200037417,190005680 , 190001593, 220009855, 210033036

ID5059 P2: Group Project Report

This report is an investigation into the Playground Series Credit Card Fraud Detection on Kaggle. The dataset was generated from a deep learning model trained on the Credit Card Fraud Detection. Our goal is to construct several simple machine learning models which can predict the Class attribute using the other attributes included in the dataset. This variable is binary, therefore, making this a binary classification problem. First we investigated the properties of the dataset. Next, we selected a method for imputing missing data values and prepared the dataset for use in modelling. Following this, a number of models were assessed, these were: Stochastic Gradient Descent, Logistic Regression, Random Forest, Decision Tree, and Histogram-Gradient Boosting classifiers**.** We assessed each model with a performance measure and analysed the confusion matrices to assess the accuracy, precision, and recall of each model. This report will explain the process for each section and conclude the research findings at the end.

Data Exploration, Cleaning, & Imputation

The first step taken in this model fitting procedure was an exploration of both the training dataset, which were used to fit our models. The first feature we examined was the Class variable. This variable takes the value 1 indicates a transaction was fraudulent and this variable taking the value 0 indicates a transaction was not fraudulent. We first wanted to examine what proportion of transactions were fraudulent and non-fraudulent transactions for this variable in each dataset.

Table 1 gives the proportion of each Class outcome present in both the training and testing datasets. It can be seen from this table that there is a very small proportion of fraudulent transactions in our training data compared to non-fraudulent transactions. This means we are training our models on very few fraudulent cases in comparison to non-fraudulent cases which may lead to our models having difficulty producing accurate results. We will return to this in the Models and Performance Section of this report.

Class	Proportion
0	218660 entries ~ 99.786%
1	469 entries ~ 0.214%

Table 1: The proportion of each Class outcome in the training and testing datasets

The correlations between the different variables within the training dataset were also examined. *Figure 1* is a correlation plot of the training dataset. The plot indicates that the V2 and amount variables in the dataset are highly negatively correlated, e.g. if one is high the other is low, as well as indicating that Amount and V20 are highly positively correlated, e.g. if one is high the other is also high and vice versa.

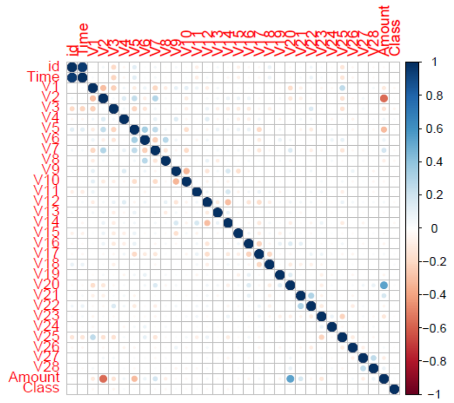


Figure 1: Correlation Plot of the training dataset

It was of particular interest to examine what variables are most correlated with class in the training dataset. *Figure 2* is a bar graph that gives the correlation of all variables in the dataset and has Class in descending order by correlation. It can be seen from this that the variables V3, V1 and V14 are the variables with the highest correlation with Class.

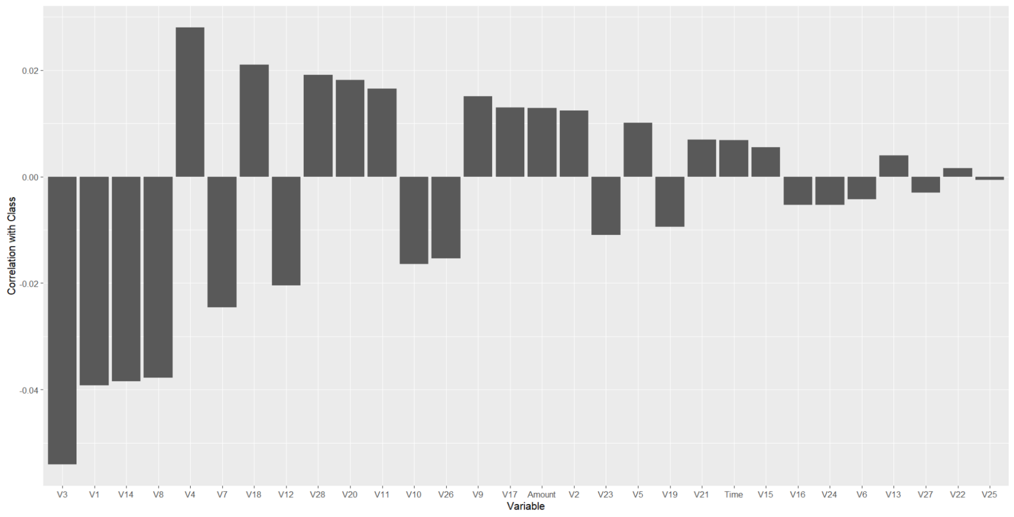


Figure 2: Bar Graph of the correlations between variables

In order to allow the models to have a better predictive power, outliers were removed from the data before training the models. It was decided that an outlier was an observation that was more than 6 standard deviations away from the mean for at least one variable. These outliers were removed as they may have influenced our models to assume there were correlations between variables where there were no significant correlations. No observations of transactions that were fraudulent were removed, however, as there was such a small proportion of them in the original training data.

Initially, the dataset has no missing values, but we will deliberately introduce missing data into the dataset. To create a realistic scenario for implementing and comparing imputation methods, we generated random missing data for the 'V1' column. The analysis includes Mean, Median, and Mode Imputations, K-Nearest Neighbour (KNN) Imputation, and Multivariate Imputation by Chained Equation (MICE). The performance of each method is assessed and compared to identify the most effective approach for dealing with missing data.

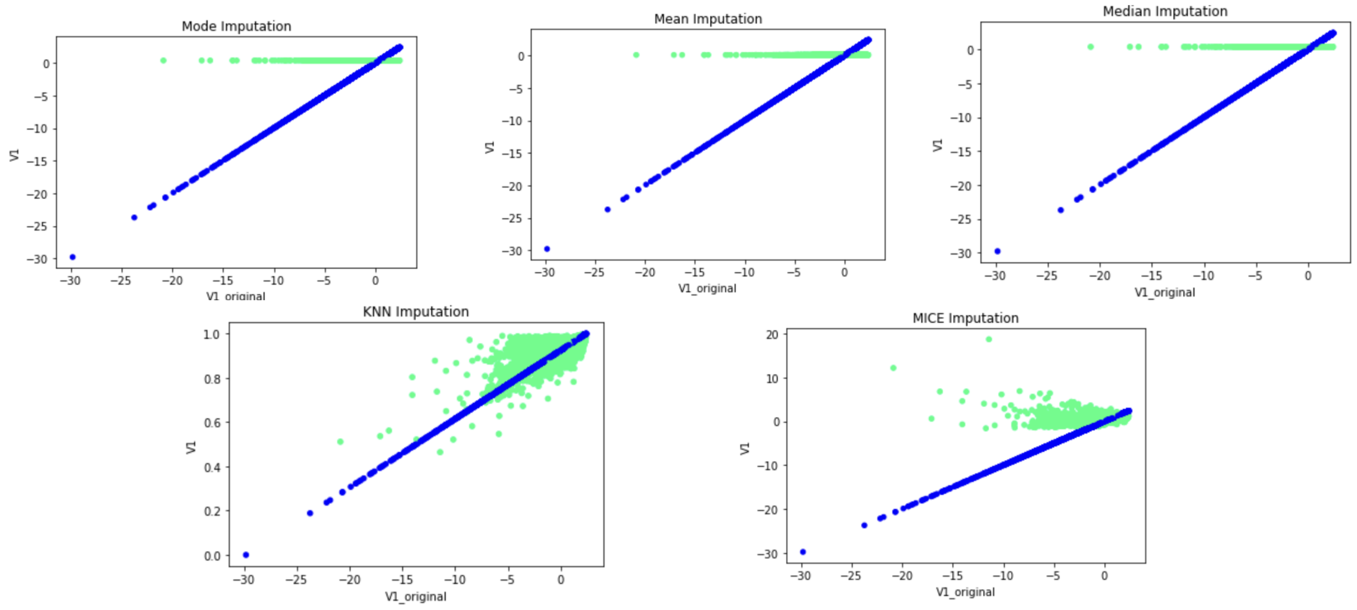


Figure 3: Scatter plots of the imputed values compared for Mode (top-left), Mean (top-middle), Median (top right), MICE (bottom left), KNN (bottom right)

From *Figure 3,* we see the Mean, Mode, and Median imputations with green points representing the imputed data and blue points representing the original data. The KNN Imputation method was implemented using a subset of the dataset containing 'Time', 'V1', 'V2', and 'V3' columns. The data was scaled between 0 and 1 before applying the KNN imputer with 5 nearest neighbours. MICE imputation was applied to the same subset of the dataset as KNN Imputation. A Bayesian Ridge estimator was used for the iterative imputation process. The imputed values were visualised in a scatter plot, similar to the other methods. The performance of the imputation methods was compared by assessing how closely the imputed values aligned with the original dataset. *Figure 3* displays the following observations:

- Mean, Median, and Mode Imputations are simple techniques but can introduce biases, particularly in cases of skewed distributions.
- MICE Imputation performs better than Mean, Median, and Mode Imputations, providing a more accurate representation of the original data. However, it still falls short when compared to KNN Imputation.
- KNN Imputation demonstrates the best performance among the compared methods, providing imputed data that closely aligns with the original dataset.

KNN Imputation demonstrates the most effective performance among the compared methods for dealing with missing data. While Mean, Median, and Mode Imputations offer a straightforward approach, they can introduce biases and inaccuracies. MICE Imputation performs better than these basic techniques but is outperformed by KNN Imputation. Therefore, KNN Imputation is recommended for optimal performance when handling missing data in similar contexts.

Model Training & Selection

In this section we will develop models to determine how likely a transaction is fraudulent. We employ Stochastic Gradient Descent (SGD), Logistic Regression Classifier, Random Forest Classifier (RF), Decision Tree Classifier, and a Histogram-based Gradient-Boosting Classification Tree. Our target column has a significant imbalance with fraudulent cases in the minority so performance measures such as accuracy could not be reliable, so we use F1 scores (harmonic mean of precision and recall) and ROC curves as our main measures of model performance. Lastly, we try to improve the model through means such as hyperparameter-tuning. We make a slight adjustment with the time column so its hours range from 0-24. This will reduce variance within the column and hopefully work better for training. We can also investigate whether the time of day a transaction is made has any effect on the probability that it is fraudulent. Lastly, we standardize the hour and amount column which will make for better inputs to train our models with. Recall, there is an imbalance between fraudulent and non-fraudulent cases in our data. This will be due to the lack of data for fraudulent transactions and will affect our model performance. We test this first by training a simple linear model.

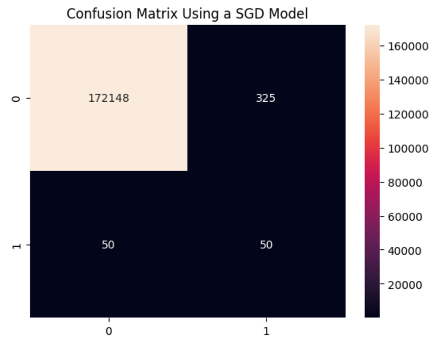


Figure 4: Confusion Matrix using an SGD Model

Figure 4 shows that our model does not accurately classify fraudulent transactions. The high accuracy for non-fraudulent cases is a consequence of the high proportion of non-fraudulent transactions in the dataset. The F-1 score for SGD on training is 23.56% which shows the model cannot predict if a transaction is fraudulent or not. What can we do to tackle this issue? The two main methods for dealing with imbalanced classes are oversampling and undersampling. For this work, we will use the oversampling approach using SMOTE (Synthetic Minority Oversampling Technique).

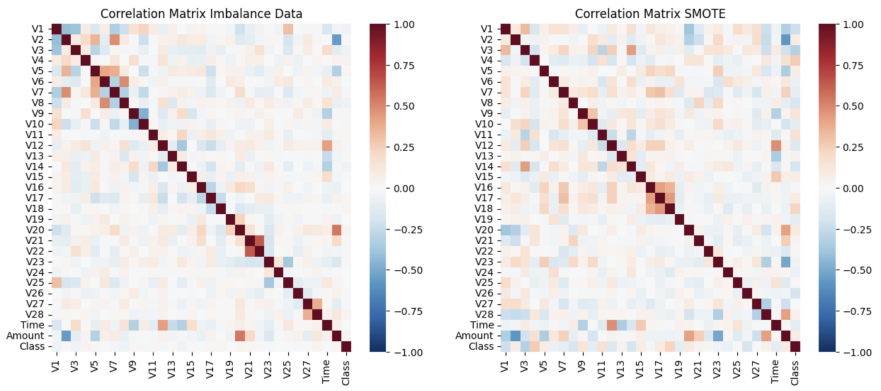


Figure 5: Correlation Matrix for Imbalance Data VS SMOTE approach

Comparing the last row (Class) for both approaches in *Figure 5*, it is evident that using a balanced data set (SMOTE) shows the response has a higher correlation with each feature. We fit another SGD classifier model to the oversampled dataset.

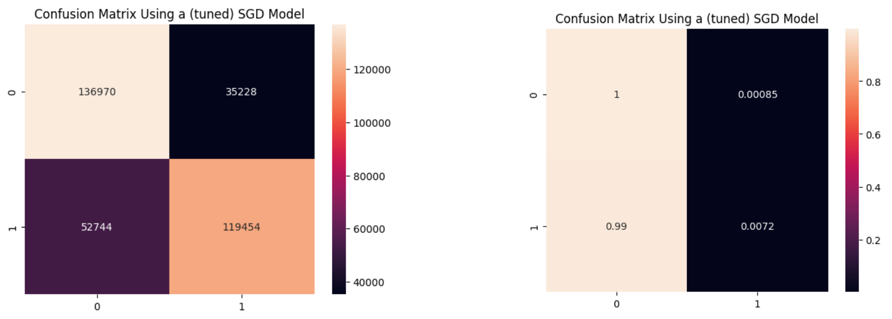


Figure 6: Confusion Matrix SGD SMOTE Training (left) VS Validation Set (right)

The confusion matrix for SGD on the balanced training dataset shown by *Figure 6* is more interpretable: We have twice the amount of false negatives than false positives which indicates the model has performance issues when a transaction is fraudulent. The F-1 score for the training set is 71.07% which is much better than the performance on the imbalanced training set. We also ran SGD on the validation set, and it shows a similar structure as the training confusion matrix. The F-1 score on the validation set is 71.04%. Next we try the Logistic Regression Classifier on the training and validation set.

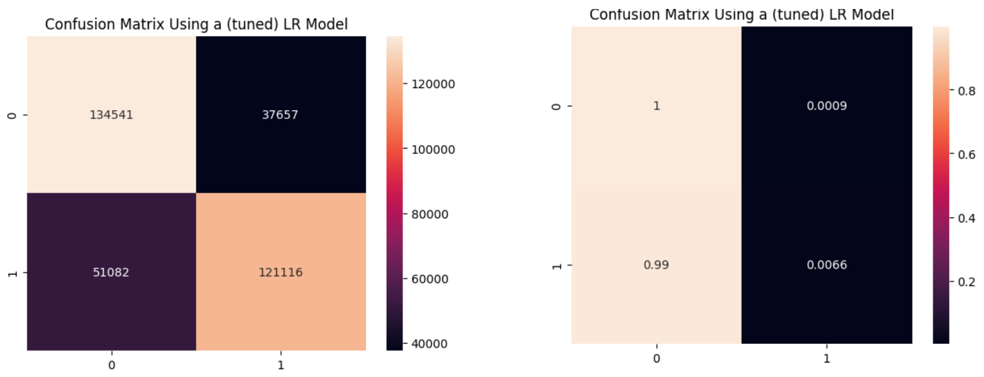


Figure 7: Confusion Matrix Logistic Regression Classifier SMOTE Training (left) VS Validation Set (right)

Figure 7 shows that the confusion matrices for the training and validation set are the same structure. The F1-score on the training (SMOTE) set is 72.44% and 72.41% on the validation set. This is good performance, but we must also look at the probabilities for each class. Next, we try a Random Forest Classifier. We start with a simple Random Forest Classifier with the following hyperparameters: max eatures = 9, max depth = 10, min samples leaf = 5. We track time spent fitting this model and it took 2 minutes. This model is intensive on time, and we will have to keep that in mind when selecting hyperparameters.

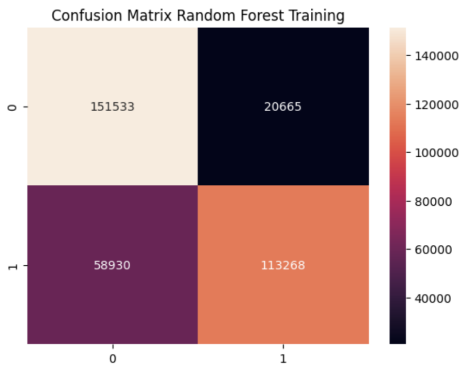


Figure 8: Confusion Matrix Random Forest Training SMOTE Training

From *Figure 8*, we notice the false positives (top right) and false negatives (bottom left) are more imbalanced than in previous models. The F-1 score is 74% which is in line with other models, but we must remember this model is very sensitive to hyperparameter values. We use grid search to find the best hyperparameters for this model. The results say we should structure with max depth = 10, max features = 10 and minimum samples leaf = 6. We run another Random Forest Classifier Model with these hyper parameters:

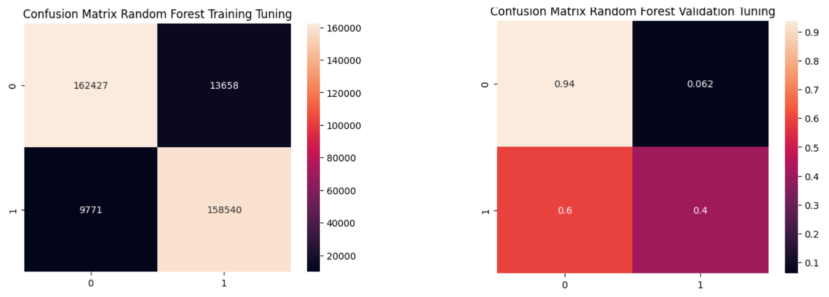


Figure 9: Confusion Matrix Random Forest Tuned SMOTE Training Set (Left) VS Validation Set (RIght)

Figure 9 shows a much better confusion matrix. The errors are more balanced between false positives and false negatives. The F-1 score for the training set with the tuned hyperparameters is 92.15% and 91.7% for the validation set.

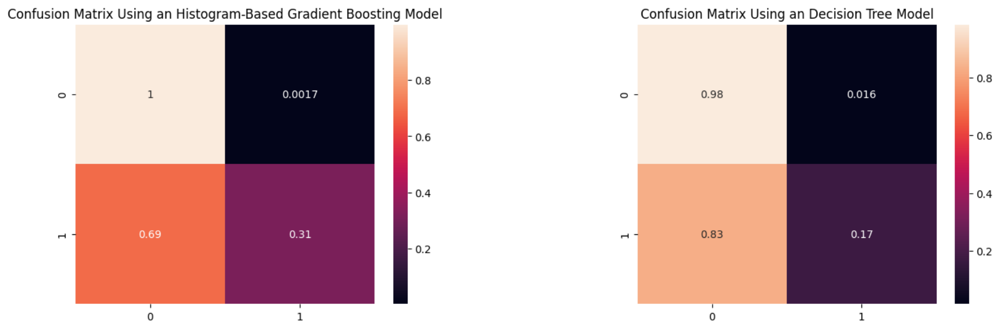


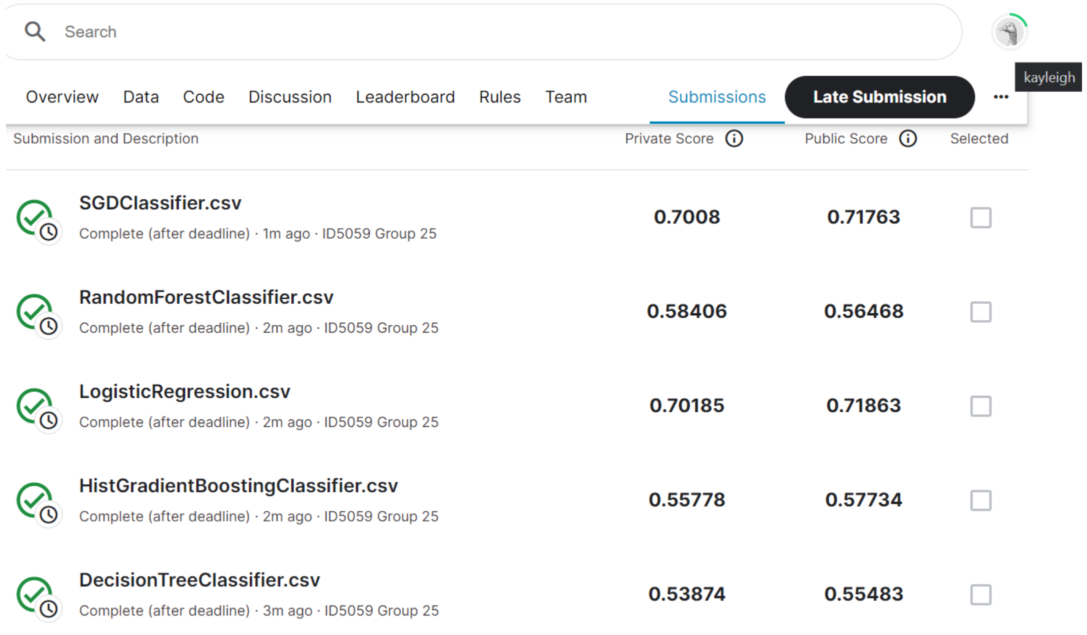
Figure 10: Confusion Matrix Histogram Based Gradient Model (left) VS Decision Tree Model (right)

Lastly, we fit a decision tree model with hypertuning parameters and a histogram-based gradient boosting model. Confusion matrix results can be seen in *Figure 10*. The decision tree model displays poor accuracy when it comes to classifying fraudulent transactions.

Results / Recommendations / Conclusions

This report demonstrated our approach to analysing the [Playground Series Credit Card Fraud Detection on Kaggle](). Through data exploration, we learned we are training our models on very few fraudulent cases in comparison to non-fraudulent cases. In our cleaning and imputation processes we removed outliers and implemented various imputation methods. In our model development, we resampled our data with SMOTE (Synthetic Minority Oversampling Technique) due to the imbalance in the data. Various machine learning models were fit including Stochastic Gradient Descent, Logistic Regression, and Random Forest Classifier, Decision Tree Classifier, and Histogram-Based Gradient-Boosting Classifier.

From testing on the dataset, we came to the conclusion that our Logistic Regression model would perform the best, however when running against the test set in order to calculate our Kaggle scores we received the following:



Submission and Description	Private Score	Public Score	Selected
SGDClassifier.csv Complete (after deadline) · 1m ago · ID5059 Group 25	0.7008	0.71763	☐
RandomForestClassifier.csv Complete (after deadline) · 2m ago · ID5059 Group 25	0.58406	0.56468	☐
LogisticRegression.csv Complete (after deadline) · 2m ago · ID5059 Group 25	0.70185	0.71863	☐
HistGradientBoostingClassifier.csv Complete (after deadline) · 2m ago · ID5059 Group 25	0.55778	0.57734	☐
DecisionTreeClassifier.csv Complete (after deadline) · 3m ago · ID5059 Group 25	0.53874	0.55483	☐

Figure 11: Kaggle Scores for Implemented Models

These results suggest that it was our Logistic Regression model which in fact performed the best. Further investigation would have ideally been carried out to determine these reasons; this would have involved a more in-depth tuning of our non-linear models (it is possible that these could have performed even better, but their potential was simply not realised).

Bibliography

Idil I. (2022). *Imputing Missing Data with Simple and Advanced Techniques*. Available at: https://towardsdatascience.com/imputing-missing-data-with-simple-and-advanced-techniques-f5c7b157fb87

Bento, C. (2021). *Decision Tree Classifier explained in real-life: picking a vacation destination*. Available at: https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575