BT2101 Decision Making Methods and Tools

SEMESTER I 2019-2020

Assessment of Machine Learning models on predicting Credit Risk

Group Project

Ernest Chng Yong Cheng A0189158M

Jeremy Sim Jin En A0182400W

Luke Izumi Lau Chuan A0190047J

Keif Tan A0180261N

Lai Jun Guo, Scott A0166840B

Lee Kai Ping A0121695A

TABLE OF CONTENTS

01 Brief introduction of data set and data modeling problem

1.1 Background

Credit Risk Assessment is an important practice of minimising losses that may have resulted from inappropriate credit approval decisions and has long been a challenge for financial institutions. As such, financial institutions have adopted credit scoring models to evaluate credit risks of individuals based on information such as their financial status, demographic or past preceding payments. Such models are to help banks in quantifying, aggregating and managing risks while doing so at an accelerated rate.

In this paper, we will be looking at different models that can be used for credit risk assessment. The ‘Credit Card Clients’ data set contains 30,000 financial records and is used to show the effectiveness and feasibilities of the various models used.

We will be considering the following 5 models and exploring each model’s accuracy thereafter.

Model	Advantages	Disadvantages
Support Vector Machine (SVM)	Effective in high-dimensional space Flexible selection of kernels for non-linear correlation	Long and inefficient training process
Decision Tree	Simple to interpret and explain	Low accuracy rate Vulnerability to overfitting
Neural Network	Good to model the non-linear data with large number of input features Flexible & adaptive model	Low explanatory (black box nature) Not probabilistic: Difficult to translate the continuous number output (e.g. a score) into a probability
Logistic Regression	Easy to interpret Output can be interpreted as a probability: you can use it for ranking instead of classification	Vulnerability to overfitting Easily outperformed by other complex models
Naive Bayes	Easy to comprehend No distribution requirements	Very strong assumption of independence class features that it makes, which is very rare in real life Cannot learn interaction between features

02 Exploratory Data Analysis (EDA)

Exploratory Data Analysis is an approach of performing initial investigations on data with the use of summary statistics and graphical representations to maximise insight into the data set, test underlying assumptions and detect outliers and anomalies.

2.1 Data overview

The original dataset comprises of 30,000 observations and 24 attributes, of which one is the dependent variable and the rest of the 23 attributes are independent.

Check for data types:

```
sapply(data, class)
```

Data types of variables are all integers. For categorical variables such as gender, the integer values are used to represent categories such as Male or Female.

Check for null/missing values:

```
apply(data, 2, function(x) any(is.null(x)|is.na(x)|is.nan(x)))
```

As the code returns false for all columns, no variable column has null/missing values.

2.2 Inconsistent values within the dataset

Taking a closer look at the range of values for categorical variables, the values are inconsistent with the legend given.

```
summary(data)
unique(data$EDUCATION) # 2 1 3 5 4 6 0
unique(data$MARRIAGE)  # 1 2 3 0
unique(data$PAY_0)     # 2 -1  0 -2  1   3   4   8   7   5   6
```

2.2.1 Education (X3)

Legend for variable Education (X3): 1 = graduate school; 2 = university; 3 = high school; 4 = others. However, the range of values found were ranging from [0, 6]. This means that there are additional values of 0, 5 and 6 that are unaccounted for.

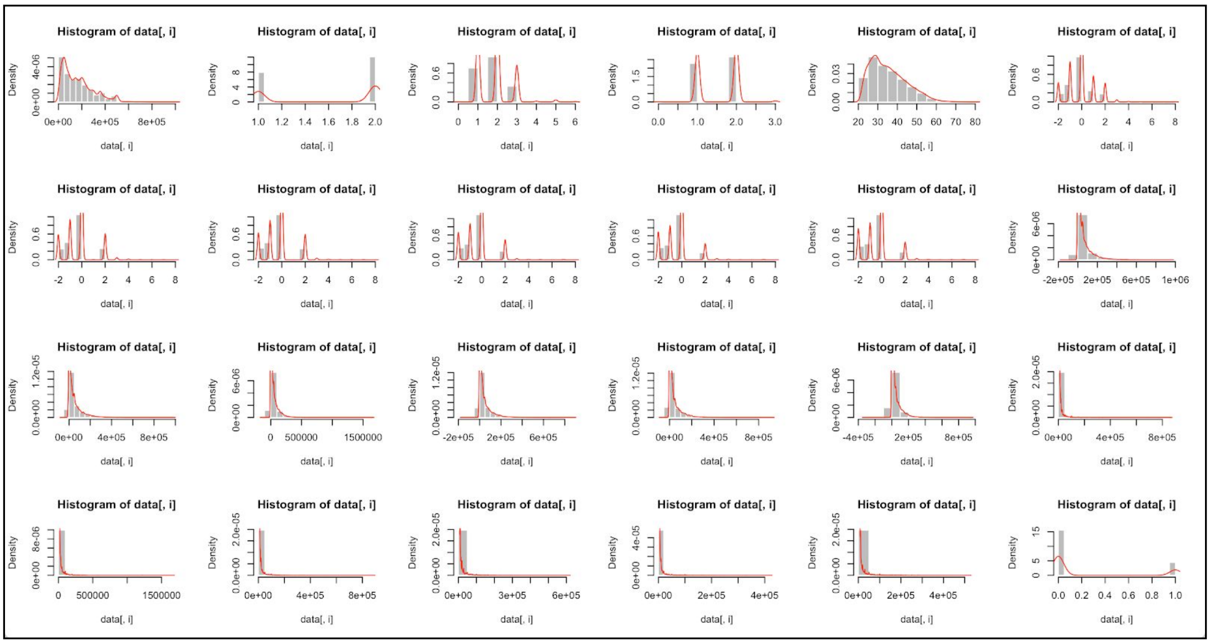
2.2.2 Marital Status (X4)

Legend for variable Marital status (X4): 1 = married; 2 = single; 3 = others. However, the range of values found were ranging from [0, 3]. This means that there is an additional value of 0 that is unaccounted for.

2.2.3 PAY_0 to PAY_6 (X6-X11)

Legend for variables PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 (X6 - X11) where 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. However, the range of values found were ranging from [-2, 8]. This means that there are additional values of -2 and 0 that are unaccounted for.

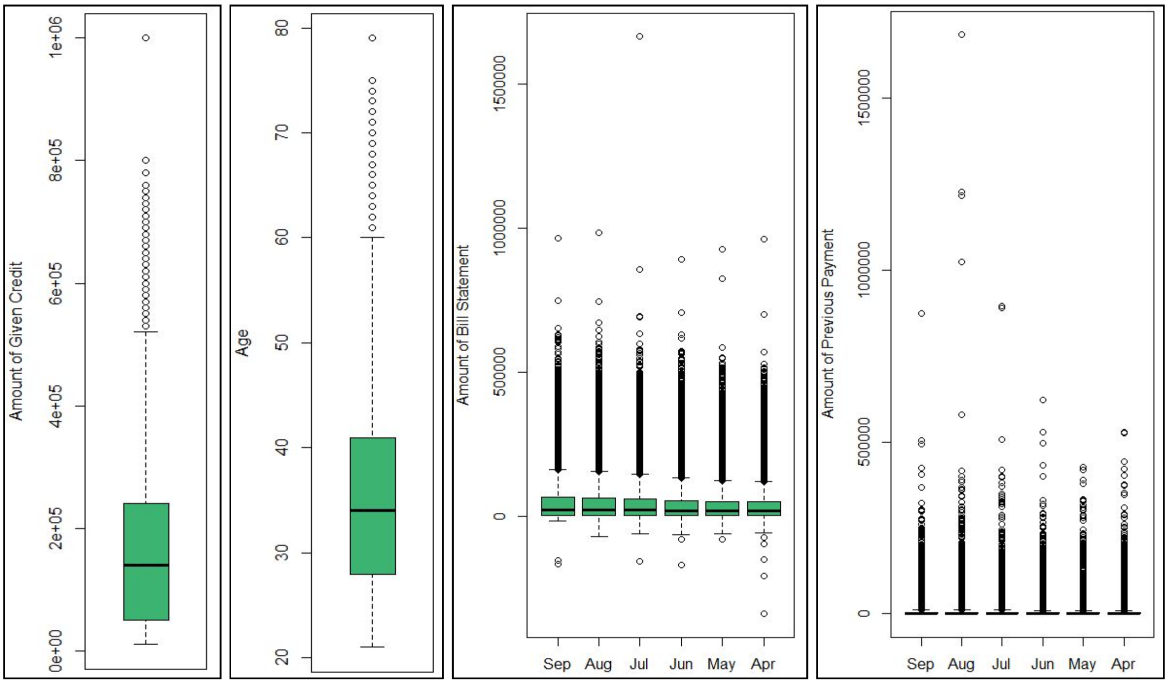
2.3 Checking for categorical data



The density plot for each attribute tells us if that attribute is categorical or not. For example, the top left attribute reflects the “LIMIT_BAL” column, and it is quantitative as there is a distribution of data. However, the attribute to its right reflecting the “SEX” column is categorical as seen by the sharp peaks in the density plot.

2.4 Distribution of values

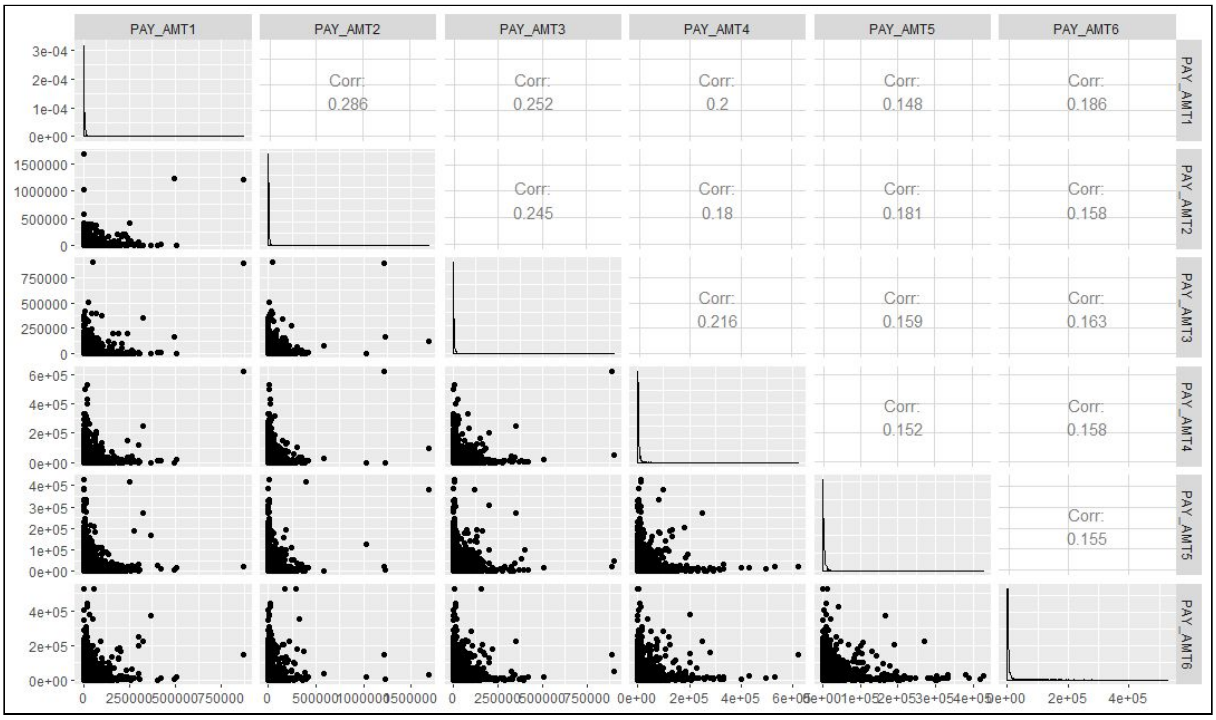
The box plots below show the distribution of the continuous variables based on the maximum and minimum values, the first and third quartile as well as the median.



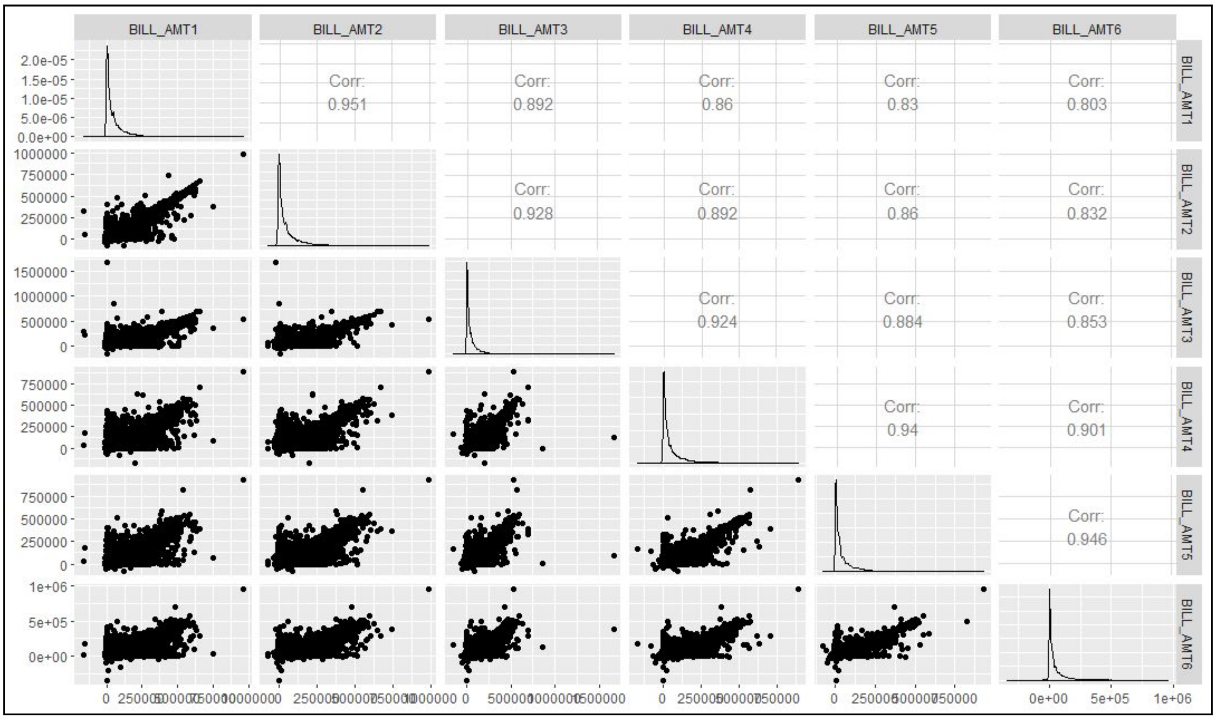
In the data set, all feature columns show outliers but these outliers are not necessarily wrong values. Hence, to determine whether we should keep or drop the outlier, we will run an analysis both with and without the outliers and examine if there's any substantial change in the results before coming to a decision.

2.5 Correlation between variables

PAY_AMT variables do not have a high correlation with one another



However, BILL_AMT variables are highly correlated with one another.



2.6 Class imbalance of dependent variable

```
#     0      1
#23364  6636
```

After analyzing the data, it was found that ~80% was classified as ‘No’ and ~20% was classified as ‘Yes’ for the dependent variable. With an imbalanced data set, the models used will not get the necessary information about the minority class to make an informed and accurate prediction as this imbalance might cause the performance of classifiers to be biased towards the majority class.

---

03 Data Pre-processing

It is a technique that involves transforming raw data into an understandable format. Real world raw data is very often not complete and thus raw data cannot be directly sent through a model as doing so would result in errors. Hence, we will be required to preprocess the data before sending it through a model.

3.1 Standardizing the data

```
data[,1] <- as.data.frame(scale(data[,1]))
data[,5] <- as.data.frame(scale(data[,5]))
data[,12:23] <- as.data.frame(scale(data[,12:23]))
```

Normalisation is executed each column to put it on a common scale which allows you to compare it to other (standardized) variables.

3.2 Filtering entries not consistent with the data source

```
data <- data %>% filter(EDUCATION %in% c(1, 2, 3, 4))
```

In the Education column, there are values “5” and “6”, which do not correspond to anything unlike the other values. No. of observations left = 29655 from original 30,000

```
data <- data %>% filter(MARRIAGE!=0)
```

In the Marriage column, there are values “0”, which do not correspond to anything unlike the other values.

PAY_0, PAY_2, PAY_3, PAY_4, PAY_5 and PAY_6 have invalid data of -2 and 0 present. However, should we choose to remove it, too many data points and potential information from other variables will be lost. Also due to the lack of ground truth, we have chosen not to make any assumptions about the data and preserve the data points.

3.3 Removing the outliers

Each of the attributes for each data point do not follow a normal distribution and some variables (or attributes) display covariance with one another. Hence, the Mahalanobis Distance can be used as a possible approach in identifying outliers. While the Mahalanobis Distance is designed with Gaussian distributions at its core, having a joint multivariate normal distribution is not a *prerequisite* for using the Mahalanobis distance as it will still improve the objective functions to a certain extent with respect to its variables/attributes.

All ordinal variables will be considering during this outlier analysis. The threshold distance is set to 10 for a data point to be an outlier.

3.4 Ignoring multicollinearity in variables

Upon analysing the data set, there are high levels of collinearity between the variables concerning Bill Amount (X12-X17) while low to moderate levels of collinearity exist for all other variables. The effect of low to moderate levels of collinearity on the accuracy with which the effects of a predictor variable on a target can be estimated is fairly minimal, but the level of precision drops substantially as the level of collinearity becomes very high. As a result, there is a need to be very concerned about high levels of collinearity, but not about moderate or low levels of collinearity.

That being said, even in the presence of high collinearity, acceptable levels of precision in determining the effect of predictors on the target can be achieved if there is a sufficient number of rows of data available to estimate a model. Hence, given the large amount of data available, we have decided not to remove variables that have shown high multicollinearity with each other.

Multicollinearity is also often addressed as it affects the coefficient estimates of the models. However, in this case, we are not interested in the coefficients themselves, but rather the accuracy of our model predictions, and checking the Variance Inflation Factor (VIF) of the variables will not answer a consequential question or change the predictive power of the model drastically.

---

04 Feature Selection

4.1 Definition

In machine learning and statistics, feature selection is the process of automatically or manually selecting a subset of relevant features that contribute most to the prediction variable or output of interest. The inclusion of irrelevant features in the data will thus decrease the overall prediction accuracy of the model.

Feature selection methods help to reduce the dimensionality of the data without much loss of the total information. It also reduces overfitting of the model. By conducting feature selection, we can speed up the training process of the machine learning algorithm. Feature selection also reduces the complexity of the model and makes it easier to interpret. With a simpler model, we can begin to understand what features are important and figure out how they contribute towards the prediction of the output variable. Lastly, it also improves the accuracy of the model if the right subset is chosen.

As Occam's Razor Principle states “the simplest models are the best”, thus we aim to achieve a model that uses the best and most relevant features while achieving the same or higher predictive accuracy.

4.2 Feature Selection using Filter Methods

4.2.1 Correlation

Correlation gives us the degree of association between two numeric variables. Features are selected based on their scores in various statistical tests for their correlation with the response variable. Generally, features with a high correlation with the response variable will be selected.

From the correlation matrix, as shown in the annex, we can see that the independent variables have a relatively low correlation coefficient score with the dependent variable (default payment next month). Thus, in this case, it might be difficult to use correlation to conduct feature selection. This may also indicate that a non-linear model might be more suitable for this dataset.

4.2.2 Hypothesis Testing (t-test and Chi-square Test)

Hypothesis testing is done to check if the independent variables have a significant relationship with the dependent variable. The t-test measures how significant the differences between the two populations are and will be used to test for association between continuous independent variables and the dependent variable. The Chi-square test is a statistical test that measures the association between two categorical variables and will be used to test for association between categorical independent variables and the dependent variable. The following describes the null and alternate hypothesis:

a. Null Hypothesis: No relationship exists
b. Alternate Hypothesis: Relationship exists

If the p-value obtained is less than the alpha-value (0.05), we reject the null hypothesis and conclude that we should include that variable in the model as it has a significant relationship with the response variable.

From the t-test, the following continuous variables, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6 have p-values of 0.0566, 0.140, 0.373, 0.538, 0.938 respectively. Since these p-values > 0.05, it suggests that these five continuous variables should not be included in our models since they do not have a significant contribution to the dependent variable. On the other hand, all the other continuous variables have p-value < 0.05 and contribute significantly to the dependent variable. Thus, the rest of the continuous variables should be included.

From the Chi-square test, other than the categorical variables SEX and MARRIAGE (p-values 0.000451 and 0.00911 respectively), the rest of the categorical variables have p-values that tends to zero. Since all these values are all clearly < 0.05, we can conclude that all the categorical variables are significant contributors to the dependent variable.

Thus, the following 18 features would be selected LIMIT_BAL, AGE, BILL_AMT1, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6, SEX, EDUCATION, MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6.

4.2.3 Information Gain

Information gain tells us how much information is given by the independent variable on the dependent variable. Features are selected based on their information gain score. Generally, features with a higher information gain score is selected to be included in the model.

From the results, we can see that the following 15 variables have non-zero attr_importance score: LIMIT_BAL, EDUCATION, AGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6.

The remaining 8 variables have a zero attri_importance score: SEX, MARRIAGE, BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6.

Thus, the following 15 variables would be selected: LIMIT_BAL, EDUCATION, AGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6.

4.3 Feature Selection using Wrapper Methods

4.3.1 Stepwise Forward and Backward Selection

Stepwise regression is a way to build a model by adding or removing predictor variables. The following are different methods of stepwise regression:

a. Forward selection - The algorithm starts with an empty model and progressively adds on significant variables to the model.
b. Backward selection - The algorithm starts with all the variables in the model and progressively deletes the least significant features
c. Stepwise selection - A hybrid of both forward and backward selection. At each iteration, a variable is considered for addition or deletion from the model.

Output:
The following variables were selected from the stepwise regression selection.

```
> print(vars_step)
 [1] "PAY_0"     "BILL_AMT1" "PAY_3"     "PAY_AMT1"  "BILL_AMT6" "PAY_AMT6"  "MARRIAGE"
 "PAY_2"     "PAY_AMT5"  "BILL_AMT2" "PAY_5"
[12] "PAY_AMT3"  "BILL_AMT5"
> print(vars_forward)
 [1] "PAY_0"     "BILL_AMT1" "PAY_3"     "PAY_AMT1"  "BILL_AMT6" "PAY_AMT6"  "MARRIAGE"
 "PAY_2"     "PAY_AMT5"  "BILL_AMT2" "PAY_5"
[12] "PAY_AMT3"  "BILL_AMT5"
> print(vars_backward)
 [1] "MARRIAGE"  "PAY_0"     "PAY_2"     "PAY_3"     "PAY_5"     "BILL_AMT1" "BILL_AMT2"
 "BILL_AMT5" "BILL_AMT6" "PAY_AMT1"  "PAY_AMT3"
[12] "PAY_AMT5"  "PAY_AMT6"
```

4.3.2 Recursive Feature Elimination (RFE) Method
A technique in which a model is constructed with all the variables initially. The algorithm then progresses to remove the least significant features one by one until it reaches the specified number of features (need to specify number of features to be included). The optimal number of features to be included can be identified using cross-validation.

From the results of RFE, as shown in the annex, the highest accuracy rate consists of the following 2 variables: PAY_0 and PAY_2.

4.4 Feature Selection using Embedded Methods

4.4.1 Least Absolute Shrinkage and Selection Operator (Lasso)

A technique that performs regularisation and feature selection. The method shrinks (regularises) the coefficients of the regression model as part of the penalisation. For feature selection, the variables which remain after the shrinkage process are included in the model.

As seen in *Figure 4.1.1* in the annex, we are unable to make inferences about the importance of the coefficients as the data has only been scaled individually and not scaled to have a common mean and standard deviation. Since our variables have different means and standard deviation, variables with larger averages will tend to have larger absolute coefficients.
Even so, we are still able to identify variables that have been definitely dropped. Any variable with a coefficient of zero has been dropped from the model, meaning that it was insignificant in prediction. The following 2 variables BILL_AMT2 and BILL_AMT5 has a coefficient of zero. Thus, the remaining 21 variables will be considered to be selected.

4.4.2 Random Forest

A technique that builds a random forest model and then extracts the list of significant variables by importance. Here, we look at the features used by machine learning algorithms such as random forest.
From the results as shown in *Output 4.4.2*, we can see that the randomForest achieved an optimal model with the following 2 variables PAY_0 and PAY_2. Additionally, the output and plot of variable importance shows quantitatively and visually the importance of these 2 variables.

4.4.3 Recursive Partitioning and Regression Trees (Rpart)

The rpart algorithm works by splitting the dataset recursively. The subsets that arise from a split are further split until a predetermined termination criterion is fulfilled. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the dependent (predicted) variable.

From the results shown in *Output 4.4.3*, we can see that the rPart model scored non-zero variable importance scores for the following 5 variables PAY_0, PAY_2, PAY_3, PAY_4, PAY_5. Thus, the rPart model confirmed these variables.

4.4.4 Boruta

Boruta algorithm is a feature selection algorithm. Boruta is a wrapper built around the random forest classification algorithm. At each iteration, the algorithm checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the features has a higher Z score than the maximum Z score of its shadow features) and constantly removes features which are deemed highly unimportant. The algorithm terminates when all features get confirmed or rejected or until it reaches a specified limit of random forest runs.

From the results as shown in *Figure 4.4.3*, we can see that the Boruta model confirmed the following 6 variables PAY_0, PAY_2, PAY_3, PAY_4, BILL_AMT4, PAY_5.

4.5 Comparison Between Methods

Type	Method	Number	Features Selected
Filter	Correlation	N.A	N.A.
	Hypothesis Testing	18	LIMIT_BAL, AGE, BILL_AMT1, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6, SEX, EDUCATION, MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6
	Information Gain	15	LIMIT_BAL, EDUCATION, AGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6.
Wrapper	Stepwise Regression	13	Forward: PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5, PAY_AMT3, BILL_AMT5
		13	Backward: MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_5, BILL_AMT1, BILL_AMT2, BILL_AMT5, BILL_AMT6, PAY_AMT1, PAY_AMT3, PAY_AMT5, PAY_AMT6
		13	Both: PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5, PAY_AMT3, BILL_AMT5
	Recursive Feature Elimination	2	PAY_0, PAY_2
Embedded	LASSO	21	LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6, BILL_AMT1, BILL_AMT3, BILL_AMT4, BILL_AMT6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6

	Random Forest	2	PAY_0, PAY_2
	Rpart	5	PAY_0, PAY_2, PAY_3, PAY_4, PAY_5
	Boruta	6	PAY_0, PAY_2, PAY_3, PAY_4, BILL_AMT4, PAY_5

We can see that we obtain very different set of features from each method. While some methods cannot provide a definite set of features to be included, they can help us filter out certain features for consideration. The table above shows the set of features that we have identified (leniently) from each model.

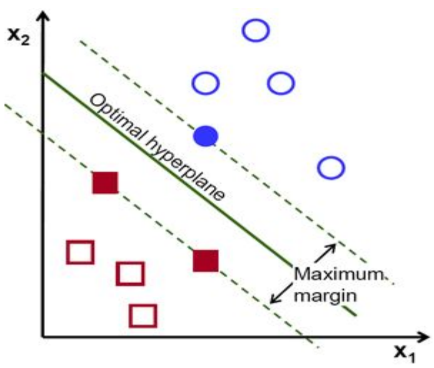
As such, we conducted several tests with the different sets of features (namely from Boruta, RFE, Rpart and Stepwise Regression since they selected the least number of features), and found that we achieved the highest predictive accuracy when using only PAY_0 and PAY_2 as our inputs (further discussed in chapter 5 of the report). Moreover, it can be noted that PAY_0 and PAY_2 has been consistently selected as one of the top few variables across all models. Since it is beneficial for us to keep the model simple (so that we may explain how the prediction of the model was done), we have decided to use PAY_0 and PAY_2 as our inputs.

---

05 Model Selection

5.1 Support Vector Machine (SVM)

In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.



SVM works very well with a clear margin of separation and effective in high dimensional spaces, making it ideal for our case. It also uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. However, it does not perform well in the case where we have huge data set because the required training time is higher.

5.1.1 Testing of the Kernels

The following model summary table displays the prediction accuracy of the neural network training based on the various kernel methods, namely Linear, Polynomial and Radial.

Output from testing all SVM kernels:

	Train Data Accuracy	**Test Data Accuracy**
Linear SVM	79.48%	79.79%
Polynomial SVM	80.20%	78.89%
Radial SVM	81.20%	80.88%

From the results above, we have decided to pick the radial kernel due to its superiority in its accuracy.

5.1.2 Measurement of Model Performance

	Train Data Accuracy	**Test Data Accuracy**
Boruta (PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4)	80.91%	81.09%
RFE and RF (PAY_0, PAY_2)	80.62%	80.94%
RPart (PAY_0, PAY_2, PAY_3, PAY_4, PAY_5)	80.82%	81.09%
Stepwise (PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5)	80.99%	80.89%

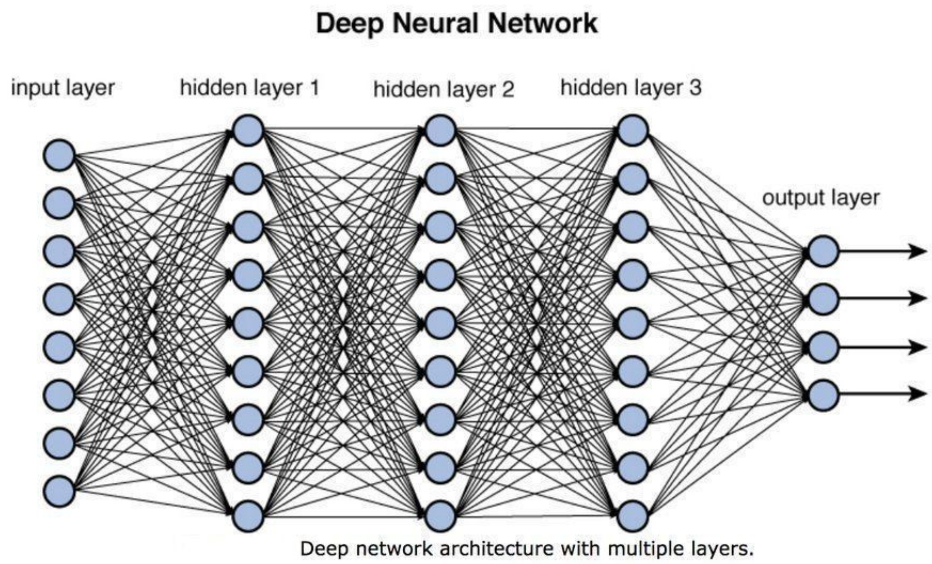
5.2 Decision Tree (RandomForest)

Random forests consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest generates a class prediction, which contributes to a vote in the final prediction. This final prediction will consider the class prediction with the most number of votes. As decision trees are very sensitive to the data they are trained on, small changes to the training set can result in significantly different tree structures. Random forests take advantage of this by allowing each individual tree to randomly sample from the dataset with replacement (bagging / bootstrap aggregation), resulting in different trees. Additionally, each tree in a random forest select features at random, which forces even more variation amongst the trees in the model, resulting in lower correlation across trees and greater diversification. Hence, uncorrelated trees are created, buffer and protect each other from potential errors that they might have inherently.

5.2.1 Measurement of Model Performance

	Training Data Accuracy	**Test Data Accuracy**
Boruta `PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4`	82.21%	81.45%
RFE and RF `PAY_0, PAY_2`	82.44%	81.6%
RPart `PAY_0, PAY_2, PAY_3, PAY_4, PAY_5`	82.12%	81.6%
Stepwise `(PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5)`	81.55%	81%

5.3 Neural Network



A Neural Networks (NN) is an advanced machine learning model that consists of nodes, also known as neurons, that are each able to make mathematical decisions. When put all together, these neurons can analyse complex problems and provide accurate predictions. There are, however, downfalls to the use of NNs. There is low explanatory power due to the "black box" nature of the network and it is also relies heavily on training data which then leads to the problem of generalisation and over-fitting.

5.3.1 Neural Network Structure

Neural Networks consist of 3 types of layers of neurons: input, hidden and output. Each input node is an independent variable and holds a weight attached to it, the hidden layer nodes combines all of these to match with the output nodes, which are also known as target or dependent variables. A shallow NN has only 1 hidden layer while a Deep NN has more than 1 hidden layer which helps to significantly improve its predictive power.
The NN model is constructed with the Mutli-layer Perceptron (MLP) supervised algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where *m* and *n* are the number of dimensions for input and output respectively. Given a set of inputs and a target, it is able to learn non-linear functions for regression or classification.

5.3.2 Measurement of Model Performance

	Training Data Accuracy	**Test Data Accuracy**
Boruta `PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4`	81.2859%	80.7167%
RFE and RF `PAY_0, PAY_2`	80.5243%	80.9164%
RPart `PAY_0, PAY_2, PAY_3, PAY_4, PAY_5`	81.0487%	80.6730%
Stepwise `PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5`	81.7478%	80.4045%

5.4 Logistic Regression

Since we are faced with a binary classification problem, a logistic regression model can be used here.
With the features selected from the RFE section, the mathematical equation for the logistic regression model is:

$$E[Y = 1|x_i] = \frac{1}{1-e^{0.47\times PAY00-1\ +\ 0.21\times PAY10\ +\ldots-\ 12.6\times PAY28}}$$

Where Y=1 is the event of a default payment, PAY00-1 is the binary variable whether PAY_0 is of the value -1, .., PAY28 is the binary variable whether PAY_2is of the value 8. When tested on the test data set, the model scored an 80.75% accuracy and had an ROC area of 0.72.

5.4.1 Measurement of Model Performance

	Training Data Accuracy	Test Data Accuracy
Boruta PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4	81.87%	80.44%
RFE and RF PAY_0, PAY_2	81.02%	80.75%
RPart PAY_0, PAY_2, PAY_3, PAY_4, PAY_5	81.90%	80.47%
Stepwise PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5	81.76%	80.45%

5.5 Naive Bayes Classifier

Naive Bayes is a Supervised Machine Learning algorithm that is based on the Bayes Theorem used to solve classification problems through a probabilistic approach. The principle behind Naive Bayes is the Bayes theorem, which is used to calculate the conditional probability - the probability of an event occurring based on information about events in the past. Mathematically, the Bayes theorem is represented as $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. A Naive Bayes model converges much quicker than other discriminative models like Logistic Regression and it requires less training data assuming the NB independence assumption holds. Even if the assumption does not hold, a NB classifier will still be effective.

5.5.1 Measurement of Model Performance

	Training Data Accuracy	Test Data Accuracy
Boruta PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4	78.90%	79.30%
RFE and RF PAY_0, PAY_2	79.40%	80.10%
RPart PAY_0, PAY_2, PAY_3, PAY_4, PAY_5	78.90%	79.30%
Stepwise PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_5	78.90%	79.30%

06 Model Evaluation

6.1 Accuracy

As the combination of variables PAY_0 and PAY_2 produce the highest accuracy among the other combinations, the variation of these accuracies will be evaluated with different folds of cross-validations.

The table below shows the accuracy of the test data with with k-fold cross validations:

Model		Accuracy			
		2-fold	5-fold	10-fold	15-fold
Support Vector Machine (SVM)	Linear	0.7754221	0.7754221	0.7754221	0.7754221
	Polynomial	0.7758147	0.7758147	0.7758147	0.7758147
	Radial	0.7758147	0.7750294	0.7754221	0.7754221
Decision Tree (Random Forest)		0.7746368	0.7726737	0.7726737	0.7734590
Neural Network		0.7738516	0.7687475	0.770318	0.7714959
Logistic Regression		0.7754221	0.7754221	0.7754221	0.7754221
Naive Bayes		0.7781704	0.7781704	0.7781704	0.7781704

As the accuracies have very little variation with different folds (cross validations), predictions with test data using models trained based on the two variables are fairly robust.

Considering the accuracies with and without k-fold cross validations being extremely close over the different models, it is not definitive which of them gives the best performance. Several future research directions also emerge.

First off, we only have one dataset to validate any proposed model in this study. In the further study, larger datasets should be collected to come to a more concrete conclusion. Perhaps also due to incorrect data inputs (especially in the PAY attributes), it had negatively influenced the accuracy of our models. Secondly, other models could be considered and tested in the next research. Thirdly, we found that testing different combinations of attributes after using the various feature selection methods produced improved results. Thus, these methods should be carried over and researched more in the next research.

07 Room for Improvement

7.1 Multi-collinearity

As discussed in chapter 3.4 and 4.2.1 of the report, we have noticed that several of the predictor variables might be highly correlated with each other. This might suggest the presence of multicollinearity in the data that could lead to impreciseness in the predictions in terms of the certainty in ascribing parameters, which although is not a major concern in machine learning, will still have an impact on the accuracy of the predictions. Thus, it might be better to remove correlated variables to improve the model.

However, even in the presence of multicollinearity, acceptable levels of precision in determining the effect of predictors on the target can be achieved since there are a sufficient number of rows of data available to estimate the model. In that sense, multicollinearity might not pose a major problem.

7.2 Class imbalance

We have also noticed class imbalance in the dataset. The given dataset has more than 80% of the recordings involving outcome = 1, leaving 20% with outcome = 0. As such, percentage error when predicting outcome = 0 is much higher than outcome = 1. We will be able to obtain a more balanced and accurate model with less bias if we can collect more data with outcome = 0.

---

08 References

Analytics Vidhya Content Team | 2016. Practical Guide to deal with Imbalanced Classification Problems in R. Retrieved from: https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

Chaitanya Sagar | 2018. Feature Selection Techniques with R. Retrieved from: https://dataaspirant.com/2018/01/15/feature-selection-techniques-r/

Mohit Sharma | 2018. Functions and Packages for Feature Selection in R. Retrieved from: https://datasciencebeginners.com/2018/11/26/functions-and-packages-for-feature-selection-in-r/

Mohit Sharma | 2018. Ultimate Practical guide to Hypothesis Testing. Retrieved from: https://datasciencebeginners.com/2018/10/04/06-ultimate-practical-guide-to-hypothesis-testing/

Steffen | 2016. Outlier Detection with Mahalanobis Distance. Retrieved from: https://www.r-bloggers.com/outlier-detection-with-mahalanobis-distance/

Christoph Bergmeir and Jose M. Benitez | 2012. RSNNS: Neural Networks using the Stuttgart Neural Network Simulator (SNNS). Retrieved from: https://rdrr.io/cran/RSNNS/src/R/mlp.R

Stephanie | 2017. Mahalanobis Distance: Simple Definition, Examples. Retrieved from: https://www.statisticshowto.datasciencecentral.com/mahalanobis-distance/

---

09 Annex

Code 2.3 Checking for categorical data

```
par(mfrow=c(4,6))
for (i in 2:25) {
  hist(data[,i],col="gray", border="white", freq=FALSE)
  d = density(data[,i])
  lines(d,col="red")
}
```

Code 2.4 Distribution of values

```
boxplot(data$LIMIT_BAL, col = 'mediumseagreen', ylab = "Amount of
       Given Credit" )
boxplot(data$AGE, col = 'mediumseagreen', ylab = "Age")
boxplot(data[,c(12:17)], col = 'mediumseagreen', ylab = c("Amount
       of Bill Statement"), xaxt="n", ann=FALSE)
axis(1, at=1:6, labels=c("Sep","Aug","Jul","Jun","May","Apr"))
boxplot(data[,c(18:23)], col = 'mediumseagreen', ylab = "Amount
of
       Previous Payment", xaxt="n", ann=FALSE)
axis(1, at=1:6, labels=c("Sep","Aug","Jul","Jun","May","Apr"))
```

Code 2.6 Class imbalance of dependent variable

```
data$`default payment next month` <-
                      as.factor(data$`default payment next
month`)
summary(data$`default payment next month`)
#    0     1
#23364  6636
```

Code 3.3 Removing the outliers

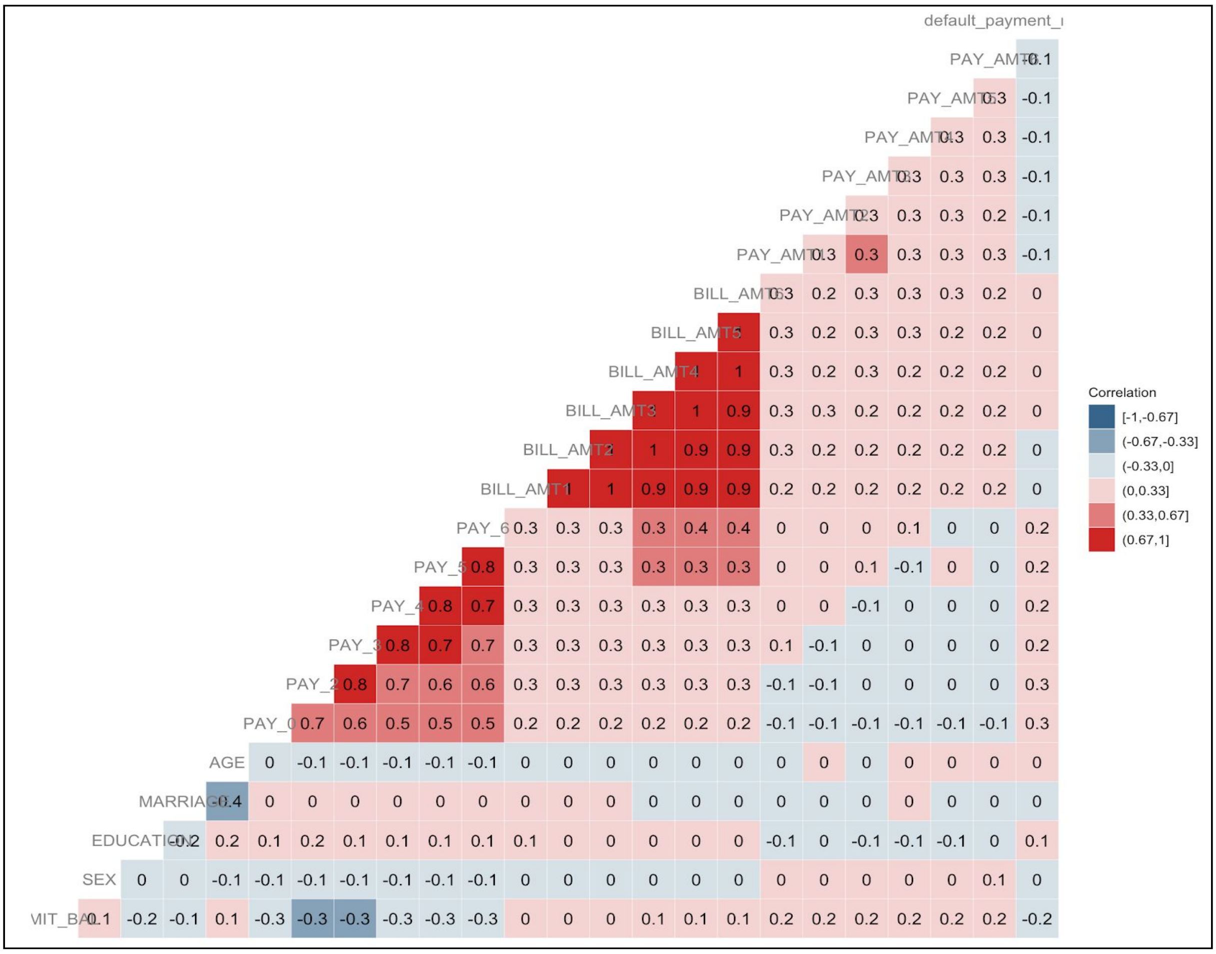
```
MD <- mahalanobis(data[,c(1,5,12:23)],
                   colMeans(data[,c(1,5,12:23)]),
                   cov(data[,c(1,5,12:23)]) )
data <- data[(which(MD <= 10)),]
```

Code 4.2.1 Correlation

```
## Correlation ##
library(GGally)
set.seed(123)

# Plotting the correlation matrix
ggcorr(train.data, nbreaks = 6, low = "steelblue4", mid =
"white",
       high = "firebrick3", label = TRUE,
```

```
        label_size = 4, color = "grey50")
```

Figure 4.2.1 Correlation



Code 4.2.2 Hypothesis Testing (t-test and Chi-square Test)

```
## Hypothesis Testing ##
set.seed(123)

# Running independent t-tests for continuous variables
cont = c(1,5,12,13,14,15,16,17,18,19,20,21,22,23)
for (i in cont) {
  print(t.test(train.data[,i], train.data$`default payment next
month`)$p.value)
}

# Running Chi-square tests for categorical variables
library(MASS)
cat = c(2,3,4,6,7,8,9,10,11)
for (i in cat) {
  tbl = table(train.data$`default payment next month`,
train.data[,i])
```

```
  print(colnames(data[i]))
  print(chisq.test(tbl)$p.value)
}
```

Output 4.2.2 Hypothesis Testing (t-test and Chi-square Test)

```
# Continuous variables
LIMIT_BAL      9.79195e-50
AGE            0.01215497
BILL_AMT1      0.003342555
BILL_AMT2      0.05659987
BILL_AMT3      0.1396438
BILL_AMT4      0.3732455
BILL_AMT5      0.5378581
BILL_AMT6      0.9380961
PAY_AMT1       2.34353e-26
PAY_AMT2       2.739637e-40
PAY_AMT3       5.054825e-29
PAY_AMT4       1.358612e-20
PAY_AMT5       1.418405e-19
PAY_AMT6       5.093679e-33

# Categorical variables
SEX            0.0004508242
EDUCATION      4.475987e-08
MARRIAGE       0.009107802
PAY_0          5.366234e-296
PAY_2          2.495613e-196
PAY_3          1.514754e-154
PAY_4          5.085097e-139
PAY_5          1.006184e-119
PAY_6          3.365915e-105
```

Code 4.2.3 Information Gain

```
## Information Gain ##
library(FSelector)
set.seed(123)

information.gain(`default payment next month`~., data=train.data)
```

Output 4.2.3 Information Gain

```
          attr_importance
LIMIT_BAL      0.017139961
SEX            0.000000000
EDUCATION      0.002146855
MARRIAGE       0.000000000
AGE            0.001324768
PAY_0          0.072439276
```

```
PAY_2         0.046543382
PAY_3         0.035829962
PAY_4         0.030849410
PAY_5         0.028389276
PAY_6         0.025464918
BILL_AMT1     0.000000000
BILL_AMT2     0.000000000
BILL_AMT3     0.000000000
BILL_AMT4     0.000000000
BILL_AMT5     0.000000000
BILL_AMT6     0.000000000
PAY_AMT1      0.015397561
PAY_AMT2      0.011786481
PAY_AMT3      0.010973511
PAY_AMT4      0.008917437
PAY_AMT5      0.006851455
PAY_AMT6      0.007333324
```

Code 4.3.1 Stepwise Forward and Backward Selection

```
## Stepwise Forward and Backward Selection ##
set.seed(123)

# Step 1: Building the base intercept only model
base.mod <- lm(`default payment next month`~1 , data=train.data)

# Step 2: Building the full model with all predictors
all.mod <- lm(`default payment next month`~. , data=train.data)

# Step 3: Perform stepwise algorithm.
Direction=forwards/backwards/both determines the method
stepMod <- step(base.mod, scope=list(lower=base.mod,
upper=all.mod), direction="both", trace=0, steps=1000)
forwardMod <- step(base.mod, scope=list(lower=base.mod,
upper=all.mod), direction="forward", trace=0, steps=1000)
backwardMod <- step(all.mod, scope=list(lower=base.mod,
upper=all.mod), direction="backward", trace=0, steps=1000)

# Step 4: Get the selected variables.
vars_step <- names(unlist(stepMod[[1]]))
vars_step <- shortlistedVars_step[!shortlistedVars_step %in%
"(Intercept)"] # remove intercept
vars_forward <- names(unlist(forwardMod[[1]]))
vars_forward <- shortlistedVars_forward[!shortlistedVars_forward
%in% "(Intercept)"] # remove intercept
vars_backward <- names(unlist(backwardMod[[1]]))
vars_backward <-
shortlistedVars_backward[!shortlistedVars_backward %in%
"(Intercept)"] # remove intercept
```

```
# Step 5: Print the selected variables.
print(vars_step)
print(vars_forward)
print(vars_backward)
```

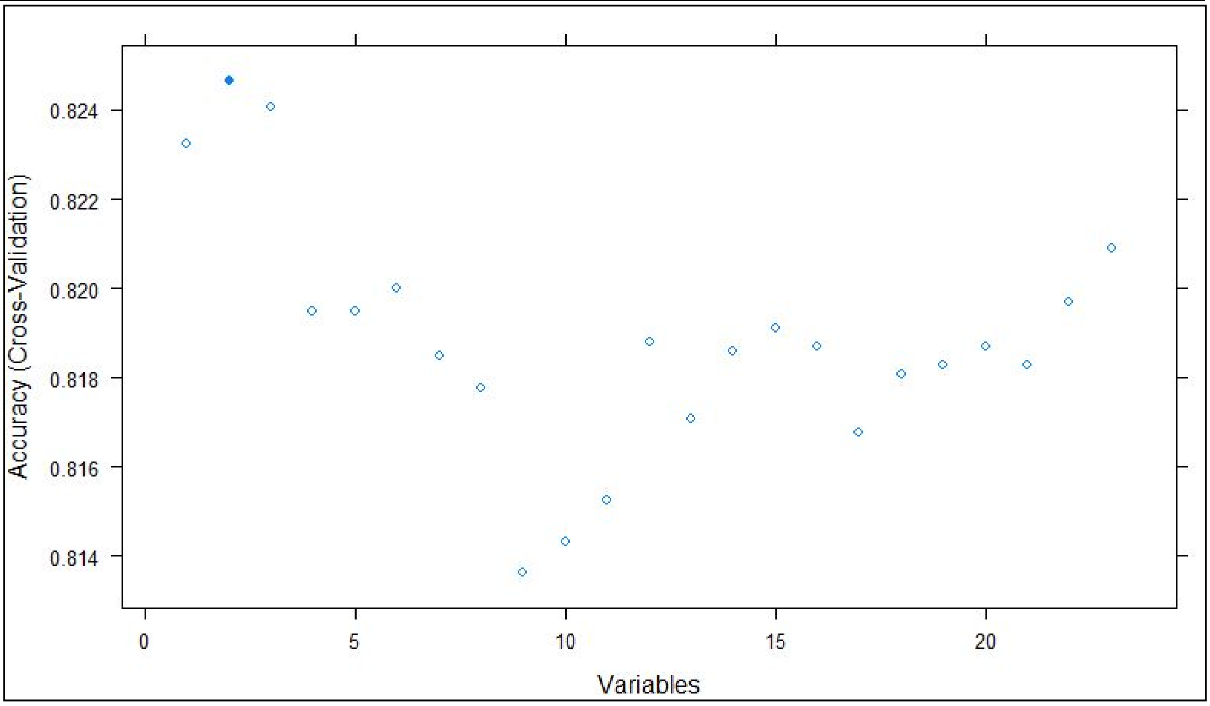
Code 4.3.2 Recursive Feature Elimination (RFE) Method

```
## Recursive Feature Elimination ##
library(caret)
set.seed(123)

control <- rfeControl(functions = rfFuncs, method = 'cv', number
= 10, allowParallel = TRUE, verbose = TRUE)
results <- rfe(train.data[,1:23], train.data$`default payment
next month`, sizes = c(1:23), rfeControl = control)

plot(results)
```

Output 4.3.2 Recursive Feature Elimination (RFE) Method

```
Recursive feature selection

Outer resampling method: Cross-Validated (10 fold)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.8232 0.3534   0.010474 0.04097
         2   0.8247 0.3572   0.010076 0.04022        *
         3   0.8241 0.3587   0.009979 0.03960
         4   0.8195 0.3482   0.011180 0.04312
         5   0.8195 0.3534   0.011450 0.04440
         6   0.8200 0.3554   0.010682 0.04080
         7   0.8185 0.3509   0.011544 0.04288
         8   0.8178 0.3495   0.011631 0.04516
         9   0.8136 0.3408   0.011613 0.04827
        10   0.8143 0.3413   0.011393 0.04627
        11   0.8152 0.3447   0.010989 0.04440
        12   0.8188 0.3551   0.009188 0.03737
        13   0.8171 0.3503   0.009018 0.03393
        14   0.8186 0.3562   0.009658 0.03572
        15   0.8191 0.3596   0.008613 0.03412
        16   0.8187 0.3576   0.009752 0.03256
        17   0.8168 0.3507   0.009629 0.03231
        18   0.8181 0.3576   0.009017 0.02901
        19   0.8183 0.3548   0.009008 0.03120
        20   0.8187 0.3568   0.009773 0.03407
        21   0.8183 0.3532   0.009353 0.03727
        22   0.8197 0.3579   0.008624 0.03347
        23   0.8209 0.3614   0.009382 0.03623
```

```
The top 2 variables (out of 2):
   PAY_0, PAY_2
```

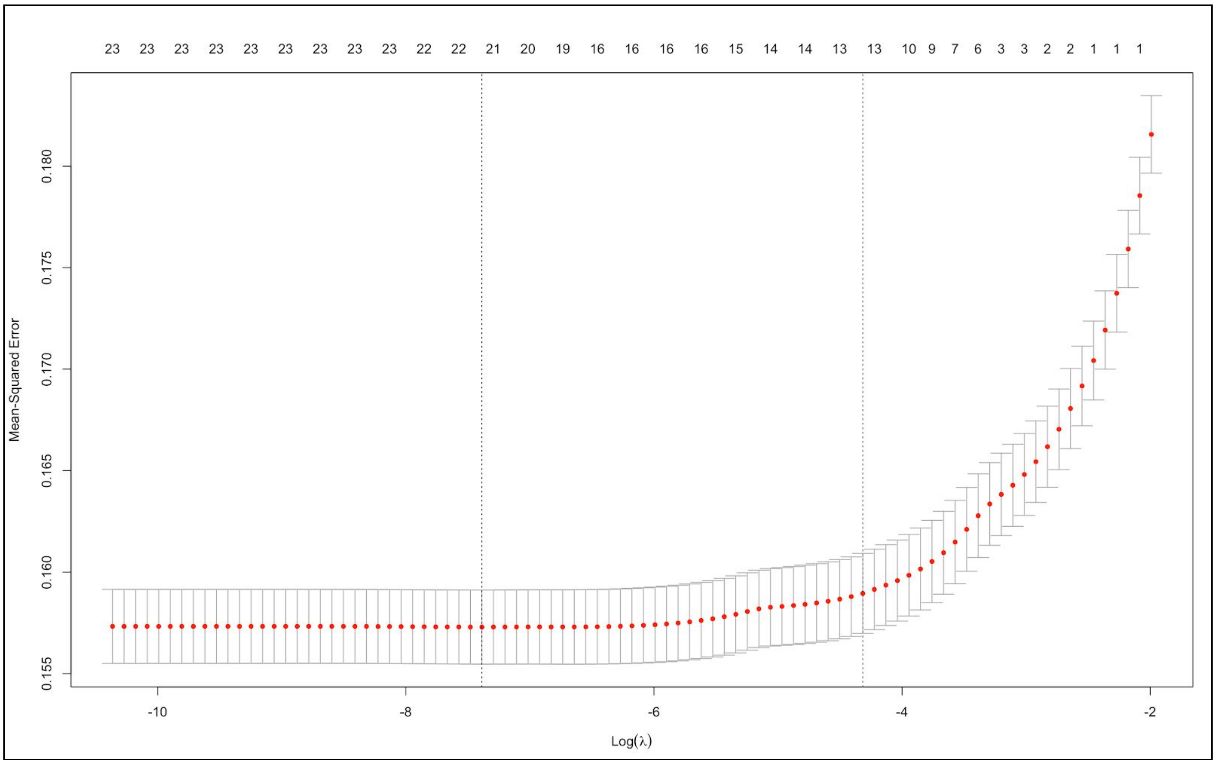


Code 4.4.1 Least Absolute Shrinkage and Selection Operator (Lasso)

```
## LASSO ##
library(glmnet)
set.seed(123)

# Building the LASSO model
feat_mod_select <- cv.glmnet(as.matrix(train.data[,2:24]) ,
train.data[, 25], standardize = TRUE, alpha =1)

# Checking coefficients with the minimum cross-validation error
as.matrix(round(coef(feat_mod_select,
feat_mod_select$lambda.min),5))

# Results
plot(feat_mod_select)
```

Figure 4.4.1 Least Absolute Shrinkage and Selection Operator (Lasso)

```
> as.matrix(round(coef(feat_mod_select, feat_mod_select$lambda.min),5))
                    1
(Intercept)   0.23957
LIMIT_BAL    -0.00327
SEX          -0.00079
EDUCATION    -0.00177
MARRIAGE     -0.02791
AGE           0.00648
PAY_0         0.08306
PAY_2         0.02271
PAY_3         0.01870
PAY_4         0.00478
PAY_5         0.01044
PAY_6        -0.00085
BILL_AMT1    -0.18546
BILL_AMT2     0.00000
BILL_AMT3     0.02615
BILL_AMT4     0.00583
BILL_AMT5     0.00000
BILL_AMT6     0.09527
PAY_AMT1     -0.06130
PAY_AMT2     -0.04563
PAY_AMT3     -0.03680
PAY_AMT4     -0.02567
PAY_AMT5     -0.03957
PAY_AMT6     -0.05584
```



Code 4.4.2 Random Forest

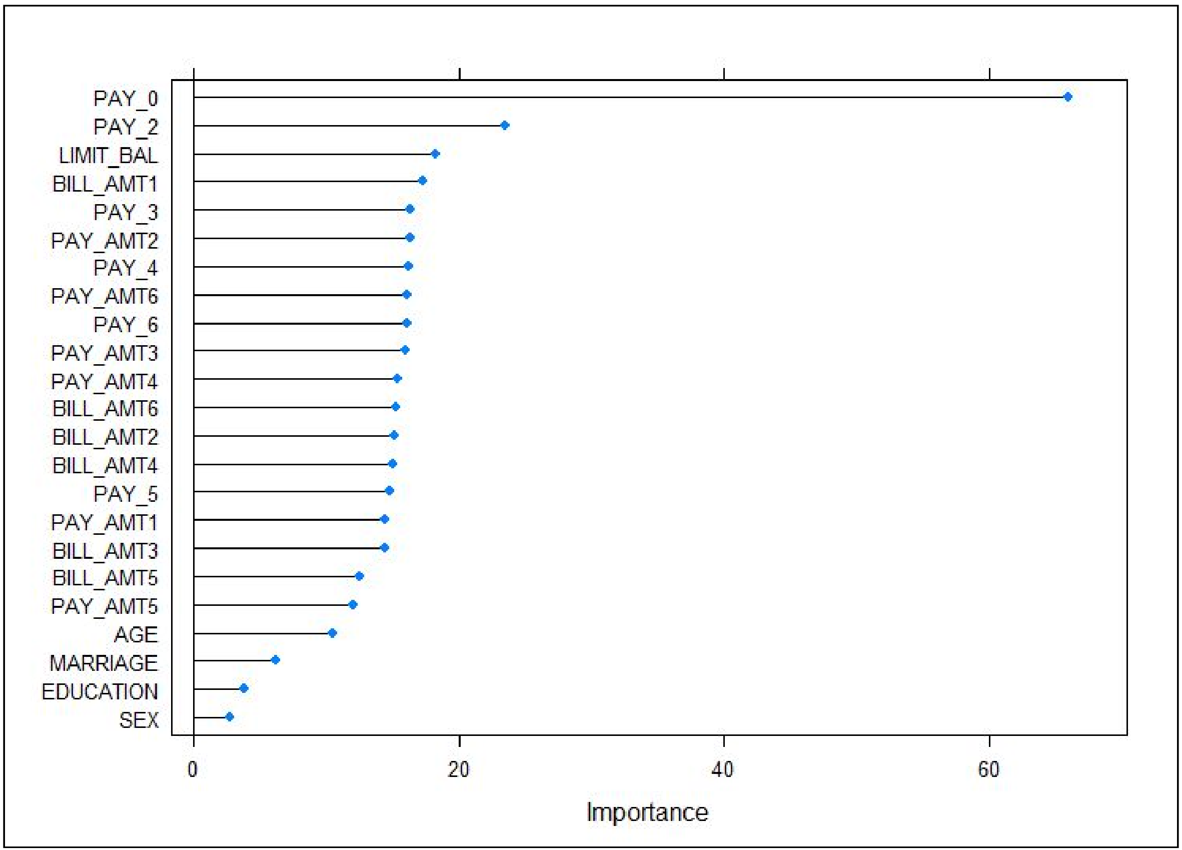
```
## RANDOM FOREST ##
library(caret)

control <- trainControl(method="cv", number=10,
                        verboseIter = TRUE, allowParallel = TRUE)
rfMod <- train(y=data[,24], x=data[,1:23], data=train.data,
               method="rf", preProcess = "scale", do.trace=TRUE,
               importance=T, ntrees=500, trControl=control)

# Checking Accuracy with test data
postResample(predict(rfMod, newdata = test.data[,1:23]),
                               test.data[,24])
```

Output 4.4.2 Random Forest

```
> rfMod
Random Forest

24029 samples
   23 predictor
    2 classes: '0', '1'

Pre-processing: scaled (23)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 21627, 21626, 21625, 21626, 21626,
21627, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8061097  0.3637849
  12    0.8045701  0.3744960
  23    0.8026968  0.3722652

Accuracy was used to select the optimal model using the largest
value.
The final value used for the model was mtry = 2.

> rfImp
rf variable importance

  only 20 most important variables shown (out of 23)

          Importance
PAY_0          65.95
PAY_2          23.50
LIMIT_BAL      18.24
BILL_AMT1      17.29
PAY_3          16.36
PAY_AMT2       16.29
```

```
PAY_4          16.28
PAY_AMT6       16.15
PAY_6          16.14
PAY_AMT3       15.99
PAY_AMT4       15.38
BILL_AMT6      15.32
BILL_AMT2      15.19
BILL_AMT4      15.01
PAY_5          14.83
PAY_AMT1       14.48
BILL_AMT3      14.39
BILL_AMT5      12.57
PAY_AMT5       12.05
AGE            10.46

> postResample(predict(rfMod, newdata = test.data[,1:23]),
                              test.data[,24])
 Accuracy     Kappa
0.7954292 0.1946404
```

Figure 4.4.2 Random Forest



Code 4.4.3 Recursive Partitioning and Regression Trees (Rpart)

```
## Rpart ##
library(caret)
```

```
set.seed(123)

# Building the rPart model
rPartMod <- train(`default payment next month` ~ .,
data=train.data, method="rpart")

# Showing the variable importance in the rPart model
rpartImp <- varImp(rPartMod)
print(rpartImp)
```

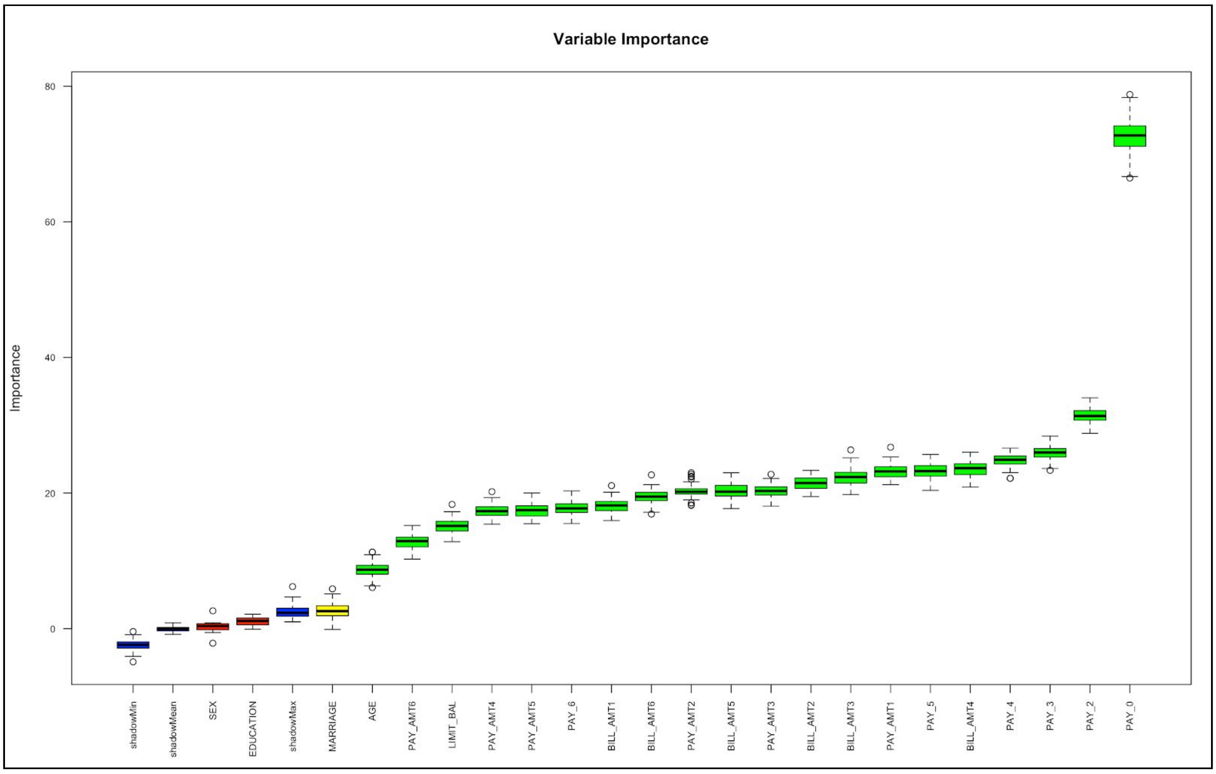
Output 4.4.3 Recursive Partitioning and Regression Trees (Rpart)

```
> print(rpartImp)
rpart variable importance

  only 20 most important variables shown (out of 23)

          Overall
PAY_0      100.00
PAY_2       88.68
PAY_3       70.60
PAY_4       62.86
PAY_5       53.43
PAY_AMT3     0.00
EDUCATION    0.00
SEX          0.00
MARRIAGE     0.00
LIMIT_BAL    0.00
PAY_AMT6     0.00
BILL_AMT6    0.00
PAY_AMT5     0.00
BILL_AMT5    0.00
PAY_6        0.00
BILL_AMT2    0.00
PAY_AMT2     0.00
BILL_AMT1    0.00
BILL_AMT4    0.00
PAY_AMT1     0.00
```

```
> print(boruta_signif)
 [1] "LIMIT_BAL" "AGE"       "PAY_0"     "PAY_2"     "PAY_3"     "PAY_4"     "PAY_5"
 "PAY_6"     "BILL_AMT1"
[10] "BILL_AMT2" "BILL_AMT3" "BILL_AMT4" "BILL_AMT5" "BILL_AMT6" "PAY_AMT1"  "PAY_AMT2"
 "PAY_AMT3"  "PAY_AMT4"
[19] "PAY_AMT5"  "PAY_AMT6"
```

```
> head(imps2[order(-imps2$meanImp), ])  # descending sort
           meanImp  decision
PAY_0     72.73111 Confirmed
PAY_2     31.46626 Confirmed
PAY_3     25.96638 Confirmed
PAY_4     24.87056 Confirmed
BILL_AMT4 23.54899 Confirmed
PAY_5     23.23604 Confirmed
```

Figure 4.4.3 Recursive Partitioning and Regression Trees (Rpart)



Code 4.4.4 Boruta

```
## Boruta ##
library(Boruta)
set.seed(123)

# Perform Boruta search
boruta_output <- Boruta(`default payment next month` ~ .,
data=na.omit(train.data), doTrace=0)

names(boruta_output)

# Get significant variables including tentatives
boruta_signif <- getSelectedAttributes(boruta_output,
withTentative = TRUE)
print(boruta_signif)

# Do a tentative rough fix
```

```
roughFixMod <- TentativeRoughFix(boruta_output)
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)

# Variable Importance Scores
imps <- attStats(roughFixMod)
imps2 = imps[imps$decision != 'Rejected', c('meanImp',
'decision')]
head(imps2[order(-imps2$meanImp), ])  # descending sort

# Plot variable importance
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable
Importance")
```

Code 4.4.5 Logistic Regression

```
library(caret)
set.seed(123)

# Build a logistic regression model
fit_glm <- glm(`default.payment.next.month`~., data = train.data,
               family = "binomial"(link='logit'))
Fit_glm
```

Output 4.4.5 Logistic Regression

```
Call:  glm(formula = default.payment.next.month ~ ., family =
binomial(link = "logit"),
    data = train.data)

Coefficients:
(Intercept)    LIMIT_BAL         SEX2   EDUCATION2
  -1.916607    -0.122635    -0.116570     0.059854
 EDUCATION3   EDUCATION4    MARRIAGE2    MARRIAGE3
   0.103213   -13.785407    -0.179043     0.125694
        AGE      PAY_0-1       PAY_00       PAY_01
  -0.006629     0.458762    -0.364332     0.576967
     PAY_02       PAY_03       PAY_04       PAY_05
   1.998281     2.137052     0.871128     0.079047
     PAY_06       PAY_07       PAY_08      PAY_2-1
  15.735629   -14.658609   -14.478463    -0.289800
     PAY_20       PAY_21       PAY_22       PAY_23
  -0.009256   -14.031990     0.005602     0.103743
     PAY_24       PAY_25       PAY_26       PAY_27
  -0.348061     0.381120    18.211195    15.759725
     PAY_28      PAY_3-1       PAY_30       PAY_31
 -30.484057     0.213140     0.160804    -0.249973
     PAY_32       PAY_33       PAY_34       PAY_35
   0.493012     0.799837     0.816343    -1.457895
     PAY_36       PAY_37      PAY_4-1       PAY_40
```

```
         NA    -0.355171    -0.010981    -0.031030
     PAY_42       PAY_43       PAY_44       PAY_45
   0.198684    -0.342907    -0.713820    -1.554402
     PAY_46       PAY_47       PAY_48      PAY_5-1
         NA    -1.617672   -31.670928    -0.139770
     PAY_50       PAY_52       PAY_53       PAY_54
   0.130763     0.485750     0.233759     1.251816
     PAY_55       PAY_57      PAY_6-1       PAY_60
  16.257247           NA    -0.031172    -0.307894
     PAY_62       PAY_63       PAY_64       PAY_65
  -0.089511     0.486351    -0.498760    -1.149247
     PAY_66       PAY_67    BILL_AMT1    BILL_AMT2
  15.670601     2.186298    -1.144227     0.735211
  BILL_AMT3    BILL_AMT4    BILL_AMT5    BILL_AMT6
   0.814144    -0.190303    -0.184521     0.076000
   PAY_AMT1     PAY_AMT2     PAY_AMT3     PAY_AMT4
  -0.340086    -0.593540    -0.646257    -0.194770
   PAY_AMT5     PAY_AMT6
  -0.415992    -0.377711

Degrees of Freedom: 8009 Total (i.e. Null);  7939 Residual
Null Deviance:      8830
Residual Deviance: 7145    AIC: 7287

round(varImp(fit_glm, scale = FALSE),2)
           Overall
LIMIT_BAL     2.63
SEX2          1.89
EDUCATION2    0.83
EDUCATION3    1.09
EDUCATION4    0.06
MARRIAGE2     2.59
MARRIAGE3     0.44
AGE           0.19
PAY_0-1       2.29
PAY_00        1.65
PAY_01        3.67
PAY_02       10.10
PAY_03        6.60
PAY_04        1.51
PAY_05        0.09
PAY_06        0.02
PAY_07        0.01
PAY_08        0.01
PAY_2-1       1.37
PAY_20        0.03
PAY_21        0.01
PAY_22        0.03
PAY_23        0.31
PAY_24        0.56
```

```
PAY_25        0.21
PAY_26        0.01
PAY_27        0.01
PAY_28        0.02
PAY_3-1       1.02
PAY_30        0.65
PAY_31        0.00
PAY_32        2.00
PAY_33        1.77
PAY_34        1.01
PAY_35        0.97
PAY_37        0.27
PAY_4-1       0.05
PAY_40        0.13
PAY_42        0.78
PAY_43        0.70
PAY_44        0.79
PAY_45        1.02
PAY_47        0.00
PAY_48        0.02
PAY_5-1       0.70
PAY_50        0.57
PAY_52        1.89
PAY_53        0.51
PAY_54        1.29
PAY_55        0.03
PAY_6-1       0.21
PAY_60        1.86
PAY_62        0.46
PAY_63        1.08
PAY_64        0.41
PAY_65        0.64
PAY_66        0.02
PAY_67        0.00
BILL_AMT1     2.75
BILL_AMT2     1.28
BILL_AMT3     1.75
BILL_AMT4     0.47
BILL_AMT5     0.37
BILL_AMT6     0.20
PAY_AMT1      1.84
PAY_AMT2      2.54
PAY_AMT3      2.99
PAY_AMT4      1.06
PAY_AMT5      2.29
PAY_AMT6      2.29
```

Code 5.1.1 Testing of the Kernels

```
##Support Vector Machine ##
library(e1071)
library(Metrics)
set.seed(123)

# Linear SVM
model_svm_linear <- svm(`default payment next month` ~. , data =
train.data, type = "C-classification", kernel = "linear")
pred_linear <- predict(model_svm_linear, test.data)
table(pred=pred_linear,actual=test.data$`default payment next
month`)
auc(pred_linear, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_linear)

# Polynomial SVM
model_svm_polynomial <- svm(`default payment next month` ~. ,
data = train.data, type = "C-classification", kernel =
"polynomial")
pred_polynomial <- predict(model_svm_polynomial, test.data)
table(pred=pred_polynomial,actual=test.data$`default payment next
month`)
auc(pred_polynomial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_polynomial)

# Radial SVM
model_svm_radial <- svm(`default payment next month` ~. , data =
train.data, type = "C-classification", kernel = "radial")
pred_radial <- predict(model_svm_radial, test.data)
table(pred=pred_radial,actual=test.data$`default payment next
month`)
auc(pred_radial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_radial)

# Sigmoid SVM
model_svm_sigmoid <- svm(`default payment next month` ~. , data =
train.data, type = "C-classification", kernel = "sigmoid")
pred_sigmoid <- predict(model_svm_sigmoid, test.data)
table(pred=pred_sigmoid,actual=test.data$`default payment next
month`)
auc(pred_sigmoid, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_sigmoid)
```

Code 5.1.2 Measurement of Model Performance

```
#Feature selection
#Boruta
model_svm_radial <- svm(`default payment next month` ~
`PAY_0`+`PAY_2`+`PAY_3`+`PAY_4`+`PAY_5`+`BILL_AMT4` , data =
train.data, type = "C-classification", kernel = "radial")
pred_radial <- predict(model_svm_radial, test.data)
```

```
table(pred=pred_radial,actual=test.data$`default payment next
month`)
auc(pred_radial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_radial)

#RFE
model_svm_radial <- svm(`default payment next month` ~
`PAY_0`+`PAY_2` , data = train.data, type = "C-classification",
kernel = "radial")
pred_radial <- predict(model_svm_radial, test.data)
table(pred=pred_radial,actual=test.data$`default payment next
month`)
auc(pred_radial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_radial)

#Rpart
model_svm_radial <- svm(`default payment next month` ~
`PAY_0`+`PAY_2`+`PAY_3`+`PAY_4`+`PAY_5` , data = train.data, type
= "C-classification", kernel = "radial")
pred_radial <- predict(model_svm_radial, test.data)
table(pred=pred_radial,actual=test.data$`default payment next
month`)
auc(pred_radial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_radial)

# vars_step
model_svm_radial <- svm(`default payment next month` ~
`PAY_0`+`BILL_AMT1`+`PAY_3`+`PAY_AMT1`+`BILL_AMT6`+`PAY_AMT6`+`MA
RRIAGE`+`PAY_2`+`PAY_AMT5`+`BILL_AMT2`+`PAY_5`, data =
train.data, type = "C-classification", kernel = "radial")
pred_radial <- predict(model_svm_radial, test.data)
table(pred=pred_radial,actual=test.data$`default payment next
month`)
auc(pred_radial, test.data$`default payment next month`)
mean(test.data$`default payment next month` == pred_radial)
```

Code 5.2 Decision Tree (Random Forest)

```
## Random Forest ##
library(randomForest)
set.seed(123)

# Feature Selection from Boruta Algorithm
rf_boruta <- randomForest(y = train.data[,24], x =
train.data[,c(6:10,15)], ytest = test.data[,24], xtest =
test.data[,c(6:10,15)], importance = TRUE)

# Feature Selection from RFE and RF (SAME)
rfMod_rfe <- train(y=data[,24], x=data[,c(6,7)], data=test.data,
method="rf", preProcess = "scale", do.trace=TRUE, importance=T,
```

```
ntrees=500, trControl=control)

rf_rfe <- randomForest(y = train.data[,24], x = train.data[,6:7],
ytest = test.data[,24], xtest = test.data[,6:7], importance =
TRUE)

# Feature Selection from RPart
rf_rpart <- randomForest(y = train.data[,24], x =
train.data[,6:10], ytest = test.data[,24], xtest =
test.data[,6:10], importance = TRUE)

# Feature Selection from STEPWISE
rf_step <- randomForest(y = train.data[,24], x =
train.data[,c(4,6,7,8,10,12,13,17,18,22,23)], ytest =
test.data[,24], xtest =
test.data[,c(4,6,7,8,10,12,13,17,18,22,23)], importance = TRUE)
```

Code 5.3 Neural Network

```
set.seed(123)
library(RSNNS)

##BORUTA
nnBoruta = mlp(train.data[,c(6:10,15)],train.data[,24],
               size = c(500), maxit = 800)
predictions <- predict(nnBoruta, train.in[,c(6:10,15)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == train.target) # 0.8128589
predictions <- predict(nnBoruta, test.in[,c(6:10,15)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == test.target) # 0.8071665

##RFE
nnRFE = mlp(train.data[,c(6:7)],train.data[,24],
            size = c(500), maxit = 800)
predictions <- predict(nnRFE, train.in[,c(6:7)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == train.target) # 0.8052434
predictions <- predict(nnRFE, test.in[,c(6:7)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == test.target) # 0.8091641

##RPART
nnRPART = mlp(train.data[,c(6:10)],train.data[,24],
              size = c(500), maxit = 800)
predictions <- predict(nnRPART, train.in[,c(6:10)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == train.target) # 0.8104869
predictions <- predict(nnRPART, test.in[,c(6:10)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
```

```
mean(pred.class == test.target) # 0.8067295

##STEPWISE
nnSTEP = mlp(train.data[,c(4,6:8,12,13,17,18,22,23)],
             train.data[,24],size = c(500), maxit = 800)
predictions <- predict(nnSTEP,
                       train.in[,c(4,6:8,12,13,17,18,22,23)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == train.target) # 0.8174782
predictions <- predict(nnSTEP,
              test.in[,c(4,6:8,12,13,17,18,22,23)])
pred.class = ifelse(predictions >= 0.5, 1 ,0)
mean(pred.class == test.target) # 0.8040452
```

Code 5.4 Logistic Regression

```
library(readr)
library(InformationValue)
library(dplyr)

set.seed(14)
n = nrow(data[,1])
index <- 1:nrow(data)
testindex <- sample(index, trunc(2*n)/3)
test.data <- data[testindex,]
train.data <- data[-testindex,]

# RFE
# PAY_0, PAY_2
fit_glm.RFE<- glm(default.payment.next.month~ PAY_0+ PAY_2,
                  data=train.data, family = "binomial")
summary(fit_glm.RFE)

train.pred <- predict(fit_glm.RFE, data=(train.data %>%
select(PAY_0, PAY_2)),
                      type="response")
plotROC(actuals=train.data[,24], predictedScores=train.pred)
optcut <- optimalCutoff(train.data[,24], train.pred,
optimiseFor="misclasserror")
train.binpred<-ifelse(train.pred < optcut,0,1)
   train.binpred
      0    1
  0 5789  298
  1 1222  701
table(train.data$default.payment.next.month, train.binpred)
mean(train.data[,24] == train.binpred) # training accuracy of
81.02%
0.8102372
test.pred<-predict(fit_glm.RFE, (test.data %>% select(PAY_0,
```

```
PAY_2)),
                   type="response")
plotROC(actuals=test.data[24], predictedScores=test.pred)
test.binpred <-ifelse(test.pred < optcut,0,1)
table(test.data$default.payment.next.month,test.binpred)
   test.binpred
        0     1
  0 11640   587
  1  2497  1295
mean(test.data[24] == test.binpred) # test accuracy of 80.75%
0.8074786


# Boruta
# PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4
set.seed(4)
n = nrow(data[,1])
index <- 1:nrow(data)
testindex <- sample(index, trunc(2*n)/3)
test.data <- data[testindex,]
train.data <- data[-testindex,]

fit_glm.boruta <- glm(default.payment.next.month~ PAY_0+ PAY_2+
PAY_3+ PAY_4+ PAY_5+ BILL_AMT4,
                      data=train.data, family = "binomial")
summary(fit_glm.boruta)

train.pred <- predict(fit_glm.boruta, data=(train.data %>%
select(PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4)),
                      type="response")
plotROC(actuals=train.data[,24], predictedScores=train.pred)
optcut <- optimalCutoff(train.data[,24], train.pred,
optimiseFor="misclasserror")
train.binpred<-ifelse(train.pred < optcut,0,1)
table(train.data$default.payment.next.month, train.binpred)
   train.binpred
       0    1
  0 5802  348
  1 1104  756
mean(train.data[,24] == train.binpred) # training accuracy 81.87%
0.8187266
test.pred<-predict(fit_glm.boruta, (test.data %>% select(PAY_0,
PAY_2, PAY_3, PAY_4, PAY_5, BILL_AMT4)),
                   type="response")
plotROC(actuals=(test.data$default.payment.next.month),
predictedScores=test.pred)
test.binpred <-ifelse(test.pred < optcut,0,1)
table((test.data)$default.payment.next.month,test.binpred)
   test.binpred
        0     1
```

```
  0 11417   747
  1  2387  1468
mean((test.data)[24] == test.binpred) # test accuracy 80.44%
0.8043573

# RPART
# PAY_0, PAY_2, PAY_3, PAY_4, PAY_5
fit_glm.RPART<- glm(default.payment.next.month~ PAY_0+
PAY_2+PAY_3+ PAY_4+ PAY_5,
                    data=train.data, family = "binomial")
summary(fit_glm.RPART)

train.pred <- predict(fit_glm.RPART, data=(train.data %>%
select(PAY_0, PAY_2, PAY_3, PAY_4, PAY_5)),
                      type="response")
plotROC(actuals=train.data[,24], predictedScores=train.pred)
optcut <- optimalCutoff(train.data[,24], train.pred,
optimiseFor="misclasserror")
train.binpred<-ifelse(train.pred < optcut,0,1)
table(train.data$default.payment.next.month, train.binpred)
   train.binpred
       0    1
  0 5851  299
  1 1151  709
mean(train.data[,24] == train.binpred)
 # training accuracy of 81.90%
0.8189763
test.pred<-predict(fit_glm.RPART, (test.data %>% select(PAY_0,
PAY_2, PAY_3, PAY_4, PAY_5)),
                   type="response")
plotROC(actuals=test.data[24], predictedScores=test.pred)
test.binpred <-ifelse(test.pred < optcut,0,1)
table(test.data$default.payment.next.month, test.binpred)
   test.binpred
        0     1
  0 11517   647
  1  2481  1374
mean(test.data[24] == test.binpred) # test accuracy of 80.47%
0.8047319


# STEPWISE
# PAY_0, BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6, PAY_AMT6,
MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2, PAY_AMT5
fit_glm.STEPWISE<- glm(default.payment.next.month~ PAY_0+
BILL_AMT1+ PAY_3+ PAY_AMT1+ BILL_AMT6+ PAY_AMT6
                       + MARRIAGE+ PAY_2+ PAY_AMT5+ BILL_AMT2+
PAY_AMT5, data=train.data, family = "binomial")
summary(fit_glm.STEPWISE)
```

```
train.pred <- predict(fit_glm.STEPWISE, data=(train.data %>%
select(PAY_0, BILL_AMT1, PAY_3, PAY_AMT1,

BILL_AMT6, PAY_AMT6, MARRIAGE, PAY_2,

PAY_AMT5, BILL_AMT2, PAY_AMT5)),
                      type="response")
plotROC(actuals=train.data[,24], predictedScores=train.pred)
optcut <- optimalCutoff(train.data[,24], train.pred,
optimiseFor="misclasserror")
train.binpred<-ifelse(train.pred < optcut,0,1)
table(train.data$default.payment.next.month, train.binpred)
   train.binpred
       0    1
  0 5811  339
  1 1122  738
mean(train.data[,24] == train.binpred) # accuracy of 81.76%
0.817603

test.pred<-predict(fit_glm.STEPWISE, (test.data %>% select(PAY_0,
BILL_AMT1, PAY_3, PAY_AMT1, BILL_AMT6,

PAY_AMT6, MARRIAGE, PAY_2, PAY_AMT5, BILL_AMT2,

PAY_AMT5)),
                   type="response")
plotROC(actuals=test.data[24], predictedScores = test.pred)
test.binpred <-ifelse(test.pred < optcut,0,1)
table(test.data$default.payment.next.month,test.binpred)
   test.binpred
        0     1
  0 11445   719
  1  2413  1442
mean(test.data[24] == test.binpred) # truncated test accuracy of
80.45%
0.8044822
```

Code 5.5 Naive Bayes Classifier

```
library(e1071)

NBclassfier=naiveBayes(as.factor(default.payment.next.month)~.,
data=train.data)
print(NBclassfier)

printALL=function(model){
  trainPred=predict(model, newdata = train.data, type = "class")
  trainTable=table(train.data$default.payment.next.month,
trainPred)
  testPred=predict(NBclassfier, newdata=test.data, type="class")
```

```
  testTable=table(test.data$default.payment.next.month, testPred)
  trainAcc=(trainTable[1,1]+trainTable[2,2])/sum(trainTable)
  testAcc=(testTable[1,1]+testTable[2,2])/sum(testTable)
  message("Contingency Table for Training Data")
  print(trainTable)
  message("Contingency Table for Test Data")
  print(testTable)
  message("Accuracy")
  print(round(cbind(trainAccuracy=trainAcc,
testAccuracy=testAcc),3))
}
printALL(NBclassfier)

Contingency Table for Training Data
   trainPred
       0    1
  0 2301  738
  1  412  578
Contingency Table for Test Data
   testPred
        0     1
  0 11382  3893
  1  1876  2849
Accuracy
     trainAccuracy testAccuracy
[1,]         0.715        0.712
```

Code 6.1 Accuracy

```
library(Metrics)
library(caret)
control <- trainControl(method="cv", number = k, verboseIter =
TRUE,
                        allowParallel = TRUE, classProbs = TRUE)
set.seed(123)

## SUPPORT VECTOR MACHINE ##
levels(train.data[,24]) <-
make.names(levels(factor(train.data[,24])))
levels(test.data[,24]) <-
make.names(levels(factor(test.data[,24])))
set.seed(123)

# LINEAR #
rf_rfe <- train(y=train.data[,24], x=train.data[,6:7],
                   method='svmLinear', preProcess =
"scale",importance=T,
                   trControl=control)
testPred <- predict(rf_rfe, newdata = test.data[,6:7])
accuracy(testPred, test.data[,24])
```

```
# POLYNOMIAL #
rf_rfe <- train(y=train.data[,24], x=train.data[,6:7],
                   method=’svmPoly’, preProcess =
"scale",importance=T,
                   trControl=control)
testPred <- predict(rf_rfe, newdata = test.data[,6:7])
accuracy(testPred, test.data[,24])


# RADIAL #
rf_rfe <- train(y=train.data[,24], x=train.data[,6:7],
                   method='svmRadial', preProcess =
"scale",importance=T,
                   trControl=control)
testPred <- predict(rf_rfe, newdata = test.data[,6:7])
accuracy(testPred, test.data[,24])

## RANDOM FOREST ##
rf_rfe <- train(y=train.data[,24], x=train.data[,c(6,7)],
method="rf",
                preProcess = "scale", do.trace=TRUE,
importance=T,
                ntrees=500, trControl=control)
postResample(predict(rf_rfe, newdata = test.data[,1:23]),
test.data[,24])

## NEURAL NETWORK ##
rf_rfe <- train(y=train.data[,24], x=train.data[,c(6,7)],
method=”mlp",
                preProcess = "scale", do.trace=TRUE,
importance=T,
                ntrees=500, trControl=control)
postResample(predict(rf_rfe, newdata = test.data[,1:23]),
test.data[,24])

## LOGISTIC REGRESSION ##
rf_rfe <- train(y=train.data[,24], x=train.data[,6:7],
method='glm',
                             family = ‘binomial’,
trControl=control)
testPred <- predict(rf_rfe, newdata = test.data[,6:7])
accuracy(testPred, test.data[,24])

## NAIVE BAYES ##
rf_rfe <- train(y=train.data[,24], x=train.data[,6:7],
                                 method='naive_bayes',
trControl=control)
testPred <- predict(rf_rfe, newdata = test.data[,6:7])
accuracy(testPred, test.data[,24])
```