# BT2103 Optimisation Methods in Business Analytics

Liew Wei Brian (A0239041M), Leng Jin De Joel(A0234179Y), Seah Ding Xuan (A0240134X)

## 1. Brief Introduction Of Data Set And Data Modeling Problem

The dataset "card.csv" has been obtained from UCI repository: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

It contains payment information of 30,000 credit card holders obtained from a bank in Taiwan.

### Dataset Description

ID: Unique dataset identification number for consumer

**Feature Attributes**

LIMIT_BAL: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit (Continuous)

SEX: Gender (1 = male; 2 = female) (Categorical)

EDUCATION: Education (0 = unknown, 1 = graduate school; 2 = university; 3 = high school; 4 = others, 5 = unknown, 6 = unknown) (Categorical)

MARRIAGE: Marital status (0 = unknown, 1 = married; 2 = single; 3 = others) (Categorical)

AGE: Age (in years) (Continuous)

PAY_0 - PAY_6: History of past payment. Past monthly payment records were tracked (from April 2005 to September 2005, where PAY_0 = the repayment status in September 2005 and PAY_6 = the repayment status in April 2005). The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above. (Categorical)

BILL_AMT1 - BILL_AMT6: Amount of bill statement (NT dollar) (from April 2005 to September 2005, where BILL_AMT1 = amount of bill statement in September 2005 and BILL_AMT6 = amount of bill statement in April 2005). (Continuous)

PAY_AMT1 - PAY_AMT6: Amount of previous payment (NT dollar), where PAY_AMT1 = amount paid in September 2005 and PAY_AMT6 = amount paid in April 2005. (Continuous)

**Target Variable**

default_payment_next_month: target values of whether payment is defaulted (Yes = 1, No = 0) (Categorical)

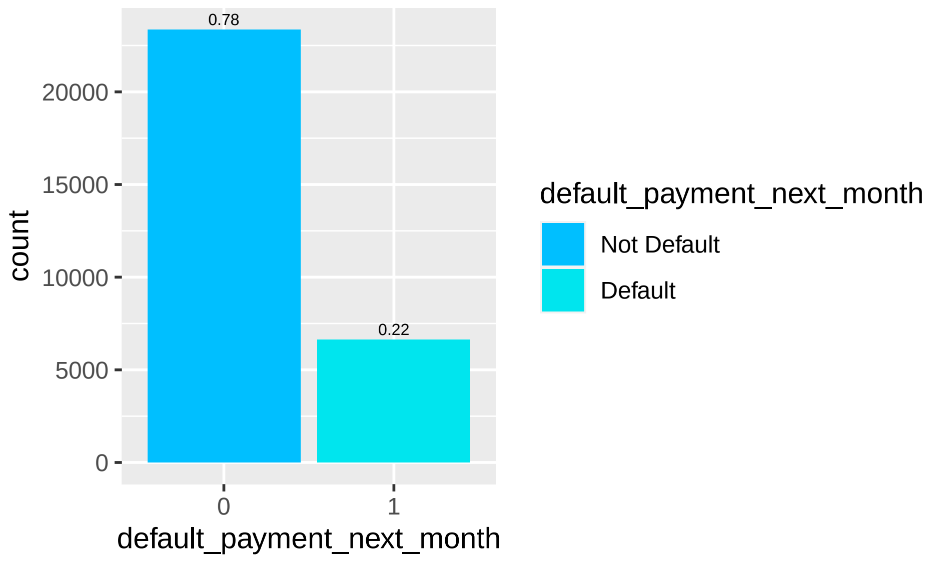
In this report, we aim to evaluate supervised machine learning algorithms in their ability to predict whether a customer will default on their credit card payment. From the perspective of risk management for the bank, predicting whether or not a customer is likely to default payment is important for banks as it poses a financial risk to them.

We will be using the following classification algorithms:

1) Logistic Regression
2) Support Vector Machines (SVM)
3) Naive Bayes Classification
4) Decision Tree (Conditional Inference Tree)
5) Random Forest
6) Neural Networks

## 2. Exploratory Data Analysis

Before we perform initial investigations on data to discover patterns and spot anomalies/inconsistencies in our data, we will first take a look at our target variable, default_payment_next_month.



As we can see, we have a binary classification problem as shown in the figure, with values set to "0" for consumers who did not default on payment and "1" for consumers who defaulted on payment. Out of 30,000 samples, 6636 or 22.1% of clients has defaulted on payment.

### Data Cleaning

a. **Handling missing data**

```
## [1] "Number Of Missing Values In Dataset: 0"
```

There are no missing values in the dataset.

b. **Categorical Feature Analysis**

Firstly, column for the repayment status in September 2005 is wrongly labelled as PAY_0. To be consistent with "BILL_AMT1" and PAY_AMT1", the "PAY_0" column will be re-labelled as "PAY_1" instead.

Secondly, we also found unknown values for certain categorical features.

These errors in the data set can be addressed in two ways:

1. Removing the rows which contain the error

2. Replace the wrong attribute class with a correction

MARRIAGE

According to the UCI repository, MARRIAGE: Marital status (1 = married; 2 = single; 3 = others). There are data points of MARRIAGE that are labelled as "0", which do not correspond to any of the above-mentioned categories.

For the MARRIAGE feature attribute, we noticed that 54 observations has a value of "0", which is not in any of the categories as defined in the variable.

We decided to remove this observations because we cannot assume that these entries belong to any of the other 3 defined categories. Also, since there are 30000 customer data points in the data set, and thus these data points will not affect the performance and evaluation of our model severely, with the change seen in the table below.

```
##   MARRIAGE     n
## 1        0    54
## 2        1 13659
## 3        2 15964
## 4        3   323
```

```
##   MARRIAGE     n
## 1        1 13659
## 2        2 15964
## 3        3   323
```

EDUCATION

According to the UCI repository, EDUCATION: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

For the EDUCATION feature attribute, we noticed that 346 observations has a value of "0", "5" or "6", which are not in any of the categories as defined in the variable.

We decided to remove this observations because we cannot assume that these entries belong to any of the other 4 defined categories. Also, since there are 30000 customer data points in the data set, and thus these data points will not affect the performance and evaluation of our model severely, with the change seen in the table below.

```
##   EDUCATION     n
## 1         0    14
## 2         1 10581
## 3         2 14024
## 4         3  4873
## 5         4   123
## 6         5   280
## 7         6    51
```

```
##   EDUCATION     n
## 1         1 10581
## 2         2 14024
## 3         3  4873
## 4         4   123
```

PAY_N

| cat | pay_1 | Pay_2 | Pay_3 | Pay_4 |
|---|---|---|---|---|
| -2 | 2708 | 3722 | 4027 | 4287 |
| -1 | 5633 | 5990 | 5863 | 5617 |
| 0 | 14499 | 15476 | 15518 | 16204 |
| 1 | 3662 | 28 | 4 | 2 |
| 2 | 2640 | 3904 | 3802 | 3142 |
| 3 | 320 | 326 | 237 | 180 |
| 4 | 76 | 97 | 76 | 69 |
| 5 | 24 | 25 | 21 | 35 |
| 6 | 11 | 12 | 23 | 5 |
| 7 | 9 | 20 | 27 | 58 |
| 8 | 19 | 1 | 3 | 2 |

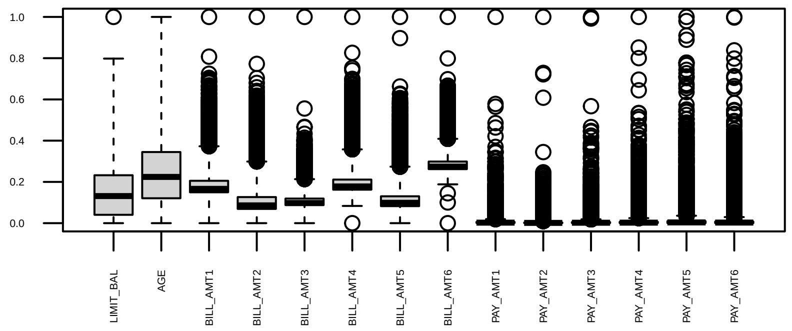
| cat | Pay_5 | Pay_6 |
|---|---|---|
| -2 | 4479 | 4806 |
| -1 | 5480 | 5674 |
| 0 | 16684 | 16053 |
| 2 | 2617 | 2756 |
| 3 | 177 | 183 |
| 4 | 84 | 49 |
| 5 | 17 | 13 |
| 6 | 4 | 19 |
| 7 | 58 | 46 |
| 8 | 1 | 2 |

According to the UCI repository, for PAY_N: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

However, There are data points of PAY_N that are labelled as “-2” and “0”, which do not correspond to any of the above-mentioned categories. Upon further research on the dataset, we have found that the author who created the data set clarified that “-2”: No consumption and “0”: The use of revolving credit, as seen in https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset/discussion/34608.

Therefore, there are no errors in the PAY_N columns with respect to the categories of PAY_N and no further action is required.

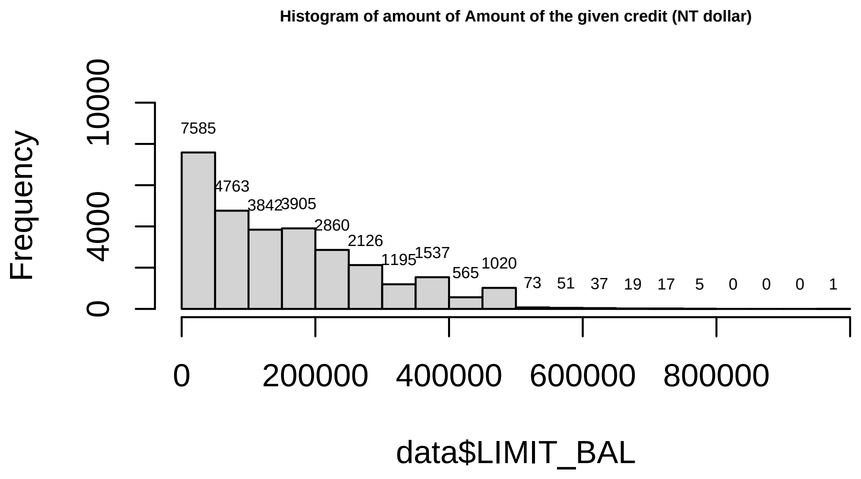
c. **Checking For Outliers (For Numerical Data)**



Input variables may have different units of different scales and magnitude; for this reason before drawing a boxplot, a Min-Max standardisation is applied in order to scale the features between the range of 0 to 1.

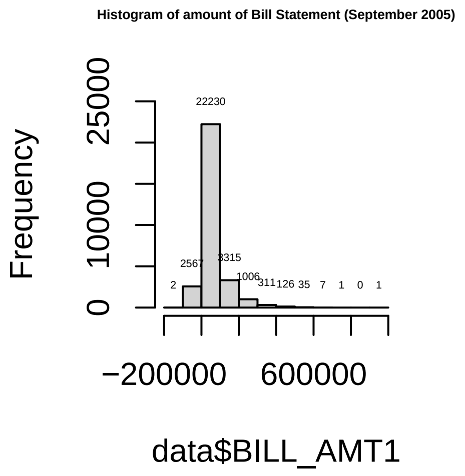
From the boxplot shown, we can see that there is a significant amount of outliers present in the dataset. Due to this significant number, we choose not to remove these outliers since they may contain valuable information that can improve our model, and we do not have enough concrete evidence to prove that they are anomalies in observations.

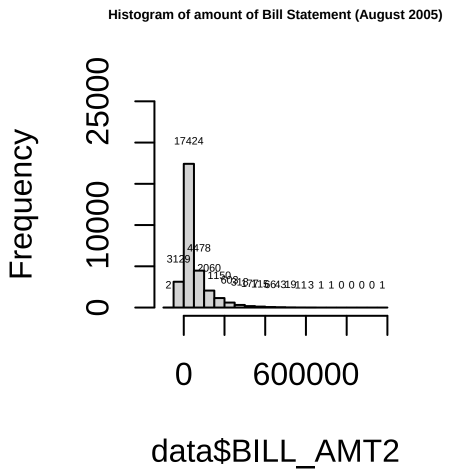
Due to the high number of outliers, we plotted the histograms of each respective feature to understand the distribution of each feature.

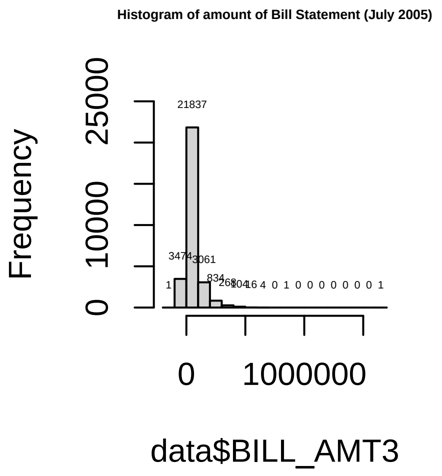


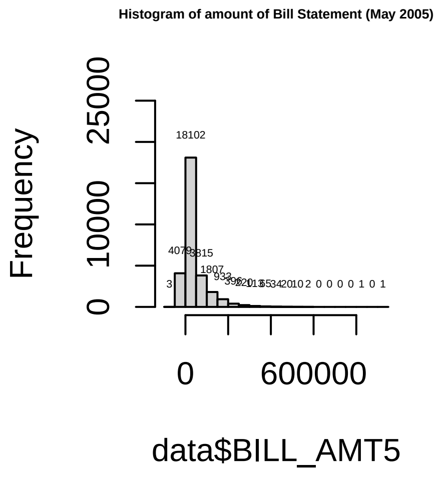
Looking at the distribution of LIMIT_BAL, the perceived outliers do not stray away from the observed distribution as seen from the histogram.

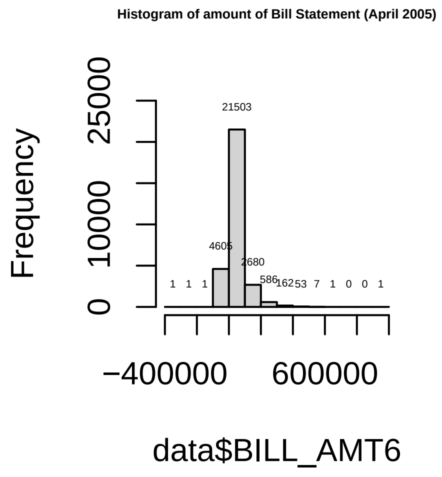
Hence, we decided not to remove any data points of LIMIT_BAL as they are plausible values of the amount of the given credit (NT dollar).





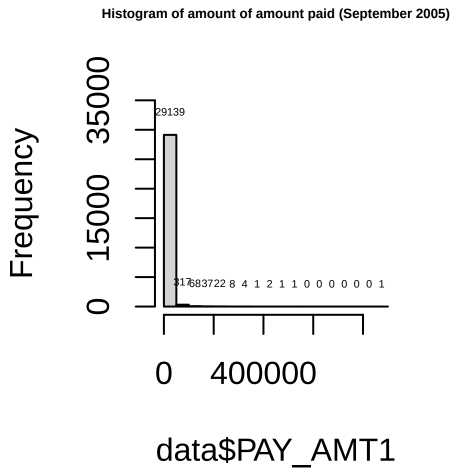


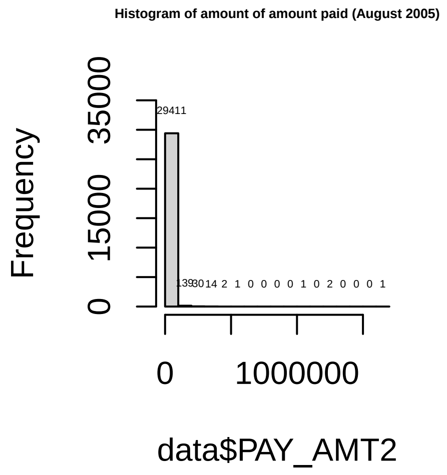


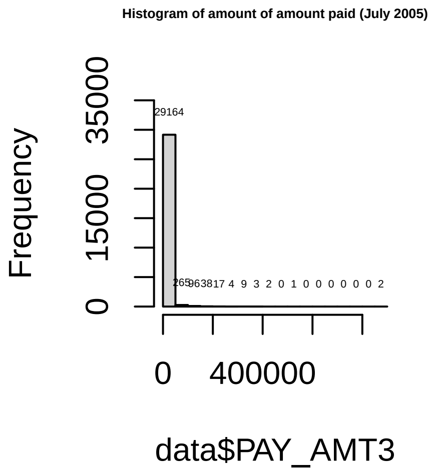


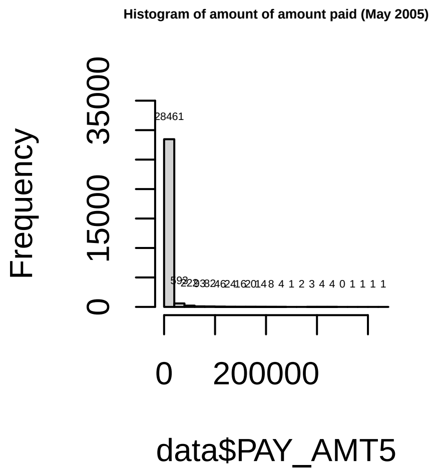
Looking at the distribution of BILL_AMTN, the perceived outliers do not stray away from the observed distribution as seen from the histograms.

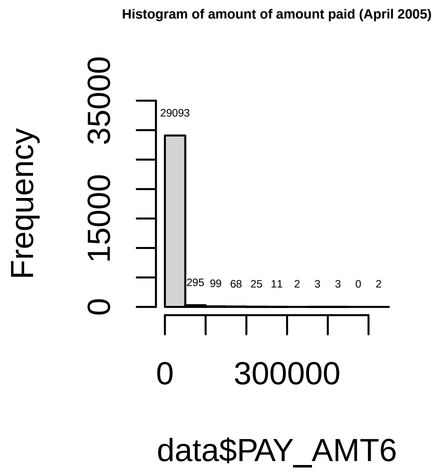
Hence, we decided not to remove any data points of BILL_AMTN as they are plausible values of the amount of the amount of bill statement (NT dollar).





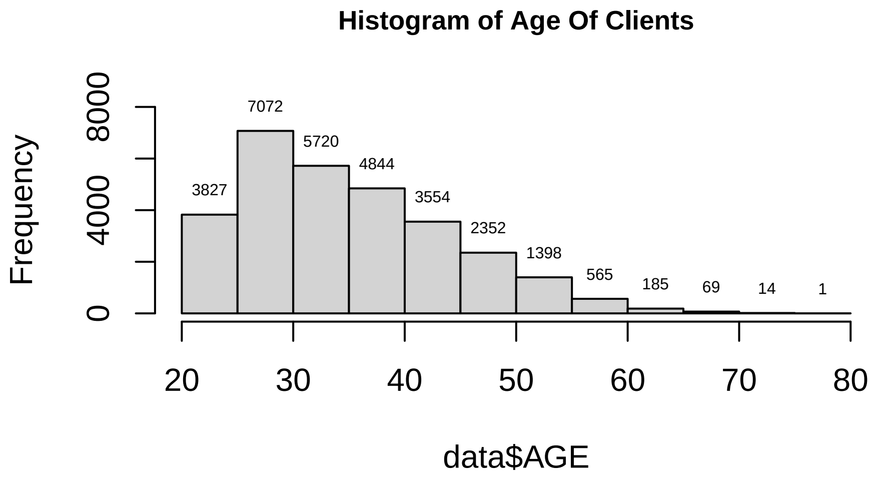






Looking at the distribution of PAY_AMTN, the perceived outliers do not stray away from the observed distribution as seen from the histograms.

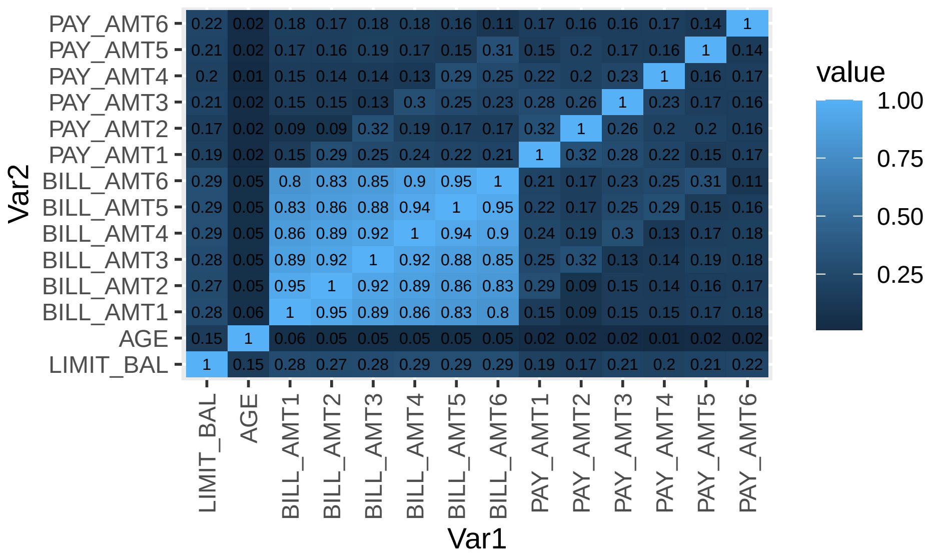
Hence, we decided not to remove any data points of PAY_AMTN as they are plausible values of the amount of the amount of previous payment (NT dollar).



No outliers detected as the range of values in as shown in the histograms of AGE are plausible values of the amount of age in years

d. **Correlation Between Features**

One factor that could affect machine learning classification performances is correlation between features. This is because if features are strongly correlated to each other, classification algorithms which assume that the features are all independent may result in a poor classification performance. Reducing the number of dimensions of feature vectors could lead to better classification performances.



Features that have high correlation to each other ($\rho > 0.9$):

BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6

5 out of 6 of this variables will later be removed in the data pre-processing phase as they likely will not be useful in predicting the target values of whether payment is defaulted (default_payment_next_month) as they may impact the performance of the machine learning models.

## 3. Data Pre-Processing

The data pre-processing section involves transforming raw data into an understandable and usable format.

### a. One-Hot Encoding

Categorical features must be changed in the data pre-processing phase since machine learning models require numeric input values. One-hot encoding will be used for categorical data with no ordinal relationship in our data set. One-hot encoding creates new binary dummy variables for each class in every categorical feature. For example, categorical features such as EDUCATION has been split into dummy variables such as EDUCATION_1 to EDUCATION_4.

### b. Feature Scaling

It's a common case that in most data sets, features are not on the same scale. Machine learning models are, however, largely Euclidean distant-based. Without feature scaling, features with large units will dominate those with small units when it comes to calculation of the Euclidean distance, hence features with small units may be neglected. To prevent this from happening, we need to scale our numerical features.

In this project, we will be using be using caret library to pre-process and scale the data through Min-Max Scaling method to scale the data values to a range of 0 to 1.

### c. Train-Test Split

Each dataset is split into 2 parts, the training set and test set, in the proportion 3:1. To reduce any leaking of information into the testset, we will be doing all feature selection techniques on the trainset only. This includes testing for high correlation and implementing oversampling and undersampling techniques.

## 4. Feature Selection

Feature selection is the process where features which contribute most to the prediction variable are selected.

### a. Removal of highly correlated variables

As mentioned above, features that have high correlation to each other will be removed as they likely will not be useful in predicting the target values of whether payment is defaulted (default_payment_next_month) as they may impact the performance of the machine learning models. Some classification models such as the Naive Bayes Classification are also reliant on the assumption that features used to predict the target variable are linearly independent.

### b. Principal Component Analysis (PCA)

PCA is particularly useful for dimensionality reduction when the dimensions of the data are high. Since there is a large number of feature attributes in our dataset, to reduce the number of features used in the model, we can also use PCA to find a set of features ("Principal Components") that explain the most variance in data. Since PCA is designed to minimize variance (squared deviations), we applied PCA to only the continuous variables in our dataset as the concept of squared deviations break down when considering binary variables.

We can then choose a subset of the PCs, e.g. the first k PCs, perhaps based on the cumulative variance explained. (e.g, the first 7 PCs may account for 99% of the variance). For our project, we chose PCA as a feature selection because we do not require interpretability of the selected features.

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.1626 0.1295 0.06199 0.03624 0.03199 0.02467 0.01992
## Proportion of Variance 0.5201 0.3297 0.07558 0.02583 0.02012 0.01197 0.00780
## Cumulative Proportion  0.5201 0.8498 0.92541 0.95124 0.97136 0.98333 0.99113
##                            PC8     PC9
## Standard deviation     0.01688 0.01288
## Proportion of Variance 0.00561 0.00326
## Cumulative Proportion  0.99674 1.00000
```

From the summary of the PCA done, Principal components 1 to 7 together explain 99.113% of the variance in the data. Thus, we will use PCA 1 to 7 in our machine learning models.

Using the Principal Components derived in our PCA analysis from our trainset, we would be projecting it to our test set. This way the points in both train and test sets end up in the same dimension space, and we would not be using any knowledge about our test set during training, hence preventing leaking.

### c. Oversampling and Undersampling techniques

For the target values of whether payment is defaulted (default_payment_next_month) in training data set, data set is unbalanced as 22.31881% of clients has defaulted on payment, as compared to 77.68119% who did not default on payment, as seen in the summary table below.

```
## # A tibble: 2 x 2
##   default_payment_next_month `Percentage of Total`
##   <fct>                                      <dbl>
## 1 0                                           77.7
## 2 1                                           22.3
```

To prevent the training process from biasing towards a certain class over another, over-sampling and under-sampling techniques will be used.

Over-sampling generates more observations from the minority class to ensure the dataset is balanced. Under-sampling reduces the observations from the majority class to ensure the dataset is balanced.

Data generated from over-sampling is expected to have repeated observations and data generated from under-sampling is expected to be deprived of important information from the original data. This would result in inaccuracies in the resulting performance of the machine learning algorithms.

To prevent this from happening, the ROSE package helps to generate data synthetically.

However, it is noteworthy that synthetic samples do not formally belong to the target distribution hence should not be used as a test sample.

PCA feature selection was applied before oversampling and undersampling. By doing so, we are able to project the data on the axes/plane where the variance of the data is the highest, hence leveraging the maximisation of within data variability. Running oversampling and undersampling on when the variability within the data is high creates synthetic samples closer to the actual data.

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.1626 0.1295 0.06199 0.03624 0.03199 0.02467 0.01992
## Proportion of Variance 0.5201 0.3297 0.07558 0.02583 0.02012 0.01197 0.00780
## Cumulative Proportion  0.5201 0.8498 0.92541 0.95124 0.97136 0.98333 0.99113
##                            PC8     PC9
## Standard deviation     0.01688 0.01288
## Proportion of Variance 0.00561 0.00326
## Cumulative Proportion  0.99674 1.00000
```

# 5. Model Selection

We will now run a variety of machine learning models to test and evaluate which one performs the best in predicting defaulters among credit card customers.

The models we will be using are:

1) Logistic Regression
2) Support Vector Machines (SVM)
3) Naive Bayes Classification
4) Decision Tree (Conditional Inference Tree)
5) Random Forest
6) Neural Networks

We will be training each model on datasets with the different preprocessing techniques using the train set. The different datasets are:

1) Original trainset after removing highly correlated variables

2) Original trainset after removing highly correlated variables + Oversampling and undersampling techniques
3) PCA after removing highly correlated variables
4) PCA after removing highly correlated variables + Oversampling and undersampling techniques

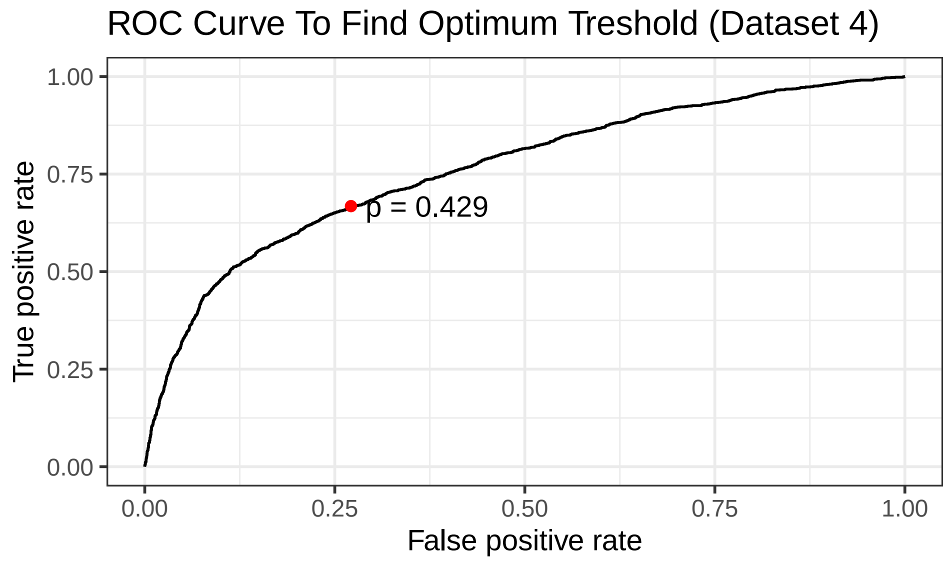
After which, we will be evaluating each respective model using the test set.

## 1. Logistic Regression

Logistic regression is a classification algorithm used to find the probability of event success and failure. Since outcome variable default__payment__next__month: target values of whether payment is defaulted (Yes = 1, No = 0) is binary, logistic regression is used.

To find optimum threshold for the logistic regression classification, we have to decide how much True Positive Rate(TPR) and False Positive Rate(FPR) we wish to have. If we were to increase the TPR, FPR will likely to increase as well. For this set of data, we wish to minimise the number of false negatives as far as possible, hence we choose a threshold that increases TPR and reduces FPR.

To find the probability threshold to give us the best performance, we looked into the point on the ROC curve that will maximise TPR while minimising FPR at the same time. For instance, we plotted the ROC curve for test results of dataset 4 (PCA after removing highly correlated variables + Oversampling and undersampling techniques), which gave us an optimum threshold of 0.429, as seen below.



```
##                               precision    recall  accuracy        f1
## corr                          0.8083692 0.7189565 0.6491892 0.7610457
## corr+under/oversampling       0.8716850 0.7431304 0.7154054 0.8022906
## corr+PCA                      0.8374452 0.6316522 0.6185135 0.7201348
## corr+under/oversampling+PCA 0.8842661 0.7281739 0.7147297 0.7986648
##                               avg_accuracy       auc
## corr                             0.5625086 0.5625086
## corr+under/oversampling          0.6809592 0.6809592
## corr+PCA                         0.6021897 0.6021897
## corr+under/oversampling+PCA      0.6980264 0.7632454
```

Table 1: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 4134 | 1616 | 4273 | 1477 | 3632 | 2118 | 4187 | 1563 |
| 1 | 980 | 670 | 629 | 1021 | 705 | 945 | 548 | 1102 |

## 2. Support Vector Model (SVM)

Support Vector Machines is a supervised machine learning algorithm which uses classification algorithms for two-group classification problems.

SVM is an appropriate model to use in this report since we do not need to interpret the model, and merely need the results of its classification. Furthermore, SVM works well with high dimensional data, and is able to solve complex problems conveniently using the kernel functions.

The type of *svm* used is C-classification. For a binary classification as in this case where there are only 2 classes $defaulter$ and $non-defaulter$, the $C$ parameter comes into the picture when the dataset is linearly non separable. It accounts for the slack variables that are introduced into the linear constraints so that the model becomes feasible. $C > 0$ for all $C$.

Cost (C) is a regularisation parameter, it tells the optimizer what it should optimise more, the distance between data inputs to the hyperplane or the penalty of mis-classification. A large cost would tell the optimiser to minimise misclassification since C is the scalar weight of the penalty of mis-classification.

We iterated different values of C into the svm models to find the C that gives us the best classification performance. We then used the optimal value of C to train each respective dataset.

For kernel types, we iterated through different kernel types and found that the linear kernel gives the best classification performance on the dataset. We then used the optimal kernel type to train each respective dataset.

```
##                              precision    recall  accuracy        f1
## corr                         0.8314776 0.9610435 0.8183784 0.8915779
## corr+under/oversampling      0.8597721 0.8530435 0.7777027 0.8563946
## corr+PCA                     0.8314776 0.9610435 0.8183784 0.8915779
## corr+under/oversampling+PCA 0.8583834 0.8939130 0.8029730 0.8757880
##                              avg_accuracy       auc
## corr                            0.6411278 0.6411278
## corr+under/oversampling         0.6840975 0.6840975
## corr+PCA                        0.6411278 0.6411278
## corr+under/oversampling+PCA     0.6899868 0.6899868
```

Table 2: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5526 | 224 | 4905 | 845 | 5526 | 224 | 5140 | 610 |
| 1 | 1120 | 530 | 800 | 850 | 1120 | 530 | 848 | 802 |

## 3. Naive Bayes Classification

This is a supervised machine learning classification technique that works off the Bayes' Theorem, which works on the principle of conditional probaility and assumes that predictors are independent of each other. Naives Bayes Classification is fast for making real-time predictions and is observed to be effective for large datasets.

However, if categorical variable has a category (in test data set), which was not observed in training data set, then model will assign its probability to equal 0.

To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

We chosen Naive Bayes classifier as one of our classifying algorithms as it is not sensitive to irrelevant features. Although we have conducted feature selection and dimensionality reduction to reduce irrelevant and redundant features, there may still be correlation between features which might affect classification performance for other unsupervised learning algorithms. Naive Bayes classifier therefore could be a suitable classifier to overcome the issue of irrelevant features.

However Naive Bayes has a limited parameter set, hence hyperparameter tuning is not the most valid method to improve accuracy. To increase the performance of our Naive Bayes classifier, it is important to handle missing data values and remove correlated features as we might double count features, overestimating the importance of the feature, degrading the performance of the Naive Bayes classifer.

We have concluded in our data pre-processing phase that there are no missing values and have removed highly correlated features, therefore we have optimised our Naive Bayes classifer and our results are as shown below.

```
##                              precision    recall  accuracy        f1
## corr                         0.8016339 0.9726957 0.7917568 0.8789188
## corr+under/oversampling      0.7917604 0.9793043 0.7837838 0.8756026
## corr+PCA                     0.8022857 0.9766957 0.7948649 0.8809412
## corr+under/oversampling+PCA  0.7912472 0.9841739 0.7859459 0.8772283
##                              avg_accuracy       auc
## corr                            0.5669539 0.5669539
## corr+under/oversampling         0.5408643 0.5408643
## corr+PCA                        0.5689539 0.5689539
## corr+under/oversampling+PCA     0.5396627 0.5396627
```

Table 3: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5593 | 157 | 5631 | 119 | 5616 | 134 | 5659 | 91 |
| 1 | 1384 | 266 | 1481 | 169 | 1384 | 266 | 1493 | 157 |

## 4. Decision Tree (Conditional Inference Tree)

Decision Trees are a type of supervised machine learning algorithm where data is continuously split based on a certain parameter. The trees contain 2 entities - decision nodes and leaves. Decision nodes are where data is split and leaves are the decisions or final outcomes.

The decision tree performs recursive partitioning of data, thus breaking the data down into smaller subsets to eventually predict outcomes of the target variable.

The decision tree classification method works by testing the global null hypothesis of independence between any of the input variables and the target. It selects the input variable with strongest association to the target. This association is measured by finding the largest p-values with respect to a test for the partial null hypothesis between each input variable and the response variable.

Using this rank of features, the decision tree algorithm then checks through each feature and moves from feature to feature to finally predict the target class, forming a tree like model. In addition, we chose decision tree as it is compatible with categorical and continuous variables, and is also robust to outliers.

We iterated through different values of mincriterion to input into the decision tree models to find the value that gives us the best classification performance. (1 - mincriterion) is the significance level for each hypothesis test conducted that must be exceeded in order for a binary split to be implemented.

```
##                              precision    recall  accuracy        f1
## corr                         0.9514783 0.8378254 0.8191892 0.8910423
## corr+under/oversampling      0.8267826 0.8698994 0.7693243 0.8477931
## corr+PCA                     0.9516522 0.8375938 0.8190541 0.8909875
## corr+under/oversampling+PCA 0.3335652 0.8551048 0.4382432 0.4799199
##                              avg_accuracy       auc
## corr                            0.7585679 0.6548300
## corr+under/oversampling         0.6775853 0.6979368
## corr+PCA                        0.7584739 0.6543109
## corr+under/oversampling+PCA     0.5560185 0.5682978
```

Table 4: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5471 | 1059 | 4754 | 711 | 5472 | 1061 | 1918 | 325 |
| 1 | 279 | 591 | 996 | 939 | 278 | 589 | 3832 | 1325 |

## 5. Random Forest

The random forest is essentially an ensemble of randomly created decision trees, where each node in different trees are working on a random subset of features to calculate the output. Output of each decision tree is then combined to give the final prediction of the target class.

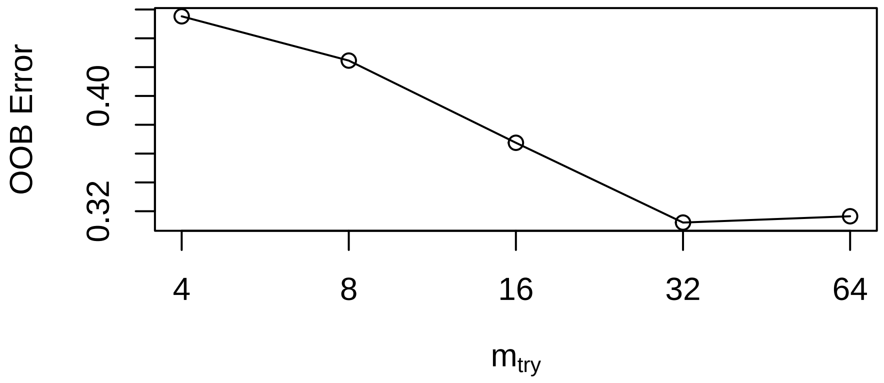
To optimise the performance of our random forest model, we tuned the mtry parameter, which refers to the number of variables randomly sampled as candidates at each split. This was done by find the mtry value that returns the lowest out of bag error (OOB). OOB refers to the errors achieved on out of bag samples, which are observations in the training data that are not used in training the model since the random forest algorithm utilises bootstrapping which allows for a sample to be selected several times and some to be excluded.

Since the random forest classification combines the outcomes of several decision trees, it reduces the risk of overfitting. Furthermore, the random forest algorithm is compatible with categorical and continuous variables, and is also robust to outliers.

```
## -0.00772297 0.05
## -0.01145989 0.05
```

```
## 0.01166117 0.05
## -0.2354235 0.05
## -0.01211971 0.05
## -0.00222607 0.05
```

```
## 0.1343236 0.05
## 0.1506312 0.05
## -0.01399711 0.05
## -0.4584416 0.05
```



```
##                               precision    recall  accuracy        f1
## corr                          0.8352333 0.9556522 0.8190541 0.8913943
## corr+under/oversampling       0.7780035 0.9989565 0.7777027 0.8747430
## corr+PCA                      0.8358732 0.9539130 0.8186486 0.8910006
## corr+under/oversampling+PCA 0.8808511 0.1800000 0.3439189 0.2989170
##                               avg_accuracy       auc
## corr                             0.6493412 0.6493412
## corr+under/oversampling          0.5028116 0.5028116
## corr+PCA                         0.6505929 0.6505929
## corr+under/oversampling+PCA      0.5475758 0.5475758
```

Table 5: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5495 | 255 | 5744 | 6 | 5485 | 265 | 1035 | 4715 |
| 1 | 1084 | 566 | 1639 | 11 | 1077 | 573 | 140 | 1510 |

## 6. Neural Networks

Neural Networks (diff variations of neural networks) The neural network algorithm comprises of 3 layers, the input layer, hidden layer and output layer. Each node in every layers connect to another, and has an associated weight and threshold. When output for a individual node is higher than the threshold, that node is activated and data is passed on to the next layer. For the input layer, there are as many nodes as there are attributes in the training data. For the output layer, since our target class is binary, we will only have one node. For the hidden layer, there can be multiple hidden layer but in our case we choose to only use 1 hidden layer due to the heavy computational cost and long training time required.

To select the number of nodes in the hidden layer, we iterated a range of values for the size of each layer and computed the accuracy rate achieved with each value. We thus selected the hidden layer sizes that returned the higher accuracy rates.

To select the learning rates, we set a minimum of 0.06 due to the long computation times associated with a low learning rate. We then iterated a range of values to find the learning rate that returns the best performance.

To speed up the learning process, we set the minimum threshold at 0.05. This means that if the change in the error between each iteration is less than 5%, the neural network model will stop optimising its weights.

Table 6: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5462 | 1048 | 4783 | 707 | 5329 | 962 | 5455 | 1041 |
| 1 | 288 | 602 | 967 | 943 | 421 | 688 | 295 | 609 |

```
##                              precision    recall  accuracy        f1
## corr                         0.9499130 0.8390169 0.8194595 0.8910277
## corr+under/oversampling      0.8318261 0.8712204 0.7737838 0.8510676
## corr+PCA                     0.9267826 0.8470831 0.8131081 0.8851424
## corr+under/oversampling+PCA 0.9486957 0.8397475 0.8194595 0.8909032
##                              avg_accuracy       auc
## corr                            0.7577107 0.7495964
## corr+under/oversampling         0.6824688 0.7484972
## corr+PCA                        0.7337309 0.7527483
## corr+under/oversampling+PCA     0.7567101 0.7525197
```

## 6. Model Evaluation

### Precision

Among all positive predictions, the proportion of actual positive instances. For our dataset, precision is a measure of the bank clients we correctly identify as individuals who defaulted on payment out of all the patients who defaulted payment.

Precision is also a measure of quality; a higher precision suggests that the classification algorithm returns more relevant results. Having a high precision is important because the bank would lose low-risk customers if the clients were deemed as defaulters based on the model's prediction.

### Recall

Among all positive instances, the proportion of correct predictions. For our dataset, recall is a measure of for all clients who have actually defaulted on payment, recall tells us how many clients we have correctly identified. Recall is a measure of quantity; a higher recall suggests that the classification algorithm returns most of the relevant results. We would like to avoid the situation where the client is a defaulter but our model fails to identify the client as a defaulter, as accepting these clients as defaulters may bring additional financial risk to the bank.

**Accuracy**

Accuracy is the ratio of the number of correct predictions over the total number of predictions. For our dataset, accuracy is a measure of the clients who are correctly predicted as defaulters and non-defaulters respectively out of all the predictions. Accuracy is important to the bank as it is important to be able to correctly identify the defaulters and non-defaulters out of all the clients. However, accuracy alone is a bad metric in the case of unbalanced datasets such as our dataset.

**Average Class Accuracy**

Average class accuracy measures the average accuracy per class, and is particularly beneficial when the data is imbalanced as classification accuracy can mask poor performance where data is imbalanced. In this case, average class reduces the accuracy bias for unbalanced datasets on a certain class, where in this case non-defaulters are the majority class and defaulters are the minority class.

**F1 Score**

F1 score is defined as the harmonic mean between precision and recall. F1 score is more useful in evaluating classification model performance in the case of unbalanced datasets such as our dataset as compared to precision and recall as F1 score gives an equal weight to precision and recall.

**Area Under ROC Curve**

AUC is a measure of separability between the classes of the target variable, which in our class is defaulters and non-defaulters. The higher the AUC, the better the classification model is at predicting defaulters as defaulters and non-defaulters as non-defaulters.

Based on our dataset and models selected, we have decided to evaluate our models based on

***F1 Score, Average Class Accuracy, AUC.***

Table 7: F1-Score, Average Class Accuracy, AUC of each model

| | log_reg | | | svm | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.7610457 | 0.5625086 | 0.5625086 | 0.8915779 | 0.6411278 | 0.6411278 |
| corr+under/oversampling | 0.8022906 | 0.6809592 | 0.6809592 | 0.8563946 | 0.6840975 | 0.6840975 |
| corr+PCA | 0.7201348 | 0.6021897 | 0.6021897 | 0.8915779 | 0.6411278 | 0.6411278 |
| corr+under/oversampling+PCA | 0.7986648 | 0.6980264 | 0.7632454 | 0.8757880 | 0.6899868 | 0.6899868 |

Table 8: F1-Score, Average Class Accuracy, AUC of each model

| | naive_bayes | | | decision_tree | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.8789188 | 0.5669539 | 0.5669539 | 0.8910423 | 0.7585679 | 0.6548300 |
| corr+under/oversampling | 0.8756026 | 0.5408643 | 0.5408643 | 0.8477931 | 0.6775853 | 0.6979368 |
| corr+PCA | 0.8809412 | 0.5689539 | 0.5689539 | 0.8909875 | 0.7584739 | 0.6543109 |
| corr+under/oversampling+PCA | 0.8772283 | 0.5396627 | 0.5396627 | 0.4799199 | 0.5560185 | 0.5682978 |

Table 9: F1-Score, Average Class Accuracy, AUC of each model

| | random_forest | | | neural_network | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.8913943 | 0.6493412 | 0.6493412 | 0.8910277 | 0.7577107 | 0.7495964 |
| corr+under/oversampling | 0.8747430 | 0.5028116 | 0.5028116 | 0.8510676 | 0.6824688 | 0.7484972 |
| corr+PCA | 0.8910006 | 0.6505929 | 0.6505929 | 0.8851424 | 0.7337309 | 0.7527483 |
| corr+under/oversampling+PCA | 0.2989170 | 0.5475758 | 0.5475758 | 0.8909032 | 0.7567101 | 0.7525197 |

As a whole, using our selected evaluation metrics of ***F1 Score, Average Class Accuracy, AUC***, the neural networks classification algorithm has shown significantly higher F1 Score, Average Class Accuracy and AUC.

We can identify that the neural networks classification algorithm applied on the dataset with oversampling & undersampling techniques applied, has its highly correlated features removed and PCA feature selection techniques applied to it is the best performing model.

Generally, an F1 score is considered good when it is closer to 1. For our selected model and dataset it is trained on, F1 score is 0.891, and is thus considered a strong performance. For average class accuracy, a value of 0.5 implies that the model predicts classes correctly with a 50% chance. Thus, anything above 0.5 is considered okay, with a value of above 0.7 to be considered good. For our selected model, average class accuracy rate is 0.757, and is thus considered a good classification model. Finally, the AUC score is considered good generally when it is above 0.7. For our selected model, AUC score is 0.753.

## 7. Areas For Improvement

1. **Grid search can be used to identify optimal hyperparameters for the models**

Due to the complexity and high computation time, we are not able to consider all possible combinations of hyperparameters for each respective model. Grid search is a cross-validation technique that would allow us to consider all possible combinations of hyperparameters to find the model with a range of values for each hyperparameter, which would better allow us to selection the best hyperparameters for the classification.

2. **Check for over-fitting**

Over-fitting of the models can be identified by evaluating the models on the train set and the holdout test set separately. Should the performance of the model be significantly better on the train set as compared to the test set, the model may have over-fitted the train dataset. Over-fitting can be identified on learning curve plots where validation metrics such as accuracy and loss continues to improve on the train set while the test set starts to get worse.

# ANNEX

```
suppressMessages(library(dplyr))
suppressMessages(library(caret))
suppressMessages(library(lessR))
suppressMessages(library(ggplot2))
suppressMessages(library(Hmisc))
suppressMessages(library(corrplot))
suppressMessages(library(mltools))
suppressMessages(library(data.table))
suppressMessages(library(cowplot))
suppressMessages(library(e1071))
suppressMessages(library(class))
suppressMessages(library(ROSE))
suppressMessages(library(ROCR))
suppressMessages(library(rpart))
suppressMessages(library(rpart.plot))
suppressMessages(library(randomForest))
suppressMessages(library(party))
suppressMessages(library(pROC))
suppressMessages(library(knitr))
suppressMessages(library(kableExtra))
```

## 1. Brief Introduction Of Data Set And Data Modeling Problem

The dataset "card.csv" has been obtained from UCI repository: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

It contains payment information of 30,000 credit card holders obtained from a bank in Taiwan.

### Dataset Description

ID: Unique dataset identification number for consumer

**Feature Attributes**

LIMIT_BAL: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit (Continuous)

SEX: Gender (1 = male; 2 = female) (Categorical)

EDUCATION: Education (0 = unknown, 1 = graduate school; 2 = university; 3 = high school; 4 = others, 5 = unknown, 6 = unknown) (Categorical)

MARRIAGE: Marital status (0 = unknown, 1 = married; 2 = single; 3 = others) (Categorical)

AGE: Age (in years) (Continuous)

PAY_0 - PAY_6: History of past payment. Past monthly payment records were tracked (from April 2005 to September 2005, where PAY_0 = the repayment status in September 2005 and PAY_6 = the repayment status in April 2005). The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above. (Categorical)

BILL_AMT1 - BILL_AMT6: Amount of bill statement (NT dollar) (from April 2005 to September 2005, where BILL_AMT1 = amount of bill statement in September 2005 and BILL_AMT6 = amount of bill statement in April 2005). (Continuous)

PAY_AMT1 - PAY_AMT6: Amount of previous payment (NT dollar), where PAY_AMT1 = amount paid in September 2005 and PAY_AMT6 = amount paid in April 2005. (Continuous)

**Target Variable**

default_payment_next_month: target values of whether payment is defaulted (Yes = 1, No = 0) (Categorical)

In this report, we aim to evaluate supervised machine learning algorithms in their ability to predict whether a customer will default on their credit card payment. From the perspective of risk management for the bank, predicting whether or not a customer is likely to default payment is important for banks as it poses a financial risk to them.

We will be using the following classification algorithms:

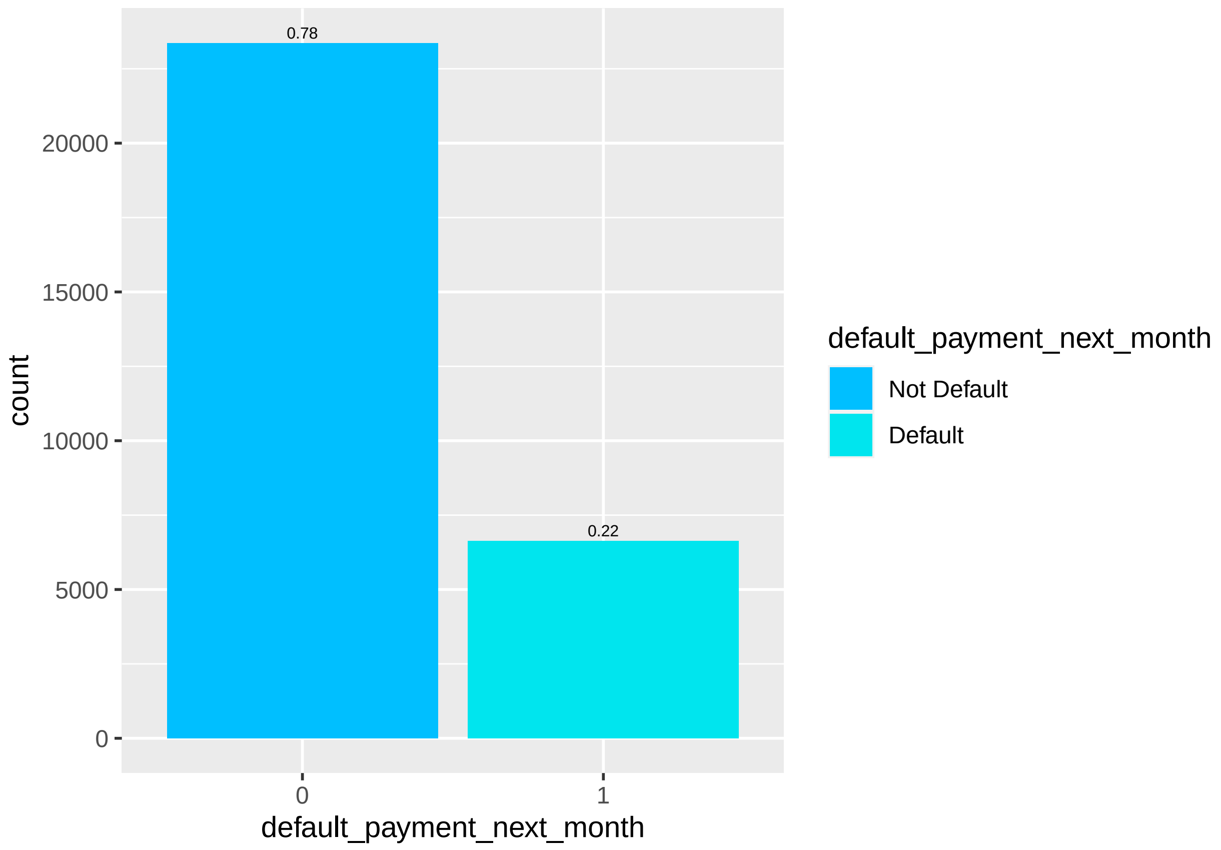
1) Logistic Regression
2) Support Vector Machines (SVM)
3) Naive Bayes Classification
4) Decision Tree (Conditional Inference Tree)
5) Random Forest
6) Neural Networks

```r
data <- read.table("card.csv",sep=",",skip=2,header=FALSE)
header <- scan("card.csv",sep=",",nlines=2,what=character())

#labeling column names
colnames(data) <- c("ID", "LIMIT_BAL", "SEX", "EDUCATION", "MARRIAGE", "AGE",
                    "PAY_0", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6",
                    "BILL_AMT1", "BILL_AMT2", "BILL_AMT3", "BILL_AMT4",
                    "BILL_AMT5", "BILL_AMT6", "PAY_AMT1", "PAY_AMT2",
                    "PAY_AMT3", "PAY_AMT4", "PAY_AMT5", "PAY_AMT6",
                    "default_payment_next_month")

#converting categorical data into factors
data$SEX <- as.factor(data$SEX)
data$EDUCATION <- as.factor(data$EDUCATION)
data$MARRIAGE <- as.factor(data$MARRIAGE)
data$PAY_0 <- as.factor(data$PAY_0)
data$PAY_2 <- as.factor(data$PAY_2)
data$PAY_3 <- as.factor(data$PAY_3)
data$PAY_4 <- as.factor(data$PAY_4)
data$PAY_5 <- as.factor(data$PAY_5)
data$PAY_6 <- as.factor(data$PAY_6)
data$default_payment_next_month <- as.factor(data$default_payment_next_month)
```

## 2. Exploratory Data Analysis

Before we perform initial investigations on data to discover patterns and spot anomalies/inconsistencies in our data, we will first take a look at our target variable, default_payment_next_month.

```r
ggplot(data, aes(default_payment_next_month,
        fill = default_payment_next_month)) +
        geom_bar() +
        scale_fill_manual(values = c("deepskyblue", "turquoise2"),
                          labels=c('Not Default', 'Default'))  +
        geom_text(aes(label=round(..count../30000,2)), stat = "count",
                  vjust = -0.4, colour = "black", size=2)
```



As we can see, we have a binary classification problem as shown in the figure, with values set to "0" for consumers who did not default on payment and "1" for consumers who defaulted on payment. Out of 30,000 samples, 6636 or 22.1% of clients has defaulted on payment.

### Data Cleaning

a. **Handling missing data**

```r
paste("Number Of Missing Values In Dataset:", sum(is.na(data)))
```

```
## [1] "Number Of Missing Values In Dataset: 0"
```

There are no missing values in the dataset.

b. **Categorical Feature Analysis**

Firstly, column for the repayment status in September 2005 is wrongly labelled as PAY_0. To be consistent with "BILL_AMT1" and PAY_AMT1", the "PAY_0" column will be re-labelled as "PAY_1" instead.

```
colnames(data)[7] = "PAY_1"
```

Secondly, we also found unknown values for certain categorical features.

These errors in the data set can be addressed in two ways:

1. Removing the rows which contain the error
2. Replace the wrong attribute class with a correction

MARRIAGE

According to the UCI repository, MARRIAGE: Marital status (1 = married; 2 = single; 3 = others). There are data points of MARRIAGE that are labelled as "0", which do not correspond to any of the above-mentioned categories.

For the MARRIAGE feature attribute, we noticed that 54 observations has a value of "0", which is not in any of the categories as defined in the variable.

We decided to remove this observations because we cannot assume that these entries belong to any of the other 3 defined categories. Also, since there are 30000 customer data points in the data set, and thus these data points will not affect the performance and evaluation of our model severely, with the change seen in the table below.

```
marriage1<-count(data, MARRIAGE)
marriage1
```

```
##   MARRIAGE     n
## 1        0    54
## 2        1 13659
## 3        2 15964
## 4        3   323
```

```
data = data[!(data$MARRIAGE == 0),]
data$MARRIAGE = droplevels(data$MARRIAGE)
marriage2 <- count(data, MARRIAGE)
marriage2
```

```
##   MARRIAGE     n
## 1        1 13659
## 2        2 15964
## 3        3   323
```

EDUCATION

According to the UCI repository, EDUCATION: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

For the EDUCATION feature attribute, we noticed that 346 observations has a value of “0”, “5” or “6”, which are not in any of the categories as defined in the variable.

We decided to remove this observations because we cannot assume that these entries belong to any of the other 4 defined categories. Also, since there are 30000 customer data points in the data set, and thus these data points will not affect the performance and evaluation of our model severely, with the change seen in the table below.

```
edu1 = count(data, EDUCATION)
edu1
```

```
##   EDUCATION     n
## 1         0    14
## 2         1 10581
## 3         2 14024
## 4         3  4873
## 5         4   123
## 6         5   280
## 7         6    51
```

```
data = data[!(data$EDUCATION == 0 | data$EDUCATION == 5 | data$EDUCATION == 6),]
data$EDUCATION = droplevels(data$EDUCATION)
edu2 = count(data, EDUCATION)
edu2
```

```
##   EDUCATION     n
## 1         1 10581
## 2         2 14024
## 3         3  4873
## 4         4   123
```

PAY_N

```
x = cbind(count(data, PAY_1),count(data, PAY_2)$n,count(data, PAY_3)$n,count(data, PAY_4)$n)
colnames(x) <- c('cat','pay_1','Pay_2','Pay_3','Pay_4')
y = cbind(count(data, PAY_5),count(data, PAY_6)$n)
colnames(y) <- c('cat','Pay_5','Pay_6')

x
```

```
##    cat pay_1 Pay_2 Pay_3 Pay_4
## 1   -2  2708  3722  4027  4287
## 2   -1  5633  5990  5863  5617
## 3    0 14499 15476 15518 16204
## 4    1  3662    28     4     2
## 5    2  2640  3904  3802  3142
## 6    3   320   326   237   180
## 7    4    76    97    76    69
## 8    5    24    25    21    35
## 9    6    11    12    23     5
## 10   7     9    20    27    58
## 11   8    19     1     3     2
```

```
y
```

```
##    cat Pay_5 Pay_6
## 1   -2  4479  4806
## 2   -1  5480  5674
## 3    0 16684 16053
## 4    2  2617  2756
## 5    3   177   183
## 6    4    84    49
## 7    5    17    13
## 8    6     4    19
## 9    7    58    46
## 10   8     1     2
```

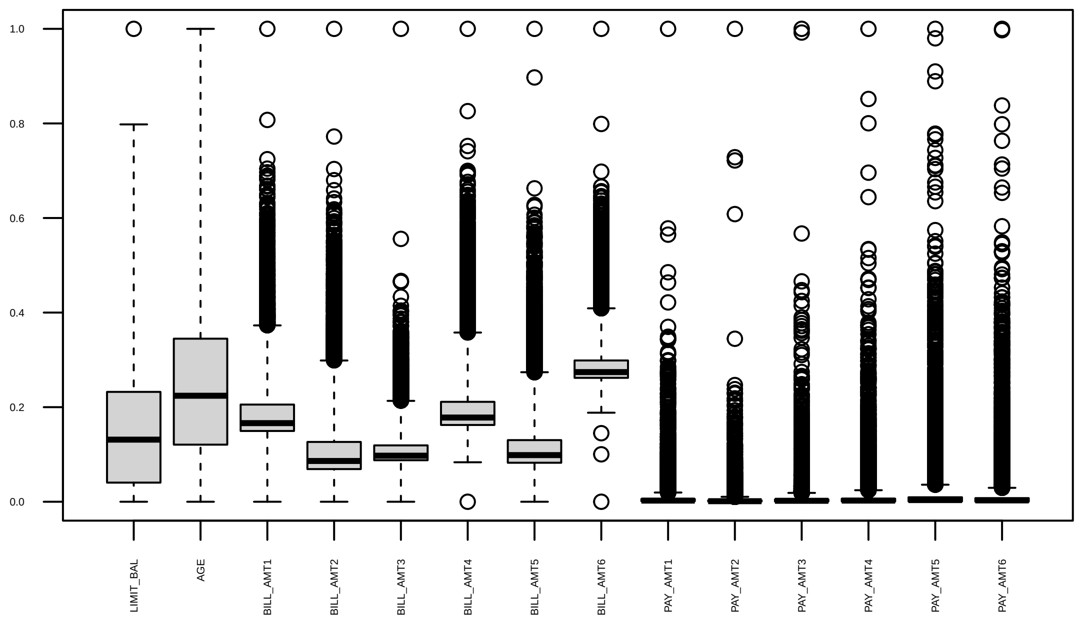
According to the UCI repository, for PAY_N: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

However, There are data points of PAY_N that are labelled as "-2" and "0", which do not correspond to any of the above-mentioned categories. Upon further research on the dataset, we have found that the author who created the data set clarified that "-2": No consumption and "0": The use of revolving credit, as seen in https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset/discussion/34608.

Therefore, there are no errors in the PAY_N columns with respect to the categories of PAY_N and no further action is required.

c. **Checking For Outliers (For Numerical Data)**

```
process <- caret::preProcess(data[,c(2,6,13,14,15,16,17,18,19,20,21,22,23,24)], method=c("range"))

norm_scale <- predict(process,data[,c(2,6,13,14,15,16,17,18,19,20,21,22,23,24)])

boxplot1<- boxplot(norm_scale,range= 3, cex.axis=0.3, las=2)
```

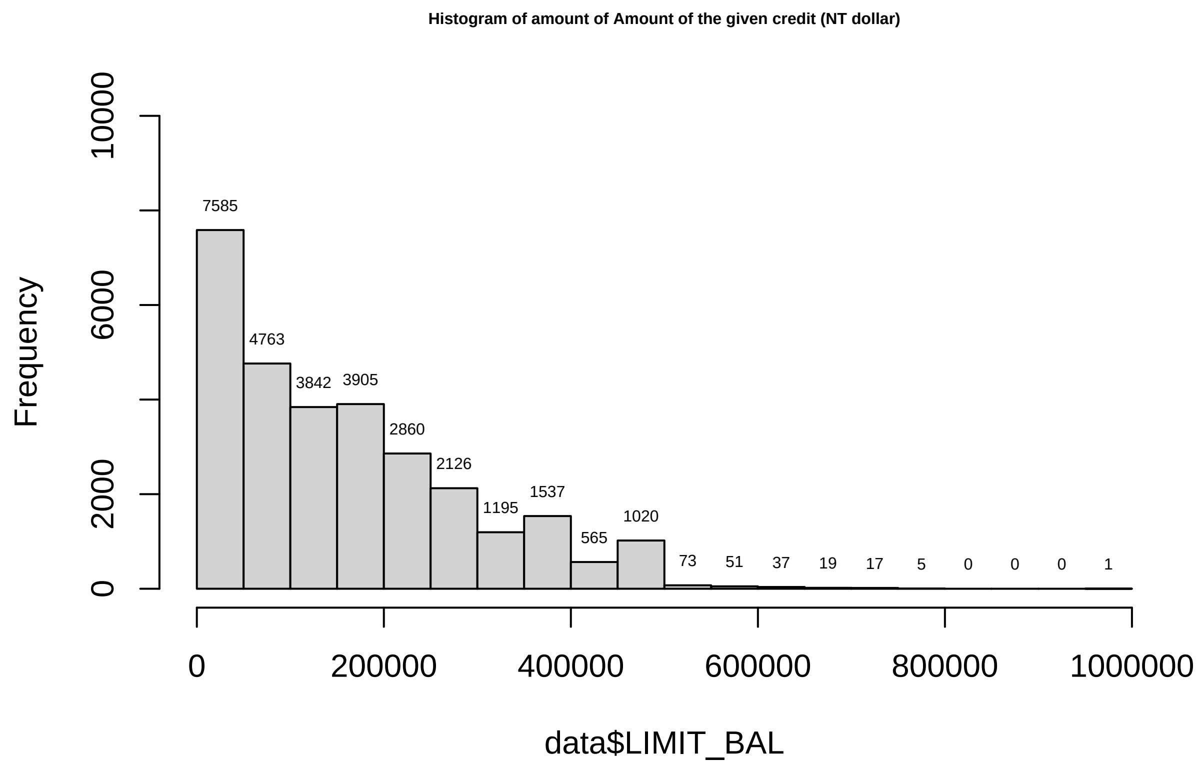


Input variables may have different units of different scales and magnitude; for this reason before drawing a boxplot, a Min-Max standardisation is applied in order to scale the features between the range of 0 to 1.

From the boxplot shown, we can see that there is a significant amount of outliers present in the dataset. Due to this significant number, we choose not to remove these outliers since they may contain valuable information that can improve our model, and we do not have enough concrete evidence to prove that they are anomalies in observations.

Due to the high number of outliers, we plotted the histograms of each respective feature to understand the distribution of each feature.

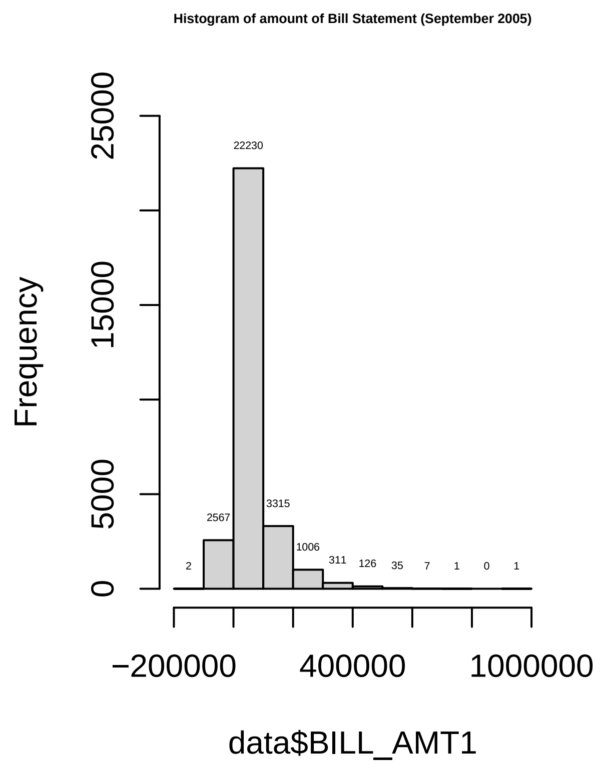
```
#Histogram of Amount of the given credit (NT dollar)
a = hist(data$LIMIT_BAL, main="Histogram of amount of Amount of the given credit (NT dollar)",
         cex.main = 0.5, breaks = "Sturges", labels=F, ylim = c(0,10000))
text(x = a$mids, y = a$counts, labels = a$counts, cex = 0.5, pos = 3)
```

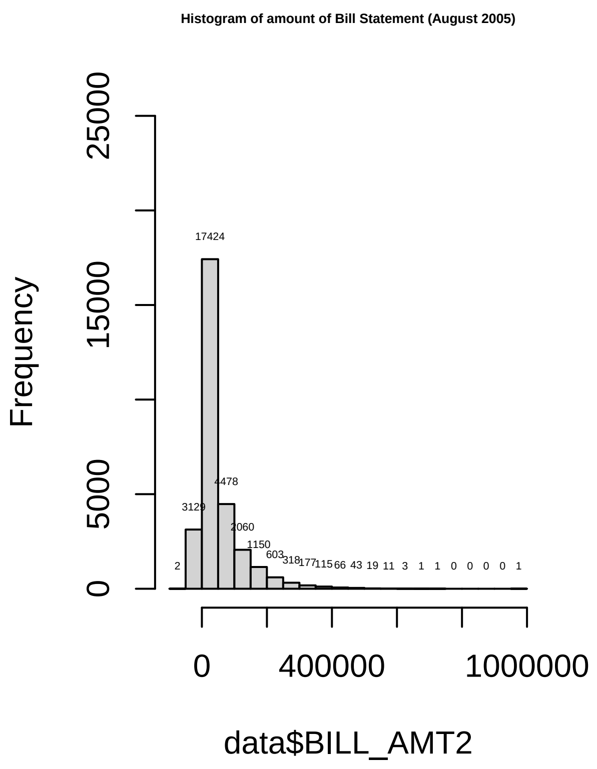


Looking at the distribution of LIMIT_BAL, the perceived outliers do not stray away from the observed distribution as seen from the histogram.

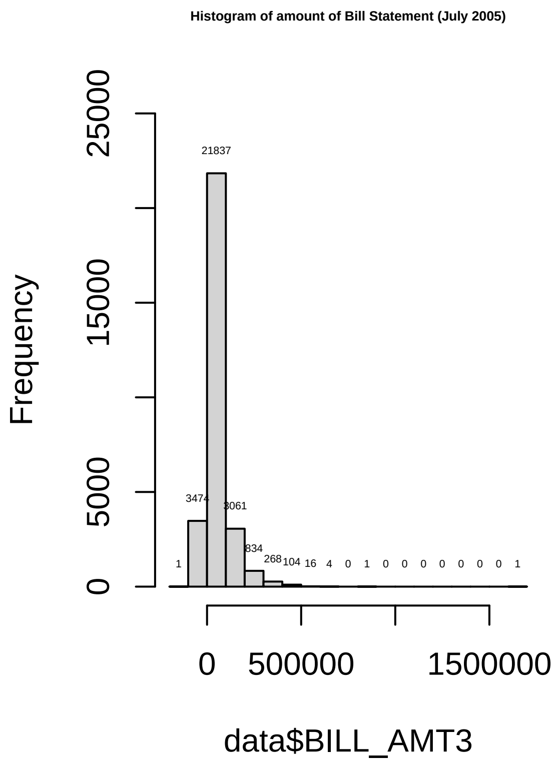
Hence, we decided not to remove any data points of LIMIT_BAL as they are plausible values of the amount of the given credit (NT dollar).

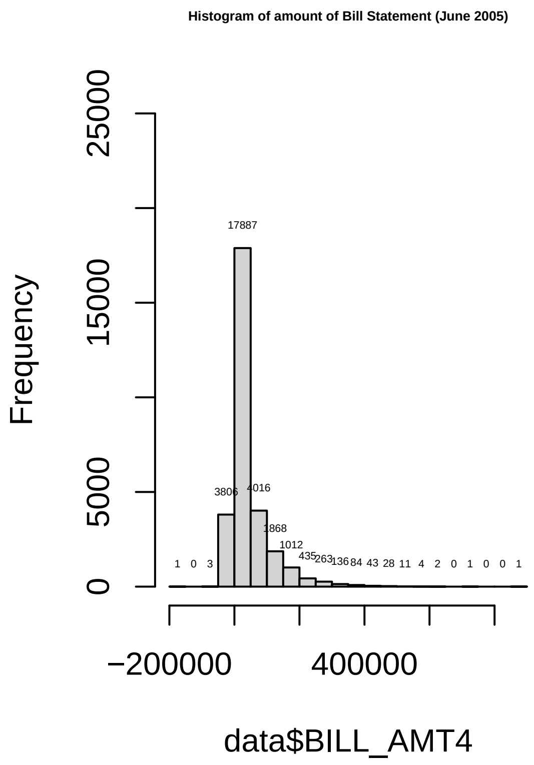
```r
#Histograms of Bill Statements

par(mfrow=c(1,2))
a = hist(data$BILL_AMT1, main="Histogram of amount of Bill Statement (September 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = a$mids, y = a$counts, labels = a$counts, cex = 0.35, pos = 3)

b = hist(data$BILL_AMT2, main="Histogram of amount of Bill Statement (August 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = b$mids, y = b$counts, labels = b$counts, cex = 0.35, pos = 3)
```



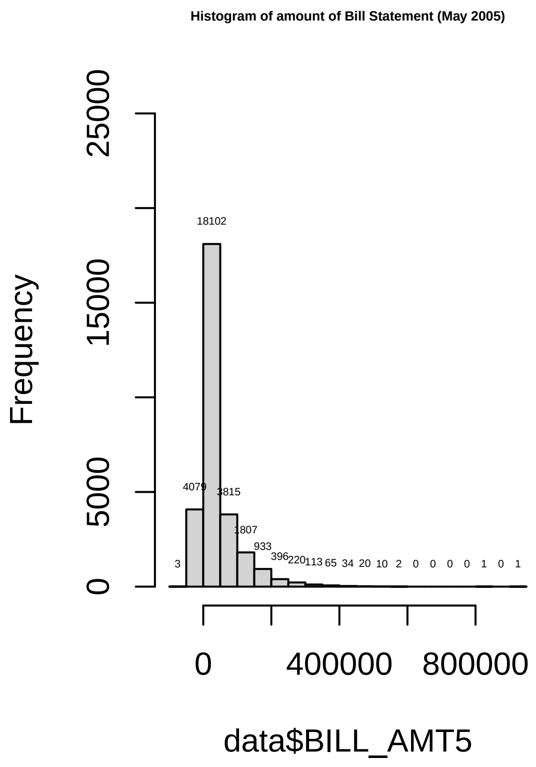


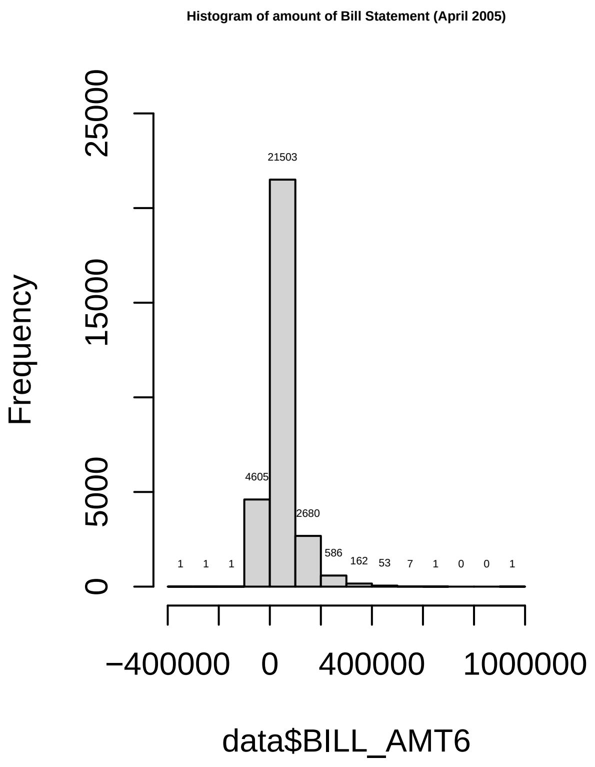
```
par(mfrow=c(1,2))
c = hist(data$BILL_AMT3, main="Histogram of amount of Bill Statement (July 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = c$mids, y = c$counts, labels = c$counts, cex = 0.35, pos = 3)

d = hist(data$BILL_AMT4, main="Histogram of amount of Bill Statement (June 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = d$mids, y = d$counts, labels = d$counts, cex = 0.35, pos = 3)
```





```
par(mfrow=c(1,2))
e = hist(data$BILL_AMT5, main="Histogram of amount of Bill Statement (May 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = e$mids, y = e$counts, labels = e$counts, cex = 0.35, pos = 3)
f = hist(data$BILL_AMT6, main="Histogram of amount of Bill Statement (April 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,25000))
text(x = f$mids, y = f$counts, labels = f$counts, cex = 0.35, pos = 3)
```

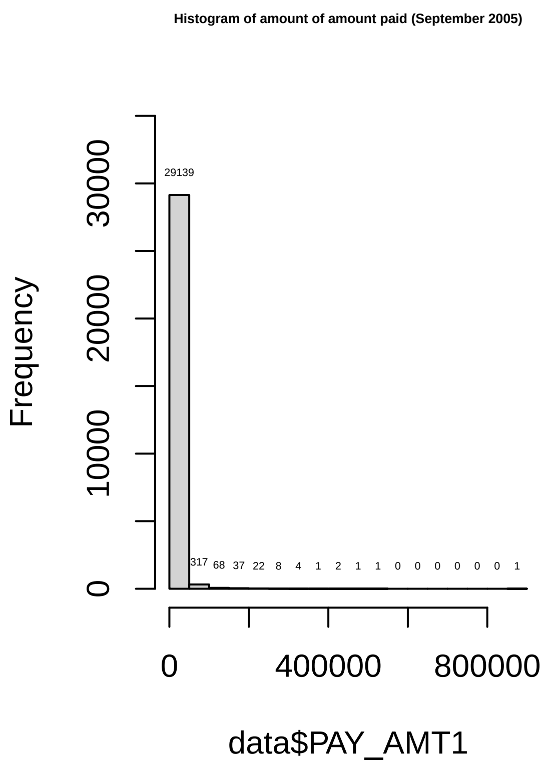


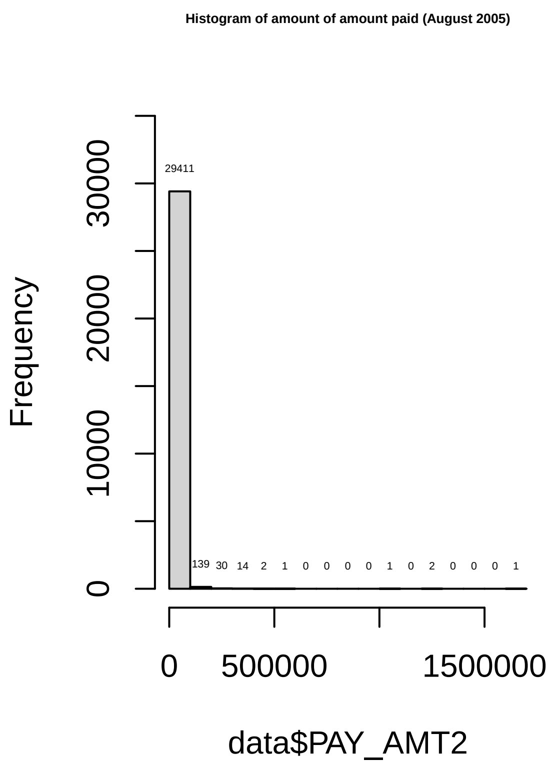


Looking at the distribution of BILL_AMTN, the perceived outliers do not stray away from the observed distribution as seen from the histograms.

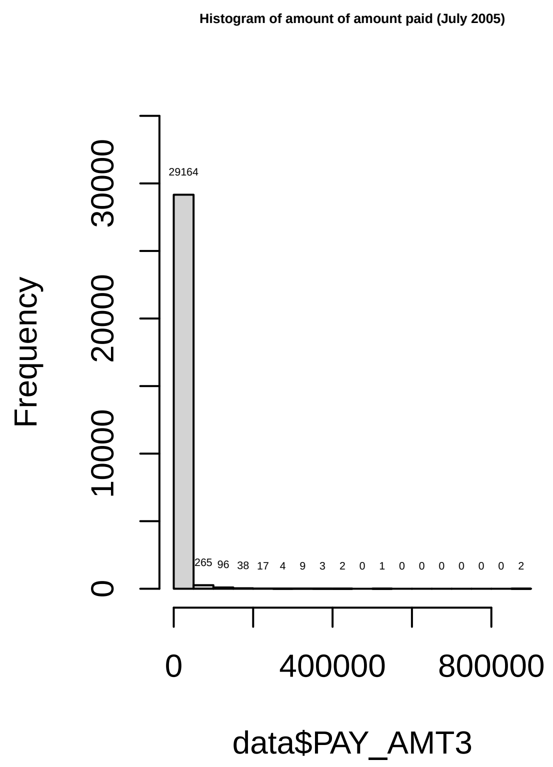
Hence, we decided not to remove any data points of BILL_AMTN as they are plausible values of the amount of the amount of bill statement (NT dollar).

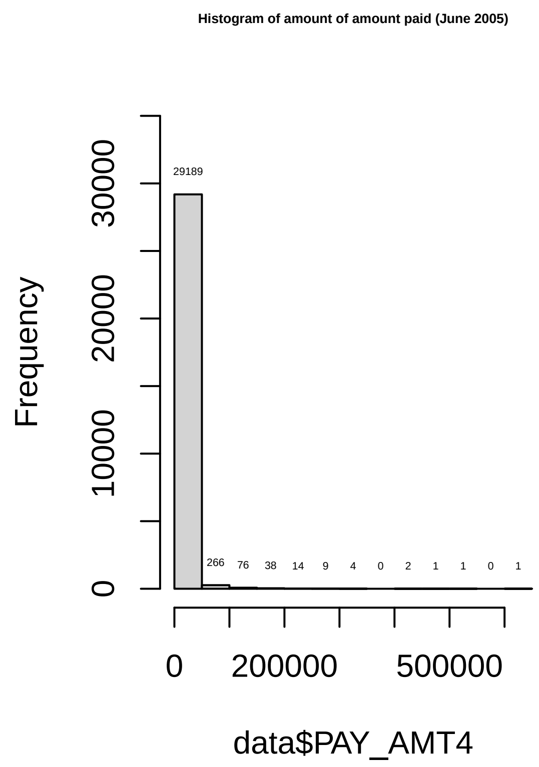
```r
#Histograms of Amount of Previous Payment
par(mfrow=c(1,2))
a = hist(data$PAY_AMT1, main="Histogram of amount of amount paid (September 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = a$mids, y = a$counts, labels = a$counts, cex = 0.35, pos = 3)
b = hist(data$PAY_AMT2, main="Histogram of amount of amount paid (August 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = b$mids, y = b$counts, labels = b$counts, cex = 0.35, pos = 3)
```



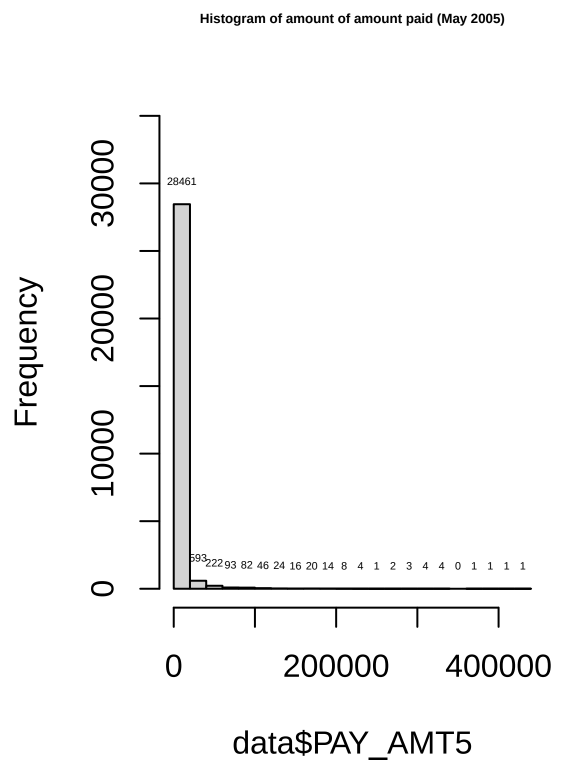


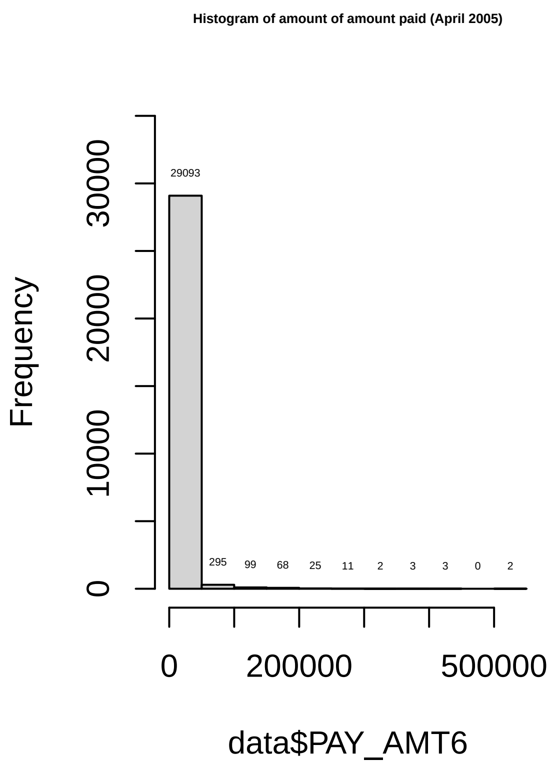
```r
par(mfrow=c(1,2))
c = hist(data$PAY_AMT3, main="Histogram of amount of amount paid (July 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = c$mids, y = c$counts, labels = c$counts, cex = 0.35, pos = 3)
d = hist(data$PAY_AMT4, main="Histogram of amount of amount paid (June 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = d$mids, y = d$counts, labels = d$counts, cex = 0.35, pos = 3)
```





```
par(mfrow=c(1,2))
e = hist(data$PAY_AMT5, main="Histogram of amount of amount paid (May 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = e$mids, y = e$counts, labels = e$counts, cex = 0.35, pos = 3)
f = hist(data$PAY_AMT6, main="Histogram of amount of amount paid (April 2005)",
         cex.main = 0.4, breaks = "Sturges", labels=F, ylim = c(0,35000))
text(x = f$mids, y = f$counts, labels = f$counts, cex = 0.35, pos = 3)
```

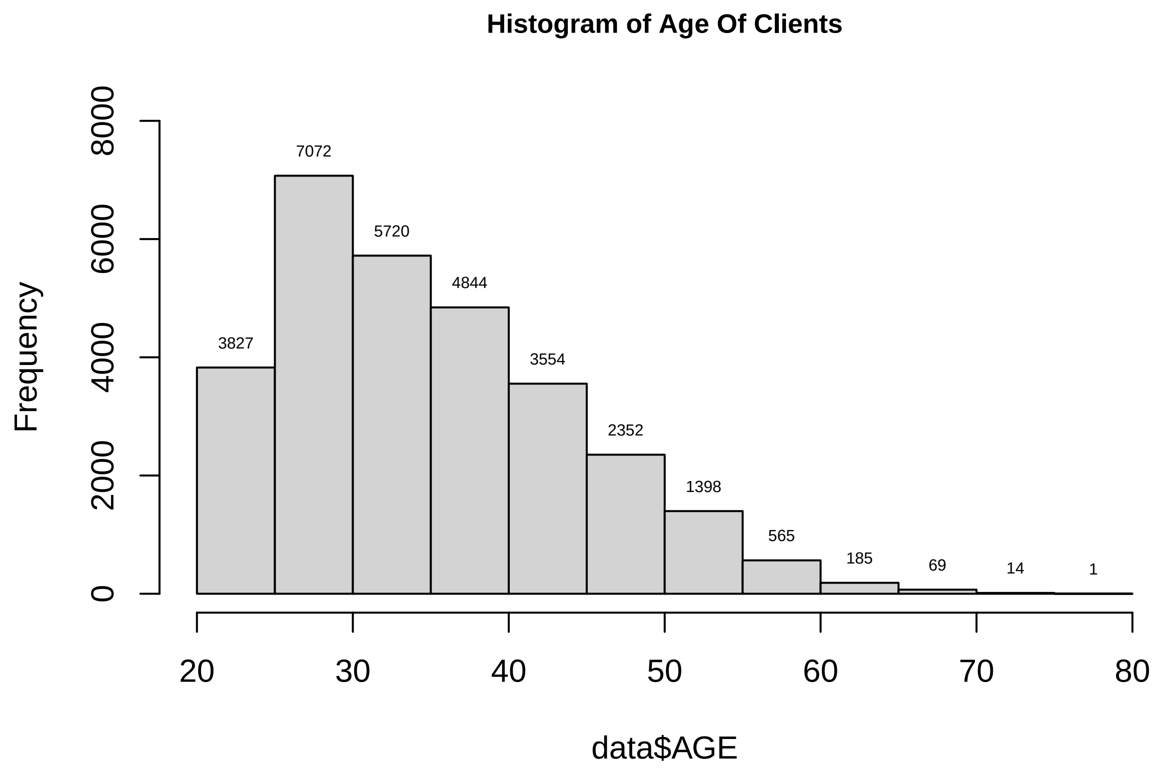




Looking at the distribution of PAY_AMTN, the perceived outliers do not stray away from the observed distribution as seen from the histograms.

Hence, we decided not to remove any data points of PAY_AMTN as they are plausible values of the amount of the amount of previous payment (NT dollar).

```r
#Histogram of age
f = hist(data$AGE, main="Histogram of Age Of Clients", cex.main = 0.8,
         breaks = "Sturges", labels=F, ylim = c(0,8000))
text(x = f$mids, y = f$counts, labels = f$counts, cex = 0.5, pos = 3)
```



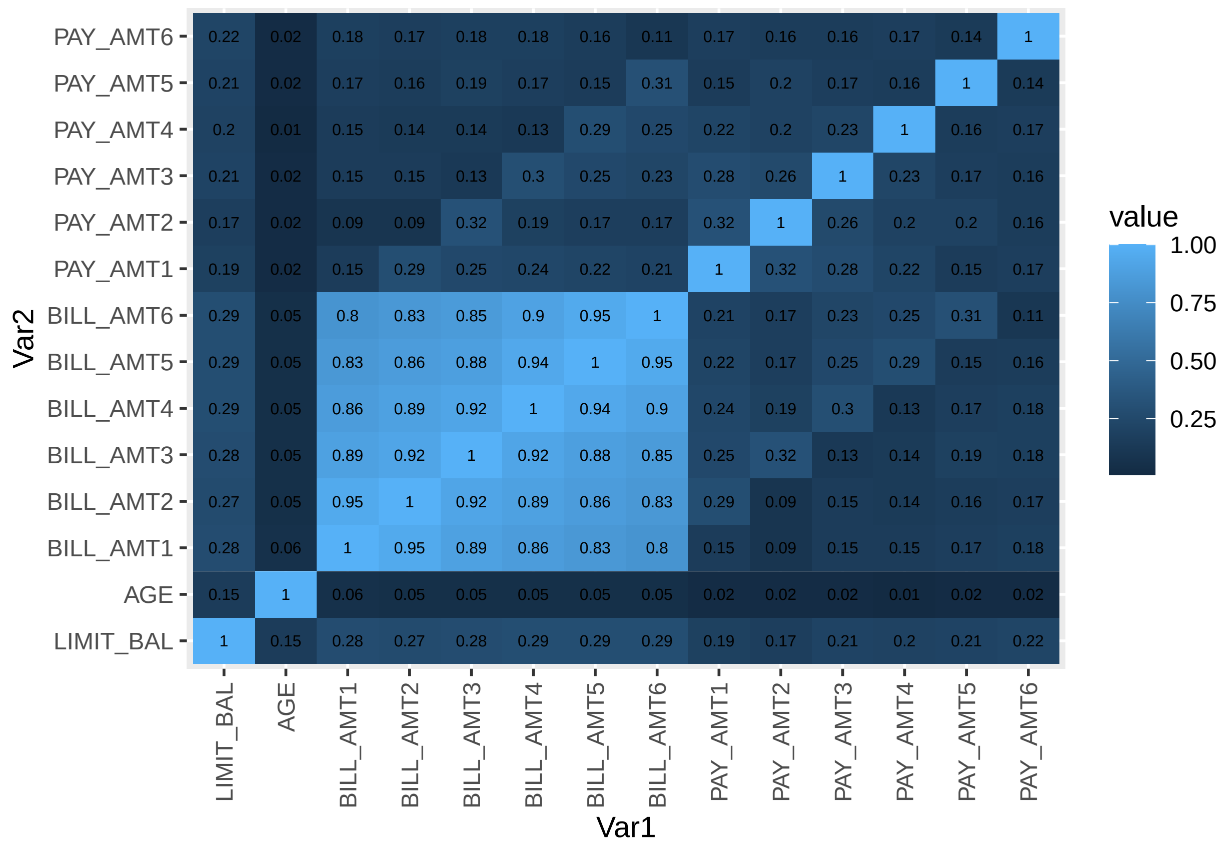
No outliers detected as the range of values in as shown in the histograms of AGE are plausible values of the amount of age in years

d. **Correlation Between Features**

One factor that could affect machine learning classification performances is correlation between features. This is because if features are strongly correlated to each other, classification algorithms which assume that the features are all independent may result in a poor classification performance. Reducing the number of dimensions of feature vectors could lead to better classification performances.

```
#splitting dataset into training set(train.data) & test set(test.data)

set.seed(1234)
n = length(data$ID)
index <- 1:nrow(data)
testindex <- sample(index, trunc(n)/4)
test.data <- data[testindex,]
train.data <- data[-testindex,]

#using train data, taking out default_payment_next_month and the
#categorical feature attributes
#(remove ID, SEX, MARRIAGE, PAY_N, default_payment_next_month)

train.data_2 <- cor(train.data[,-c(1,3,4,5,7,8,9,10,11,12,25)])
corr_mat <- round(train.data_2,2)
melted_corr_mat <- reshape2::melt(corr_mat)
```

```r
ggplot(data = melted_corr_mat, aes(x=Var1, y=Var2,
                                   fill=value)) +
geom_tile() +
geom_text(aes(Var2, Var1, label = value),
          color = "black", size = 2) +
scale_x_discrete(guide = guide_axis(angle = 90))
```



Features that have high correlation to each other ($\rho > 0.9$):

BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6

5 out of 6 of this variables will later be removed in the data pre-processing phase as they likely will not be useful in predicting the target values of whether payment is defaulted (default_payment_next_month) as they may impact the performance of the machine learning models.

## 3. Data Pre-Processing

The data pre-processing section involves transforming raw data into an understandable and usable format.

### a. One-Hot Encoding

Categorical features must be changed in the data pre-processing phase since machine learning models require numeric input values. One-hot encoding will be used for categorical data with no ordinal relationship in our data set. One-hot encoding creates new binary dummy variables for each class in every categorical feature. For example, categorical features such as EDUCATION has been split into dummy variables such as EDUCATION_1 to EDUCATION_4.

```
#one hot encoding for train data
train.data_without_target = train.data[-c(25)]
train.data_without_target_encoded <- one_hot(as.data.table(train.data_without_target))
train_data_encoded = cbind(train.data_without_target_encoded, train.data[c(25)])

#one hot encoding for test data
test.data_without_target <- test.data[-c(25)]
test.data_without_target_encoded <- one_hot(as.data.table(test.data_without_target))
test_data_encoded = cbind(test.data_without_target_encoded, test.data[c(25)])
```

### b. Feature Scaling

It's a common case that in most data sets, features are not on the same scale. Machine learning models are, however, largely Euclidean distant-based. Without feature scaling, features with large units will dominate those with small units when it comes to calculation of the Euclidean distance, hence features with small units may be neglected. To prevent this from happening, we need to scale our numerical features.

In this project, we will be using be using caret library to pre-process and scale the data through Min-Max Scaling method to scale the data values to a range of 0 to 1.

```
#scaling training data set
raw_train_data <- train_data_encoded
processed_train_data <- train_data_encoded

df <- preProcess(processed_train_data, method=c("range"))
processed_train_data <- predict(df, as.data.frame(processed_train_data))
#processed_train_data

#scaling test data set
raw_test_data <- test_data_encoded
processed_test_data <- test_data_encoded

df <- preProcess(processed_test_data, method=c("range"))
processed_test_data <- predict(df, as.data.frame(processed_test_data))
#processed_test_data
```

### c. Train-Test Split

Each dataset is split into 2 parts, the training set and test set, in the proportion 3:1. To reduce any leaking of information into the testset, we will be doing all feature selection techniques on the trainset only. This includes testing for high correlation and implementing oversampling and undersampling techniques.

## 4. Feature Selection

Feature selection is the process where features which contribute most to the prediction variable are selected.

## a. Removal of highly correlated variables

As mentioned above, features that have high correlation to each other will be removed as they likely will not be useful in predicting the target values of whether payment is defaulted (default_payment_next_month) as they may impact the performance of the machine learning models. Some classification models such as the Naive Bayes Classification are also reliant on the assumption that features used to predict the target variable are linearly independent.

```r
processed_train_data.corr <- select(processed_train_data,
                                    -c('ID','BILL_AMT2','BILL_AMT3',
                                       'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
processed_test_data.corr <- select(processed_test_data,
                                   -c('ID','BILL_AMT2','BILL_AMT3',
                                      'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
setnames(processed_test_data.corr,
         old = c('PAY_1_-1','PAY_2_-1','PAY_3_-1','PAY_4_-1','PAY_5_-1',
                 'PAY_6_-1','PAY_1_-2','PAY_2_-2','PAY_3_-2','PAY_4_-2',
                 'PAY_5_-2','PAY_6_-2'),
         new = c('PAY_1_n1','PAY_2_n1','PAY_3_n1','PAY_4_n1','PAY_5_n1',
                 'PAY_6_n1','PAY_1_n2','PAY_2_n2','PAY_3_n2',
                 'PAY_4_n2','PAY_5_n2','PAY_6_n2'))
setnames(processed_train_data.corr, old = c('PAY_1_-1','PAY_2_-1','PAY_3_-1',
                                            'PAY_4_-1','PAY_5_-1','PAY_6_-1',
                                            'PAY_1_-2','PAY_2_-2','PAY_3_-2',
                                            'PAY_4_-2','PAY_5_-2','PAY_6_-2'),
         new = c('PAY_1_n1','PAY_2_n1','PAY_3_n1','PAY_4_n1','PAY_5_n1',
                 'PAY_6_n1','PAY_1_n2','PAY_2_n2','PAY_3_n2','PAY_4_n2',
                 'PAY_5_n2','PAY_6_n2'))


processed_train_data_rosepca.corr <- select(processed_train_data,
                                            -c('ID','BILL_AMT2','BILL_AMT3',
                                            'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
processed_test_data_rosepca.corr <- select(processed_test_data,
                                           -c('ID','BILL_AMT2','BILL_AMT3',
                                              'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
setnames(processed_test_data_rosepca.corr, old = c('PAY_1_-1','PAY_2_-1','PAY_3_-1',
                                                   'PAY_4_-1','PAY_5_-1','PAY_6_-1',
                                                   'PAY_1_-2','PAY_2_-2','PAY_3_-2',
                                                   'PAY_4_-2','PAY_5_-2','PAY_6_-2'),
         new = c('PAY_1_n1','PAY_2_n1','PAY_3_n1','PAY_4_n1','PAY_5_n1','PAY_6_n1',
                 'PAY_1_n2','PAY_2_n2','PAY_3_n2','PAY_4_n2','PAY_5_n2','PAY_6_n2'))
setnames(processed_train_data_rosepca.corr, old = c('PAY_1_-1','PAY_2_-1',
                                                    'PAY_3_-1','PAY_4_-1','PAY_5_-1',
                                                    'PAY_6_-1','PAY_1_-2','PAY_2_-2',
                                                    'PAY_3_-2','PAY_4_-2','PAY_5_-2',
                                                    'PAY_6_-2'),
         new = c('PAY_1_n1','PAY_2_n1','PAY_3_n1','PAY_4_n1',
                 'PAY_5_n1','PAY_6_n1','PAY_1_n2','PAY_2_n2',
                 'PAY_3_n2','PAY_4_n2','PAY_5_n2','PAY_6_n2'))
```

## b. Principal Component Analysis (PCA)

PCA is particularly useful for dimensionality reduction when the dimensions of the data are high. Since there is a large number of feature attributes in our dataset, to reduce the number of features used in the model, we can also use PCA to find a set of features ("Principal Components") that explain the most variance in data. Since PCA is designed to minimize variance (squared deviations), we applied PCA to only the continuous variables in our dataset as the concept of squared deviations break down when considering binary variables.

We can then choose a subset of the PCs, e.g. the first k PCs, perhaps based on the cumulative variance explained. (e.g, the first 7 PCs may account for 99% of the variance). For our project, we chose PCA as a feature selection because we do not require interpretability of the selected features.

```
set.seed(1234)
pca1 <- prcomp(select(processed_train_data.corr,
                      c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                        'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
               center=T)

#We did not include scale = True since the variables are already scaled.
summary(pca1)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.1626 0.1295 0.06199 0.03624 0.03199 0.02467 0.01992
## Proportion of Variance 0.5201 0.3297 0.07558 0.02583 0.02012 0.01197 0.00780
## Cumulative Proportion  0.5201 0.8498 0.92541 0.95124 0.97136 0.98333 0.99113
##                            PC8     PC9
## Standard deviation     0.01688 0.01288
## Proportion of Variance 0.00561 0.00326
## Cumulative Proportion  0.99674 1.00000
```

From the summary of the PCA done, Principal components 1 to 7 together explain 99.113% of the variance in the data. Thus, we will use PCA 1 to 7 in our machine learning models.

Using the Principal Components derived in our PCA analysis from our trainset, we would be projecting it to our test set. This way the points in both train and test sets end up in the same dimension space, and we would not be using any knowledge about our test set during training, hence preventing leaking.

```
processed_train_data.corr$PC1 = pca1$x[,"PC1"]
processed_train_data.corr$PC2 = pca1$x[,"PC2"]
processed_train_data.corr$PC3 = pca1$x[,"PC3"]
processed_train_data.corr$PC4 = pca1$x[,"PC4"]
processed_train_data.corr$PC5 = pca1$x[,"PC5"]
processed_train_data.corr$PC6 = pca1$x[,"PC6"]
processed_train_data.corr$PC7 = pca1$x[,"PC7"]
#mapping pca 1 to 7 weights to form pca 1 to 7 in test set

pca_test= predict(pca1,processed_test_data.corr)
processed_test_data.corr = cbind(processed_test_data.corr,pca_test[,-c(8,9)])
```

## c. Oversampling and Undersampling techniques

For the target values of whether payment is defaulted (default_payment_next_month) in training data set, data set is unbalanced as 22.31881% of clients has defaulted on payment, as compared to 77.68119% who did not default on payment, as seen in the summary table below.

```r
#check class distribution

processed_train_data %>%
    group_by(default_payment_next_month) %>%
    summarise("Percentage of Total" = 100*n()/nrow(processed_train_data ))
```

```
## # A tibble: 2 x 2
##   default_payment_next_month ‘Percentage of Total‘
##   <fct>                                       <dbl>
## 1 0                                            77.7
## 2 1                                            22.3
```

To prevent the training process from biasing towards a certain class over another, over-sampling and under-sampling techniques will be used.

Over-sampling generates more observations from the minority class to ensure the dataset is balanced. Under-sampling reduces the observations from the majority class to ensure the dataset is balanced.

Data generated from over-sampling is expected to have repeated observations and data generated from under-sampling is expected to be deprived of important information from the original data. This would result in inaccuracies in the resulting performance of the machine learning algorithms.

To prevent this from happening, the ROSE package helps to generate data synthetically.

However, it is noteworthy that synthetic samples do not formally belong to the target distribution hence should not be used as a test sample.

```r
#1 preprocessing
train_data.corr <- select(train.data,
                          -c('ID','BILL_AMT2','BILL_AMT3',
                             'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
test_data.corr <- select(test.data,
                         -c('ID','BILL_AMT2','BILL_AMT3',
                            'BILL_AMT4','BILL_AMT5','BILL_AMT6'))
data.rose <- ROSE(default_payment_next_month ~ .,
                  data = train_data.corr,
                  seed = 1)$data
rose_train.data_without_target.corr = data.rose[-c(19)]
rose_train.data_without_target_encoded.corr <- one_hot(as.data.table(
  rose_train.data_without_target.corr))

rose_train_data_encoded.corr = cbind(rose_train.data_without_target_encoded.corr,
                                     data.rose[c(19)])

rose_processed_train_data.corr <- rose_train_data_encoded.corr

df <- preProcess(rose_processed_train_data.corr, method=c("range"))

rose_processed_train_data.corr <- predict(df,
                                          as.data.frame(rose_processed_train_data.corr))

setnames(rose_processed_train_data.corr,
         old = c('PAY_1_-1','PAY_2_-1','PAY_3_-1','PAY_4_-1',
                 'PAY_5_-1','PAY_6_-1','PAY_1_-2','PAY_2_-2',
                 'PAY_3_-2','PAY_4_-2','PAY_5_-2','PAY_6_-2'),
         new = c('PAY_1_n1','PAY_2_n1','PAY_3_n1','PAY_4_n1',
                 'PAY_5_n1','PAY_6_n1','PAY_1_n2','PAY_2_n2',
                 'PAY_3_n2','PAY_4_n2','PAY_5_n2','PAY_6_n2'))
```

PCA feature selection was applied before oversampling and undersampling. By doing so, we are able to project the data on the axes/plane where the variance of the data is the highest, hence leveraging the maximisation of within data variability. Running oversampling and undersampling on when the variability within the data is high creates synthetic samples closer to the actual data.

```
#4 preprocessing
set.seed(1234)
rose_processed_train_data_pca.corr <- processed_train_data_rosepca.corr
pca2 <- prcomp(rose_processed_train_data_pca.corr[,c(1,11,76:82)], center=T)

#We did not include scale = True since the variables are already scaled.
summary(pca2)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.1626 0.1295 0.06199 0.03624 0.03199 0.02467 0.01992
## Proportion of Variance 0.5201 0.3297 0.07558 0.02583 0.02012 0.01197 0.00780
## Cumulative Proportion  0.5201 0.8498 0.92541 0.95124 0.97136 0.98333 0.99113
##                            PC8     PC9
## Standard deviation     0.01688 0.01288
## Proportion of Variance 0.00561 0.00326
## Cumulative Proportion  0.99674 1.00000
```

```
rose_processed_train_data_pca.corr$PC1 = pca2$x[,"PC1"]
rose_processed_train_data_pca.corr$PC2 = pca2$x[,"PC2"]
rose_processed_train_data_pca.corr$PC3 = pca2$x[,"PC3"]
rose_processed_train_data_pca.corr$PC4 = pca2$x[,"PC4"]
rose_processed_train_data_pca.corr$PC5 = pca2$x[,"PC5"]
rose_processed_train_data_pca.corr$PC6 = pca2$x[,"PC6"]
rose_processed_train_data_pca.corr$PC7 = pca2$x[,"PC7"]
#mapping pca 1 to 7 weights to form pca 1 to 7 in test set

pca_test2= predict(pca2,processed_test_data_rosepca.corr)

rose_processed_test_data_pca.corr = cbind(processed_test_data_rosepca.corr,pca_test2[,-c(8,9)])

rose_processed_train_data_pca.corr<- ROSE(default_payment_next_month ~ .,
                  data = rose_processed_train_data_pca.corr,
                  seed = 1)$data
```

## 5. Model Selection

We will now run a variety of machine learning models to test and evaluate which one performs the best in predicting defaulters among credit card customers.

The models we will be using are:

1) Logistic Regression

2) Support Vector Machines (SVM)
3) Naive Bayes Classification
4) Decision Tree (Conditional Inference Tree)
5) Random Forest
6) Neural Networks

We will be training each model on datasets with the different preprocessing techniques using the train set. The different datasets are:

1) Original trainset after removing highly correlated variables
2) Original trainset after removing highly correlated variables + Oversampling and undersampling techniques
3) PCA after removing highly correlated variables
4) PCA after removing highly correlated variables + Oversampling and undersampling techniques

After which, we will be evaluating each respective model using the test set.

## 1. Logistic Regression

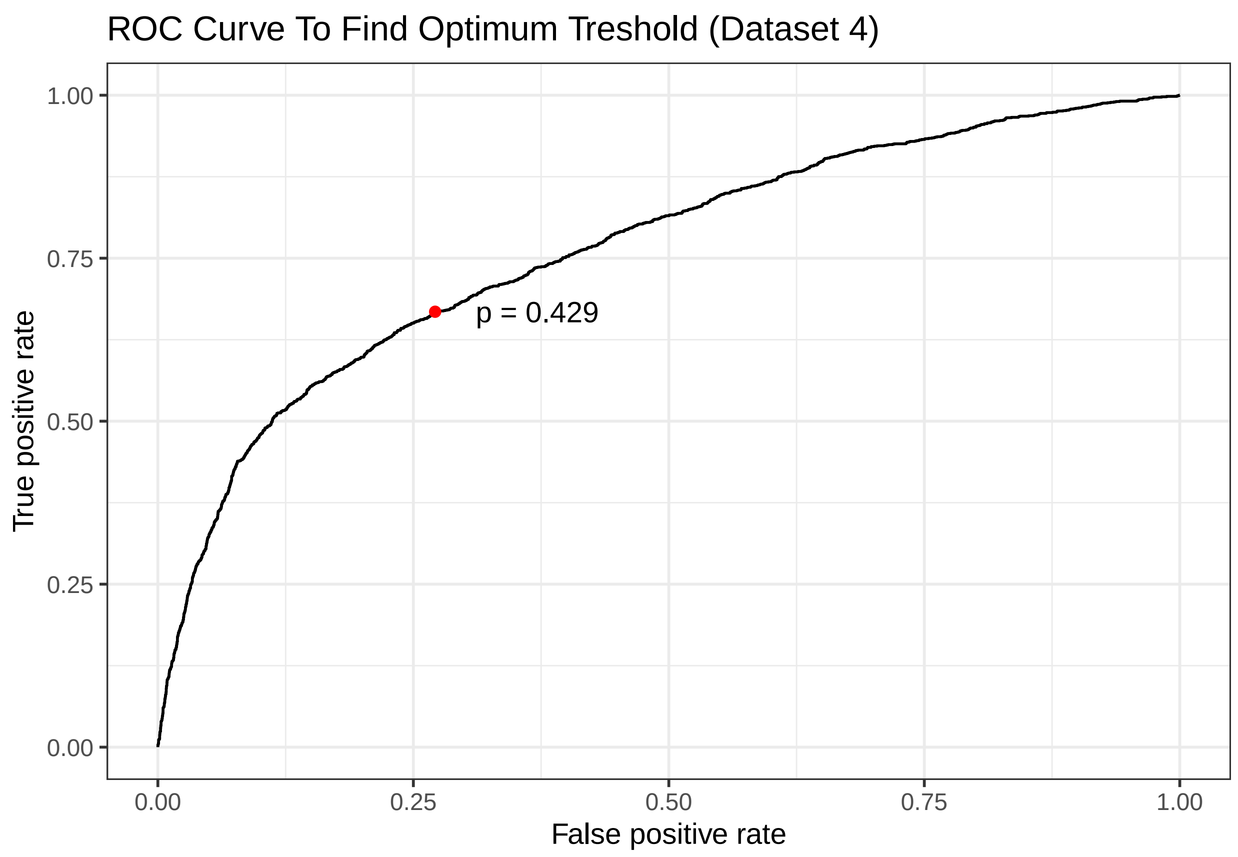
Logistic regression is a classification algorithm used to find the probability of event success and failure. Since outcome variable default__payment__next__month: target values of whether payment is defaulted (Yes = 1, No = 0) is binary, logistic regression is used.

To find optimum threshold for the logistic regression classification, we have to decide how much True Positive Rate(TPR) and False Positive Rate(FPR) we wish to have. If we were to increase the TPR, FPR will likely to increase as well. For this set of data, we wish to minimise the number of false negatives as far as possible, hence we choose a threshold that increases TPR and reduces FPR.

To find the probability threshold to give us the best performance, we looked into the point on the ROC curve that will maximise TPR while minimising FPR at the same time. For instance, we plotted the ROC curve for test results of dataset 4 (PCA after removing highly correlated variables + Oversampling and undersampling techniques), which gave us an optimum threshold of 0.429, as seen below.

```r
set.seed(1234)
#1
#processed_train_data_rosepca.corr, processed_test_data_rosepca.corr

logistic <- glm(default_payment_next_month ~ .,
                data = processed_train_data_rosepca.corr,
                family = "binomial")
#summary
#summary(logistic)

#predict test data based on model
predict_reg <- predict(logistic,
                 processed_test_data_rosepca.corr,
                 type = "response")

# prediction(predict_reg,processed_test_data_rosepca.corr$default_payment_next_month) %>%
#   performance(measure = "tpr", x.measure = "fpr") -> result
#
# plotdata <- data.frame(x = result@x.values[[1]],
#                         y = result@y.values[[1]],
#                         p = result@alpha.values[[1]])
#
# p <- ggplot(data = plotdata) +
#   geom_path(aes(x = x, y = y)) +
#   xlab(result@x.name) +
#   ylab(result@y.name) +
#   theme_bw()
#
# dist_vec <- plotdata$x^2 + (1 - plotdata$y)^2
# opt_pos <- which.min(dist_vec)
# round(plotdata[opt_pos, ]$p, 3)

prediction_data <- ifelse(predict_reg > 0.5, 1, 0)

log.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,prediction_data)
log.cm1 <- table(processed_test_data_rosepca.corr$default_payment_next_month,prediction_data)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc = auc(processed_test_data_rosepca.corr$default_payment_next_month, predict_reg, quiet=TRUE)

a = cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

#2
#rose_processed_train_data.corr rose_processed_test_data.corr

logistic <- glm(default_payment_next_month ~ .,
                data = rose_processed_train_data.corr,
                family = "binomial")
#summary
#summary(logistic)

#predict test data based on model
predict_reg <- predict(logistic,
                 processed_test_data_rosepca.corr,
                 type = "response")

# prediction(predict_reg,processed_test_data_rosepca.corr$default_payment_next_month) %>%
#   performance(measure = "tpr", x.measure = "fpr") -> result
#
# plotdata <- data.frame(x = result@x.values[[1]],
#                         y = result@y.values[[1]],
#                         p = result@alpha.values[[1]])
#
# p <- ggplot(data = plotdata) +
#   geom_path(aes(x = x, y = y)) +
#   xlab(result@x.name) +
#   ylab(result@y.name) +
#   theme_bw()
#
# dist_vec <- plotdata$x^2 + (1 - plotdata$y)^2
# opt_pos <- which.min(dist_vec)
# round(plotdata[opt_pos, ]$p, 3)

prediction_data <- ifelse(predict_reg > 0.5, 1, 0)

log.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,prediction_data)
log.cm2 <- table(processed_test_data_rosepca.corr$default_payment_next_month,prediction_data)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc = auc(processed_test_data_rosepca.corr$default_payment_next_month, predict_reg, quiet=TRUE)

b = cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


#3
#processed_train_data.corr, processed_train_data.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')

logistic <- glm(default_payment_next_month ~ .,
                data = select(processed_train_data.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1',
                                 'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                 'PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                family = "binomial")
#summary
#summary(logistic)

#predict test data based on model
predict_reg <- predict(logistic,
                 processed_test_data.corr,
                 type = "response")

# prediction(predict_reg,processed_test_data.corr$default_payment_next_month) %>%
#   performance(measure = "tpr", x.measure = "fpr") -> result
#
# plotdata <- data.frame(x = result@x.values[[1]],
#                        y = result@y.values[[1]],
#                        p = result@alpha.values[[1]])
#
# p <- ggplot(data = plotdata) +
#   geom_path(aes(x = x, y = y)) +
#   xlab(result@x.name) +
#   ylab(result@y.name) +
#   theme_bw()
#
# dist_vec <- plotdata$x^2 + (1 - plotdata$y)^2
# opt_pos <- which.min(dist_vec)
# #round(plotdata[opt_pos, ]$p, 3)

prediction_data <- ifelse(predict_reg > 0.5, 1, 0)
```

```r
log.cm <- table(processed_test_data.corr$default_payment_next_month,prediction_data)
log.cm3 <- table(processed_test_data.corr$default_payment_next_month,prediction_data)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc = auc(processed_test_data.corr$default_payment_next_month, predict_reg, quiet=TRUE)

c = cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


#4
#rose_processed_train_data_pca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')
#rose_processed_test_data_pca.corr

logistic <- glm(default_payment_next_month ~ .,
                data = select(rose_processed_train_data_pca.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1',
                                 'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                 'PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                family = "binomial")
#summary
#summary(logistic)

#predict test data based on model
predict_reg <- predict(logistic,
                 rose_processed_test_data_pca.corr,
                 type = "response")

prediction_data <- ifelse(predict_reg > 0.429 , 1, 0)


prediction(predict_reg,rose_processed_test_data_pca.corr$default_payment_next_month) %>%
  performance(measure = "tpr", x.measure = "fpr") -> result

plotdata <- data.frame(x = result@x.values[[1]],
                       y = result@y.values[[1]],
                       p = result@alpha.values[[1]])

p <- ggplot(data = plotdata) +
  geom_path(aes(x = x, y = y)) +
  xlab(result@x.name) +
  ylab(result@y.name) +
  theme_bw() +
  ggtitle("ROC Curve To Find Optimum Treshold (Dataset 4)")

dist_vec <- plotdata$x^2 + (1 - plotdata$y)^2
opt_pos <- which.min(dist_vec)

p +
  geom_point(data = plotdata[opt_pos, ],
             aes(x = x, y = y), col = "red") +
  annotate("text",
           x = plotdata[opt_pos, ]$x + 0.1,
           y = plotdata[opt_pos, ]$y,
           label = paste("p =", round(plotdata[opt_pos, ]$p, 3)))
```



```
log.cm <- table(rose_processed_test_data_pca.corr$default_payment_next_month,prediction_data)
log.cm4 <- table(rose_processed_test_data_pca.corr$default_payment_next_month,prediction_data)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc = auc(rose_processed_test_data_pca.corr$default_payment_next_month, predict_reg, quiet=TRUE)

d = cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

final_log<-rbind(a,b,c,d)
rownames(final_log) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_log
```

```
##                              precision    recall  accuracy        f1
## corr                         0.8083692 0.7189565 0.6491892 0.7610457
## corr+under/oversampling      0.8716850 0.7431304 0.7154054 0.8022906
## corr+PCA                     0.8374452 0.6316522 0.6185135 0.7201348
## corr+under/oversampling+PCA 0.8842661 0.7281739 0.7147297 0.7986648
##                              avg_accuracy       auc
## corr                            0.5625086 0.5625086
## corr+under/oversampling         0.6809592 0.6809592
## corr+PCA                        0.6021897 0.6021897
## corr+under/oversampling+PCA     0.6980264 0.7632454
```

```
tm<-cbind(log.cm1,log.cm2,log.cm3,log.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA      "=2,"corr+under/ove
  kable_styling(latex_options =c("repeat_header"))
```

Table 1: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 4134 | 1616 | 4273 | 1477 | 3632 | 2118 | 4187 | 1563 |
| 1 | 980 | 670 | 629 | 1021 | 705 | 945 | 548 | 1102 |

## 2. Support Vector Model (SVM)

Support Vector Machines is a supervised machine learning algorithm which uses classification algorithms for two-group classification problems.

SVM is an appropriate model to use in this report since we do not need to interpret the model, and merely need the results of its classification. Furthermore, SVM works well with high dimensional data, and is able to solve complex problems conveniently using the kernel functions.

The type of *svm* used is C-classification. For a binary classification as in this case where there are only 2 classes $defaulter$ and $non-defaulter$, the $C$ parameter comes into the picture when the dataset is linearly non separable. It accounts for the slack variables that are introduced into the linear constraints so that the model becomes feasible. $C > 0$ for all $C$.

Cost (C) is a regularisation parameter, it tells the optimizer what it should optimise more, the distance between data inputs to the hyperplane or the penalty of mis-classification. A large cost would tell the optimiser to minimise misclassification since C is the scalar weight of the penalty of mis-classification.

We iterated different values of C into the svm models to find the C that gives us the best classification performance. We then used the optimal value of C to train each respective dataset.

For kernel types, we iterated through different kernel types and found that the linear kernel gives the best classification performance on the dataset. We then used the optimal kernel type to train each respective dataset.

```
set.seed(1234)
#1 processed_train_data_rosepca.corr, processed_test_data_rosepca.corr
svm.crossmodel <- svm(default_payment_next_month ~ . ,
                      data=processed_train_data_rosepca.corr,
                      type="C-classification",
                      kernel="linear",
                      cost=1)

#svm.tuned <-  tune(svm,default_payment_next_month ~ . ,
#data=processed_train_data_rosepca.corr, type="C-classification", kernel="linear",ranges=c(1,2^2,2^4,2^

results_test_cross <- predict(svm.crossmodel, processed_test_data_rosepca.corr[,-83])

log.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test_cross)
log.cm1 <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test_cross)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)

auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(results_test_cross), quiet=TRUE)

auc = auc$auc

a = cbind(precision,recall,accuracy,f1,avg_accuracy,auc)


#2 rose_processed_train_data.corr processed_test_data_rosepca.corr

svm.crossmodel <- svm(default_payment_next_month ~ . ,
                      data=rose_processed_train_data.corr,
                      type="C-classification",
                      kernel="linear",
                      cost=1)

#svm.tuned <-  tune(svm,default_payment_next_month ~ . ,
#data=rose_processed_train_data.corr, type="C-classification", kernel="linear",ranges=c(1,2^2,2^4,2^6,2

results_test_cross <- predict(svm.crossmodel, processed_test_data_rosepca.corr[,-83])

log.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test_cross)
log.cm2 <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test_cross)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(results_test_cross), quiet=TRUE)

auc = auc$auc

b = cbind(precision,recall,accuracy,f1,avg_accuracy,auc)


#3
#(processed_train_data.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')
#processed_test_data.corr
```

```r
svm.crossmodel <- svm(default_payment_next_month ~ . ,
                      data= select(processed_train_data.corr,
                                   -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                      'PAY_AMT2','PAY_AMT3','PAY_AMT4',
                                      'PAY_AMT5','PAY_AMT6')),
                      type="C-classification",
                      kernel="linear",
                      cost=1)

#svm.tuned <-  tune(svm,default_payment_next_month ~ . ,
#data=select(processed_train_data.corr, -c('LIMIT_BAL','AGE','BILL_AMT1',
#'PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#'type="C-classification", kernel="linear",ranges=c(1,2^2,2^4,2^6,2^8,2^10))

results_test_cross <- predict(svm.crossmodel, processed_test_data.corr[,-83])

log.cm <- table(processed_test_data.corr$default_payment_next_month,results_test_cross)
log.cm3 <- table(processed_test_data.corr$default_payment_next_month,results_test_cross)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data.corr$default_payment_next_month,
           predictor = as.numeric(results_test_cross), quiet=TRUE)

auc = auc$auc

c = cbind(precision,recall,accuracy,f1,avg_accuracy,auc)


#4
#rose_processed_train_data_pca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')
#rose_processed_test_data_pca.corr

svm.crossmodel <- svm(default_payment_next_month ~ . ,
                      data=select(rose_processed_train_data_pca.corr,
                                  -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                     'PAY_AMT2','PAY_AMT3','PAY_AMT4',
                                     'PAY_AMT5','PAY_AMT6')),
                      type="C-classification",
                      kernel="linear",
                      cost=1)

#svm.tuned <-  tune(svm,default_payment_next_month ~ . ,
#data=select(rose_processed_train_data_pca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#type="C-classification", kernel="linear",ranges=c(1,2^2,2^4,2^6,2^8,2^10))

results_test_cross <- predict(svm.crossmodel, rose_processed_test_data_pca.corr[,-83])

log.cm <- table(rose_processed_test_data_pca.corr$default_payment_next_month,results_test_cross)
log.cm4 <- table(rose_processed_test_data_pca.corr$default_payment_next_month,results_test_cross)
precision <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[2])
recall <- as.matrix(log.cm)[1] / (as.matrix(log.cm)[1] + as.matrix(log.cm)[3])
accuracy <- (as.matrix(log.cm)[1] + as.matrix(log.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(log.cm)[4] /(as.matrix(log.cm)[2] + as.matrix(log.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = rose_processed_test_data_pca.corr$default_payment_next_month,
           predictor = as.numeric(results_test_cross), quiet=TRUE)

auc = auc$auc

d = cbind(precision,recall,accuracy,f1,avg_accuracy,auc)

final_svm<-rbind(a,b,c,d)
rownames(final_svm) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_svm
```

```
##                             precision    recall  accuracy        f1
## corr                        0.8314776 0.9610435 0.8183784 0.8915779
## corr+under/oversampling     0.8597721 0.8530435 0.7777027 0.8563946
## corr+PCA                    0.8314776 0.9610435 0.8183784 0.8915779
## corr+under/oversampling+PCA 0.8583834 0.8939130 0.8029730 0.8757880
##                             avg_accuracy       auc
## corr                           0.6411278 0.6411278
## corr+under/oversampling        0.6840975 0.6840975
## corr+PCA                       0.6411278 0.6411278
## corr+under/oversampling+PCA    0.6899868 0.6899868
```

```r
tm<-cbind(log.cm1,log.cm2,log.cm3,log.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA     "=2,
                     "corr+under/oversampling+PC"=2))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 2: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5526 | 224 | 4905 | 845 | 5526 | 224 | 5140 | 610 |
| 1 | 1120 | 530 | 800 | 850 | 1120 | 530 | 848 | 802 |

## 3. Naive Bayes Classification

This is a supervised machine learning classification technique that works off the Bayes' Theorem, which works on the principle of conditional probaility and assumes that predictors are independent of each other. Naives Bayes Classification is fast for making real-time predictions and is observed to be effective for large datasets.

However, if categorical variable has a category (in test data set), which was not observed in training data set, then model will assign its probability to equal 0.

To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

We chosen Naive Bayes classifier as one of our classifying algorithms as it is not sensitive to irrelevant features. Although we have conducted feature selection and dimensionality reduction to reduce irrelevant and redundant features, there may still be correlation between features which might affect classification performance for other unsupervised learning algorithms. Naive Bayes classifier therefore could be a suitable classifier to overcome the issue of irrelevant features.

However Naive Bayes has a limited parameter set, hence hyperparameter tuning is not the most valid method to improve accuracy. To increase the performance of our Naive Bayes classifier, it is important to handle missing data values and remove correlated features as we might double count features, overestimating the importance of the feature, degrading the performance of the Naive Bayes classifer.

We have concluded in our data pre-processing phase that there are no missing values and have removed highly correlated features, therefore we have optimised our Naive Bayes classifer and our results are as shown below.

```
set.seed(1234)

nb.model <- naiveBayes(default_payment_next_month ~. , data = processed_train_data_rosepca.corr)
nb.pred <-  predict(nb.model, processed_test_data_rosepca.corr[,-83])
nb.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,nb.pred)
nb.cm1 <- table(processed_test_data_rosepca.corr$default_payment_next_month,nb.pred)

precision <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[2])
recall <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[3])
accuracy <- (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nb.cm)[4] /(as.matrix(nb.cm)[2] + as.matrix(nb.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(nb.pred), quiet=TRUE)
auc = auc$auc

a<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


nb.model <- naiveBayes(default_payment_next_month ~. , data = rose_processed_train_data.corr)
nb.pred <-  predict(nb.model, processed_test_data_rosepca.corr[,-83])
nb.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,nb.pred)
nb.cm2 <- table(processed_test_data_rosepca.corr$default_payment_next_month,nb.pred)
precision <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[2])
recall <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[3])
accuracy <- (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nb.cm)[4] /(as.matrix(nb.cm)[2] + as.matrix(nb.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(nb.pred), quiet=TRUE)
auc = auc$auc

b<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


nb.model <- naiveBayes(default_payment_next_month ~. ,
                       data = select(processed_train_data.corr,
                                     -c('LIMIT_BAL','AGE','BILL_AMT1',
                                        'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                        'PAY_AMT4','PAY_AMT5','PAY_AMT6')))
nb.pred <-  predict(nb.model, processed_test_data.corr[,-83])
nb.cm <- table(processed_test_data.corr$default_payment_next_month,nb.pred)
nb.cm3 <- table(processed_test_data.corr$default_payment_next_month,nb.pred)

precision <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[2])
recall <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[3])
accuracy <- (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nb.cm)[4] /(as.matrix(nb.cm)[2] + as.matrix(nb.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data.corr$default_payment_next_month,
           predictor = as.numeric(nb.pred), quiet=TRUE)
auc = auc$auc

c<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


nb.model <- naiveBayes(default_payment_next_month ~. ,
                       data = select(rose_processed_train_data_pca.corr,
                                     -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                        'PAY_AMT2','PAY_AMT3','PAY_AMT4',
                                        'PAY_AMT5','PAY_AMT6')))
nb.pred <-  predict(nb.model, rose_processed_test_data_pca.corr[,-83])
nb.cm <- table(rose_processed_test_data_pca.corr$default_payment_next_month,nb.pred)
nb.cm4 <- table(rose_processed_test_data_pca.corr$default_payment_next_month,nb.pred)

precision <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[2])
recall <- as.matrix(nb.cm)[1] / (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[3])
accuracy <- (as.matrix(nb.cm)[1] + as.matrix(nb.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nb.cm)[4] /(as.matrix(nb.cm)[2] + as.matrix(nb.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = rose_processed_test_data_pca.corr$default_payment_next_month,
           predictor = as.numeric(nb.pred), quiet=TRUE)
auc = auc$auc

d<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


final_nb<-rbind(a,b,c,d)
#c('High Correlated variables removed','High Correlated variables removed with undersampling/oversampli
#'High Correlated variables removed followed by PCA',
#''High Correlated variables removed with undersampling/oversampling techniques followed by PCA')
rownames(final_nb) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_nb
```

```
##                             precision    recall  accuracy        f1
## corr                        0.8016339 0.9726957 0.7917568 0.8789188
## corr+under/oversampling     0.7917604 0.9793043 0.7837838 0.8756026
## corr+PCA                    0.8022857 0.9766957 0.7948649 0.8809412
## corr+under/oversampling+PCA 0.7912472 0.9841739 0.7859459 0.8772283
##                             avg_accuracy       auc
## corr                           0.5669539 0.5669539
## corr+under/oversampling        0.5408643 0.5408643
## corr+PCA                        0.5689539 0.5689539
## corr+under/oversampling+PCA    0.5396627 0.5396627
```

```
tm<-cbind(nb.cm1,nb.cm2,nb.cm3,nb.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA     "=2,
                   "corr+under/oversampling+PC"=2))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 3: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5593 | 157 | 5631 | 119 | 5616 | 134 | 5659 | 91 |
| 1 | 1384 | 266 | 1481 | 169 | 1384 | 266 | 1493 | 157 |

## 4. Decision Tree (Conditional Inference Tree)

Decision Trees are a type of supervised machine learning algorithm where data is continuously split based on a certain parameter. The trees contain 2 entities - decision nodes and leaves. Decision nodes are where data is split and leaves are the decisions or final outcomes.

The decision tree performs recursive partitioning of data, thus breaking the data down into smaller subsets to eventually predict outcomes of the target variable.

The decision tree classification method works by testing the global null hypothesis of independence between any of the input variables and the target. It selects the input variable with strongest association to the target. This association is measured by finding the largest p-values with respect to a test for the partial null hypothesis between each input variable and the response variable.

Using this rank of features, the decision tree algorithm then checks through each feature and moves from feature to feature to finally predict the target class, forming a tree like model. In addition, we chose decision tree as it is compatible with categorical and continuous variables, and is also robust to outliers.

We iterated through different values of mincriterion to input into the decision tree models to find the value that gives us the best classification performance. (1 - mincriterion) is the significance level for each hypothesis test conducted that must be exceeded in order for a binary split to be implemented.

```
set.seed(1234)

is_binary <- function(x) {
  x0 <- na.omit(x)
  is.numeric(x) && length(unique(x0)) %in% 1:2 && all(x0 %in% 0:1)
}

#processed_train_data_rosepca.corr, processed_test_data_rosepca.corr
ok2 <- sapply(processed_train_data_rosepca.corr, is_binary)
processed_train_data_rosepca.corr2 <- replace(processed_train_data_rosepca.corr,
                                              ok2, lapply(processed_train_data_rosepca.corr[ok2],
                                                          factor, levels = 0:1))
ok2 <- sapply(processed_test_data_rosepca.corr, is_binary)
processed_test_data_rosepca.corr2 <- replace(processed_test_data_rosepca.corr, ok2,
                                              lapply(processed_test_data_rosepca.corr[ok2],
                                                     factor, levels = 0:1))

dtree.model2 <- ctree(default_payment_next_month~.,
                      data = select(processed_train_data_rosepca.corr2,
                                    -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                       'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5',
                                       'PAY_AMT6')),
                      controls = ctree_control(mincriterion=0.95))
results_test11 <- predict(dtree.model2,select(processed_test_data_rosepca.corr2,-
                                                c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                                  'PAY_AMT2','PAY_AMT3','PAY_AMT4',
                                                  'PAY_AMT5','PAY_AMT6'))[,-83])
rf.cm<-table(results_test11,processed_test_data_rosepca.corr2$default_payment_next_month)
rf.cm1<-table(results_test11,processed_test_data_rosepca.corr2$default_payment_next_month)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corr2$default_payment_next_month,
           predictor = as.numeric(results_test11), quiet=TRUE)
auc = auc$auc

a<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


ok3 <- sapply(rose_processed_train_data.corr, is_binary)
rose_processed_train_data.corr2 <- replace(rose_processed_train_data.corr,
                                           ok3, lapply(rose_processed_train_data.corr[ok3],
                                                       factor, levels = 0:1))
ok3 <- sapply(processed_test_data_rosepca.corr, is_binary)
processed_test_data_rosepca.corrr2 <- replace(processed_test_data_rosepca.corr,
                                              ok3, lapply(processed_test_data_rosepca.corr[ok3],
                                                          factor, levels = 0:1))

dtree.model3 <- ctree(default_payment_next_month~.,
                      data = rose_processed_train_data.corr2,
                      controls = ctree_control(mincriterion=0.95))
results_test12 <- predict(dtree.model3,processed_test_data_rosepca.corrr2[,-83])
rf.cm<-table(results_test12,processed_test_data_rosepca.corrr2$default_payment_next_month)
rf.cm2<-table(results_test12,processed_test_data_rosepca.corrr2$default_payment_next_month)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corrr2$default_payment_next_month,
           predictor = as.numeric(results_test12), quiet=TRUE)
auc = auc$auc
b<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)
```

```r
ok <- sapply(processed_train_data.corr, is_binary)
processed_train_data.corr2 <- replace(processed_train_data.corr,
                                      ok, lapply(processed_train_data.corr[ok], factor, levels = 0:1))
ok <- sapply(processed_test_data.corr, is_binary)
processed_test_data.corr2 <- replace(processed_test_data.corr,
                                     ok, lapply(processed_test_data.corr[ok], factor, levels = 0:1))
dtree.model <- ctree(default_payment_next_month~.,
                     data = select(processed_train_data.corr2,
                                   -c('LIMIT_BAL','AGE','BILL_AMT1',
                                      'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                      'PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                      controls = ctree_control(mincriterion=0.95))
results_test13 <- predict(dtree.model,select(processed_test_data.corr2,
                                             -c('LIMIT_BAL','AGE','BILL_AMT1',
                                                'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                                'PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-83])

rf.cm<-table(results_test13,processed_test_data.corr2$default_payment_next_month)
rf.cm3<-table(results_test13,processed_test_data.corr2$default_payment_next_month)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data.corr2$default_payment_next_month,
           predictor = as.numeric(results_test13), quiet=TRUE)
auc = auc$auc
c<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


#4

ok4 <- sapply(rose_processed_train_data_pca.corr, is_binary)
i2 <- replace(rose_processed_train_data_pca.corr,
              ok, lapply(rose_processed_train_data_pca.corr[ok4], factor, levels = 0:1))
ok4 <- sapply(rose_processed_test_data_pca.corr, is_binary)
rose_processed_test_data_pca.corr2 <- replace(rose_processed_test_data_pca.corr,
                                              ok4, lapply(rose_processed_test_data_pca.corr[ok4],
                                                          factor, levels = 0:1))
dtree.model3 <- ctree(default_payment_next_month~.,
                      data = select(i2,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                          'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5',
                                          'PAY_AMT6')),
                      controls = ctree_control(mincriterion=0.95))
results_test13 <- predict(dtree.model3,
                          select(rose_processed_test_data_pca.corr2,-c('LIMIT_BAL',
                                                                       'AGE','BILL_AMT1','PAY_AMT1',
                                                                       'PAY_AMT2','PAY_AMT3','PAY_AMT4'
                                                                       'PAY_AMT5','PAY_AMT6')))
rf.cm<-table(results_test13,rose_processed_test_data_pca.corr2$default_payment_next_month)
rf.cm4<-table(results_test13,rose_processed_test_data_pca.corr2$default_payment_next_month)
```

```r
precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = rose_processed_test_data_pca.corr2$default_payment_next_month,
           predictor = as.numeric(results_test13), quiet=TRUE)
auc = auc$auc
d<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


final_dt<-rbind(a,b,c,d)
#c('High Correlated variables removed','High Correlated variables removed with
#undersampling/oversampling techniques','High Correlated variables removed followed by PCA',
#'High Correlated variables removed with undersampling/oversampling techniques followed by PCA')
rownames(final_dt) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_dt
```

```
##                             precision    recall  accuracy        f1
## corr                        0.9514783 0.8378254 0.8191892 0.8910423
## corr+under/oversampling     0.8267826 0.8698994 0.7693243 0.8477931
## corr+PCA                    0.9516522 0.8375938 0.8190541 0.8909875
## corr+under/oversampling+PCA 0.3335652 0.8551048 0.4382432 0.4799199
##                             avg_accuracy       auc
## corr                           0.7585679 0.6548300
## corr+under/oversampling        0.6775853 0.6979368
## corr+PCA                       0.7584739 0.6543109
## corr+under/oversampling+PCA    0.5560185 0.5682978
```

```r
tm<-cbind(rf.cm1,rf.cm2,rf.cm3,rf.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA     "=2,
                   "corr+under/oversampling+PC"=2))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 4: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5471 | 1059 | 4754 | 711 | 5472 | 1061 | 1918 | 325 |
| 1 | 279 | 591 | 996 | 939 | 278 | 589 | 3832 | 1325 |

## 5. Random Forest

The random forest is essentially an ensemble of randomly created decision trees, where each node in different trees are working on a random subset of features to calculate the output. Output of each decision tree is then combined to give the final prediction of the target class.

To optimise the performance of our random forest model, we tuned the mtry parameter, which refers to the number of variables randomly sampled as candidates at each split. This was done by find the mtry value that returns the lowest out of bag error (OOB). OOB refers to the errors achieved on out of bag samples, which are observations in the training data that are not used in training the model since the random forest algorithm utilises bootstrapping which allows for a sample to be selected several times and some to be excluded.

Since the random forest classification combines the outcomes of several decision trees, it reduces the risk of overfitting. Furthermore, the random forest algorithm is compatible with categorical and continuous variables, and is also robust to outliers.

```
set.seed(1234)
#1 processed_train_data_rosepca.corr, processed_test_data_rosepca.corr
rf.model = randomForest(default_payment_next_month~.,
                        data=processed_train_data_rosepca.corr,
                        mtry=9)
results_test5 = predict(rf.model, newdata = processed_test_data_rosepca.corr)
rf.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test5)
rf.cm1 <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test5)
precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)

auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(results_test5), quiet=TRUE)
auc = auc$auc
a<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)





#FINETUNING HYPERPARAMETERS
t1 <- tuneRF(processed_train_data_rosepca.corr[,-83],
            processed_train_data_rosepca.corr[,83], #exclude the response variable
            stepFactor = 0.5, #per each interations number of variables tried at each split (mtry) in i
            plot = FALSE, #plot OOB
            ntreeTry = 500, #tuning number of trees
            trace = FALSE,
            improve = 0.05)
```

```
## -0.00772297 0.05
## -0.01145989 0.05
```

```
#hist(treesize(rf.model),                       # give us the number of trees in term of number of nod
#     main = "Number of Nodes for the Trees",
#     col = "grey")

# Plotting model
#plot(rf.model)

#Variable importance plot
#varImpPlot(rf.model,
#           sort = T,
#           n.var = 10,
#           main = "Top 10 Variable Importance")
```

```r
#2 rose_processed_train_data.corr processed_test_data_rosepca.corr
rf.model = randomForest(default_payment_next_month~.,
                        data=rose_processed_train_data.corr,
                        mtry = 18)
results_test5 = predict(rf.model, newdata = processed_test_data_rosepca.corr)
rf.cm <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test5)
rf.cm2 <- table(processed_test_data_rosepca.corr$default_payment_next_month,results_test5)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)

auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(results_test5), quiet=TRUE)
auc = auc$auc
b<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

#FINETUNING HYPERPARAMETERS
t2 <- tuneRF(rose_processed_train_data.corr[,-83],
             rose_processed_train_data.corr[,83], #exclude the response variable
             stepFactor = 0.5, #per each interations number of variables tried at each split (mtry) in i
             plot = FALSE, #plot OOB
             ntreeTry = 500, #tuning number of trees
             trace = FALSE,
             improve = 0.05)
```

```
## 0.01166117 0.05
## -0.2354235 0.05
```

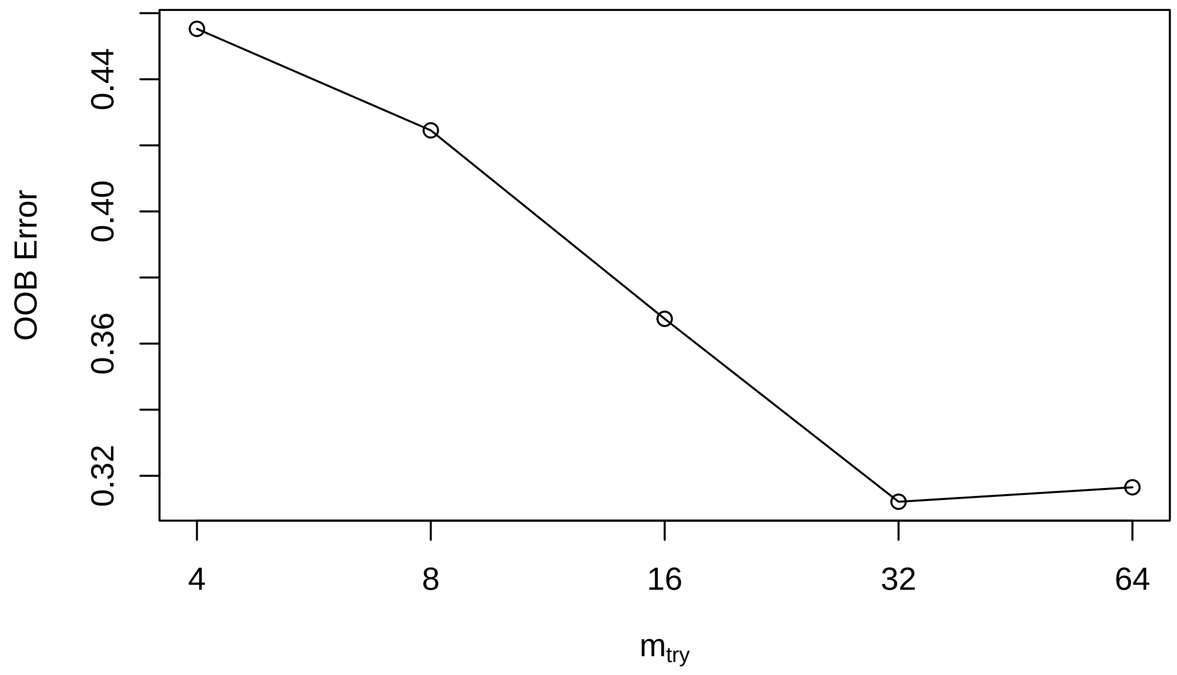
```r
# hist(treesize(rf.model),                        # give us the number of trees in term of number of no
#      main = "Number of Nodes for the Trees",
#      col = "grey")
#
# # Plotting model
# #plot(rf.model)
#
# #Variable importance plot
# varImpPlot(rf.model,
#            sort = T,
#            n.var = 10,
#            main = "Top 10 Variable Importance")


#3
#processed_train_data.corr,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4'
#processed_test_data.corr
rf.model = randomForest(default_payment_next_month~.,
                    data=select(processed_train_data.corr,
                                -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                   'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5',
                                   'PAY_AMT6')),
                    mtry = 9)
results_test5 = predict(rf.model,
                        newdata = select(processed_test_data.corr,
                                         -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                            'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5',
                                            'PAY_AMT6')))
rf.cm <- table(processed_test_data.corr$default_payment_next_month,results_test5)
rf.cm3 <- table(processed_test_data.corr$default_payment_next_month,results_test5)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)

auc <- roc(response = processed_test_data.corr$default_payment_next_month,
           predictor = as.numeric(results_test5), quiet=TRUE)
auc = auc$auc
c<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


#FINETUNING HYPERPARAMETERS
t3 <- tuneRF(processed_train_data.corr[,-83],
            processed_train_data.corr[,83], #exclude the response variable
            stepFactor = 0.5, #per each interations number of variables tried at each split (mtry) in i
            plot = FALSE, #plot OOB
            ntreeTry = 500, #tuning number of trees
            trace = FALSE,
            improve = 0.05)
```

```
## -0.01211971 0.05
## -0.00222607 0.05
```

```
# hist(treesize(rf.model),                        # give us the number of trees in term of number of no
#       main = "Number of Nodes for the Trees",
#       col = "grey")
#
# # Plotting model
# plot(rf.model)
#
# #Variable importance plot
# varImpPlot(rf.model,
#             sort = T,
#             n.var = 10,
#             main = "Top 10 Variable Importance")

#4
#rose_processed_train_data_pca.corr,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','
#rose_processed_test_data_pca.corr
```

```
rf.model2 = randomForest(default_payment_next_month~.,
                         data=select(i2,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3
                         mtry = 32)
results_test6 = predict(rf.model2, newdata = rose_processed_test_data_pca.corr2)
rf.cm <- table(rose_processed_test_data_pca.corr$default_payment_next_month,results_test6)
rf.cm4 <- table(rose_processed_test_data_pca.corr$default_payment_next_month,results_test6)

precision <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[2])
recall <- as.matrix(rf.cm)[1] / (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[3])
accuracy <- (as.matrix(rf.cm)[1] + as.matrix(rf.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(rf.cm)[4] /(as.matrix(rf.cm)[2] + as.matrix(rf.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)

auc <- roc(response = rose_processed_test_data_pca.corr$default_payment_next_month,
           predictor = as.numeric(results_test6), quiet=TRUE)
auc = auc$auc
d<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

#FINETUNING HYPERPARAMETERS
t4 <- tuneRF(select(i2,
                    -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2',
                       'PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-74],
            select(i2,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2',
                         'PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,74], #exclude the response vari
            stepFactor = 0.5, #per each interations number of variables tried at each split (mtry) in i
            plot = TRUE, #plot OOB
            ntreeTry = 500, #tuning number of trees
            trace = FALSE,
            improve = 0.05)
```

```
## 0.1343236 0.05
## 0.1506312 0.05
## -0.01399711 0.05
## -0.4584416 0.05
```



```
#hist(treesize(rf.model2),                          # give us the number of trees in term of number of no
#     main = "Number of Nodes for the Trees",
#     col = "grey")

# Plotting model
#plot(rf.model2)

#Variable importance plot
#varImpPlot(rf.model2,
#           sort = T,
#           n.var = 10,
#           main = "Top 10 Variable Importance")



final_rf<-rbind(a,b,c,d)


#c('High Correlated variables removed','High Correlated variables removed with undersampling/oversampli
rownames(final_rf) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_rf
```

```
##                              precision    recall  accuracy        f1
## corr                         0.8352333 0.9556522 0.8190541 0.8913943
## corr+under/oversampling      0.7780035 0.9989565 0.7777027 0.8747430
## corr+PCA                       0.8358732 0.9539130 0.8186486 0.8910006
## corr+under/oversampling+PCA 0.8808511 0.1800000 0.3439189 0.2989170
##                               avg_accuracy       auc
## corr                             0.6493412 0.6493412
## corr+under/oversampling          0.5028116 0.5028116
## corr+PCA                         0.6505929 0.6505929
## corr+under/oversampling+PCA      0.5475758 0.5475758
```

```
tm<-cbind(rf.cm1,rf.cm2,rf.cm3,rf.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA     "=2,
                     "corr+under/oversampling+PC"=2))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 5: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5495 | 255 | 5744 | 6 | 5485 | 265 | 1035 | 4715 |
| 1 | 1084 | 566 | 1639 | 11 | 1077 | 573 | 140 | 1510 |

## 6. Neural Networks

Neural Networks (diff variations of neural networks) The neural network algorithm comprises of 3 layers, the input layer, hidden layer and output layer. Each node in every layers connect to another, and has an associated weight and threshold. When output for a individual node is higher than the threshold, that node is activated and data is passed on to the next layer. For the input layer, there are as many nodes as there are attributes in the training data. For the output layer, since our target class is binary, we will only have one node. For the hidden layer, there can be multiple hidden layer but in our case we choose to only use 1 hidden layer due to the heavy computational cost and long training time required.

To select the number of nodes in the hidden layer, we iterated a range of values for the size of each layer and computed the accuracy rate achieved with each value. We thus selected the hidden layer sizes that returned the higher accuracy rates.

To select the learning rates, we set a minimum of 0.06 due to the long computation times associated with a low learning rate. We then iterated a range of values to find the learning rate that returns the best performance.

To speed up the learning process, we set the minimum threshold at 0.05. This means that if the change in the error between each iteration is less than 5%, the neural network model will stop optimising its weights.

```
suppressMessages(library(neuralnet))
suppressMessages(library(keras))



set.seed(1234)

# mylist <- list()
# for (i in 10:15) {
#   nn <- neuralnet(default_payment_next_month~., data = select(processed_train_data_rosepca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#hidden = i, linear.output = F, lifesign = 'full',rep=1,learningrate =  0.06,threshold = 0.05,act.fct =
#
#   nn.pred <- compute(nn,select(processed_test_data_rosepca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,
#
#   nn.pred1 = rep("0", dim(processed_test_data_rosepca.corr)[1])
#   #Look for optimal threshold : AUC
#
#   nn.pred1[nn.pred$net.result[,2]>.5] = "1"
#   nn.cm<- table(nn.pred1, processed_test_data_rosepca.corr$default_payment_next_month)
#
#
#   mylist<- append(mylist, mean(nn.pred1==processed_test_data_rosepca.corr$default_payment_next_month)
#   print(mylist)
# }

#choosing hidden = 1,
nn <- neuralnet(default_payment_next_month~.,
                data = select(processed_train_data_rosepca.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                 'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                hidden = 1, linear.output = F,rep=1,learningrate =  0.06,threshold = 0.05,act.fct = 'log

nn.pred <- compute(nn,select(processed_test_data_rosepca.corr,
                             -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-83])

nn.pred1 = rep("0", dim(processed_test_data_rosepca.corr)[1])
   #Look for optimal threshold : AUC
nn.pred1[nn.pred$net.result[,2]>.5] = "1"
nn.cm<- table(nn.pred1, processed_test_data_rosepca.corr$default_payment_next_month)
nn.cm1<- table(nn.pred1, processed_test_data_rosepca.corr$default_payment_next_month)
precision <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[2])
recall <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[3])
accuracy <- (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nn.cm)[4] /(as.matrix(nn.cm)[2] + as.matrix(nn.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = rose_processed_test_data_pca.corr2$default_payment_next_month,
           predictor = as.numeric(nn.pred$net.result[,2]), quiet=TRUE)
auc = auc$auc
a<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

# mylist2 <- list()
# for (i in 1:6) {
#   nn <- neuralnet(default_payment_next_month~.,
#data = select(rose_processed_train_data.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#hidden = i, linear.output = T, lifesign = 'full',rep=1,learningrate =  0.06,threshold = 0.05)
#
#   nn.pred <- compute(nn,select(processed_test_data_rosepca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,
#
#   nn.pred2 = rep("0", dim(processed_test_data_rosepca.corr)[1])
#   #Look for optimal threshold : AUC
#
#   nn.pred2[nn.pred$net.result[,2]>.5] = "1"
#   nn.cm2<- table(nn.pred2, processed_test_data_rosepca.corr$default_payment_next_month)
#
#
#   mylist2<- append(mylist2, mean(nn.pred2==processed_test_data_rosepca.corr$default_payment_next_mont
#   print(mylist2)
# }

#choosing hidden = 1
nn <- neuralnet(default_payment_next_month~.,
                data = select(rose_processed_train_data.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2',
                                 'PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                hidden = 1, linear.output = T,rep=1,learningrate =  0.06,threshold = 0.05)

nn.pred <- compute(nn,select(processed_test_data_rosepca.corr,
                             -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-83])
nn.pred2 = rep("0", dim(processed_test_data_rosepca.corr)[1])
#Look for optimal threshold : AUC
nn.pred2[nn.pred$net.result[,2]>.5] = "1"
nn.cm2<- table(nn.pred2, processed_test_data_rosepca.corr$default_payment_next_month)
nn.cm<- table(nn.pred2, processed_test_data_rosepca.corr$default_payment_next_month)
precision <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[2])
recall <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[3])
accuracy <- (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nn.cm)[4] /(as.matrix(nn.cm)[2] + as.matrix(nn.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data_rosepca.corr$default_payment_next_month,
           predictor = as.numeric(nn.pred$net.result[,2]), quiet=TRUE)
auc = auc$auc
b<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)


# mylist3 <- list()
# for (i in 1:6) {
# nn <- neuralnet(default_payment_next_month~.,
#data = select(processed_train_data.corr,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
#'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#hidden = i, linear.output = F,act.fct = 'logistic', lifesign = 'full',rep=1,learningrate =  0.06,thres
# head(processed_test_data.corr2)
# nn.pred <- compute(nn,select(processed_test_data.corr,-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PA
#
# nn.pred3 = rep("0", dim(processed_test_data.corr)[1])
# #Look for optimal threshold : AUC
#
# nn.pred3[nn.pred$net.result[,2]>.5] = "1"
# nn.cm3<- table(nn.pred2, processed_test_data.corr$default_payment_next_month)
#
#
# mylist3<- append(mylist3, mean(nn.pred3==processed_test_data.corr$default_payment_next_month))
```

```r
# print(mylist3)
# }

#choosing hidden = 4
nn <- neuralnet(default_payment_next_month~.,
                data = select(processed_train_data.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2',
                                 'PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                hidden = 4, linear.output = F,act.fct = 'logistic',
                rep=1,learningrate =  0.06,threshold = 0.05)
#head(processed_test_data.corr2)
nn.pred <- compute(nn,select(processed_test_data.corr,
                             -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-83])

nn.pred3 = rep("0", dim(processed_test_data.corr)[1])
#Look for optimal threshold : AUC

nn.pred3[nn.pred$net.result[,2]>.5] = "1"
nn.cm3<- table(nn.pred3, processed_test_data.corr$default_payment_next_month)
nn.cm<- table(nn.pred3, processed_test_data.corr$default_payment_next_month)
precision <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[2])
recall <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[3])
accuracy <- (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nn.cm)[4] /(as.matrix(nn.cm)[2] + as.matrix(nn.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = processed_test_data.corr$default_payment_next_month,
           predictor = as.numeric(nn.pred$net.result[,2]), quiet=TRUE)
auc = auc$auc
c<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

# mylist4 <- list()
# for (i in 1:6) {
# nn <- neuralnet(default_payment_next_month~.,
#data = select(rose_processed_train_data_pca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6')),
#hidden = i, linear.output = F,act.fct = 'logistic', lifesign = 'full',rep=1,learningrate =  0.1,thresh
# nn.pred <- compute(nn,select(rose_processed_test_data_pca.corr,
#-c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,
#
# nn.pred4 = rep("0", dim(rose_processed_test_data_pca.corr)[1])
# #Look for optimal threshold : AUC
#
# nn.pred4[nn.pred$net.result[,2]>.5] = "1"
# nn.cm4<- table(nn.pred4, rose_processed_test_data_pca.corr$default_payment_next_month)
#
#
# mylist4<- append(mylist4, mean(nn.pred4==rose_processed_test_data_pca.corr$default_payment_next_month
# print(mylist4)
# }

#choosing hidden = 1
nn <- neuralnet(default_payment_next_month~.,
                data = select(rose_processed_train_data_pca.corr,
                              -c('LIMIT_BAL','AGE','BILL_AMT1',
                                 'PAY_AMT1','PAY_AMT2','PAY_AMT3',
                                 'PAY_AMT4','PAY_AMT5','PAY_AMT6')),
                hidden = 1, linear.output = F,act.fct = 'logistic',
                rep=1,learningrate =  0.1,threshold = 0.05)
nn.pred <- compute(nn,select(rose_processed_test_data_pca.corr,
                             -c('LIMIT_BAL','AGE','BILL_AMT1','PAY_AMT1',
                                'PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6'))[,-83])

nn.pred4 = rep("0", dim(rose_processed_test_data_pca.corr)[1])
#Look for optimal threshold : AUC

nn.pred4[nn.pred$net.result[,2]>.5] = "1"
nn.cm4<- table(nn.pred4, rose_processed_test_data_pca.corr$default_payment_next_month)
nn.cm<- table(nn.pred4, rose_processed_test_data_pca.corr$default_payment_next_month)
precision <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[2])
recall <- as.matrix(nn.cm)[1] / (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[3])
accuracy <- (as.matrix(nn.cm)[1] + as.matrix(nn.cm)[4]) / 7400
avg_accuracy <- (recall+(as.matrix(nn.cm)[4] /(as.matrix(nn.cm)[2] + as.matrix(nn.cm)[4])))/2
f1 <- 2* (precision*recall) / (precision+recall)
auc <- roc(response = rose_processed_test_data_pca.corr$default_payment_next_month,
           predictor = as.numeric(nn.pred$net.result[,2]), quiet=TRUE)
auc = auc$auc
d<-cbind(precision,recall,accuracy,f1,avg_accuracy, auc)

tm<-cbind(nn.cm1,nn.cm2,nn.cm3,nn.cm4)
kable(tm,longtable =T,booktabs =T,caption ="Confusion matrices of each model")%>%
  add_header_above(c(" ","corr        "=2,"corr+under/oversampling"=2,"corr+PCA     "=2,
                   "corr+under/oversampling+PC"=2))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 6: Confusion matrices of each model

| | corr | | corr+under/oversampling | | corr+PCA | | corr+under/oversampling+PC | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 5462 | 1048 | 4783 | 707 | 5329 | 962 | 5455 | 1041 |
| 1 | 288 | 602 | 967 | 943 | 421 | 688 | 295 | 609 |

```
final_nn<-rbind(a,b,c,d)

rownames(final_nn) = c('corr','corr+under/oversampling','corr+PCA','corr+under/oversampling+PCA')
final_nn
```

```
##                             precision    recall  accuracy        f1
## corr                        0.9499130 0.8390169 0.8194595 0.8910277
## corr+under/oversampling     0.8318261 0.8712204 0.7737838 0.8510676
## corr+PCA                    0.9267826 0.8470831 0.8131081 0.8851424
## corr+under/oversampling+PCA 0.9486957 0.8397475 0.8194595 0.8909032
##                             avg_accuracy       auc
## corr                           0.7577107 0.7495964
## corr+under/oversampling        0.6824688 0.7484972
## corr+PCA                       0.7337309 0.7527483
## corr+under/oversampling+PCA    0.7567101 0.7525197
```

## 6. Model Evaluation

### Precision

Among all positive predictions, the proportion of actual positive instances. For our dataset, precision is a measure of the bank clients we correctly identify as individuals who defaulted on payment out of all the patients who defaulted payment.

Precision is also a measure of quality; a higher precision suggests that the classification algorithm returns more relevant results. Having a high precision is important because the bank would lose low-risk customers if the clients were deemed as defaulters based on the model's prediction.

### Recall

Among all positive instances, the proportion of correct predictions. For our dataset, recall is a measure of for all clients who have actually defaulted on payment, recall tells us how many clients we have correctly identified. Recall is a measure of quantity; a higher recall suggests that the classification algorithm returns most of the relevant results. We would like to avoid the situation where the client is a defaulter but our model fails to identify the client as a defaulter, as accepting these clients as defaulters may bring additional financial risk to the bank.

### Accuracy

Accuracy is the ratio of the number of correct predictions over the total number of predictions. For our dataset, accuracy is a measure of the clients who are correctly predicted as defaulters and non-defaulters respectively out of all the predictions. Accuracy is important to the bank as it is important to be able to correctly identify the defaulters and non-defaulters out of all the clients. However, accuracy alone is a bad metric in the case of unbalanced datasets such as our dataset.

### Average Class Accuracy

Average class accuracy measures the average accuracy per class, and is particularly beneficial when the data is imbalanced as classification accuracy can mask poor performance where data is imbalanced. In this case, average class reduces the accuracy bias for unbalanced datasets on a certain class, where in this case non-defaulters are the majority class and defaulters are the minority class.

### F1 Score

F1 score is defined as the harmonic mean between precision and recall. F1 score is more useful in evaluating classification model performance in the case of unbalanced datasets such as our dataset as compared to precision and recall as F1 score gives an equal weight to precision and recall.

### Area Under ROC Curve

AUC is a measure of separability between the classes of the target variable, which in our class is defaulters and non-defaulters. The higher the AUC, the better the classification model is at predicting defaulters as defaulters and non-defaulters as non-defaulters.

Based on our dataset and models selected, we have decided to evaluate our models based on

***F1 Score, Average Class Accuracy, AUC.***

```r
a = final_log[,4:6]
b = final_svm[,4:6]
c = final_nb[,4:6]
```

```
d = final_dt[,4:6]
e = final_rf[,4:6]
f = final_nn[,4:6]

tm<-cbind(a,b)
kable(tm,longtable =T,booktabs =T,caption ="F1-Score, Average Class Accuracy, AUC of each model")%>%
  add_header_above(c(" ","log_reg         "=3,"svm"=3))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 7: F1-Score, Average Class Accuracy, AUC of each model

| | log_reg | | | svm | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.7610457 | 0.5625086 | 0.5625086 | 0.8915779 | 0.6411278 | 0.6411278 |
| corr+under/oversampling | 0.8022906 | 0.6809592 | 0.6809592 | 0.8563946 | 0.6840975 | 0.6840975 |
| corr+PCA | 0.7201348 | 0.6021897 | 0.6021897 | 0.8915779 | 0.6411278 | 0.6411278 |
| corr+under/oversampling+PCA | 0.7986648 | 0.6980264 | 0.7632454 | 0.8757880 | 0.6899868 | 0.6899868 |

```
tm<-cbind(c,d)
kable(tm,longtable =T,booktabs =T,caption ="F1-Score, Average Class Accuracy, AUC of each model")%>%
  add_header_above(c(" ","naive_bayes     "=3,"decision_tree"=3))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 8: F1-Score, Average Class Accuracy, AUC of each model

| | naive_bayes | | | decision_tree | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.8789188 | 0.5669539 | 0.5669539 | 0.8910423 | 0.7585679 | 0.6548300 |
| corr+under/oversampling | 0.8756026 | 0.5408643 | 0.5408643 | 0.8477931 | 0.6775853 | 0.6979368 |
| corr+PCA | 0.8809412 | 0.5689539 | 0.5689539 | 0.8909875 | 0.7584739 | 0.6543109 |
| corr+under/oversampling+PCA | 0.8772283 | 0.5396627 | 0.5396627 | 0.4799199 | 0.5560185 | 0.5682978 |

```
tm<-cbind(e,f)
kable(tm,longtable =T,booktabs =T,caption ="F1-Score, Average Class Accuracy, AUC of each model")%>%
  add_header_above(c(" ","random_forest"=3,"neural_network"=3))%>%
  kable_styling(latex_options =c("repeat_header"))
```

Table 9: F1-Score, Average Class Accuracy, AUC of each model

| | random_forest | | | neural_network | | |
|---|---|---|---|---|---|---|
| | f1 | avg_accuracy | auc | f1 | avg_accuracy | auc |
| corr | 0.8913943 | 0.6493412 | 0.6493412 | 0.8910277 | 0.7577107 | 0.7495964 |
| corr+under/oversampling | 0.8747430 | 0.5028116 | 0.5028116 | 0.8510676 | 0.6824688 | 0.7484972 |
| corr+PCA | 0.8910006 | 0.6505929 | 0.6505929 | 0.8851424 | 0.7337309 | 0.7527483 |
| corr+under/oversampling+PCA | 0.2989170 | 0.5475758 | 0.5475758 | 0.8909032 | 0.7567101 | 0.7525197 |

As a whole, using our selected evaluation metrics of ***F1 Score, Average Class Accuracy, AUC***, the neural networks classification algorithm has shown significantly higher F1 Score, Average Class Accuracy and AUC.

We can identify that the neural networks classification algorithm applied on the dataset with oversampling & undersampling techniques applied, has its highly correlated features removed and PCA feature selection techniques applied to it is the best performing model.

Generally, an F1 score is considered good when it is closer to 1. For our selected model and dataset it is trained on, F1 score is 0.891, and is thus considered a strong performance. For average class accuracy, a value of 0.5 implies that the model predicts classes correctly with a 50% chance. Thus, anything above 0.5 is considered okay, with a value of above 0.7 to be considered good. For our selected model, average class accuracy rate is 0.757, and is thus considered a good classification model. Finally, the AUC score is considered good generally when it is above 0.7. For our selected model, AUC score is 0.753.

# 7. Areas For Improvement

1. **Grid search can be used to identify optimal hyperparameters for the models**

Due to the complexity and high computation time, we are not able to consider all possible combinations of hyperparameters for each respective model. Grid search is a cross-validation technique that would allow us to consider all possible combinations of hyperparameters to find the model with a range of values for each hyperparameter, which would better allow us to selection the best hyperparameters for the classification.

2. **Check for over-fitting**

Over-fitting of the models can be identified by evaluating the models on the train set and the holdout test set separately. Should the performance of the model be significantly better on the train set as compared to the test set, the model may have over-fitted the train dataset. Over-fitting can be identified on learning curve plots where validation metrics such as accuracy and loss continues to improve on the train set while the test set starts to get worse.

# APPENDIX

<u>**References**</u>

1. 10. Decision trees. (n.d.). Scikit-Learn. Retrieved November 18, 2022, from https://scikit-learn/stable/modules/tree.html

Brownlee, J. (2016, February 4). Tune machine learning algorithms in r(Random forest case study). MachineLearningMastery.Com. https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

Ctree function—Rdocumentation. (n.d.). Retrieved November 18, 2022, from https://www.rdocumentation.org/packages/partykit/versions/1.2-16/topics/ctree

Danasingh, A. A. G. S., Subramanian, A. alias B., & Epiphany, J. L. (2020). Identifying redundant features using unsupervised learning for high-dimensional data. SN Applied Sciences, 2(8), 1367. https://doi.org/10.1007/s42452-020-3157-6

Decision trees for classification: A machine learning algorithm. (n.d.). Xoriant. Retrieved November 18, 2022, from https://www.xoriant.com/blog/decision-trees-for-classification-a-machine-learning-algorithm

Default of credit card clients dataset. (n.d.). Retrieved November 18, 2022, from https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

Dey, V. (2021, August 25). When to use one-hot encoding in deep learning? Analytics India Magazine. https://analyticsindiamag.com/when-to-use-one-hot-encoding-in-deep-learning/

finnstats. (2021, April 10). Deep neural network in r | r-bloggers. https://www.r-bloggers.com/2021/04/deep-neural-network-in-r/

finnstats. (2021, April 13). Random forest in r | r-bloggers. https://www.r-bloggers.com/2021/04/random-forest-in-r/

How to create correlation heatmap in r. (2022, January 24). GeeksforGeeks. https://www.geeksforgeeks.org/how-to-create-correlation-heatmap-in-r/

Huilgol, P. (2020, September 3). Precision vs recall | precision and recall machine learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

Lee, I. (2020, April 19). Determine the threshold “p” in the logistic regression. Issac Lee. https://theissaclee.com/post/logistic-regression-beta/

Logistic regression essentials in r—Articles—Sthda. (n.d.). Retrieved November 18, 2022, from http://www.sthda.com/english/articles/36-classification-methods-essentials/151-logistic-regression-essentials-in-r/

Logistic regression in r programming. (2020, June 1). GeeksforGeeks. https://www.geeksforgeeks.org/logistic-regression-in-r-programming/

Loukas, S. (2021, October 8). PCA clearly explained — How, when, why to use it and feature importance: A guide in Python. Medium. https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e

Muralidhar, K. S. V. (2021, February 22). What is stratified cross-validation in machine learning? Medium. https://towardsdatascience.com/what-is-stratified-cross-validation-in-machine-learning-8844f3e7ae8e

pascalwhoop. (2019, July 19). An overview of categorical input handling for neural networks. Medium. https://towardsdatascience.com/an-overview-of-categorical-input-handling-for-neural-networks-c172ba552dee

Pramoditha, R. (2021, September 28). Principal component analysis (Pca) with scikit-learn. Medium. https://towardsdatascience.com/principal-component-analysis-pca-with-scikit-learn-1e84a0c731b0

Preprocessing of categorical predictors in svm, knn and kdc(Contributed by xi cheng). (2017, October 23). Statistics LibreTexts.
https://stats.libretexts.org/Bookshelves/Computing_and_Modeling/RTG%3A_Classification_Methods/4%3A_Numerical_Experiments_and_Real_Data_Analysis/Preprocessing_of_categorical_predictors_in_SVM%2C_KNN_and_KDC_(contributed_by_Xi_Cheng)

R - decision tree. (n.d.). Retrieved November 18, 2022, from
https://www.tutorialspoint.com/r/r_decision_tree.htm#:~:text=Decision%20tree%20is%20a%20graph,Data%20Mining%20applications%20using%20R

Rajaratne, M. (2018, December 2). Data pre processing techniques you should know. Medium. https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6

Random forests® vs neural networks: Which is better, and when? (n.d.). KDnuggets. Retrieved November 18, 2022, from https://www.kdnuggets.com/random-forest-vs-neural-network-classification-tabular-data.html

Ray, S. (2017, September 11). Learn naive bayes algorithm | naive bayes classifier examples. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Schmelzer, C. H., Martin Arnold, Alexander Gerber and Martin. (n.d.). Introduction to econometrics with r. Retrieved November 18, 2022, from https://www.econometrics-with-r.org/ivr.html

Sharma, A. (2020, May 11). Decision tree vs. Random forest—Which algorithm should you use? Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Support vector machines (Svm) algorithm explained. (2017, June 22). MonkeyLearn Blog. https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

Support vector machines using svm() function. (n.d.). Retrieved November 18, 2022, from https://rstudio-pubs-static.s3.amazonaws.com/271792_96b51b7fa2af4b3f808d04f3f3051516.html

Using grid search to optimize hyperparameters. (n.d.). Engineering Education (EngEd) Program | Section. Retrieved November 18, 2022, from https://www.section.io/engineering-education/grid-search/

Using logistic regression to model and predict categorical values. (n.d.). Retrieved November 18, 2022, from http://rstudio-pubs-static.s3.amazonaws.com/74431_8cbd662559f6451f9cd411545f28107f.html

What are neural networks? (n.d.). Retrieved November 18, 2022, from https://www.ibm.com/cloud/learn/neural-networks

What is logistic regression? - Definition from searchbusinessanalytics. (n.d.). SearchBusinessAnalytics. Retrieved November 18, 2022, from https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression

Yolanda, R. (2019, June 26). Ann classification with ‘nnet’ package in r. Medium. https://medium.com/@yolandawiyono98/ann-classification-with-nnet-package-in-r-3c4dc14d1f14