Group Name: Algorithmic Bond (Group-2)

Group Members (First name, Surname, URN):

Arjoo Gupta (URN: 6657774)
Ememobong Ekpenyong (URN: 6676807)
Ashwin Baiju (URN: 6707489)
Judith Etiobhio (URN: 6347971)
Patrick Oramah (URN: 6677861)
Ibukun Omojola (URN: 6694081)

Credit card Fraud Detection

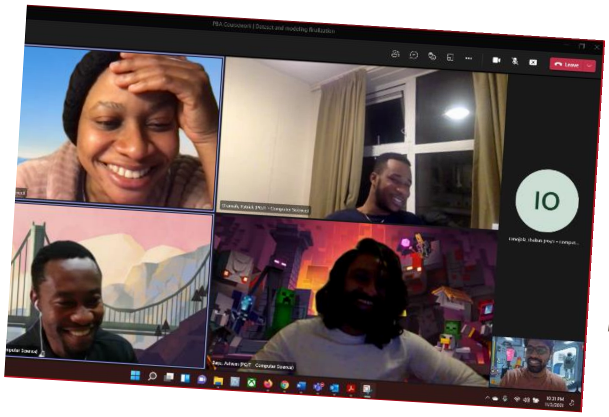


Table of Contents

1. INTRODUCTION

1.1. Problem Definition

In recent times, with the rapid advances in electronic commerce, there has been an unprecedented use of credit cards for purchases on the internet which has become convenient and necessary. As a result, credit card transactions have become widely accepted as the de facto standard for web-based e-commerce (Chan, 1999). According to UK Finance (2021), there were 312 million credit card transactions in July 2019, 19.1% more than July 2020, and a total spend of £69.4 billion from the United Kingdom (UK) and overseas countries. However, the total number of growing transactions provides more opportunities to defraud cardholders of their money. In the UK, total annual value of fraud losses on credit cards issued as of 2020 reached a value of £574.2million (Statista, 2021). Therefore, credit cards have become a target by fraudsters to perpetuate their illicit schemes.

Fraud is the process of obtaining services/goods and/or money by unethical means, and it is becoming a growing concern around the world. A fraud can occur with any type of credit products, such as home loans, personal loans, and retail (Delamaire et al, 2009). Additionally, credit card fraud has broader implications, as such provides funding for international narcotics trafficking, organised crime, and terrorist financing (Bhattacharyya et al., 2011).

The face of fraud has changed drastically during the last few decades as technologies have evolved and developed. As a result, the perpetrators of fraud have also been evolving their fraud schemes to avoid detection. Therefore, it is crucial to help businesses, financial institutions including banks to take steps to prevent fraud and deal with it efficiently and effectively. Credit card fraud detection techniques need constant innovation and intelligent mechanisms to combat the illicit schemes of fraudsters. (Anderson, 2007). Hence, early fraud detection is critical to significantly reduce the illegitimate use of credit cards (Chan et al., 1999).

In earlier studies, some approaches have been proposed to design solutions to detect fraud using supervised, unsupervised learning and hybrid approaches. Some examples of such studies in this regard are as follows:

	Author	**Contributions**	**Limitations**
1	Khan et al.(2014a)	Proposed a real time fraud detection technique	The misclassification rate is low, low classification speed and long training time.
2	Dornadula and Geetha (2019)	Proposed a novel solution which aims to overcome the problem of concept drift, which means when card transactions are unfamiliar with previous transactions made by a customer.	The data imbalance was an issue
3	Sahin and Duman (2011)	An SVM and decision tree algorithm was employed to detect credit card fraud.	The data imbalance was an issue
4	Shukar and Kurnaz (2019)	Proposed a solution using supervised and unsupervised learning algorithms: Logistic Regression, K-means clustering model, and a Neutral Network to build a classifier in detecting fraud.	Complexity increases with more trees
5	Modi et al.(2013)	A novel and improved solution was proposed to detect a credit card detection solution	The Artificial neural network (ANN)training time is slow

Table 1: *Contributions and limitations of reviewed techniques (Adewumi and Akinyelu, 2016)*

According to Trivedi et al., (2005), the characteristics of a good fraud detection system are: 1) It should identify the frauds accurately 2) It should detect the frauds in a timely manner 3) It should not classify a legitimate transaction as fraud.

1.2. Objective

This project aims to build a classifier with various machine learning models and statistical techniques that will distinguish between a potentially legitimate transaction from a fraudulent transaction with a certain degree of accuracy. A solution will be recommended to the business based on business requirements as we expect the number of correctly classified frauds (True Positive) and number of incorrectly classified frauds (False Positive) to vary per model.

The Cross Industry Standard for Data Mining (CRISP-DM) methodology process model will be employed as the template for this project.

1.3. Overview of the Dataset

The observations or records in the dataset are credit card transactions that took place within 48 hours. Each row depicts a credit card transaction and consists of 31 columns, each describing an aspect of the transaction. There are 284,807 such transactions recorded in the dataset which has been collected over a timespan of two days in September 2013 by European cardholders, out of which 492 are frauds. On average, 246 fraudulent transactions can occur in a day, which translates to 89,790 transactions in a year (365 days). A total of €60,127.97 was lost in two days, resulting in €30,063 per day and in a year about €10.9M could be lost to fraud if a solution is not deployed to address this issue.

1.4. Dataset description

The dataset contains sensitive information about credit card transactions and 28 out of 31 features have been transformed with Principal Component Analysis (PCA) and these transformed features are named sequentially from "V1" to "V28" for confidentiality purposes.

The "Time" feature represents a value of time in seconds that has elapsed since the first transaction in the dataset has been made. The last row of the dataset has a time value which is about 48 hours after the first record/transaction in the dataset.

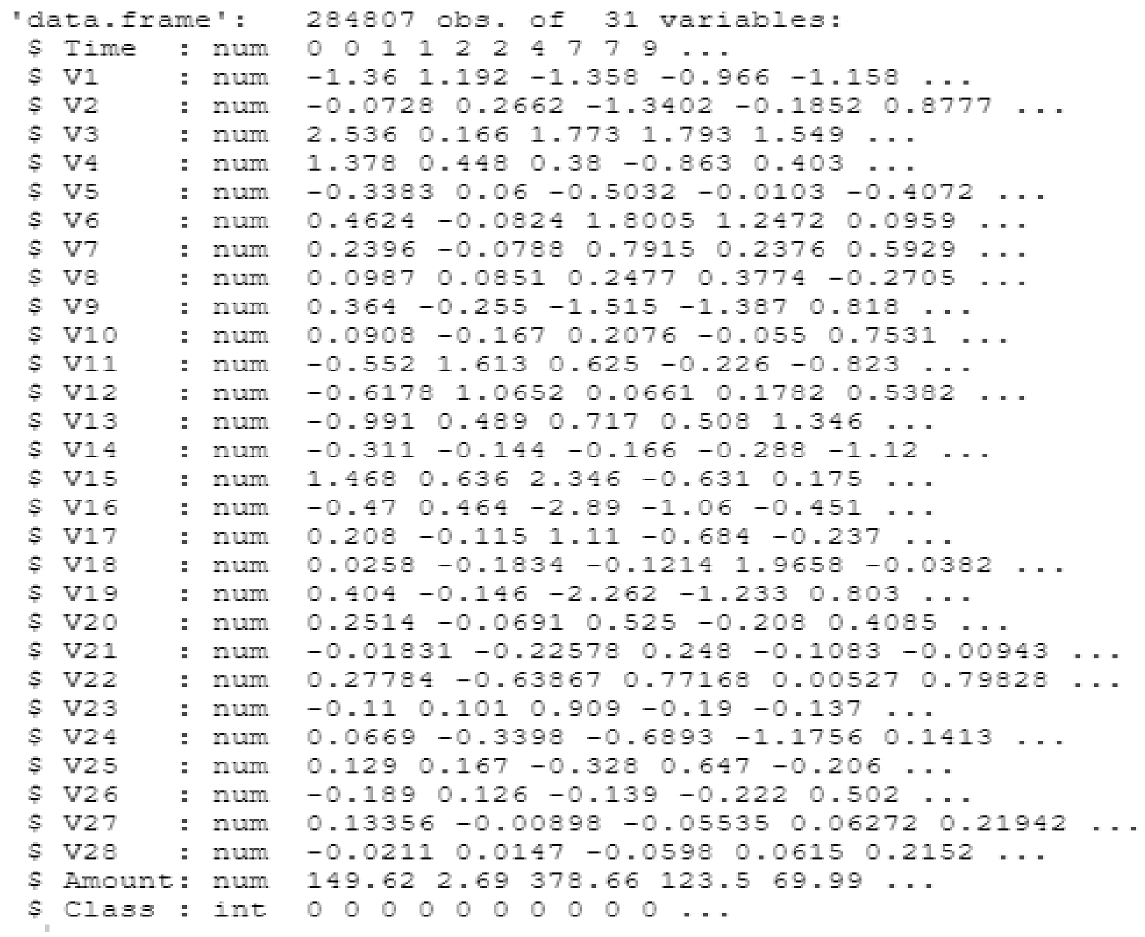
V1 to V28 features have normal distributions with mean approximately equal to 0 and standard deviation approximately equal to 1.

The "Amount" and "Class" features are not normalized. Amount represents transaction value. Class is a discrete feature with just two values, 0 and 1. 0 means the transaction was legitimate (non-fraudulent) - Negative case and 1 means the transaction was fraudulent - Positive case.

2. METHODS

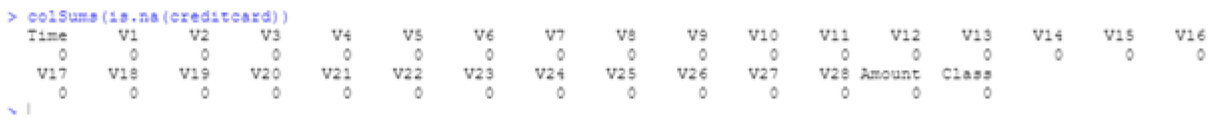
2.1. Exploratory Data Analysis

The dataset contains 30 numeric features which are the results of Principal Component Analysis (PCA) transformation to protect the confidentiality of customers' data, except 'Amount' and 'Time' features and the Class (or target) is captured as a factor variable. The Class feature contains categories '0' (Negative case) for non-fraud transactions and '1' (positive case) for fraud transactions. The figure below displays the data type and the first few records for all 31 features in our dataset. This will guide our decision-making process on the required preprocessing and feature engineering techniques to be implemented before feeding the inputs into our models.



```
'data.frame':   284807 obs. of  31 variables:
 $ Time  : num  0 0 1 1 2 2 4 7 7 9 ...
 $ V1    : num  -1.36 1.192 -1.358 -0.966 -1.158 ...
 $ V2    : num  -0.0728 0.2662 -1.3402 -0.1852 0.8777 ...
 $ V3    : num  2.536 0.166 1.773 1.793 1.549 ...
 $ V4    : num  1.378 0.448 0.38 -0.863 0.403 ...
 $ V5    : num  -0.3383 0.06 -0.5032 -0.0103 -0.4072 ...
 $ V6    : num  0.4624 -0.0824 1.8005 1.2472 0.0959 ...
 $ V7    : num  0.2396 -0.0788 0.7915 0.2376 0.5929 ...
 $ V8    : num  0.0987 0.0851 0.2477 0.3774 -0.2705 ...
 $ V9    : num  0.364 -0.255 -1.515 -1.387 0.818 ...
 $ V10   : num  0.0908 -0.167 0.2076 -0.055 0.7531 ...
 $ V11   : num  -0.552 1.613 0.625 -0.226 -0.823 ...
 $ V12   : num  -0.6178 1.0652 0.0661 0.1782 0.5382 ...
 $ V13   : num  -0.991 0.489 0.717 0.508 1.346 ...
 $ V14   : num  -0.311 -0.144 -0.166 -0.288 -1.12 ...
 $ V15   : num  1.468 0.636 2.346 -0.631 0.175 ...
 $ V16   : num  -0.47 0.464 -2.89 -1.06 -0.451 ...
 $ V17   : num  0.208 -0.115 1.11 -0.684 -0.237 ...
 $ V18   : num  0.0258 -0.1834 -0.1214 1.9658 -0.0382 ...
 $ V19   : num  0.404 -0.146 -2.262 -1.233 0.803 ...
 $ V20   : num  0.2514 -0.0691 0.525 -0.208 0.4085 ...
 $ V21   : num  -0.01831 -0.22578 0.248 -0.1083 -0.00943 ...
 $ V22   : num  0.27784 -0.63867 0.77168 0.00527 0.79828 ...
 $ V23   : num  -0.11 0.101 0.909 -0.19 -0.137 ...
 $ V24   : num  0.0669 -0.3398 -0.6893 -1.1756 0.1413 ...
 $ V25   : num  0.129 0.167 -0.328 0.647 -0.206 ...
 $ V26   : num  -0.189 0.126 -0.139 -0.222 0.502 ...
 $ V27   : num  0.13356 -0.00898 -0.05535 0.06272 0.21942 ...
 $ V28   : num  -0.0211 0.0147 -0.0598 0.0615 0.2152 ...
 $ Amount: num  149.62 2.69 378.66 123.5 69.99 ...
 $ Class : int  0 0 0 0 0 0 0 0 0 0 ...
```

***Figure 1**: Data Type representation of dataset*

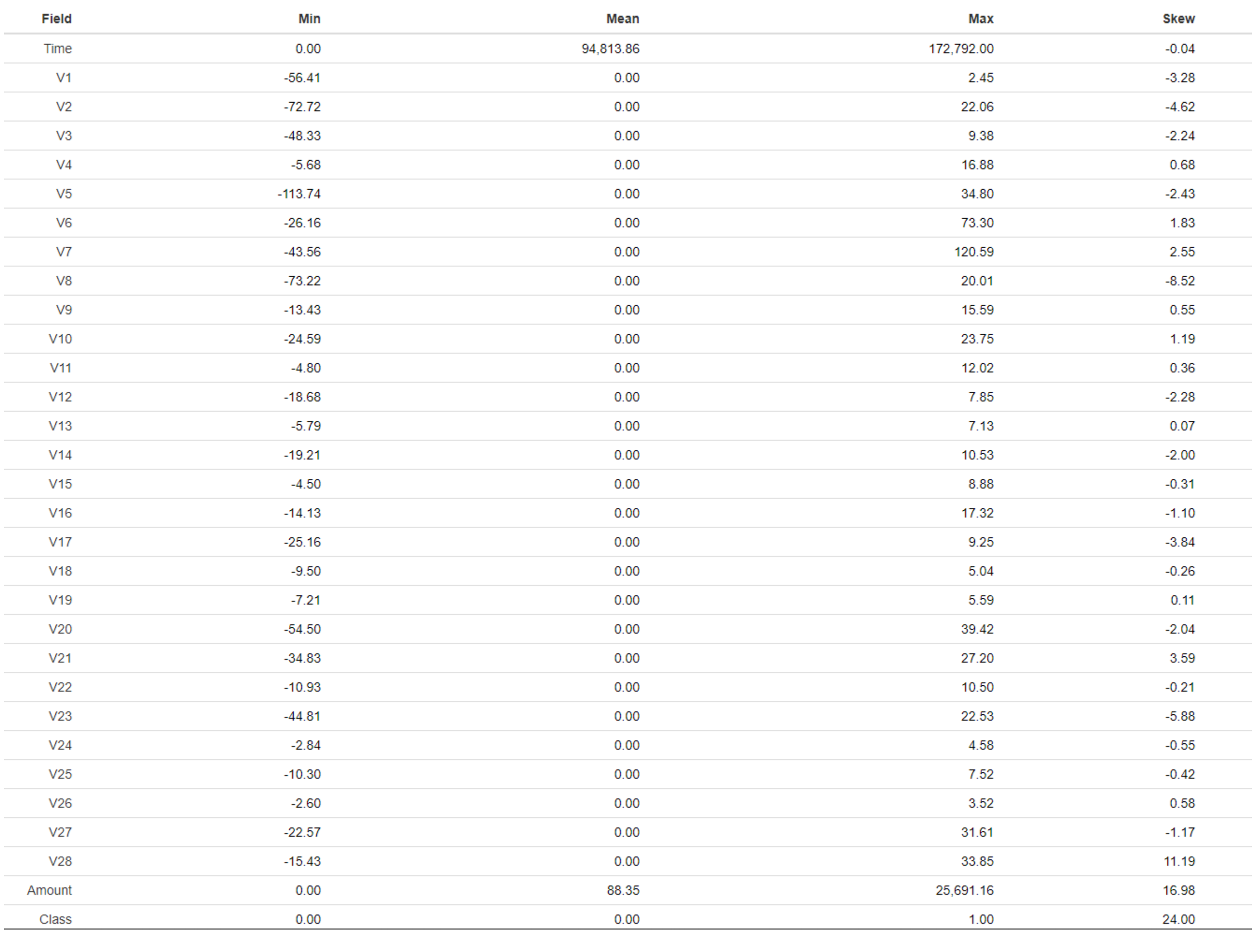


```
> colSums(is.na(creditcard))
  Time     V1     V2     V3     V4     V5     V6     V7     V8     V9    V10    V11    V12    V13    V14    V15    V16
     0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
   V17    V18    V19    V20    V21    V22    V23    V24    V25    V26    V27    V28 Amount  Class
     0      0      0      0      0      0      0      0      0      0      0      0      0      0
```

***Figure 2**: Dataset Null Values Evaluation*

As shown above, the dataset does not contain null values, hence data imputation not required before modelling.

The key aspect of any analytics problem is to analyze the data based on certain statistical metrics. From the figure 3 below, it can be inferred that the 'Amount' and 'Time' features do not have any negative values. Secondly, since all other parameters are normalized using Principal Component Analysis (PCA), V1-V28 variables have mean 0 and some values are negative as about half of the values will lie before and after the mean since the variables are normally distributed.



Field	Min	Mean	Max	Skew
Time	0.00	94,813.86	172,792.00	-0.04
V1	-56.41	0.00	2.45	-3.28
V2	-72.72	0.00	22.06	-4.62
V3	-48.33	0.00	9.38	-2.24
V4	-5.68	0.00	16.88	0.68
V5	-113.74	0.00	34.80	-2.43
V6	-26.16	0.00	73.30	1.83
V7	-43.56	0.00	120.59	2.55
V8	-73.22	0.00	20.01	-8.52
V9	-13.43	0.00	15.59	0.55
V10	-24.59	0.00	23.75	1.19
V11	-4.80	0.00	12.02	0.36
V12	-18.68	0.00	7.85	-2.28
V13	-5.79	0.00	7.13	0.07
V14	-19.21	0.00	10.53	-2.00
V15	-4.50	0.00	8.88	-0.31
V16	-14.13	0.00	17.32	-1.10
V17	-25.16	0.00	9.25	-3.84
V18	-9.50	0.00	5.04	-0.26
V19	-7.21	0.00	5.59	0.11
V20	-54.50	0.00	39.42	-2.04
V21	-34.83	0.00	27.20	3.59
V22	-10.93	0.00	10.50	-0.21
V23	-44.81	0.00	22.53	-5.88
V24	-2.84	0.00	4.58	-0.55
V25	-10.30	0.00	7.52	-0.42
V26	-2.60	0.00	3.52	0.58
V27	-22.57	0.00	31.61	-1.17
V28	-15.43	0.00	33.85	11.19
Amount	0.00	88.35	25,691.16	16.98
Class	0.00	0.00	1.00	24.00

Figure 3: *Summary of the data types*

2.2. Data Visualization of the Target Variable

Imbalanced data refers to the presence of an unequal number of instances for each category of the target variable. The visualization shown below further reflects the imbalance of non-fraud and fraud transactions in the dataset. We have class (0 — No fraud, 1 — fraud) on the X-axis and the count of instances plotted on Y-axis. It is observed that the dataset is highly unbalanced with respect to the class distribution (Fraud: 492 and Non-fraud: 284,315).

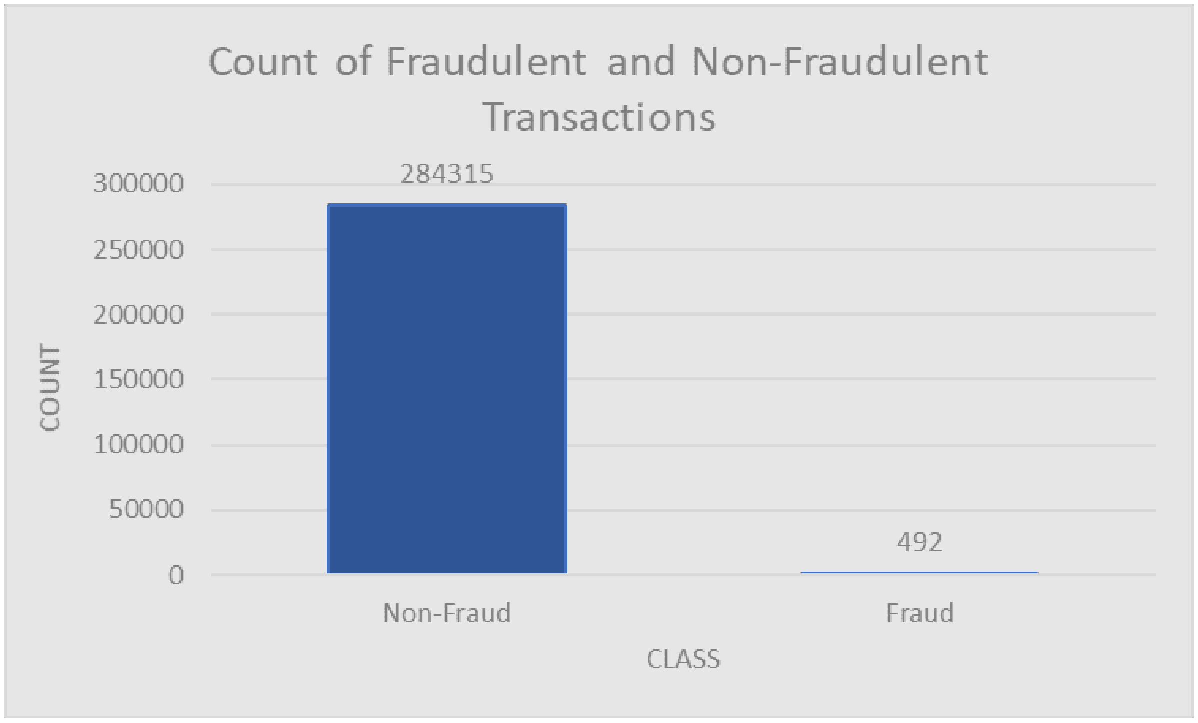


Figure 4: *Bar plot plot of the target variable showing imbalance in the dataset*

2.3. Correlation of features

There was no noticeable correlation between the features. This is because most of the features were transformed using Principal Component Analysis (PCA) before making the dataset public. The features V1 to V28 are the Principal Components results after propagating the real features through PCA. The features are not correlated with one another.

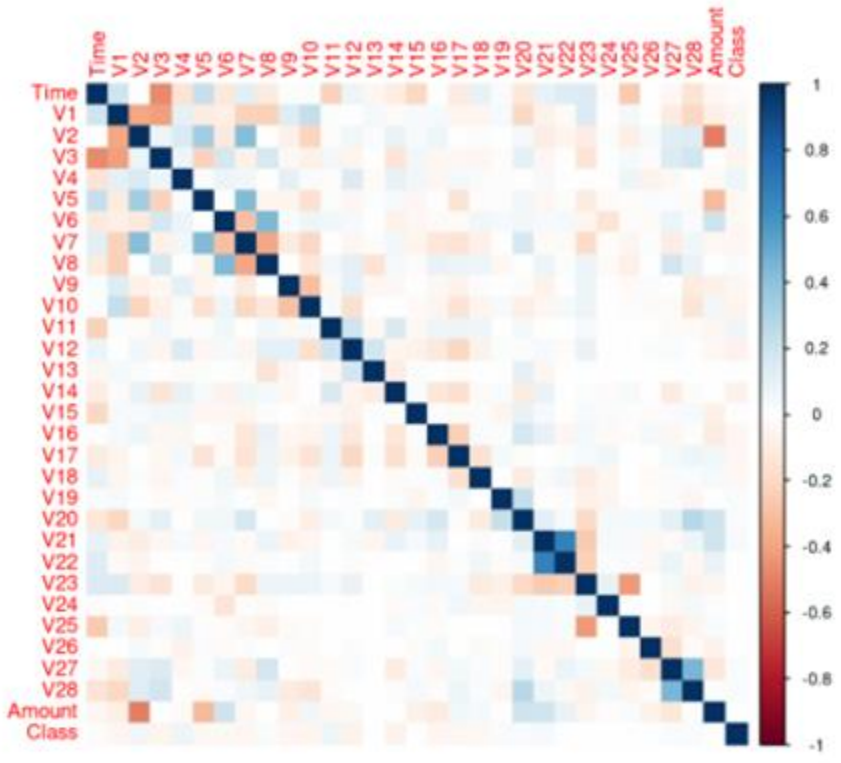


Figure 5: *Correlation plot between different features*

This is important because some of the models, e.g., logistic regression, do not perform well with highly correlated variables. This correlation plot was necessary to check for the presence of multicollinearity in the dataset.

2.4. Scatter plots of features

To understand the relationship between the input features and the target variable, the scatter plots between the target variable, Class and different predictor variables were plotted. As observed in Figure 6a and 6b below, fraudulent class is somewhat aligned towards the negative side of V14, V16 and V17 predictor variables, while non-fraudulent class is aligned towards the positive side of the axis. This information may provide useful insights and help the model to correctly distinguish between a fraud and non-fraud transaction.

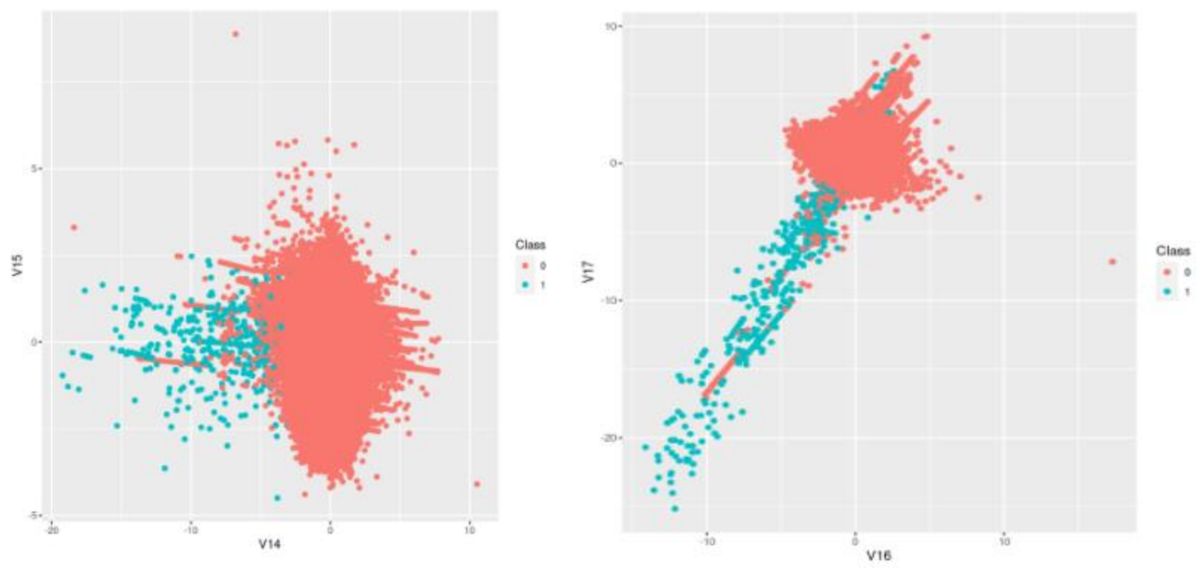


Figure 6a: *Scatter plot showing the spread of the target variable for V14, V15,V16,V17*

Additionally, in the scatter plots below, more fraudulent instances are observed on the negative axes of V12 and V18 than non-fraudulent instances.

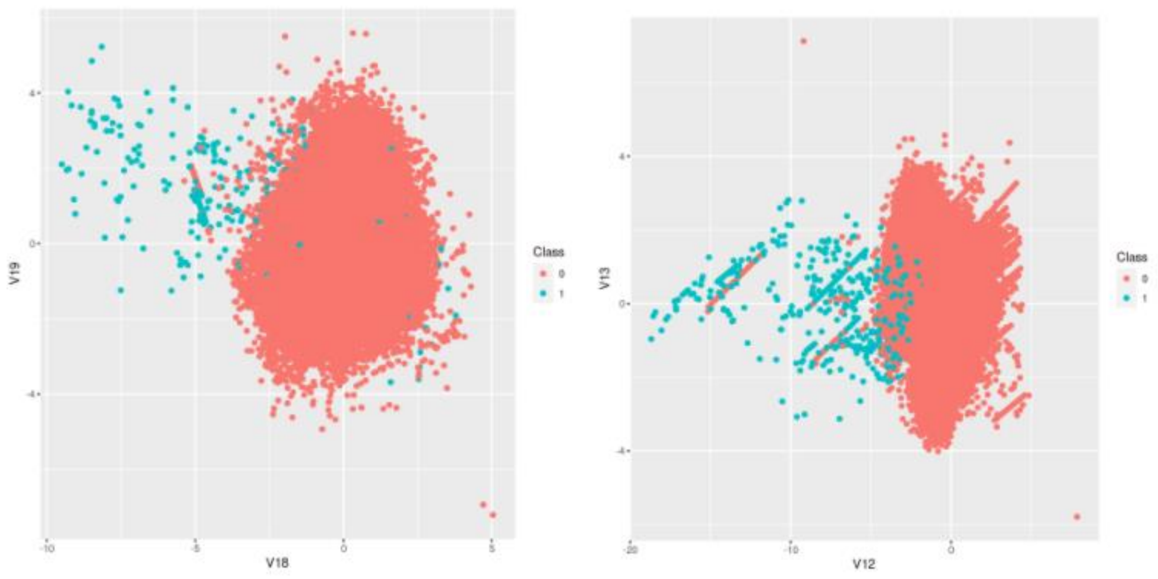


Figure 6b: *Scatter plot showing the spread of the target variable for V12, V13,V18,V19*

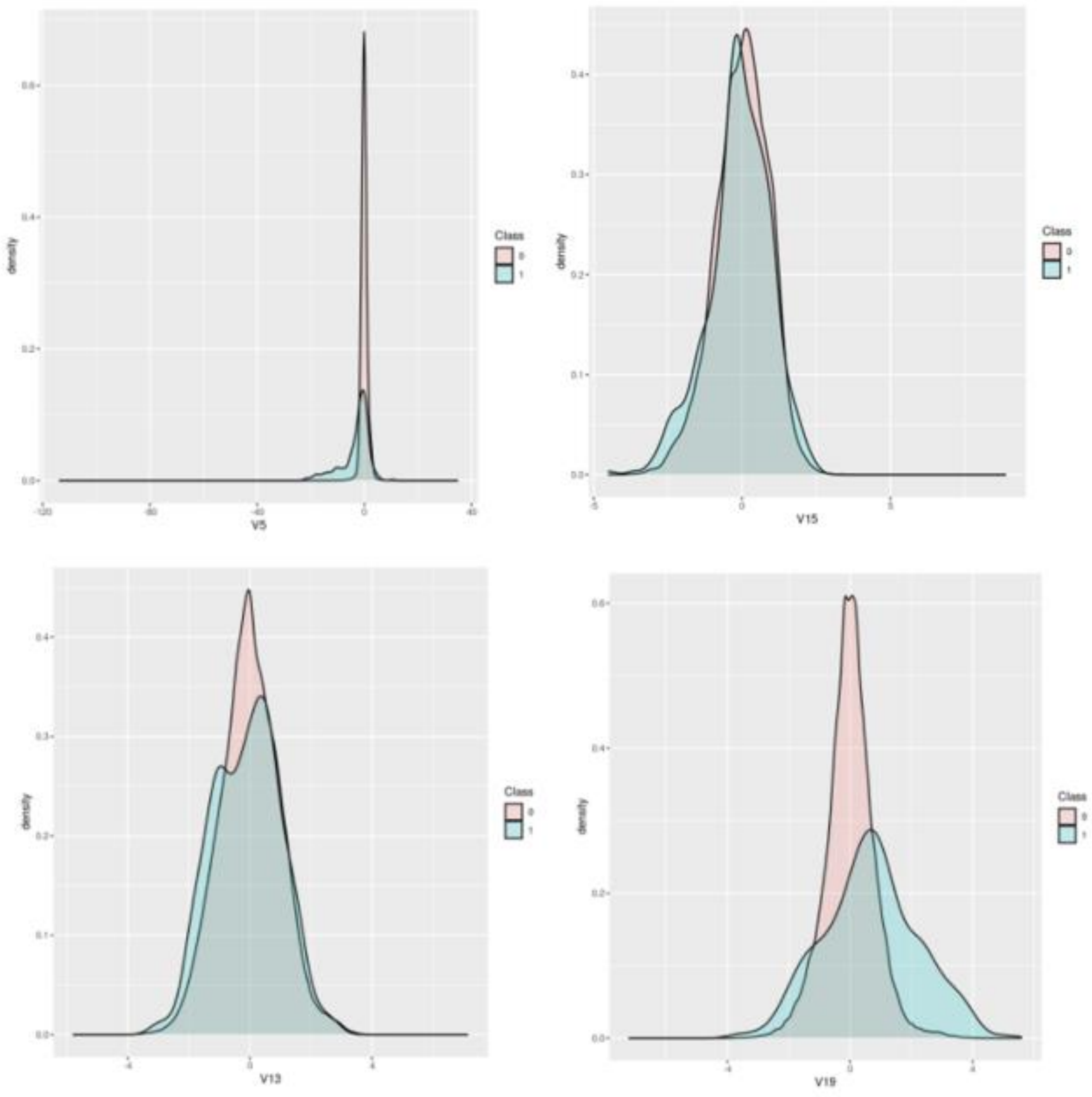


***Figure 7**: Density Distribution of the target for variables V5, V15, V13, V19*

Figure 7 shows that the fraudulent and non-fraudulent distributions are normally distributed, and the distributions are not entirely the same which speaks to the class imbalance in the dataset.

2.5. Amount Feature

The boxplot below shows that there are no outliers or extreme values among the fraudulent instances in the dataset, on the contrary a few extreme values can be observed for the non-fraudulent instances, but the effect is low since the number of instances considered to have extreme values is small.

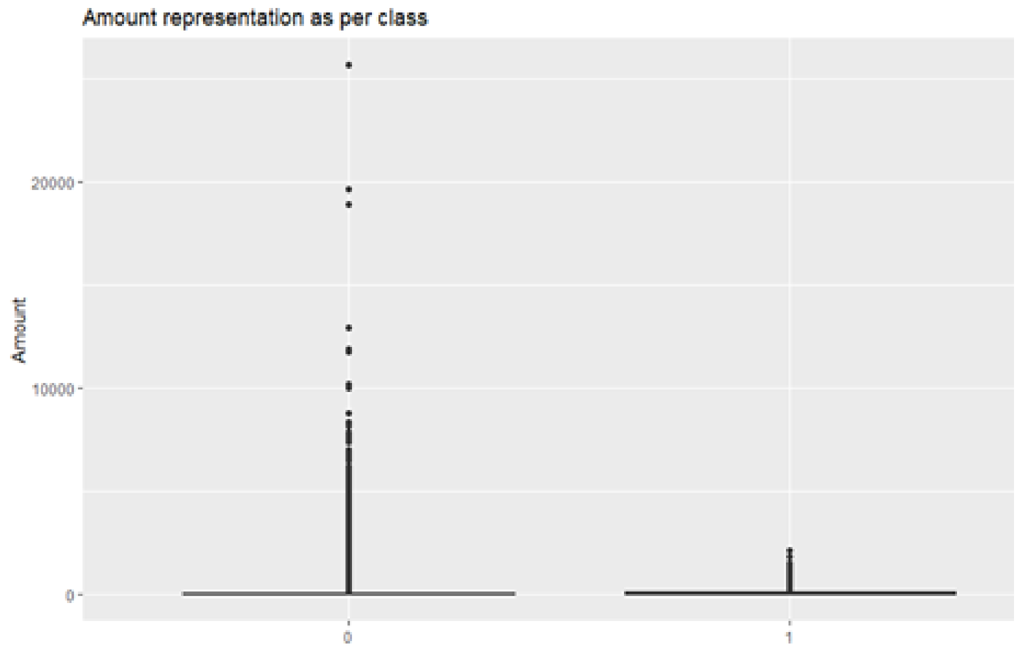


***Figure 8**: Boxplot for outliers in Amount variable*

2.6. Data Preprocessing and Feature Engineering

Below are some of the strategies adopted in preparing the data for the modelling phase to improve the performance of the model. Due to the high degree of imbalance in the number of instances for both categories of the dependent variable (i.e., 0 - Negative case and 1 - Positive case), undersampling and oversampling techniques were employed to bridge this gap. The expectation is that these data level preprocessing methods will improve the performance of our models during training. Also, the RStudio kept on aborting whenever an attempt was made to train the model on the entire dataset without undersampling.

2.6.1. Undersampling the training dataset

Undersampling technique was employed to address the imbalance in training dataset in a bid to improve the performance of the models. The undersampling was performed such that for every training example in the minority class, there are 20 observations in the majority class. This reduced the number of training examples in the majority class from 213,247 instances to 6,802 instances. After undersampling, the total number of training observations in the dataset was reduced to 7,160 (6802 - majority class - Non frauds and 358 - minority class - frauds). Of course, the number of fraudulent training examples remained unchanged after undersampling the non-fraudulent training examples.
It is important to note that the train/test split was performed before undersampling and we only undersampled the training dataset, the test dataset was not undersampled and is preserved for testing the performance of our models. This is good practice as the main objective of this task is to build a classifier that will generalise on unseen data. Ovun.sample() is the function in the ROSE library that performs undersampling, and N represents the number of training examples of the negative class required to remain after undersampling is executed. The code below implies that 20 training examples remain in the negative class (non-fraud) for every training example in the positive class (fraud). This significantly improved the training time across all models and improvement in performance was also recorded.

2.6.2. Oversampling of the training dataset

Oversampling is a technique used to address the problem with an imbalanced dataset. In this method, the oversampling of the minority class is required to match up the number of training examples in the majority class. The ‘smotefamily’ library is very useful for implementing Synthetic Minority Oversampling Technique (SMOTE), instead of replicating exact copies of the minority class during oversampling, SMOTE produces synthetic copies of the minority class thereby achieving a balance between the minority and majority classes and improving the performance of the models.

2.7. Feature Engineering

2.7.1. Time Feature

The Time variable was converted to represent the hour of the day in which the transaction took place. Considering that these transactions occurred within a 48-hour window, the new Hour feature was obtained by dividing the Time variable by the total number of seconds in a day and taking the modulus of 24. It can be seen from the plot below that more fraudulent transactions occurred between the hours of 12 midnight and 8am than non-fraudulent transactions as shown by the density plot below.

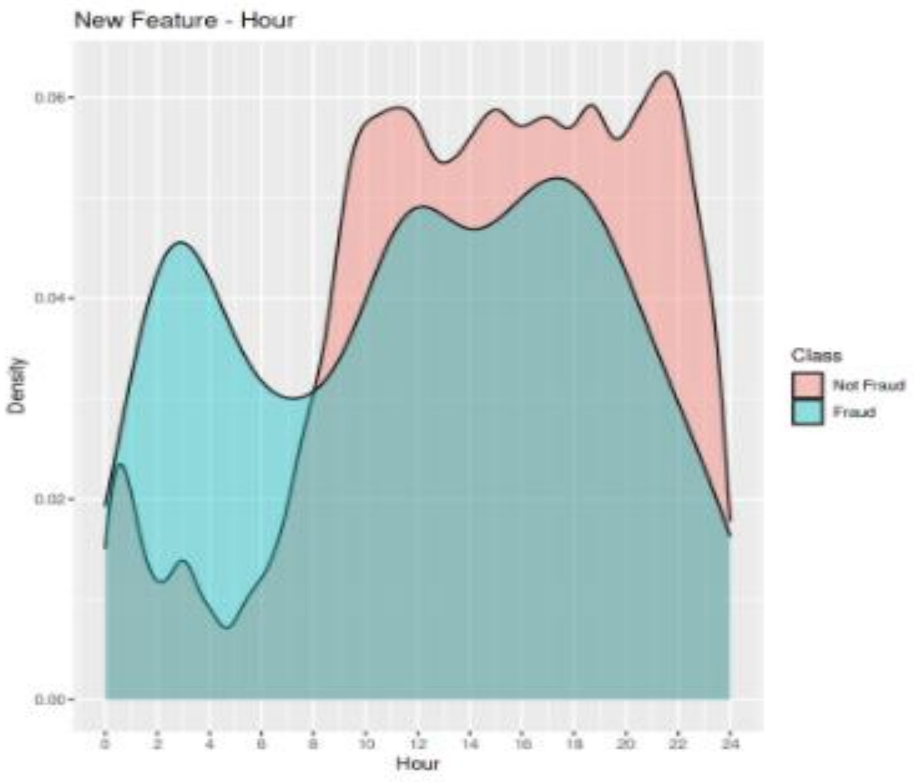


Figure 9*: Density plot showing the distribution of the target variable within a 24-hour period*

2.7.2. Normalized Amount Feature

The amount column in the dataset is not normalized and it is given in the original currency values of the transactions. In a bid to identify if the amount of money has any impact on the transaction being fraudulent or not, and to make all input variables uniform for the models, the Amount variable was normalized.

2.8. Modelling Methods

The goal is to design a machine learning model that will correctly distinguish between a fraudulent and non-fraudulent transaction on an unseen dataset. After concluding the data exploration analysis, it is evident that the data is distributed in clear clusters and in certain features between fraud and non-fraud. The results from the exploratory data analysis informed the choice of the 6 models for the reasons below:

S/N	Models	Reasons for Adoption
1	Logistic Regression	**Strengths** • Logistic regression is very robust with its intuitive assumptions, i.e., it makes no assumptions about scattering of classes in feature space. • Training time is fast, and this will be helpful for our dataset considering the high number of records. **Weaknesses** • It can only solve the problem linearly and does not perform well for non-linear problems. There may be some elements of non-linearity in our dataset, and this will not be captured by the linear model
2	Random Forest	**Strengths** • Random Forest uses the bagging approach- bootstrap aggregation, and because of this, the datasets are split into samples and different trees are built on those samples. This helps a lot in generalisation, which helps the model performance. • Feature selection is not required, trains model quickly. **Weaknesses** • This technique may not perform well for an imbalanced dataset.
3	Support Vector Regression (SVM)	**Strengths** • SVM is very good at generalising, and this means that there is a lesser chance of overfitting with this algorithm. **Weaknesses** • It consumes time for larger datasets • It is slow to train

4	K-Nearest Neighbours (KNN)	**Strength** • KNN is very simple and easily comprehensible algorithm that when users observe, they can find the different clustering characteristics of all the input variables **Weaknesses** • KNN becomes increasingly slower as the dataset becomes larger. For large datasets, the processing power becomes a bottleneck. • Does not perform well if the features are not scaled.
5	XGBoost	**Strengths** • Its high execution speed and fast interpretation abilities makes it a favourable choice. • No scaling or normalization required, which means it does not require so much feature engineering to perform well. **Weaknesses** • Identifying the optimal hyperparameters can prove difficult. Requires lots of tuning to perform well
6	Multi-Layer Perceptrons (Neutrons Network)	**Strengths** • MLP is considered because of the complexity of the dataset and the ability of the neural network to identify complex patterns in a dataset. **Weaknesses** • It is very slow to train • It requires a lot of hyperparameter tuning due to the high number of hyperparameters.

Table 2: *Strengths and weaknesses of ML techniques (Sadineni, 2020)*

2.9. Evaluation Metrics

The following metrics were used because they are reliable for measuring the performance of a model for an imbalanced classification task. Accuracy is not reliable for this dataset as an accuracy of over 99% was recorded without training a model.

- **Precision:** This captures the fraction of holdout/test data assigned to the positive class that belongs to the positive class. TP / (TP + FN)
- **Sensitivity (Recall):** This outlines the performance of the model on the positive class, and this is an important metric for the task as the aim is to design a classifier that will correctly identify fraudulent transactions (positive class).
- **F1 Score:** This is the combination of precision and recall with the aim of achieving a balance between both metrics.
- **ROC_AUC Score**: This measures the capability of the model to distinguish classes. This will help understand how the binary classifier model is performing in general. The ROC AUC curve shows visually the performance or strength of the model compared with the no-skill or untrained model or chance model.

The strategies employed for the evaluation is designed to understand the models' performance under the five constructs which are as follows:

- Evaluating the models after training with the undersampled dataset only.
- Evaluating the models after training with undersampled + oversampled dataset
- Evaluating the models after converting the Time feature to hour of the day + undersampling + oversampling the dataset.
- Evaluating the models after normalizing the Amount Feature + Hour of the day feature + undersampling + oversampling the dataset

A spreadsheet with the detailed result from this experiment is attached in the appendix.
The idea is to track the contributions of these preprocessing techniques to the overall performance of the models. Additionally, we have used the RoC curves as well to determine the AUC and consequently the best fitted model. RoC curve is a function of sensitivity and specificity; hence it is a good recommendation to use RoC for imbalance data. All the algorithms used for training the models are from CRAN. The model parameters used for training the algorithms are as follows:

Models	**Parameters**
Logistic Regression (glm function)	• **Family** - 'binomial("logit")' (2-class classification problem - Fraud and Not Fraud)

	• **maxit** - Number of iterations = 500 (Lower values did not make the model converge)
Random Forest (randomForest function From the randomForest library in CRAN)	• **ntree** = 100 - Number of trees employed (Testing values from 50 to 150 showed that the models did not perform well towards the far end of the spectrum) • **importance** - TRUE - 'Importance of input features is to be returned after training' (Used to create feature importance plot) • **keep.forest** = TRUE - 'Forest is to be kept for output'
XGBoost (Xgboost function from the xgboost library in CRAN)	• **label** = Class column - 'output variable in numeric' • **nrounds** = 200 - 'Number of iterations' • **verbose** = 0 - 'Stops from printing information of training steps' • **objective** = binary:logistic - 'Binary classification problem' • **eval_metric** = logloss - 'Evaluation metric used by the model during training'
K-Nearest Neighbours (knn function from the class library in CRAN)	• **cl** = 'Class' - 'output variable in numeric' • **k** = 116 - 'Number of nearest neighbours' Why 116? • **prob** = TRUE - 'Probability of predicted class asked to be returned'
Multi-Layer Perceptron (Mlp function from the RSNNS library in CRAN)	• **y** = 'Class' - 'output variable in numeric' • **size** = 25 - 'Number of hidden layers' • **maxit** = 2000 - 'Number of iterations to learn' • **learnFunc** = rprop - 'Resilient Backpropagation type of neural network learning'. The model updates the weights by calculating the derivatives of the lost function with respect to the model's parameters. • **linOut** = TRUE - 'Output value decided to be in classification'
Support Vector Machine (Svm function from the e1071 library in CRAN)	• **x** = 'All input features including time and normalised amount' • **y** = 'Class' - 'Output variable in numeric' • **probability** = TRUE - 'Probability predictions by the model are allowed'

Table 3: *Model's Parameters Table*

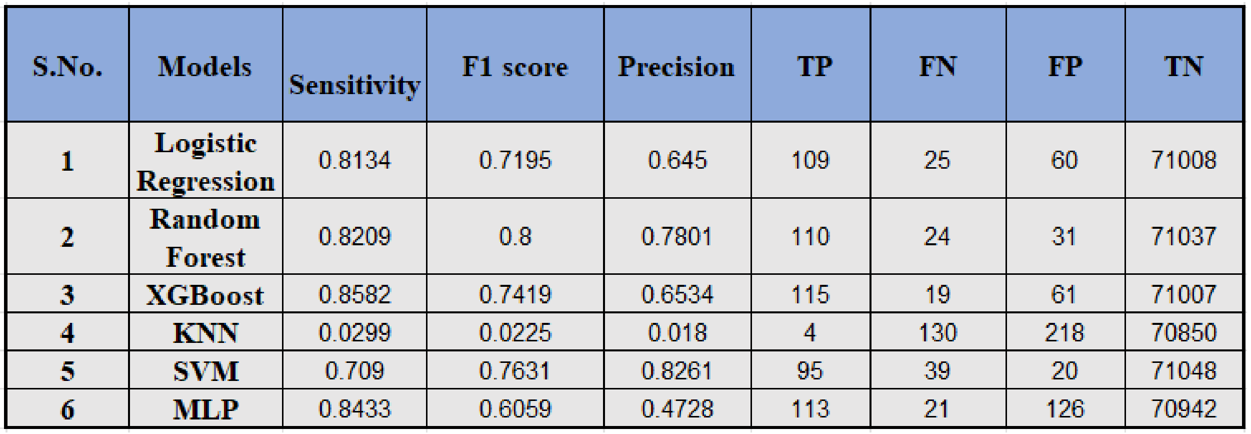
3. RESULTS

Our approach was to test the performance of the models after each preprocessing step. This will give us a sense of the contribution of each preprocessing and feature engineering technique to the overall performance of the model. Recall/Sensitivity and F1-score have been identified as very reliable metrics for this task. Recall/Sensitivity focuses on the number of correctly identified frauds (i.e., positive cases) while F1-Score is a combination of precision and recall.

3.1. Results after undersampling of the training dataset

See below the performance of all six models after undersampling only. Logistic Regression, Random Forest, XGBoost, MLP all performed relatively well but XGBoost can be said to be the best performing model considering the number of correctly classified frauds although it has twice the number of false positives compared with Random Forest. Obviously, KNN did not perform well due to the class imbalance in the dataset as can be seen below.
Class distribution of the target variable in the training examples after undersampling
Number of frauds (class 1) = 358
Number of non-frauds (class 0) = 6,802

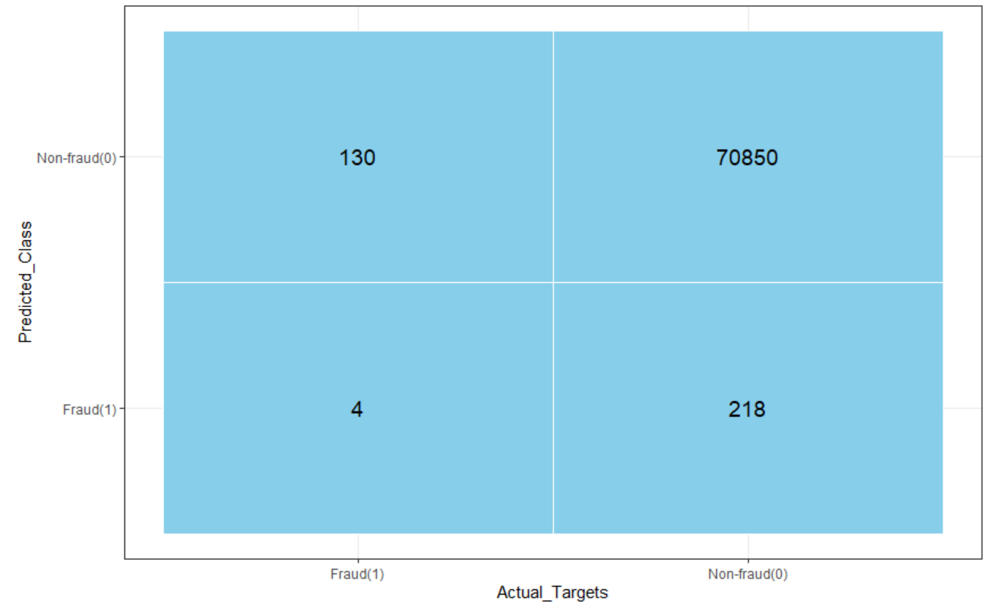
3.1.1. <u>Performance Matrix for all models after undersampling only</u>



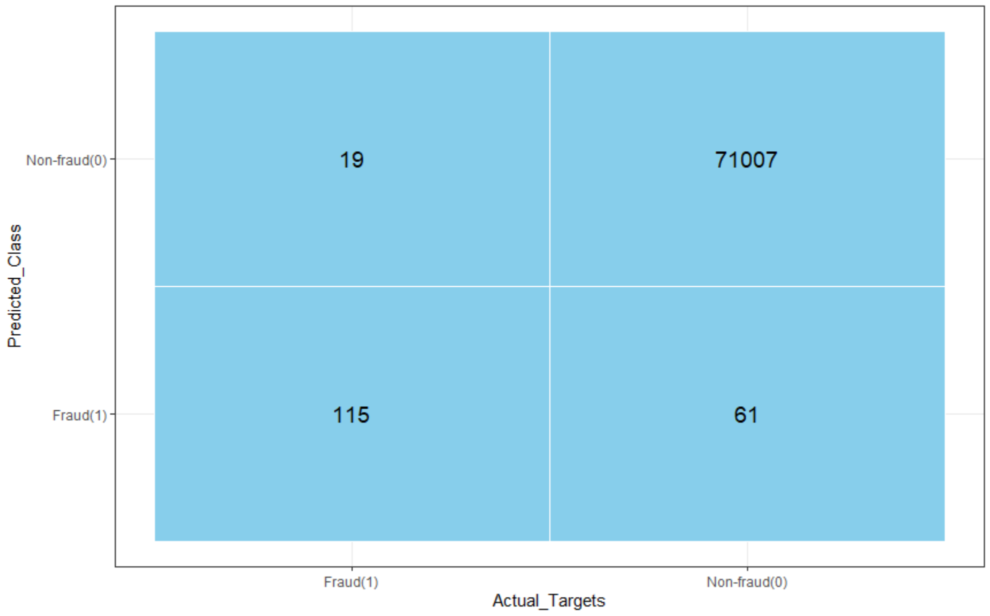
S.No.	Models	Sensitivity	F1 score	Precision	TP	FN	FP	TN
1	Logistic Regression	0.8134	0.7195	0.645	109	25	60	71008
2	Random Forest	0.8209	0.8	0.7801	110	24	31	71037
3	XGBoost	0.8582	0.7419	0.6534	115	19	61	71007
4	KNN	0.0299	0.0225	0.018	4	130	218	70850
5	SVM	0.709	0.7631	0.8261	95	39	20	71048
6	MLP	0.8433	0.6059	0.4728	113	21	126	70942

Table4: *Performance Matrix for Undersampled data*

3.1.2. Confusion Matrix for KNN and XGBoost models after undersampling



(i) Confusion matrix of worst performing model (KNN)



(ii) Confusion matrix of best performing model (XGBoost)

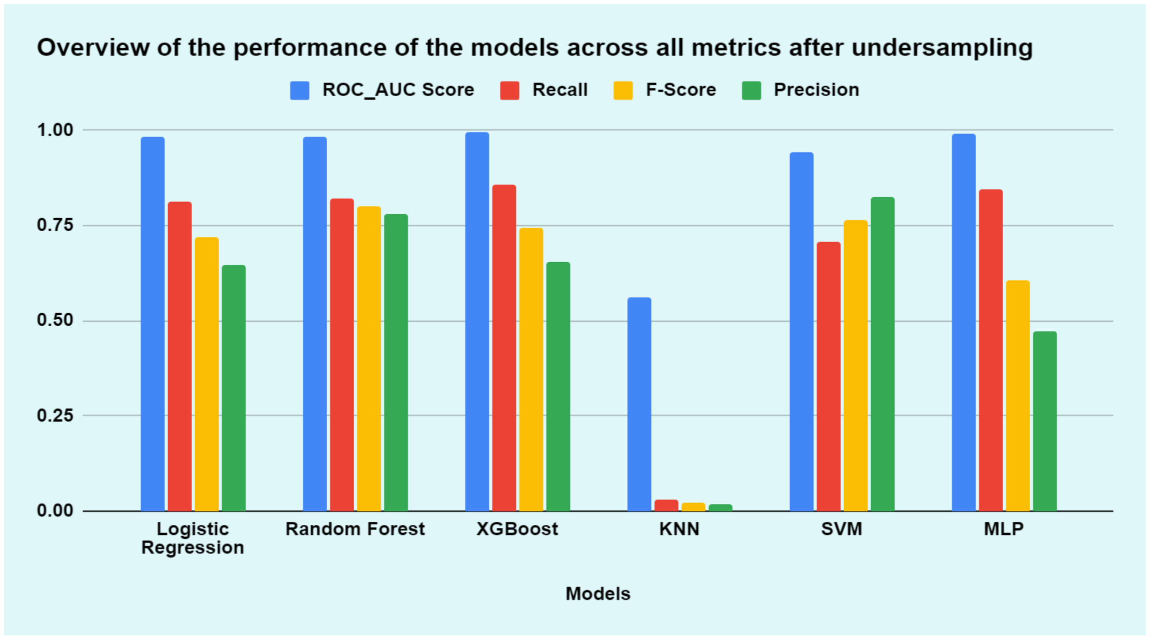


Figure 10: *Models Performance graph across all metrics*

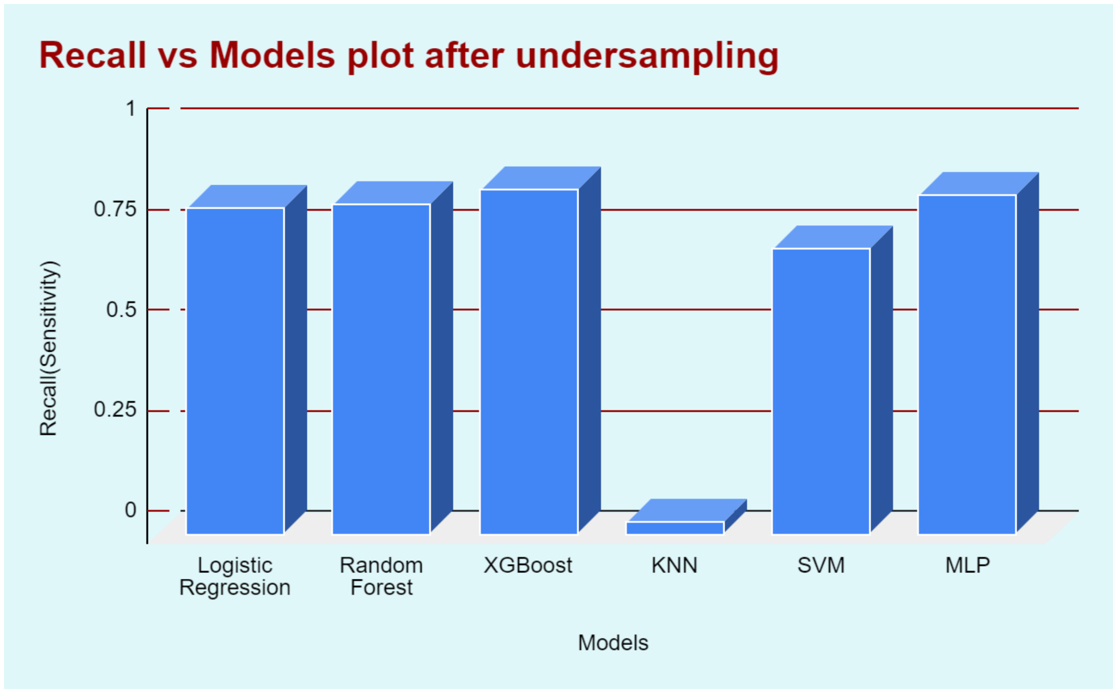


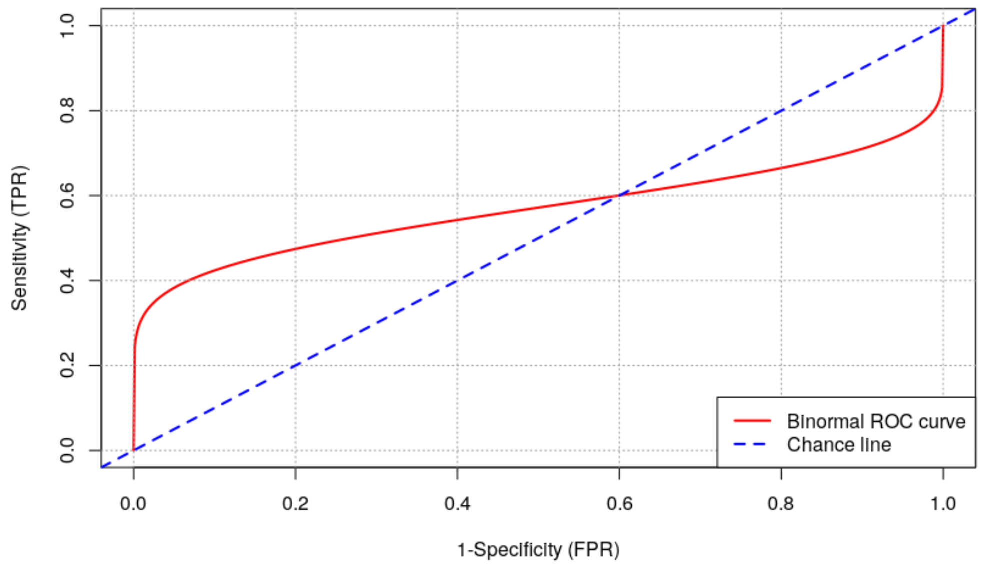
Figure 11: *Recall Plot for undersampled data*

3.1.3. Receiver Operating Characteristic Curve (ROC AUC Score)

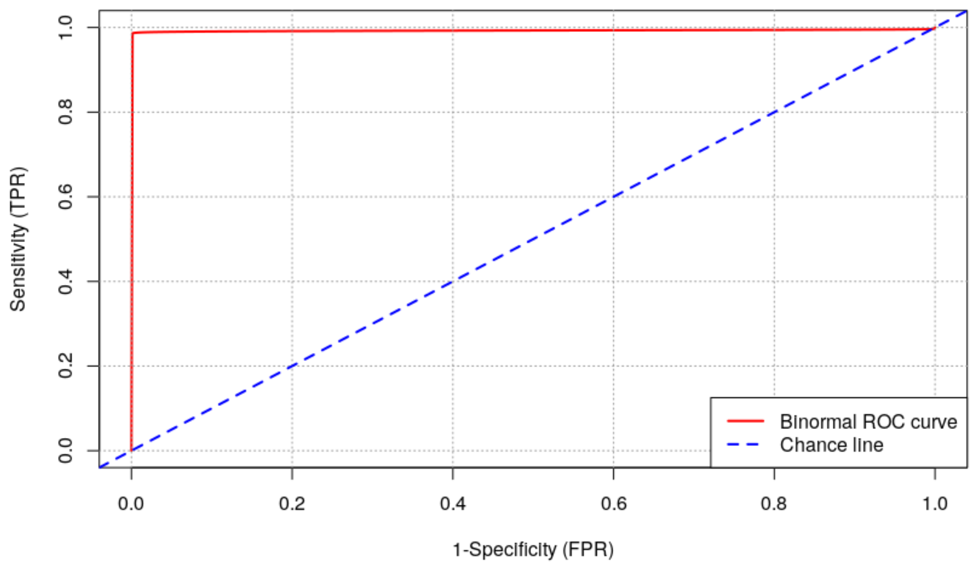
S.No.	Models	ROC_AUC Score	Specificity	Sensitivity
1	Logistic Regression	0.9812	0.9992	0.8134
2	Random Forest	0.9837	0.9996	0.8209
3	XGBoost	0.9928	0.9991	0.8582
4	KNN	0.5595	0.9969	0.0299
5	SVM	0.9405	0.9997	0.709
6	MLP	0.9893	0.9982	0.8433

Table 5: *ROC_AUC score for undersampled data*

The ROC plot below for KNN shows that at some point, the KNN model performed worse than a no skill or untrained model.



(i) ROC Curve of worst performing Model (KNN)



(ii)ROC Curve of best performing Model (XGBoost)

3.2. Results after undersampling and oversampling

After oversampling the dataset, all the models were more precise in identifying the fraudulent transactions, although the number of false positives increased slightly when compared with the undersampling results.

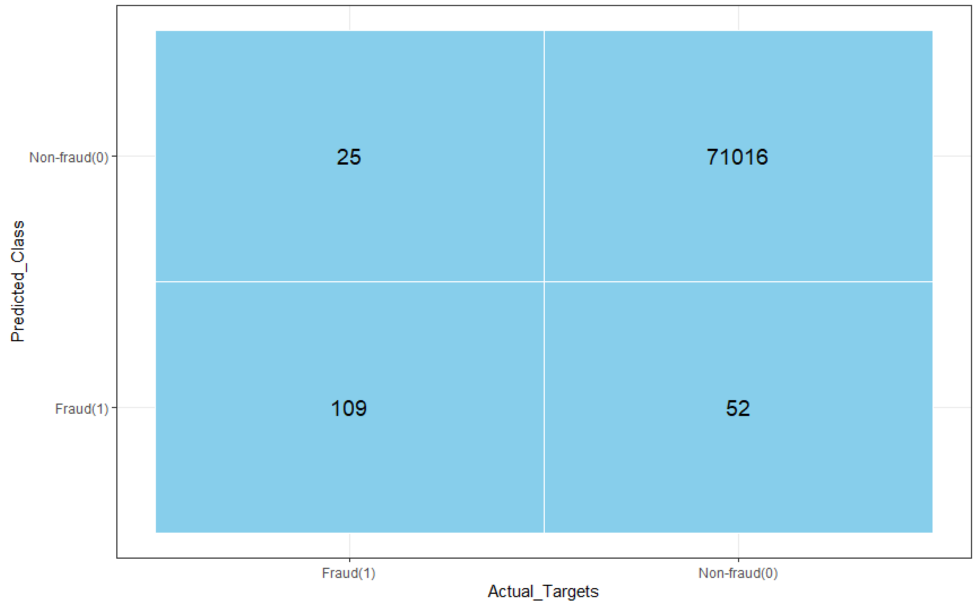
The recall score of the KNN model increased significantly from 0.0299 to 0.8209 after oversampling, this is an indication that this model performs best with a balanced dataset and is unable to extract patterns in an imbalance dataset. All other models performed slightly better in terms of the sensitivity or recall score after oversampling, but KNN was most noticeable.

3.2.1. Performance Matrix after applying undersampling and oversampling to the training dataset

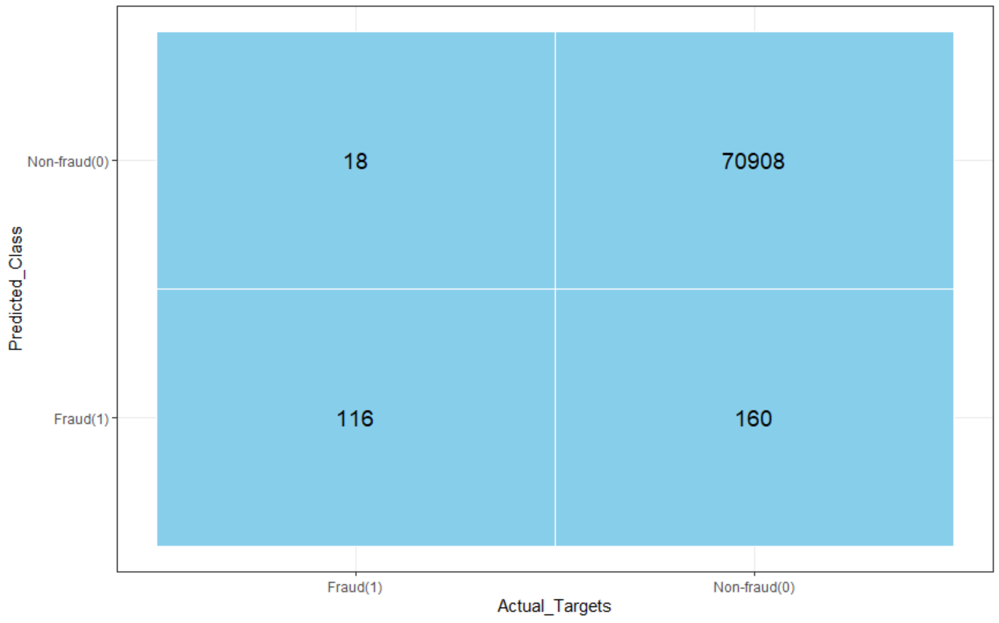
S.No.	Models	Sensitivity	F1 score	Precision	TP	FN	FP	TN
1	Logistic Regression	0.8507	0.4176	0.2767	114	20	298	70770
2	Random Forest	0.8358	0.7542	0.6871	112	22	51	71017
3	XGBoost	0.8657	0.5659	0.4203	116	18	160	70908
4	KNN	0.8209	0.4681	0.3274	110	24	226	70842
5	SVM	0.8134	0.739	0.677	109	25	52	71016
6	MLP	0.8209	0.4681	0.3274	110	24	226	70842

Table 6: Performance Matrix for Undersampled+Oversampled training data

3.2.2. Confusion Matrix for SVM and XGBoost after performing undersampling and oversampling



(i) Confusion matrix for SVM



(ii) Confusion matrix of best performing model (XGBoost)

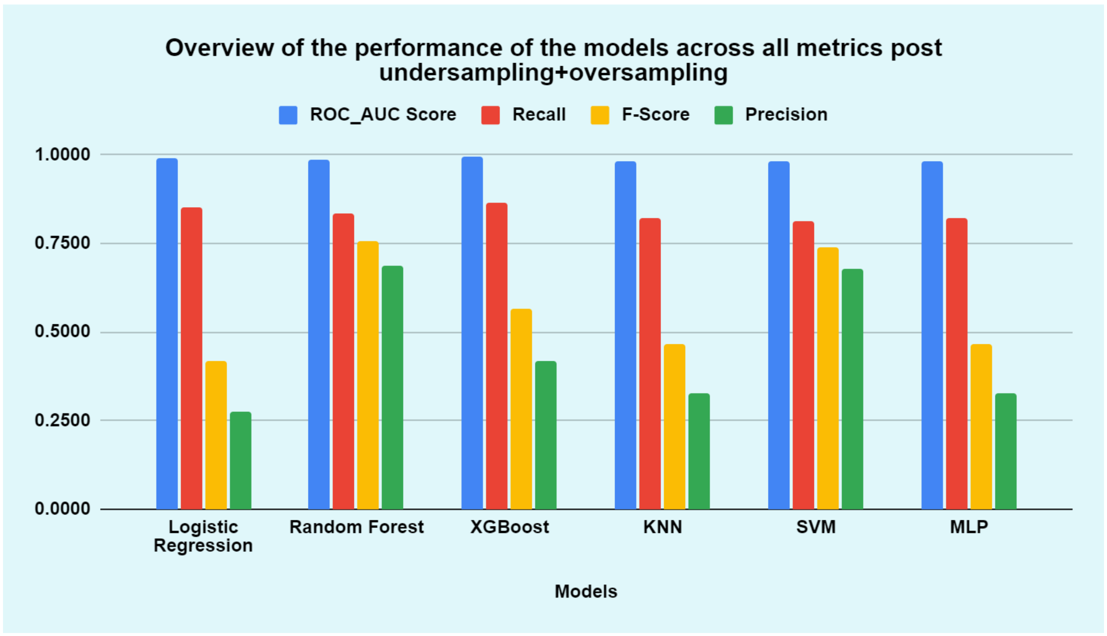


Figure 12: *Models Performance graph across all metrics (after undersampling+oversampling training data)*

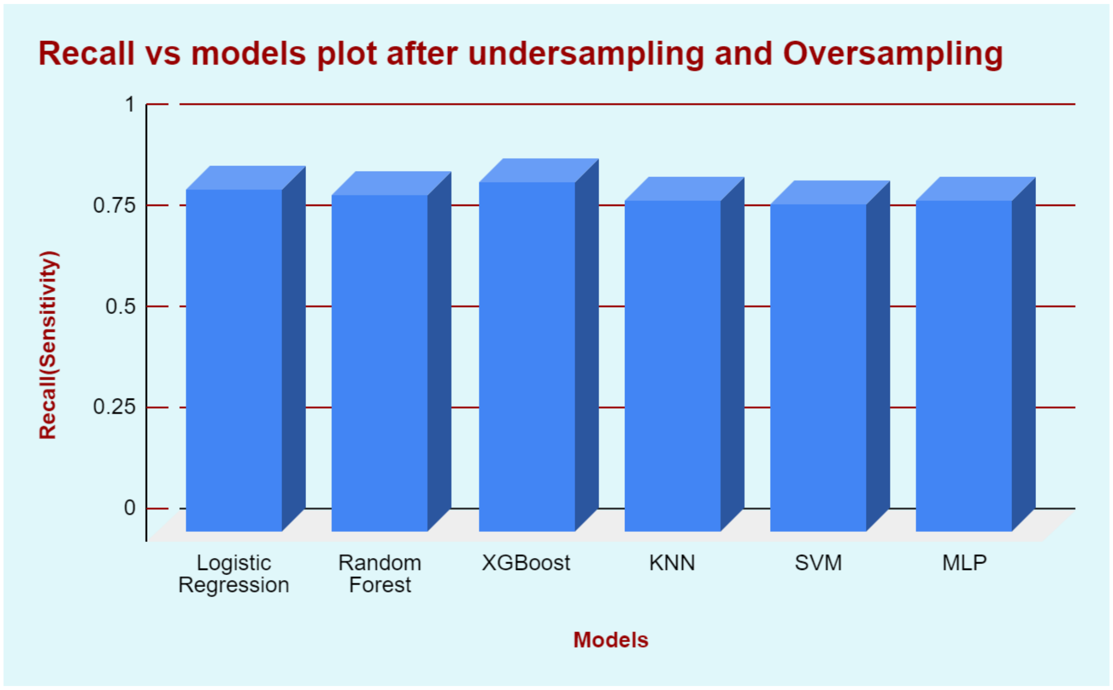
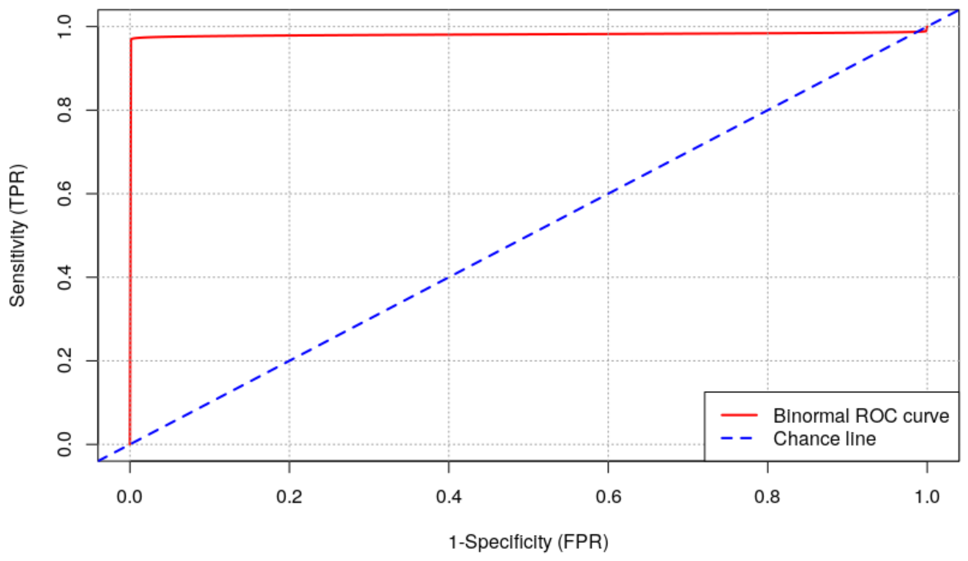


Figure 13: *Recall Plot after undersampling +oversampling training data*

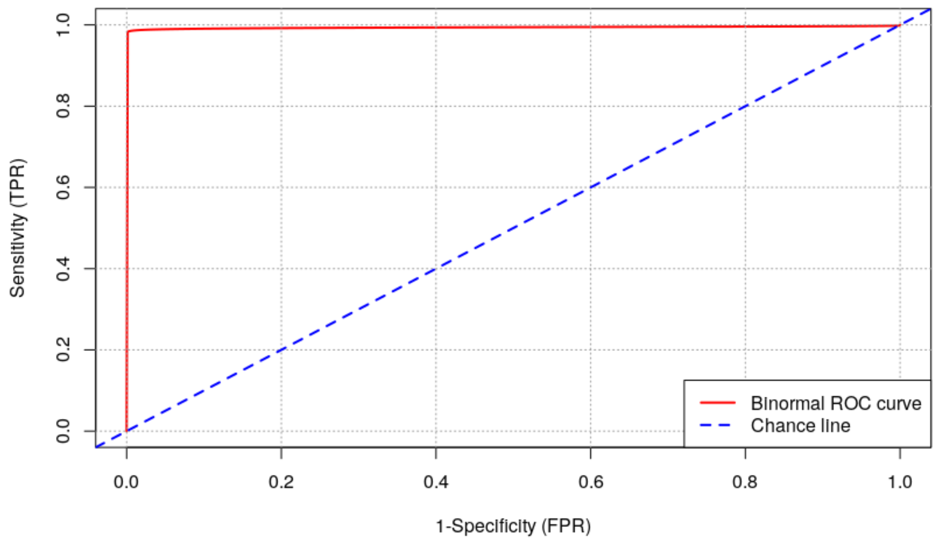
3.2.3. Receiver Operating Characteristic Curve A (ROC Curve)

S.No.	Models	ROC_AUC Score	Specificity	Sensitivity
1	Logistic Regression	0.9903	0.9958	0.8507
2	Random Forest	0.9877	0.9993	0.8358
3	XGBoost	0.9939	0.9977	0.8657
4	KNN	0.9826	0.9968	0.8209
5	SVM	0.9813	0.9993	0.8134
6	MLP	0.9826	0.9968	0.8209

Table 7: *ROC_AUC score metrics for the six classifiers*



(i) ROC Curve for SVM



(ii)ROC Curve of best performing Model (XG-Boost) after undersampling and oversampling the training data

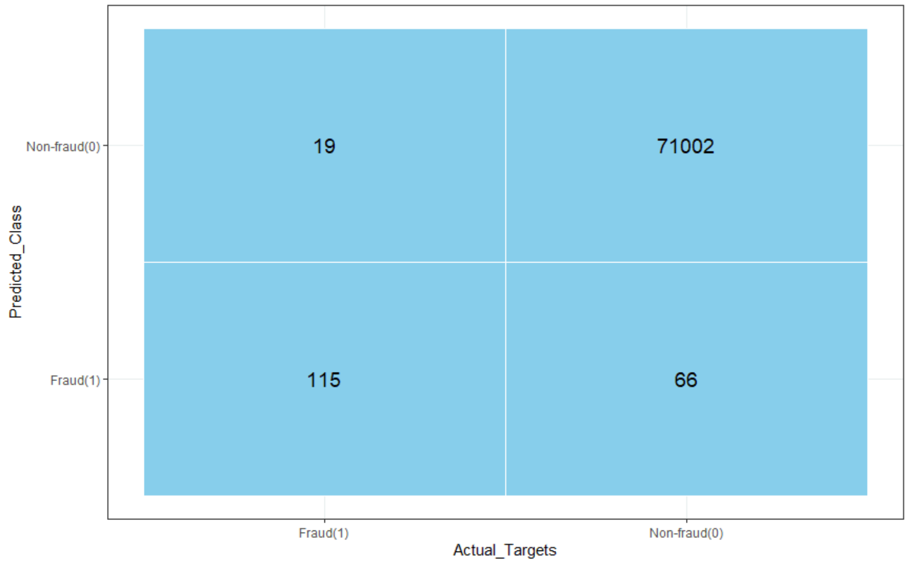
3.3. Results after Preprocessing and Feature Engineering (Time & Amount Variable Normalized)

It was observed that after performing feature engineering, which included converting the Time feature to hour of the day and normalizing the Amount variable, the models' performance improved. The models were more precise at correctly identifying the positive class (frauds).

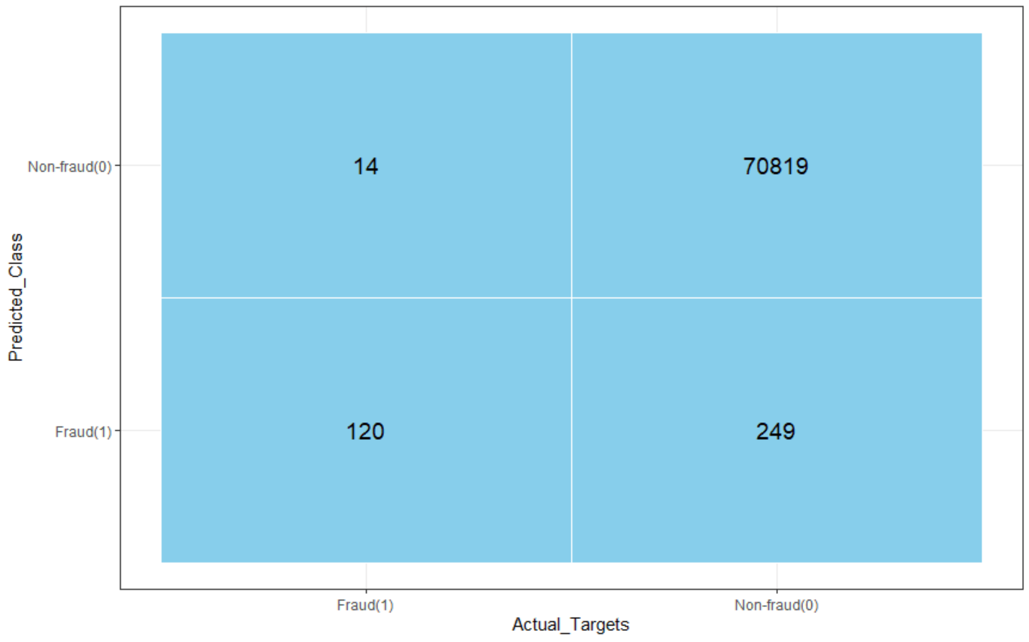
The Sensitivity of XGBoost, Logistic regression and KNN models slightly improved after feature engineering, but the number of False Positive cases also increased.

Also comparing the performance of XGBoost with that of Random Forest, Random Forest is very good in detecting the Negative and Positive classes in our dataset. KNN still is the worst performing model even though it has identified a good number of True positives, it classified about 3.55% non-fraudulent transactions as fraudulent transactions. The high number of False Positive cases may not be good for the business and hence we would not be using this model to implement a solution on this dataset for the business.

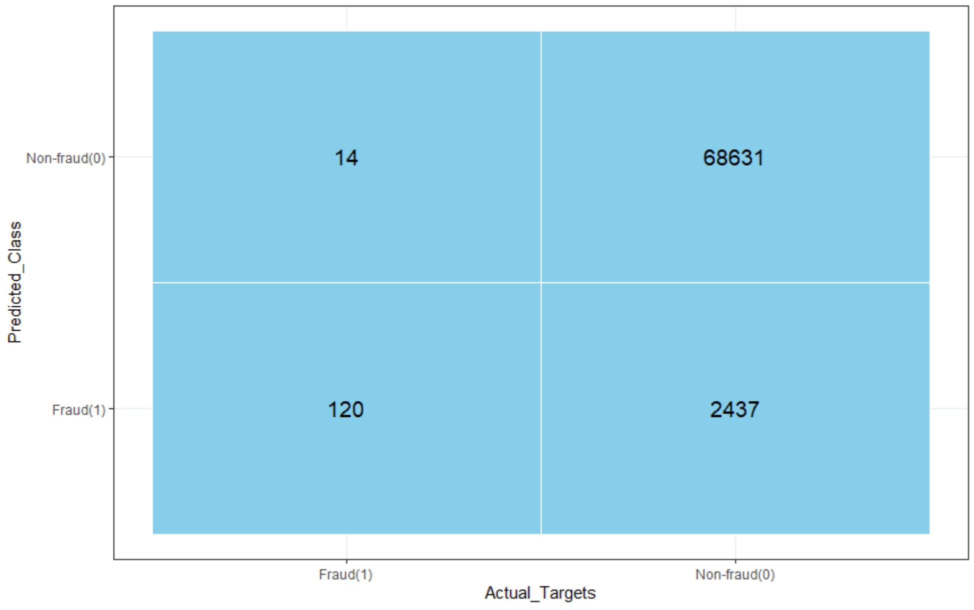
3.3.1. Confusion Matrix for Random Forest, XGBoost and KNN after undersampling + oversampling + feature engineering



(i) Best performing model for Random Forest



(ii) Best performing model for XGBoost



(iii) Worst performing model for KNN

3.3.2. Performance Matrix after undersampling, oversampling and feature engineering

S.No.	Models	Sensitivity	F1 score	Precision	TP	FN	FP	TN
1	Logistic Regression	0.8955	0.1882	0.1052	120	14	1021	70047
2	Random Forest	0.8582	0.7302	0.6354	115	19	66	71002
3	XGBoost	0.8955	0.4771	0.3252	120	14	249	70819
4	KNN	0.8955	0.0892	0.0469	120	14	2437	68631
5	SVM	0.8433	0.6366	0.5113	113	21	108	70960
6	MLP	0.8507	0.3826	0.2468	114	20	348	70720

Table 8 : *Performance matrix for undersampled+oversampled+feature_engineered dataset*

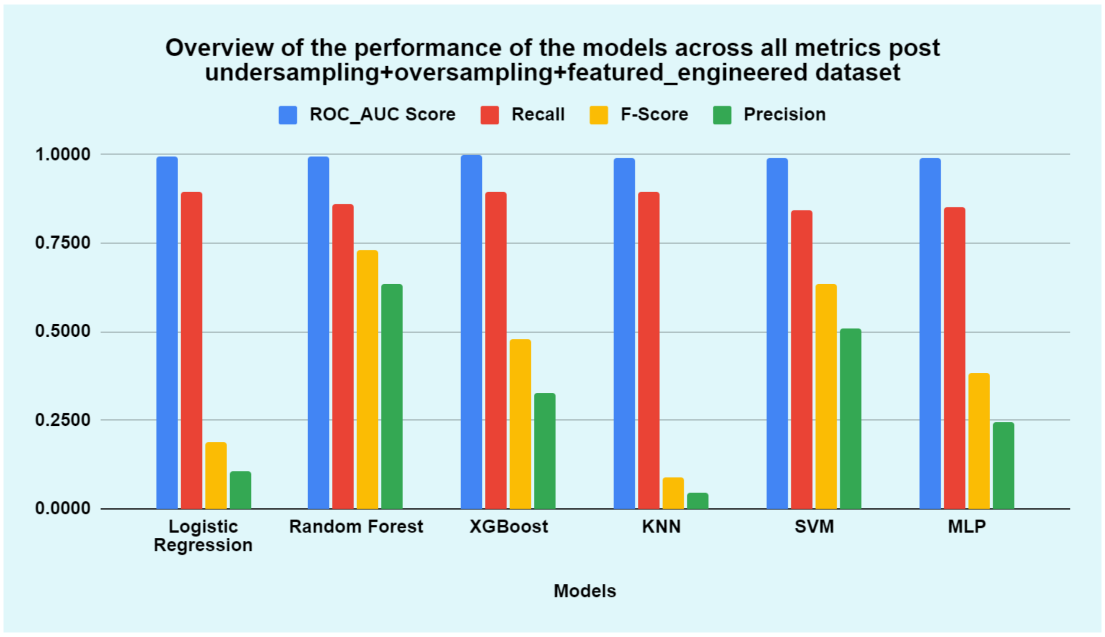


Figure 14: *Models Performance graph across all metrics (undersampled+oversampled+feature_engineered dataset)*

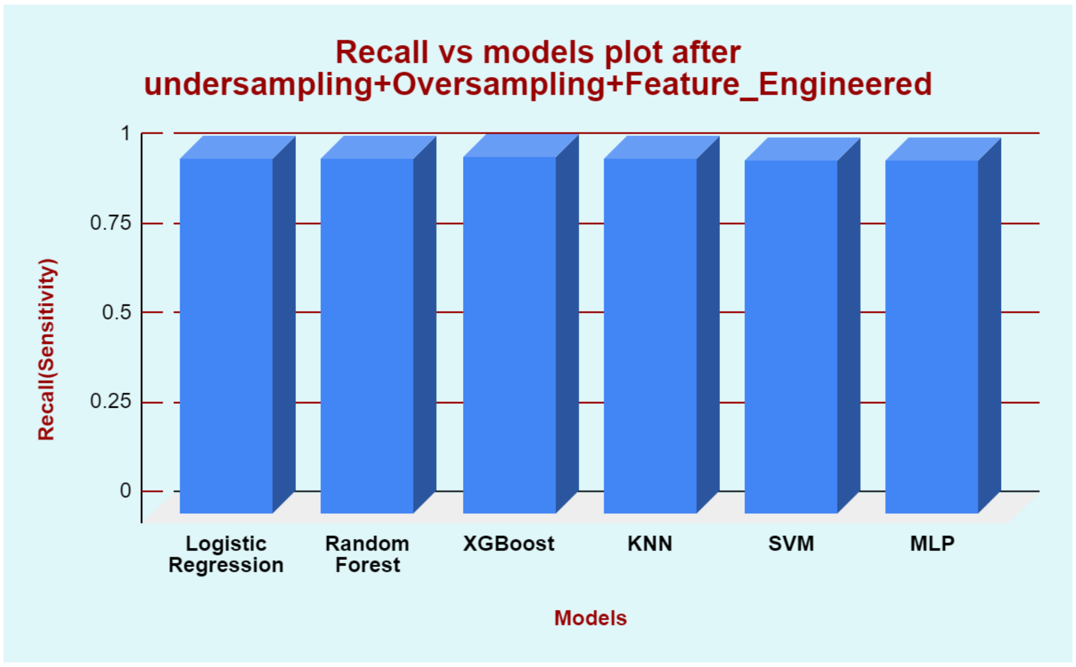
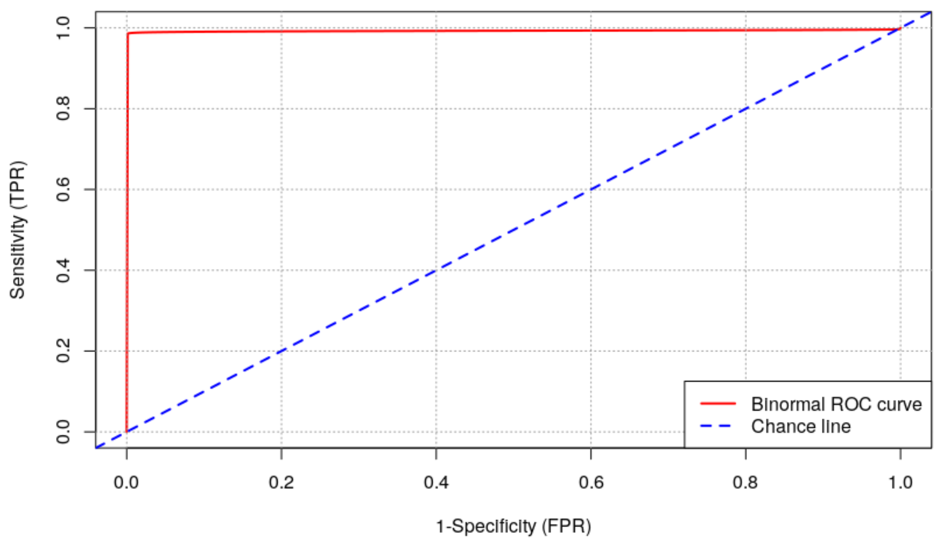


Figure 15: *Recall Plot for undersampled+oversampled+feature_engineered data*

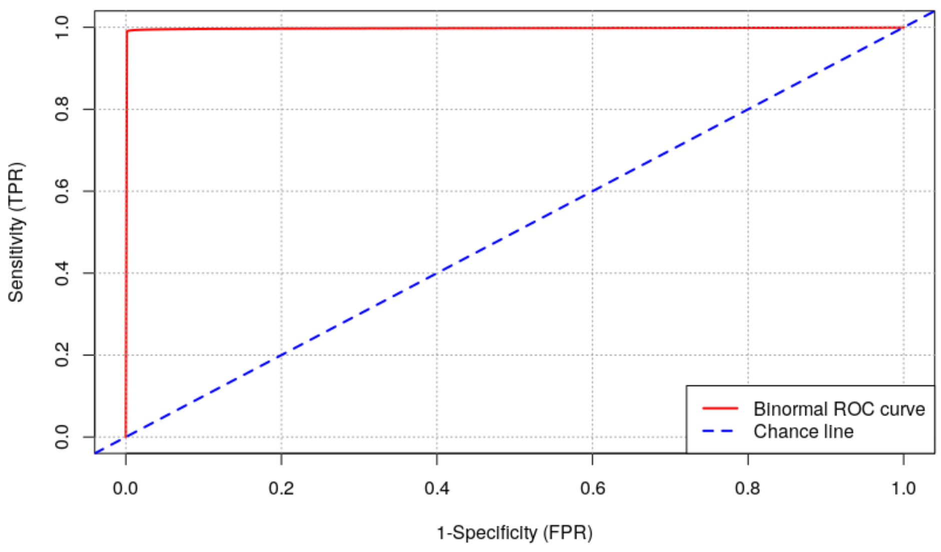
3.3.3. Receiver Operating Characteristic Curve A (ROC Curve)

S.No.	Models	ROC_AUC Score	Specificity	Sensitivity
1	Logistic Regression	0.9964	0.9856	0.8955
2	Random Forest	0.9928	0.9991	0.8582
3	XGBoost	0.9979	0.9965	0.8955
4	KNN	0.9922	0.9657	0.8955
5	SVM	0.9893	0.9985	0.8433
6	MLP	0.9901	0.9951	0.8507

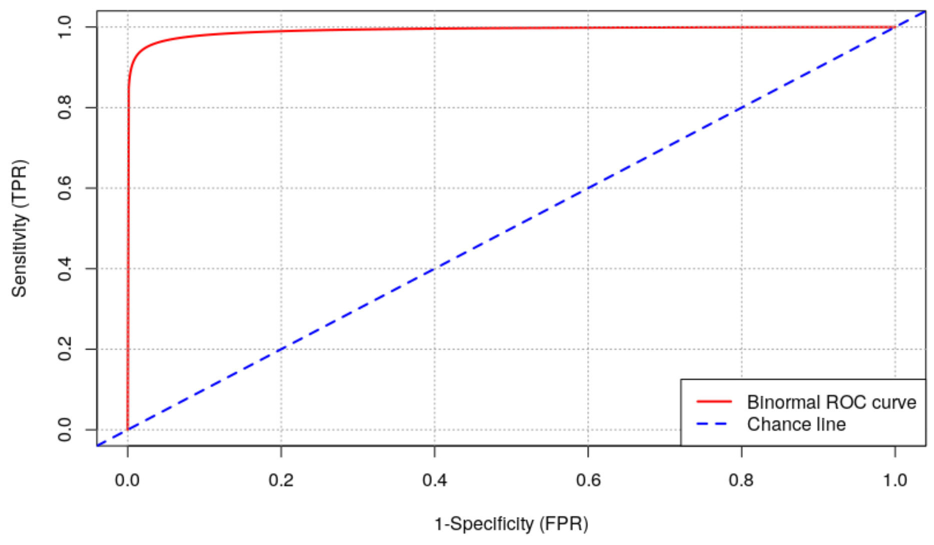
Table 9: *Showing the ROC_AUC score metrics for the six classifiers*



(i) ROC curve for Random Forest



(iii) XG-Boost ROC curve



(iv) KNN ROC Curves

4. DISCUSSION

4.1. Variable Importance for Random Forest & XGBoost

4.1.1. Variable Importance for Random Forest of the final model

The feature importance plot below shows the relative contribution of each feature or predictor to the overall performance of the Random Forest model after preprocessing and feature engineering. That is, how well the feature improved the purity of the node. V14 performed best, followed by V4 and the contribution per feature keeps decreasing until the Hour variable, which appears to have the least contribution to the Random Forest model. It will be worth going back to the business and request for the source of the encoded V-features, especially the best performing features e.g., V14, V4, V17, V12, V10. Knowing the real-world meaning or representation of these features will be a helpful guide for further feature engineering.

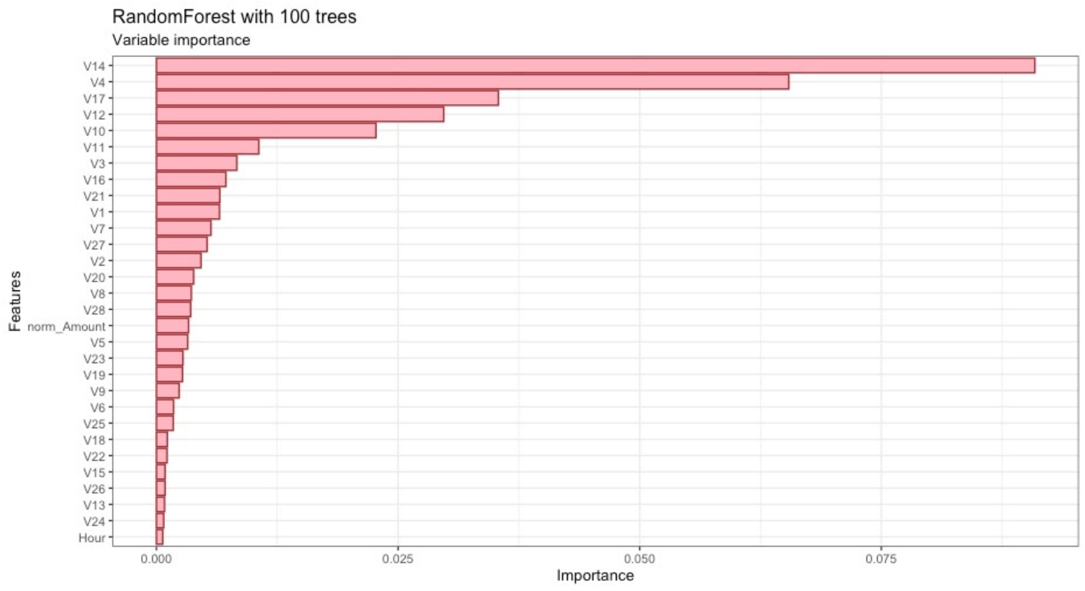


Figure 16: *Feature Importance plot for the final Random Forest model*

4.1.2. Variable Importance for XGBoost of the final model

The most important features for XGBoost are similar to that of Random Forest and this may not be unconnected to the fact that these two algorithms are tree based models. It is important to note that feature V14 was used to make key decisions in both Random Forest and XGBoost models.

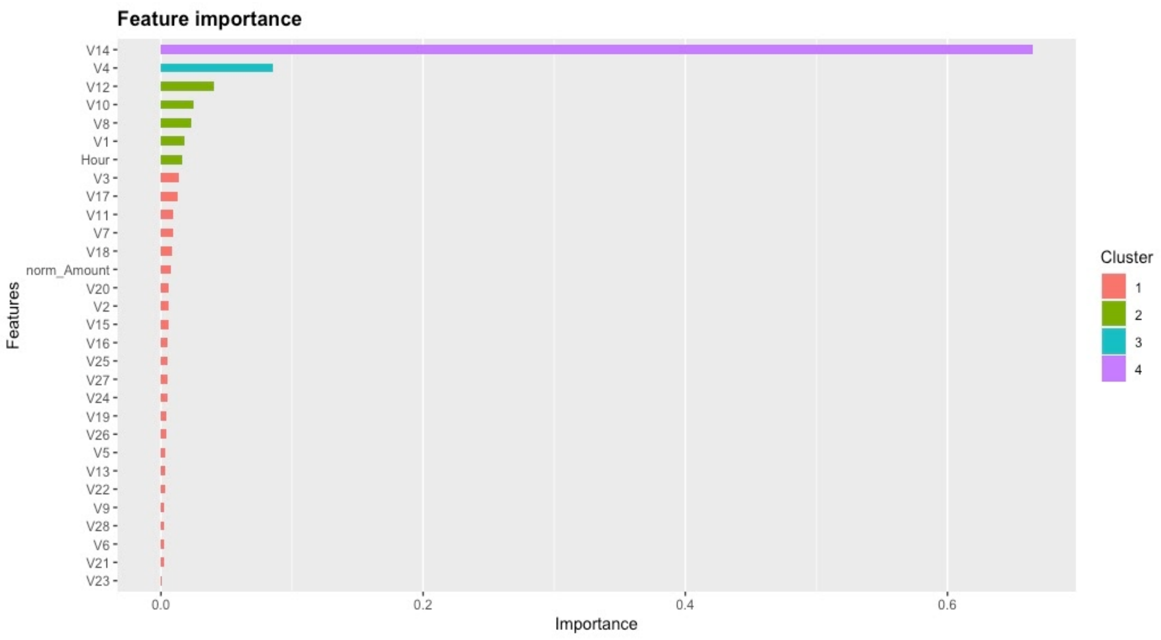


Figure 17: *Feature Importance plot for the final XGBoost model*

The sensitivity score for KNN model significantly improved after oversampling. Only 4 fraudulent transactions were detected out of 134 frauds after undersampling, however, the number of correctly classified frauds jumped to 120 after applying oversampling and feature engineering (i.e. normalizing the Amount feature and converting the Time variable to Hour of the day). In addition, the number of False Negatives increased after applying the preprocessing methodologies, keeping KNN still as the worst performing model. Below bar graph illustrates performance of KNN with different preprocessing methodologies:

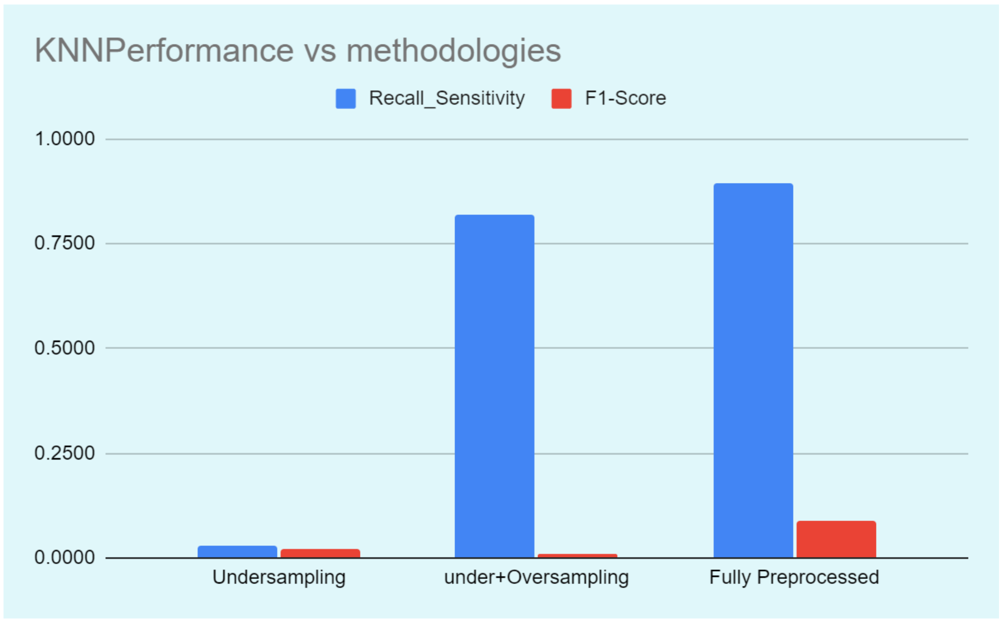


Figure 18: *KNN Performance over different methodologies*

XGBoost performed well with this dataset because of its robustness and overall high performance for tabular data. It is also the winning algorithm for most tabular Kaggle competitions.
Model sensitivity score slightly increased but F1-Score has also decreased exponentially, which resulted in overall decrease in XGBoost performance. Below graph shows the sensitivity and F1-score plot for XGBoost over the different methodologies.

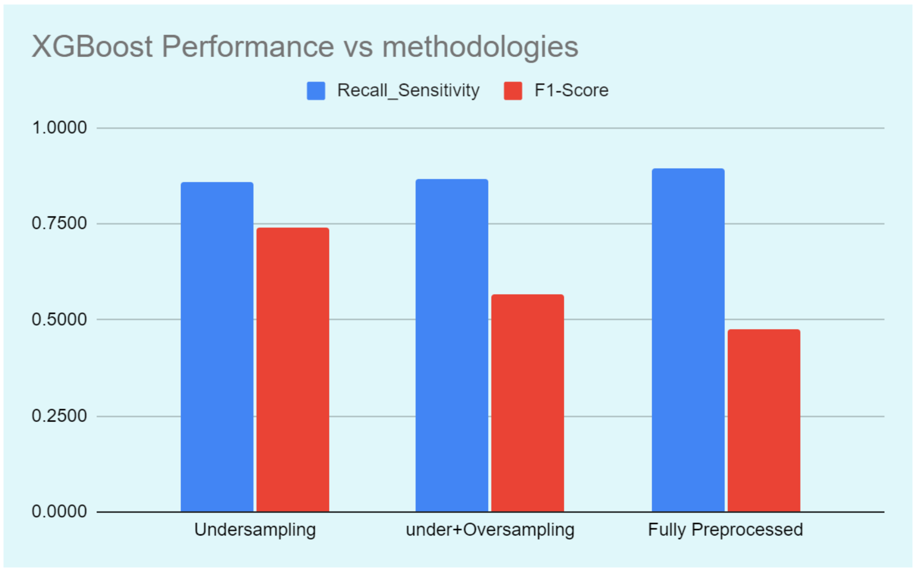


Figure 19: *XG-Boost Performance over different methodologies*

The F1-Score for Random Forest did not decrease compared to XGBoost, and sensitivity score increased as more preprocessing was done.

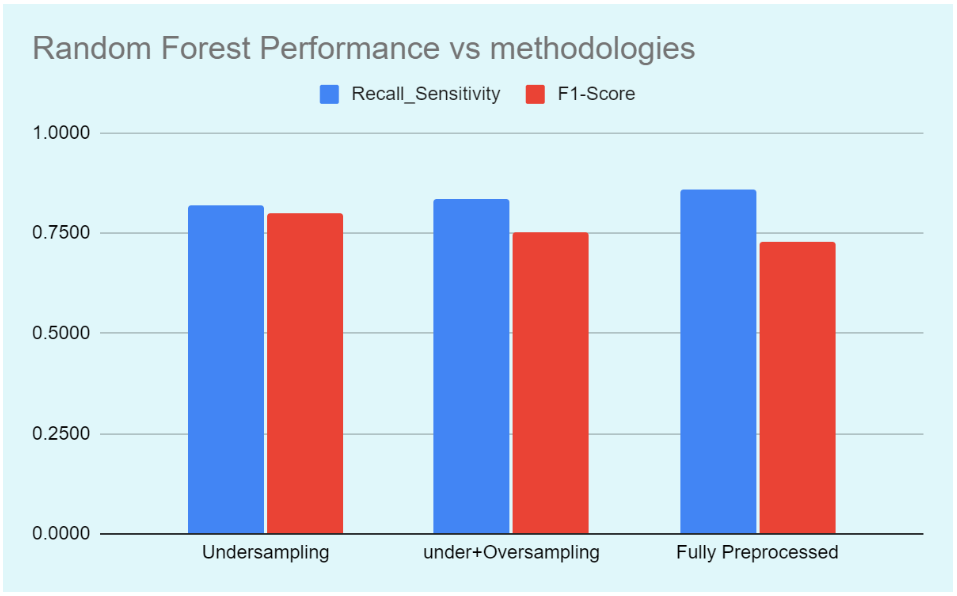


***Figure 20**: Random-Forest performance over different methodologies*

There were improvements in the performance of Logistic Regression, SVM and MLP models after preprocessing but the number of false positives generated by the models makes it unattractive to be considered as one of the best performing models. The intention is to design a solution that will yield optimal benefits to the business and as a result reducing the number of false positives is also an integral part of the solution. Therefore, we will not be recommending Logistic Regression, SVM, MLP and KNN models to the business.

5. CONCLUSION

After implementing undersampling, oversampling, normalizing the Amount feature and transforming the Time variable, it was observed that Random Forest performed best across all 4 metrics by correctly identifying 115 frauds out of a total of 134 frauds with only 66 false positives in the test dataset with 71,202 transactions followed by the XGBoost model that correctly identified 120 frauds with 249 false positives. Again, this is a decision that will be made by the business. If the business is willing to accept the fact that about 249 out of 71,202 customers will call the company and complain that their cards were blocked, then XGBoost is preferred, else Random Forest is the best option.

From the dataset, about 249 frauds occur per day amounting to about €30,063 lost each day to fraud. Let's assume the business buys our solution and uses our Random Forest model for identifying fraudulent transactions, then it will successfully identify and prevent 213 frauds from occurring and save the company from losing €28,500 per day. Assuming the same number of transactions occur every day for a year, then our solution will prevent frauds of **€9,417,122** from taking place. In addition to the savings, customers will be happier as the number of reported frauds will reduce in the company. Hence, it is a win-win solution.

The following improvements will be implemented if given more time on this project:

1. Further optimization of the models by performing a wider grid search to obtain hyperparameters that will improve the performance of the models on test data.
2. Training a deeper neural network using the dataset.
3. Collecting more data and training the models in a bid to improve the overall performance.

6. BIBLIOGRAPHY

Adewumi, A.O. and Akinyelu, A.A. (2016). A survey of machine-learning and nature-inspired based credit card fraud detection techniques. *International Journal of System Assurance Engineering and Management*, 8(S2), pp.937–953.

Ali Shukur, H. and Kurnaz, S. (2019). *Credit Card Fraud Detection using Machine Learning Methodology*. [online] http://www.ijcsmc.com. Available at: http://www.ijcsmc.com [Accessed 27 Nov. 2021].

Anderson, R. (2007). *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. Oxford University Press.

Bhattacharyya, S., Jha, S., Tharakunnel, K. and Westland, J.C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), pp.602–613.

Bodepudi, H. (2021). Credit Card Fraud Detection Using Unsupervised Machine Learning Algorithms. *International Journal of Computer Trends and Technology*, 69(8), pp.1–3.

Brownlee, J. (2016). *Crash Course On Multi-Layer Perceptron Neural Networks*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/neural-networks-crash-course/.

Chan, P.K., Fan, W., Prodromidis, A.L. and Stolfo, S.J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, 14(6), pp.67–74.

Delamaire, L., Abdou, H. and Pointon, J. (2009). Credit card fraud and detection techniques: a review. *Banks and Bank systems*, 4(2), pp.57–68.

Dornadula, V.N. and Geetha, S. (2019). Credit Card Fraud Detection using Machine Learning Algorithms. *Procedia Computer Science*, 165(ISSN:1877-0509), pp.631–641.

Gunn, S.R. (1998). Support vector machines for classification and regression. *ISIS technical report*, 14(1), pp.5–16.

KDNuggets (2021). *Machine Learning – it's all about assumptions*. [online] KDnuggets. Available at: https://www.kdnuggets.com/2021/02/machine-learning-assumptions.html [Accessed 27 Nov. 2021].

Kotsiantis, Sotiris B (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4), pp.261–283.

Machine Learning Group - ULB (2013). *Credit Card Fraud Detection*. [online] Kaggle.com. Available at: https://www.kaggle.com/mlg-ulb/creditcardfraud.

Modi, H., Lakhani, S., Patel, N. and Patel, V. (2013). Fraud detection in credit card system using web mining. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2), pp.175–179.

Prof. Nick Ryman-Tubb (2020). *NPREPROCESSING_prettyDataset*<- *function(dataset,...){ params* <- *list(...)*. University of Surrey: Prof. Nick Ryman-Tubb.

Rokach, L. and Maimon, Oded Z (2007). *Data mining with decision trees: theory and applications*. World scientific.

Statista (2021). • *Statista - The Statistics Portal for Market Data, Market Research and Market Studies*. [online] Statista.com. Available at: https://www.statista.com.

UKFinance (2021). *Card spending | UK Finance*. [online] www.ukfinance.org.uk. Available at: https://www.ukfinance.org.uk/data-and-research/data/cards/card-spending.

Zareapoor, M., Seeja.K.R, Seeja.K.R. and Afshar Alam, M. (2012). Analysis on Credit Card Fraud Detection Techniques: Based on Certain Design Criteria. *International Journal of Computer Applications*, 52(3), pp.35–42.

7. APPENDIX

- ROC Curves for Fully preprocessed dataset for all models tested here
- Confusion matrix for fully preprocessed dataset for all models tested here
- Performance matrix for all model for all pre-processing steps here