0.86 / 0.83	0.85 / 0.84	0.89
Linear SVM	0/1	0.71 / 0.76	0.78 / 0.69	0.75 / 0.72	0.80

Table 2: Model Performance using Oversampling

Firstly, we note that we have much better AUPRC values and clearly much better fits. However, this comes with an increased computational cost, especially when running Gradient Boosting (see ROC curve in notebook – much better than the one corresponding to the under sampling method). Moreover, whilst the fit of the models may have improved, we risk introducing bias and extra variance into it as well. It also appears as if we have overfit the Random Forest model. Let us lastly consider under-sampling next.

3.3 *Undersampling*

As opposed to oversampling, under sampling shrinks the data set down. We ensure, however, that no observations of the minority class are discarded since so few exist. We therefore sample the majority class and then fit models.

Undersampling appears to (see table 3 on following page) have caused a drop in overall performance compared to oversampling. Precision, recall and the F1 scores for class 1 have decreased, but for class 0 these appear to have overall increased. The AUPRC values have also decreased. Overall, for this method, we conclude that there is no improvement.

Model	Class	Precision	Recall	F1	AUPRC
Logistic Regression	0/1	0.93/ 0.40	0.81 / 0.67	0.87 / 0.50	0.56
Decision Tree	0/1	0.89/ 0.61	0.95 / 0.39	0.92 / 0.48	0.55
Random Forest	0/1	0.90 / 0.77	0.98 / 0.44	0.94 / 0.56	0.65
AdaBoost	0/1	0.90 / 0.68	0.96 / 0.46	0.93 / 0.55	0.61
Gradient Boosting	0/1	0.90 / 0.72	0.97 / 0.43	0.93 / 0.54	0.62
Linear SVM	0/1	0.90 / 0.79	0.98 / 0.38	0.94 / 0.52	0.64

Table 3: Model Performance using Undersampling

Lastly, we assess the performance by evaluating on Kaggle using the model associated with oversampling (specifically using the gradient boosting method with 100 estimators, 0.1 learning rate, a maximum depth of 3 and loss set to 'deviance'). See Figure 1 in the appendix. The overall private (calculated on 80% of the test data) and public scores (calculated on 20% of the test data) are 0.75 and 0.80, respectively. This fit might be improved by using grid_search, however, we were not able to successfully run it (time-wise).

Conclusion and Discussion

In this project, we analysed data structures and evaluated estimation methods to gain insight into credit card transaction data. By analysing the different performance of six models under three different data processing scenarios, we found methods suitable (and unsuitable) for detecting credit card fraud. The result shows that random forest and gradient boosting methods can achieve better detection of transaction fraud in the presence of data oversampling.

However, as this study only contains numerical input variables, i.e., not the actual feature name as a result of the PCA transformation, we are unable to discuss in depth the impacts of each feature beyond correlation in practice. It is important to determine which features are important for credit card fraud detection so that they can be used more effectively when building models and making predictions. Typical models only use raw transaction features, however, these approaches do not take into account the customer's spending behaviour. (Bahnsen et al., 2016)

Therefore, improvements to the data collection and processing process are needed to identify and retain the most meaningful features and to strengthen the correlation between these features. In terms of improving the data collection strategy, we suggest a transaction aggregation strategy (Whitrow et al., 2009) that takes into account the customer's consumption behaviour. It involves grouping transactions within a recent time period, first by card or account number, then by transaction type, merchant group, country or other, and then calculating the number of transactions or the total amount spent on these transactions.

In summary, in terms of data collection, the focus is on ensuring that the data is comprehensive, accurate and covers as many characteristics as possible. Also appropriate data processing will effectively improve the use of credit card fraud detection models. (Bahnsen et al., 2016)

References

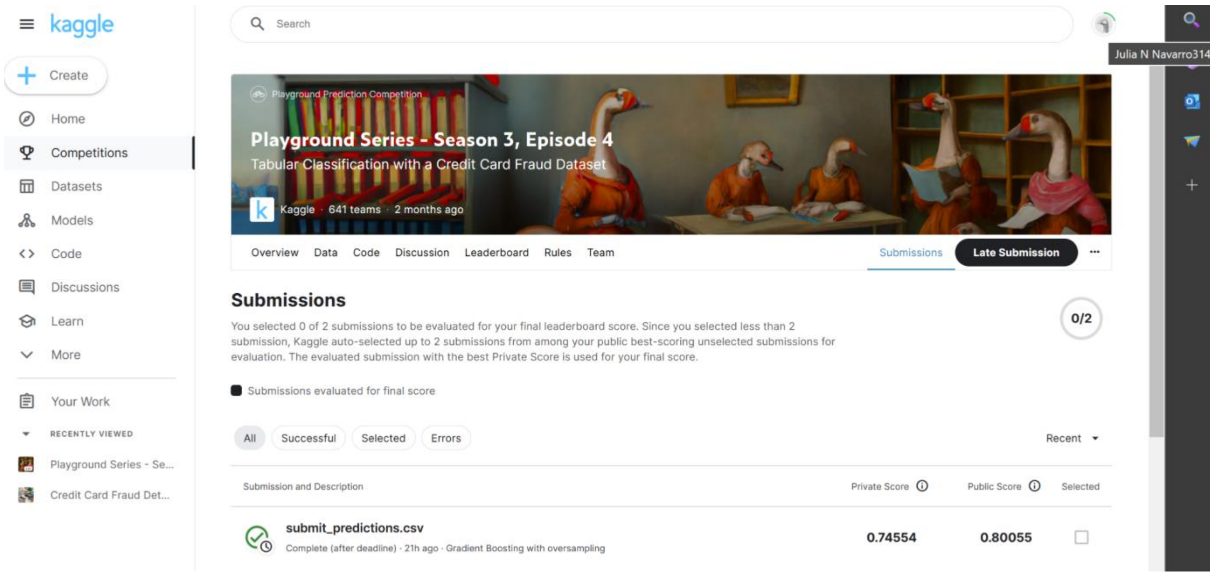
Bahnsen, A. C., Aouada, D., Stojanovic, A. and Ottersten, B. (2016) 'Feature engineering strategies for credit card fraud detection', *Expert Systems with Applications,* 51, pp. 134-142.

ULB, W. a. t. M. L. G. o. (2018) 'Credit Card Fraud Detection-Anonymized credit card transactions labeled as fraudulent or genuine'. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (Accessed April 2023)

Whitrow, C., Hand, D. J., Juszczak, P., Weston, D. and Adams, N. M. (2009) 'Transaction aggregation as a strategy for credit card fraud detection', *Data mining and knowledge discovery,* 18, pp. 30-55.

Appendix

Figure 1:
Kaggle Output for Final Model:



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

0/2

■ Submissions evaluated for final score

All · Successful · Selected · Errors · Recent

Submission and Description	Private Score	Public Score	Selected
submit_predictions.csv Complete (after deadline) · 21h ago · Gradient Boosting with oversampling	0.74554	0.80055	☐