# GROUP ASSIGNMENT ON DATA MINING

# (GROUP-3)

**Done by:**

1. Akshay Subramanian
2. Gokul
3. Priyanka Ponraj
4. Rajendran Swetha

**Tool used:** R Studio

**Submitted on:**22-02-2020

# TABLE OF CONTENTS

## ABOUT THE DATASET

Dataset name: Thera Bank-Data Set.xlsx

This dataset is all about earning interest from the loans given by “Thera Bank” to the potential customers (who are able to repay the loan). This dataset also plays a major role in finding the potential customer for disbursing the loans considering their **Income, Family size, Credit card usage, Education** and the amount of **mortgage** available in their name.

```
bank = bank%>% rename(Age = "Age (in years)", Experience = "Experience (in years)",
                      Income = "Income (in K/month)")
View(bank)
```

## PROBLEM STATEMENT

- Build a model which can help the bank to identify the customers who have high potential of purchasing the personal loan and to find the areas to be focused for making the customers to take the personal loan.

## ANALYSING THE DATASET

Since the given dataset is a *.xlsx file, to read that file, the package “**readxl**” has to be installed. On analyzing the column names of the dataset, we found that the column names **Age (in Years),Experience (in Years)**and **Income (in K/Month)** are not getting detected by the R studio during analysis as the word **in** is used as a keyword in R.

Hence, to avoid that error, the above mentioned column names have been renamed as **Age**, **Experience** and **Income** in the assigned data frame.

Number of records: 5000

Number of columns: 14

```
setwd("D:/BABI/BABI-5th Residency/Data Mining/Group Assignment")
getwd()
library(readxl) #package to read the excel file
bank <- read_excel("Thera Bank-Data Set.xlsx", sheet = "Bank_Personal_Loan_Modelling")
View(bank)
dim(bank) #checking the dimensions
```

```
> setwd("D:/BABI/BABI-5th Residency/Data Mining/Group Assignment")
> getwd()
[1] "D:/BABI/BABI-5th Residency/Data Mining/Group Assignment"
> library(readxl) #package to read the excel file
> bank <- read_excel("Thera Bank-Data Set.xlsx", sheet = "Bank_Personal_Loan_Modelling")
> View(bank)
> dim(bank) #checking the dimensions
[1] 5000   14
>
```

## INSTALLED PACKAGES

Since various models and functionalities have been used for analyzing the dataset, the following packages have been installed and loaded.

The packages are:

- readr
- deplyr
- ggplot2
- gridExtra
- lattice
- DataExplorer
- factoextra
- caret
- e1071
- rpart
- rpart.plot
- randomForest
- Metrics
- ROCit
- kableExtra

```
library(readr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(lattice)
library(DataExplorer)
library(factoextra)
library(caret)
library(e1071)
library(rpart)
library(rpart.plot)
library(randomForest)
library(Metrics)
library(ROCit)
library(kableExtra)
```

## PRELIMINARY ANALYSIS OF DATA SET

Since the first and fifth column of the dataset (**ID** and **ZIP code**) are not required for the analysis, they are dropped from the data frame.

```
bank = bank[,-c(1,5)] #dropping the first and 5th columns of the dataset
```

Summary of the data frame:

```
> summary(bank)
       ID        Age (in years)  Experience (in years) Income (in K/month)    ZIP Code    
 Min.   :   1   Min.   :23.00   Min.   :-3.0          Min.   :  8.00      Min.   : 9307  
 1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00      1st Qu.:91911  
 Median :2500   Median :45.00   Median :20.0          Median : 64.00      Median :93437  
 Mean   :2500   Mean   :45.34   Mean   :20.1          Mean   : 73.77      Mean   :93153  
 3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00      3rd Qu.:94608  
 Max.   :5000   Max.   :67.00   Max.   :43.0          Max.   :224.00      Max.   :96651  
 Family members      CCAvg           Education        Mortgage      Personal Loan   Securities Account
 Min.   :0.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0   Min.   :0.000   Min.   :0.0000    
 1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000    
 Median :2.000   Median : 1.500   Median :2.000   Median :  0.0   Median :0.000   Median :0.0000    
 Mean   :2.389   Mean   : 1.938   Mean   :1.881   Mean   : 56.5   Mean   :0.096   Mean   :0.1044    
 3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000    
 Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0   Max.   :1.000   Max.   :1.0000    
   CD Account         Online         CreditCard   
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
 Median :0.0000   Median :1.0000   Median :0.000  
 Mean   :0.0604   Mean   :0.5968   Mean   :0.294  
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
```

On checking the **Experience (in years)** column, we can find that there are negative values in that column. Hence, to make the negative values positive, the function **abs ()** was used.

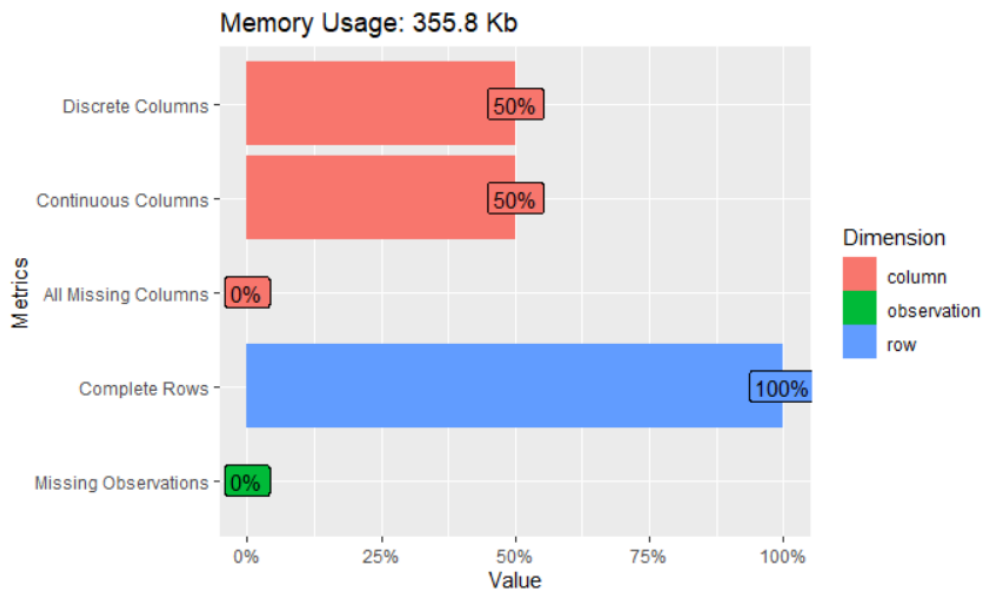
```
head(bank[bank$Experience<0,]) #checking for negative values in experience col

bank$Experience = abs(bank$Experience) #modulo function for the negative values

dim(bank)

summary(bank)
```

Summary of the data frame after applying the **abs ()** function for the **Experience** column:

```
> summary(bank)
      Age          Experience        Income       Family members       CCAvg        Education
 Min.   :23.00   Min.   : 0.00   Min.   :  8.00   Min.   :0.000   Min.   : 0.000   1:2096
 1st Qu.:35.00   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:1.000   1st Qu.: 0.700   2:1403
 Median :45.00   Median :20.00   Median : 64.00   Median :2.000   Median : 1.500   3:1501
 Mean   :45.34   Mean   :20.13   Mean   : 73.77   Mean   :2.389   Mean   : 1.938
 3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   3rd Qu.:3.000   3rd Qu.: 2.500
 Max.   :67.00   Max.   :43.00   Max.   :224.00   Max.   :4.000   Max.   :10.000
    Mortgage     Personal Loan Securities Account CD Account Online    CreditCard
 Min.   :  0.0   0:4520        0:4478             0:4698     0:2016    0:3530
 1st Qu.:  0.0   1: 480        1: 522             1: 302     1:2984    1:1470
 Median :  0.0
 Mean   : 56.5
 3rd Qu.:101.0
 Max.   :635.0
> 
```

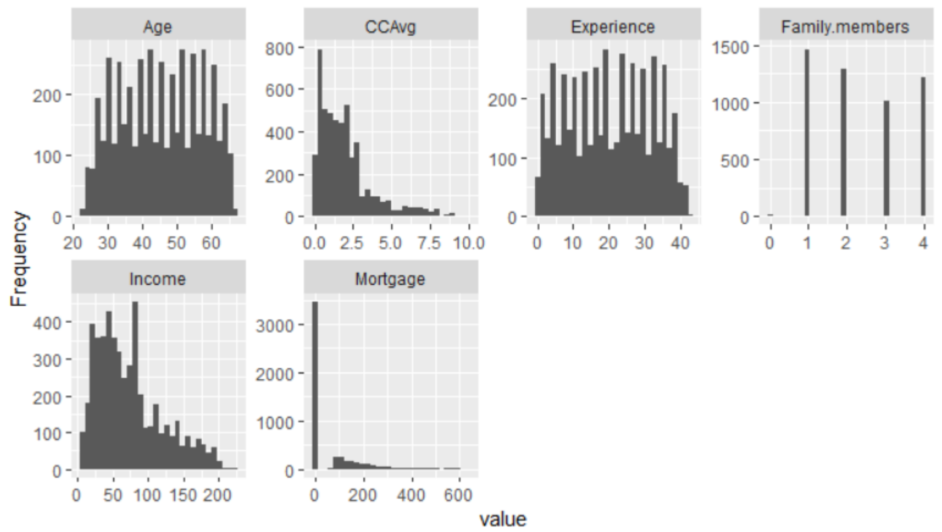
## Introduction Plot

```
plot_intro(bank)#EDA
```



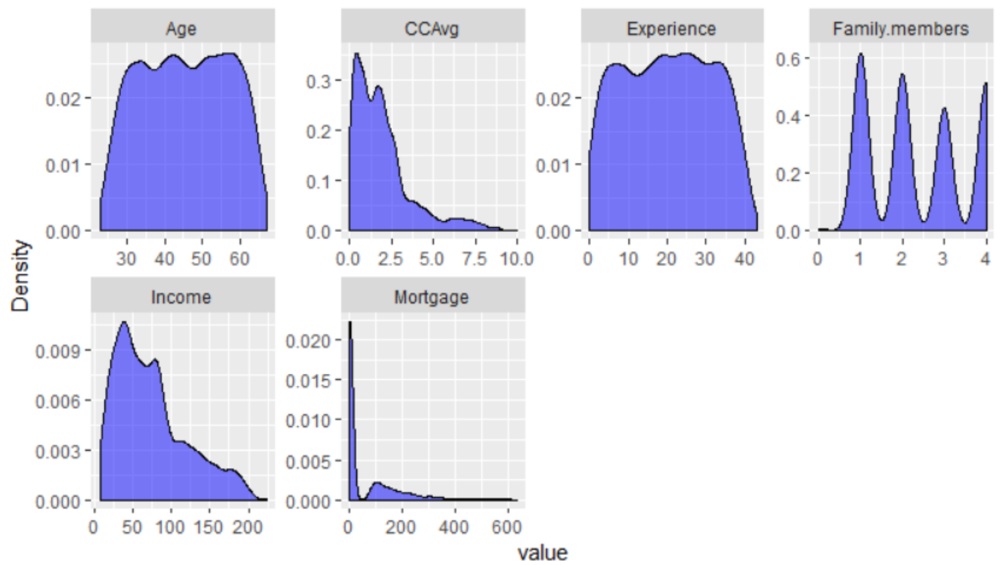

## Histogram Plot

```
plot_histogram(bank)
```




## Density Plot

```
plot_density(bank,geom_density_args = list(fill="blue", alpha=0.5))
```

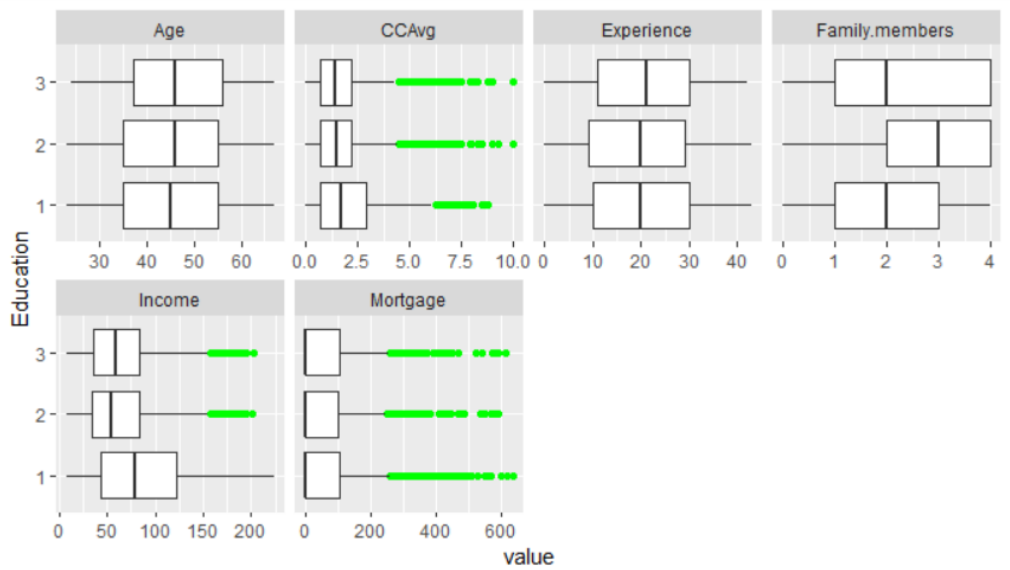



From the histograms and the density plots above we could conclude there are not much outliers in the parameters such as age, Family members and the parameters such as CCAvg and Income seem to be left skewed and with outliers.

## Boxplots

### With respect to Education

```
plot_boxplot(bank, by = "Education", geom_boxplot_args = list("outlier.color" = "green"))
```

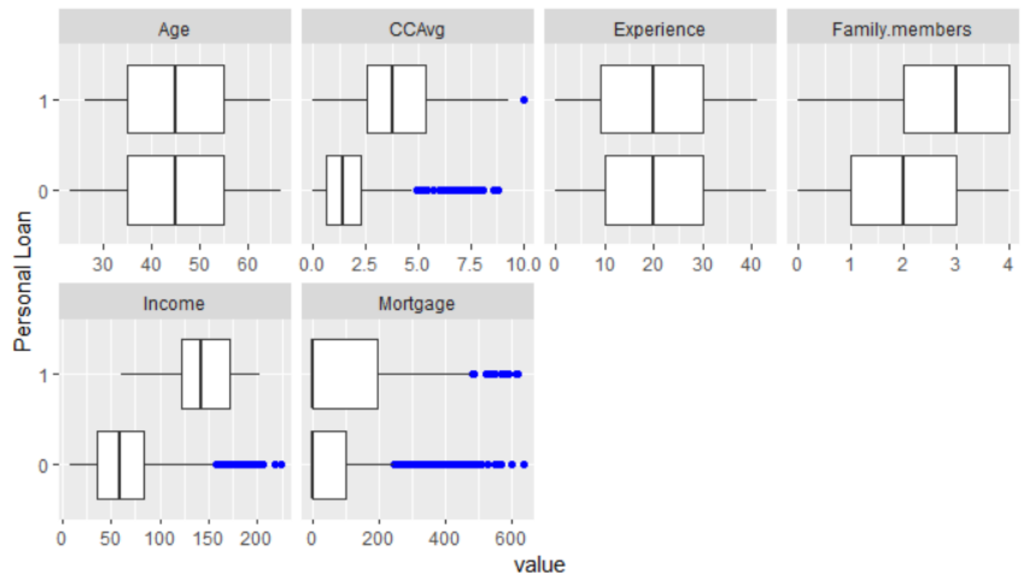



On analyzing the above box plots, we can infer that:

1. The columns **CCAvg** (Avg. spending on Credit card) and **Mortgage** have the highest number of outliers irrespective of the Education.
2. Education level **2 (Graduate)** and **3 (Advanced/Professional)** have the highest number of outliers in the **Income** column.

### With respect to Personal Loan

```
plot_boxplot(bank, by = "Personal Loan", geom_boxplot_args = list("outlier.color" = "blue"))
```




On analyzing the above box plots, we can infer that:

Lots of **class 0** (**people who don't take Personal Loan**) are present as outliers in the columns **CCAvg**, **Mortgage** and **Income** columns.Hence the bank can find more prospective loan customers in the higher income and higher mortgage values groups.

## Density, Scatter and Bar plots

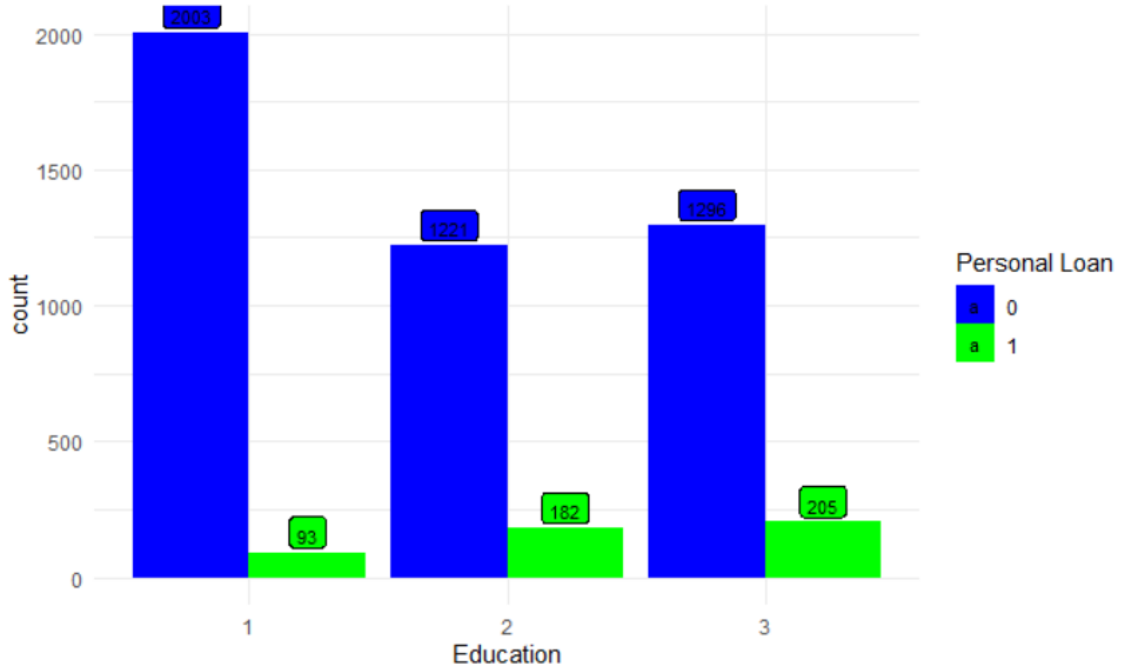
1. Income
2. Mortgage
3. Age
4. Experience
5. Income vs. Mortgage
6. Income vs. Education

```
p1 = ggplot(bank, aes(Income, fill= "Personal Loan")) + geom_density(alpha=0.5)
p2 = ggplot(bank, aes(Mortgage, fill= "Personal Loan")) + geom_density(alpha=0.5)
p3 = ggplot(bank, aes(Age, fill= "Personal Loan")) + geom_density(alpha=0.5)
p4 = ggplot(bank, aes(Experience, fill= "Personal Loan")) + geom_density(alpha=0.5)
p5 = ggplot(bank, aes(Income, fill= Education)) + geom_histogram(alpha=0.5, bins = 70)
p6 = ggplot(bank, aes(Income, Mortgage, color = "Personal Loan")) +
  geom_point(alpha = 0.5)
grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 2, nrow = 3)
```


## Bar plot for Education

```
ggplot(bank, aes(Education,fill= bank$`Personal Loan`)) +
  geom_bar(stat = "count", position = "dodge") +
  geom_label(stat = "count", aes(label= ..count..),
             size = 3, position = position_dodge(width = 0.9), vjust=-0.15)+
  scale_fill_manual("Personal Loan", values = c("0" = "blue", "1" = "green"))+
  theme_minimal()

summary(bank$`Personal Loan`)
```

```
> summary(bank$`Personal Loan`)
   0    1
4520  480
```




On analyzing the above bar graph, we can infer that:

1. Most of the customers are not willing to take up Personal loans.
2. The distribution is somewhat left-skewed and inclined more towards **class 0** (**People who don't take up Personal Loans**)
3. Also, the number of customers who take up Personal Loans increases as the Education level increases from **1** (**Undergrad**) to **3** (**Advanced/Professional**).

On looking on the above results, there seems to be a lot of prospective customer base and the bank must enlighten the customers on the perks and prospects of the Personal Loan in order to increase the revenue from the Personal Loans.

## Scatter plot for CCAvg and Income

Credit card usage can be used as a performance metric and also to manipulate a certain customer for taking a Personal Loan.

```
ggplot(bank, aes(Income,y = CCAvg, color = bank$`Personal Loan`)) +
  geom_point(size = 1)
```


On analyzing the above scatter plot, we can infer that customers:

1. Who are in the **income group** between **0K and 40K**, spend **very less** or **don't even spend** on Credit cards. Hence, there won't be a need of Personal loan for them to clear off the debts.
2. Who are in the **income group** between **40K and 90K**, spend **a considerable amount** using the Credit cards.
3. Who are in the **income group** of **greater than 100K**, spend **more** using credit cards. Hence, they will surely need a Personal loan to clear off their debts.

Hence, the bank must concentrate on the customers who are in the income group between **40K and 90K**to take up Personal loans.

## Scatter plot for Income and Mortgage

Mortgage is another good indicator with which the customers can be targeted.

```
ggplot(bank, aes(Income,y = Mortgage, color = bank$`Personal Loan`)) +
  geom_point(size = 1)
```


On analyzing the above scatter plot, we can infer that:

1. Customers having moderate Income and Mortgage are least interested in taking up Personal loans.
2. Customers having high Mortgage value take up the personal loans which have low rate of interest in order to retrieve their property from mortgage.

## CLUSTERING

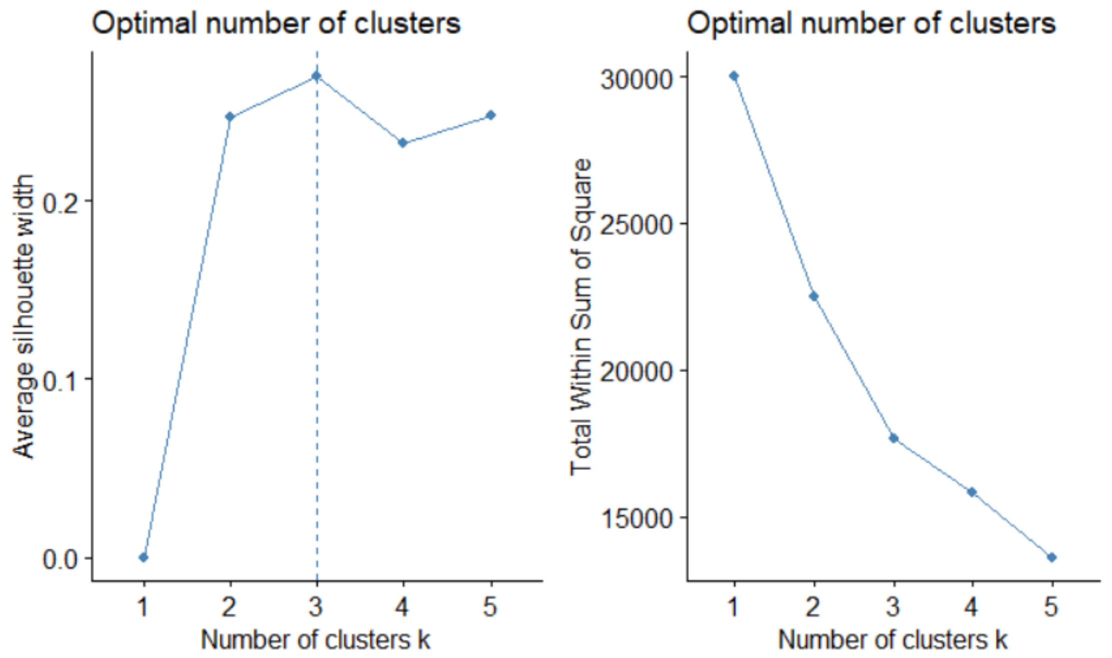
Clustering is one of the non-supervised learning techniques which are used to find similarities and dissimilarities of the pattern in the given dataset. In a dataset each row itself clustered.

Clustering is one of the non-supervised learning techniques which are very helpful in handling real-time data. It can be broadly classified into Hierarchical and Non-hierarchical.

Since our dataset has a large volume of data with 5000 observations, non-hierarchical clustering (K-means clustering) is used for analyzing.

Before starting with the clustering, the ideal cluster size can be determined with the given number of datasets.

```
bank.clust = bank %>% select_if(is.numeric)

bank.scale = scale(bank.clust, center = TRUE)  #scaling the cluster

bank.dist = dist(bank.scale, method = "euclidean") #calculating the euclidean distance

## checking optimal number of clusters to categorize dataset

p12 = fviz_nbclust(bank.scale, kmeans, method = "silhouette", k.max = 5) # k-means clustering is used
p21 = fviz_nbclust(bank.scale, kmeans, method = "wss", k.max = 5)

grid.arrange(p12, p21, ncol=2)
```

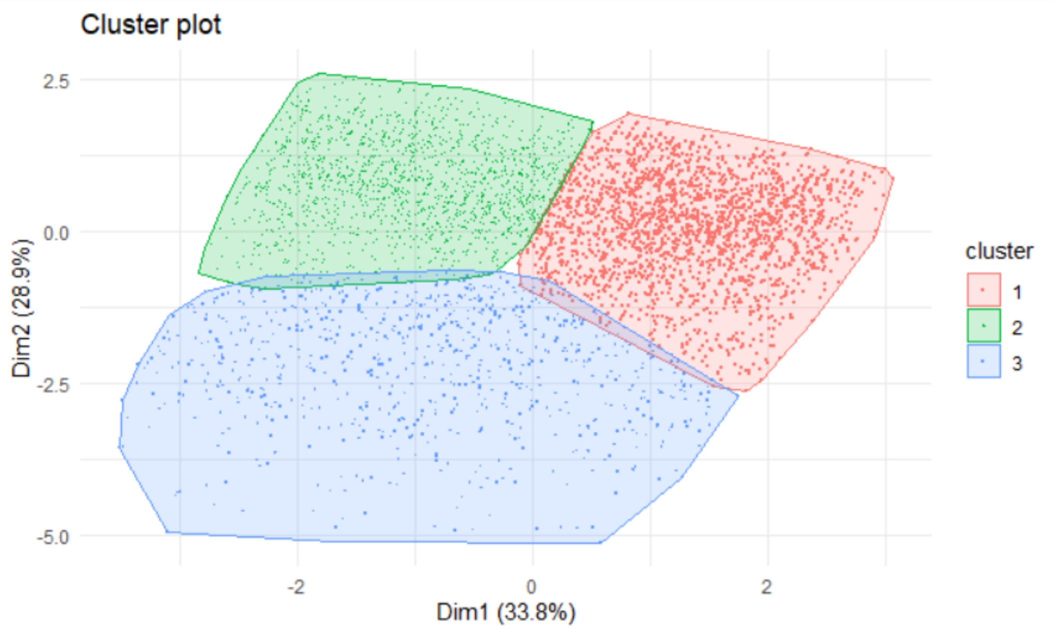



Hence, the optimal number of clusters can be formed is 3 with 3 centroids (1 for each).

Hence, the k-means clustering is performed using the scaled data since the actual values of the variables are not uniformed:

```
#k-means clustering

set.seed(8787)
bank.clusters = kmeans(bank.scale, 3, nstart = 10)

fviz_cluster(bank.clusters, bank.scale, geom = "point",
             ellipse = TRUE, pointsize = 0.2, ) + theme_minimal()
```

```
bank.clusters            List of 9
  cluster : int [1:5000] 2 2 2 2 2 2 1 1 2 3 ...
  centers : num [1:3, 1:6] 0.89 -0.891 -0.143 0.886 -0.893 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:6] "Age" "Experience" "Income" "Family members" ...
  totss : num 29994
  withinss : num [1:3] 6326 5817 5502
  tot.withinss: num 17646
  betweenss : num 12348
  size : int [1:3] 2149 2012 839
  iter : int 3
  ifault : int 0
  - attr(*, "class")= chr "kmeans"
```




From this, we came to know that:

1. The clusters that are formed with the “Euclidean distance” method co-incidentally coincide with the number of education levels (which is also 3).
2. As the education level increases, people’s financial needs increase due to increase in the personal needs. Hence, personal loans will be preferred.
3. The dataset has been split into 3 clusters of size 2149, 2012 and 839 respectively.

## SPLITTING OF DATASET

To analyze the dataset with various models, the model is split into two datasets namely:

1. Training dataset

2. Test dataset

Out of the 5000 observations, 70% of the observation is used as the training dataset and rest of them is used as test dataset. Here, we have taken the test dataset for analyzing purpose.

```
#splitting dataset into train and test

set.seed(1233)

## sampling 70% of data for training the algorithms using random sampling

bank.index = sample(1:nrow(bank), nrow(bank)*0.70)
bank.train = bank[bank.index,]
bank.test = bank[-bank.index,]

dim(bank.test)

dim(bank.train)

table(bank.train$`Personal Loan`)

table(bank.test$`Personal Loan`)
```

```
> set.seed(1233)
> bank.index = sample(1:nrow(bank), nrow(bank)*0.70)
> bank.train = bank[bank.index,]
> bank.test = bank[-bank.index,]
> dim(bank.test)
[1] 1500   12
> dim(bank.train)
[1] 3500   12
> table(bank.train$`Personal Loan`)

   0    1
3151  349
> table(bank.test$`Personal Loan`)

   0    1
1369  131
>
```

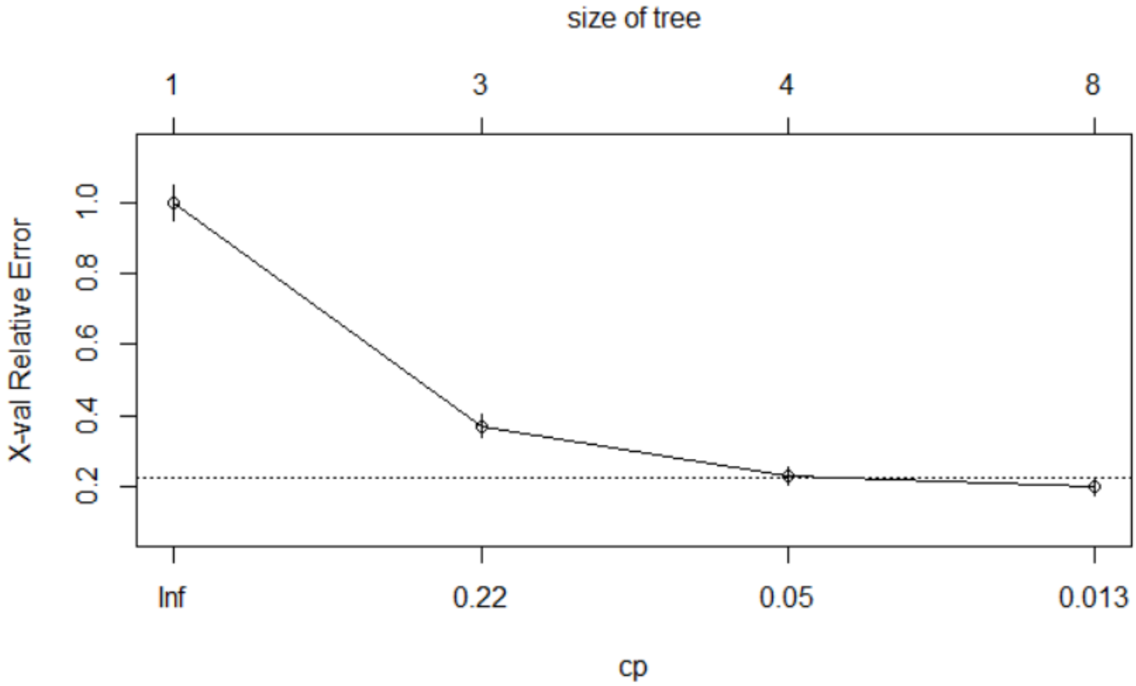
## CART MODEL

Classification tree models use GINI gain for analyzing purpose. The independent variable with the highest GINI gain is used for splitting the parent node. For running the CART model in R, the packages **rpart** and **rpart.control** are installed.

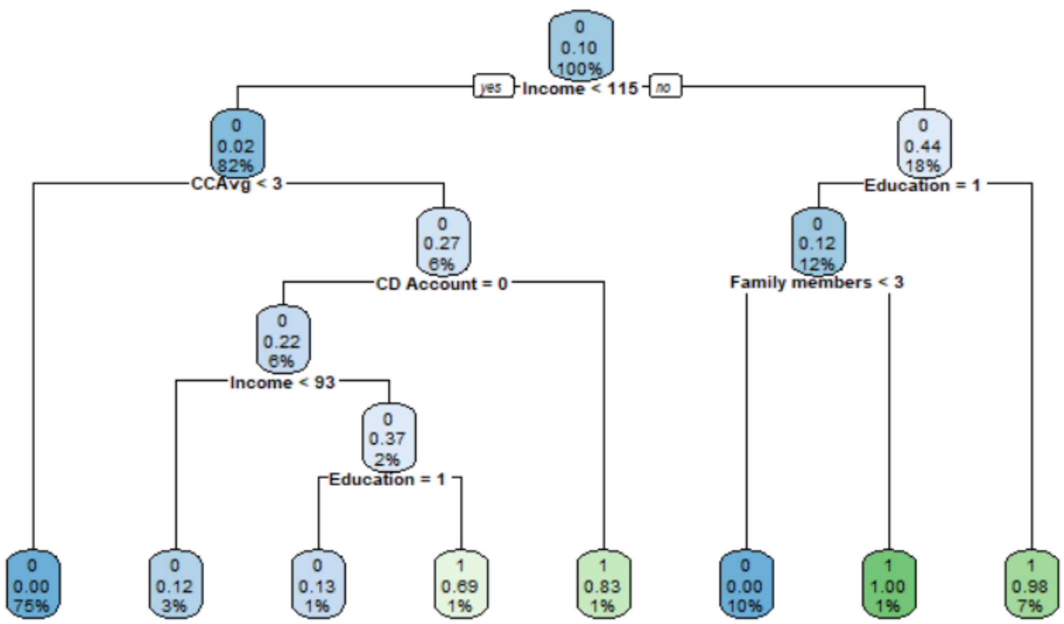
```r
#CART model

set.seed(233)

cart.model.gini = rpart(bank.train$`Personal Loan`~., data = bank.train, method = "class",
                        parms = list(split="gini"))

## checking the complexity parameter
plotcp(cart.model.gini)

## plotting the classification tree
rpart.plot(cart.model.gini, cex =0.6)

## checking the cptable to gauge the best crossvalidated error and correspoding
## Complexity paramter
cart.model.gini$cptable

## checking for the variable importance for splitting of tree
cart.model.gini$variable.importance
```

```
> set.seed(233)
> cart.model.gini = rpart(bank.train$`Personal Loan`~., data = bank.train, method = "class",
+                         parms = list(split="gini"))
> ## checking the complexity parameter
> plotcp(cart.model.gini)
> ## checking the cptable to gauge the best crossvalidated error and correspoding
> ## Complexity paramter
> cart.model.gini$cptable
          CP nsplit rel error    xerror       xstd
1 0.32521490      0 1.0000000 1.0000000 0.05078991
2 0.14326648      2 0.3495702 0.3696275 0.03193852
3 0.01719198      3 0.2063037 0.2263610 0.02517855
4 0.01000000      7 0.1346705 0.1977077 0.02356543
> ## checking for the variable importance for splitting of tree
> cart.model.gini$variable.importance
     Education         Income Family members          CCAvg     CD Account       Mortgage     Experience
    232.137107     188.541598     142.501489     106.606257      56.904176      27.306276       3.445512
           Age         Online
      3.437672       1.751040
> ## checking the complexity parameter
> plotcp(cart.model.gini)
> ## plotting the classification tree
> rpart.plot(cart.model.gini, cex =0.6)
> 
```

## Checking the complexity parameter



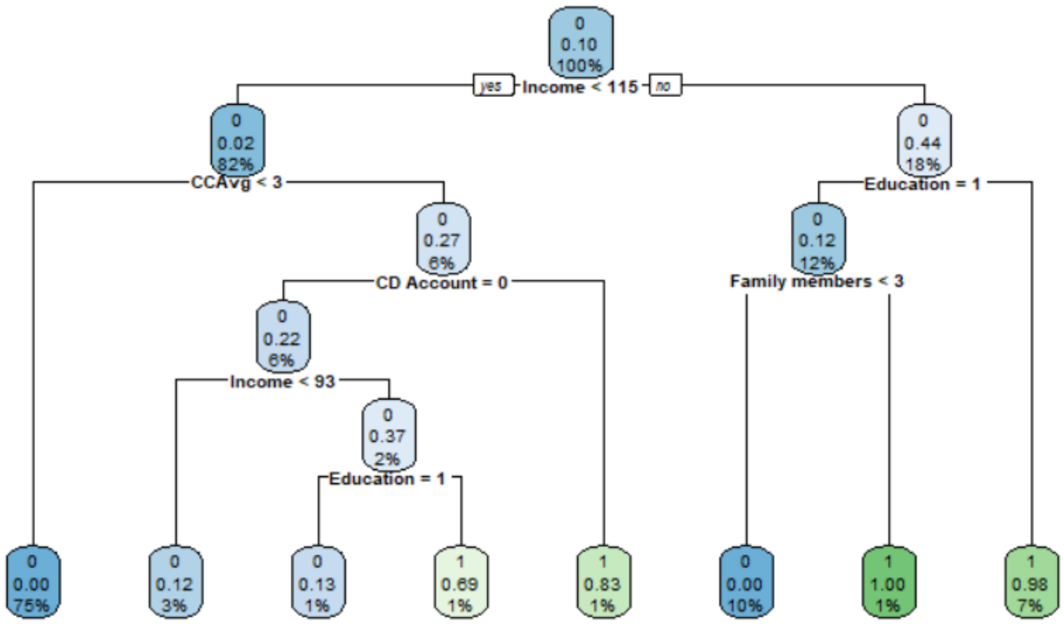

## Plotting of the CART model



## PRUNED CART MODEL

The CART model can be pruned using the complexity parameter in order to prevent the over-fitting in the model.

```
#Pruned CART Tree

## prunning the tree using the best complexity parameter
pruned.model = prune(cart.model.gini, cp = 0.015)

## plotting the prunned tree
rpart.plot(pruned.model, cex=0.65)
```




## PREDICTION USING THE CART MODEL

For predicting whether the CART model is suitable or not for analysis, the pruned model is used. Since, loan prediction is done; a threshold value is set to find the defaulters.

Here, probability value greater than **0.7** is treated as **class 1** and lesser than those are considered as **class 0**.

The confusion matrix can also be used to determine the performance of the model.

```
#CART prediction

cart.pred = predict(pruned.model, bank.test, type = "prob")

cart.pred.prob.1 = cart.pred[,1]
head(cart.pred.prob.1, 10)

## setting the threshold for probabilities to be considered as 1
threshold = 0.70

bank.test$loanprediction = ifelse(cart.pred.prob.1 >= threshold, 1, 0)

bank.test$loanprediction = as.factor(bank.test$loanprediction)

Cart.Confusion.Matrix = confusionMatrix(bank.test$loanprediction,
                                        reference = bank.test$`Personal Loan`, positive = "1")
Cart.Confusion.Matrix
```

```
> cart.pred = predict(pruned.model, bank.test, type = "prob")
> cart.pred.prob.1 = cart.pred[,1]
> head(cart.pred.prob.1, 10)
         1          2          3          4          5          6          7          8          9
0.99734244 0.99734244 0.99734244 0.99734244 0.02109705 0.31428571 0.02109705 0.99734244 0.02109705
        10
0.99734244
> ## setting the threshold for probabilities to be considered as 1
> threshold = 0.70
> bank.test$loanprediction = ifelse(cart.pred.prob.1 >= threshold, 1, 0)
> bank.test$loanprediction = as.factor(bank.test$loanprediction)
> Cart.Confusion.Matrix = confusionMatrix(bank.test$loanprediction,
+                                         reference = bank.test$`Personal Loan`, positive = "1")
> Cart.Confusion.Matrix
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0    8  121
         1 1361   10

               Accuracy : 0.012
                 95% CI : (0.0071, 0.0189)
    No Information Rate : 0.9127
    P-Value [Acc > NIR] : 1

                  Kappa : -0.1738

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.076336
            Specificity : 0.005844
         Pos Pred Value : 0.007294
         Neg Pred Value : 0.062016
             Prevalence : 0.087333
         Detection Rate : 0.006667
   Detection Prevalence : 0.914000
      Balanced Accuracy : 0.041090

       'Positive' Class : 1

> 
```

From the above calculations, we can find that the **accuracy** of the pruned CART model is less i.e., **1.2%** even after pruning with respect to the complexity parameter.Hence we cannot conclude our observations based on CART model.

## RANDOM FOREST TECHNIQUE

Random Forest is an ensemble technique in which we have used the bagging method that comprises of strong and weak learners which comprises of Decision Trees for studying the dataset, thereby providing better accuracy or output. To analyze the given dataset using the Random Forest technique in R, the package **randomForest** is used.

```
#Random Forest

set.seed(1233)

RF = randomForest(formula = bank.test$`Personal Loan`~(bank.test$Age+bank.test$Experience+
                                                       bank.test$Income+bank.test$`Family members`+
                                                       bank.test$CCAvg+bank.test$Education+
                                                       bank.test$Mortgage+
                                                       bank.test$`Securities Account`+
                                                       bank.test$`CD Account`+
                                                       bank.test$Online+bank.test$CreditCard),
                  data = bank.test)
print(RF)
```

```
> set.seed(1233)
> RF = randomForest(formula = bank.test$`Personal Loan`~(bank.test$Age+bank.test$Experience+
+                                                       bank.test$Income+bank.test$`Family members`+
+                                                       bank.test$CCAvg+bank.test$Education+
+                                                       bank.test$Mortgage+
+                                                       bank.test$`Securities Account`+
+                                                       bank.test$`CD Account`+
+                                                       bank.test$Online+bank.test$CreditCard),
+                  data = bank.test)
> print(RF)

Call:
 randomForest(formula = bank.test$`Personal Loan` ~ (bank.test$Age +      bank.test$Experience + bank.test$In
come + bank.test$`Family members` +      bank.test$CCAvg + bank.test$Education + bank.test$Mortgage +      ba
nk.test$`Securities Account` + bank.test$`CD Account` +      bank.test$Online + bank.test$CreditCard), data =
 bank.test)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 1.67%
Confusion matrix:
     0   1 class.error
0 1363   6 0.004382761
1   19 112 0.145038168
>
```

The error rate and OOB error rate for this technique are calculated as follows:

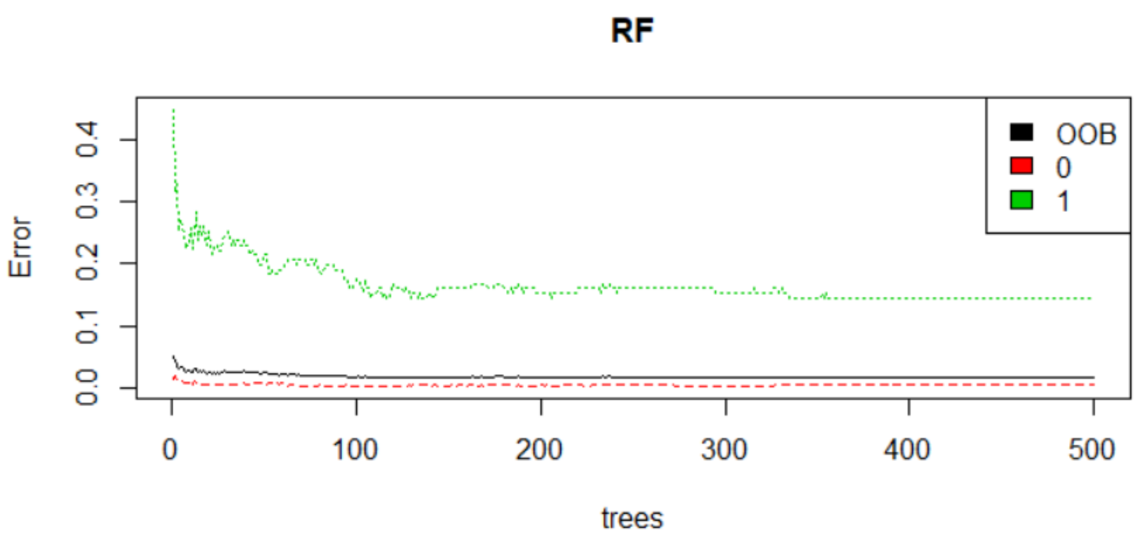
```
## Print the error rate

err = RF$err.rate
head(err)

## out of bag error
oob_err = err[nrow(err), "OOB"]
print(oob_err)  ## depicts the final out of bag error for all the samples
```

```
## plot the OOB error

plot(RF)
legend(x="topright", legend = colnames(err), fill = 1:ncol(err))
```

```
> err = RF$err.rate
> head(err)
            OOB           0         1
[1,] 0.05185185 0.012219959 0.4489796
[2,] 0.04462243 0.019975031 0.3150685
[3,] 0.03758020 0.011964108 0.3295455
[4,] 0.03064516 0.009708738 0.2523364
[5,] 0.03365744 0.010664479 0.2711864
[6,] 0.03281027 0.011700468 0.2583333
> ## out of bag error
> oob_err = err[nrow(err), "OOB"]
> print(oob_err)  ## depicts the final out of bag error for all the samples
       OOB
0.01666667
>
```

Plotting of the Random Forest graph:

```
> plot(RF)
> legend(x="topright", legend = colnames(err), fill = 1:ncol(err))
>
```




The above graph depicts the Overall OOB error, class 0 error and class 1 error.

## PREDICTION OF THE RANDOM FOREST MODEL

```
#Prediction for Random Forest package

ranfost.pred = predict(RF, bank.test, type = "prob")[,1]

bank.test$RFpred = ifelse(ranfost.pred>=0.8,"1","0")

bank.test$RFpred = as.factor(bank.test$RFpred)

levels(bank.test$RFpred)

RFConf.Matx = confusionMatrix(bank.test$RFpred, bank.test$`Personal Loan`, positive = "1")
RFConf.Matx

table(bank.test$`Personal Loan`)
```

```
> ranfost.pred = predict(RF, bank.test, type = "prob")[,1]
> bank.test$RFpred = ifelse(ranfost.pred>=0.8,"1","0")
> bank.test$RFpred = as.factor(bank.test$RFpred)
> levels(bank.test$RFpred)
[1] "0" "1"
> RFConf.Matx = confusionMatrix(bank.test$RFpred, bank.test$`Personal Loan`, positive = "1")
> RFConf.Matx
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0    5  131
         1 1364    0

               Accuracy : 0.0033
                 95% CI : (0.0011, 0.0078)
    No Information Rate : 0.9127
    P-Value [Acc > NIR] : 1

                  Kappa : -0.1896

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.000000
            Specificity : 0.003652
         Pos Pred Value : 0.000000
         Neg Pred Value : 0.036765
             Prevalence : 0.087333
         Detection Rate : 0.000000
   Detection Prevalence : 0.909333
      Balanced Accuracy : 0.001826

       'Positive' Class : 1
```

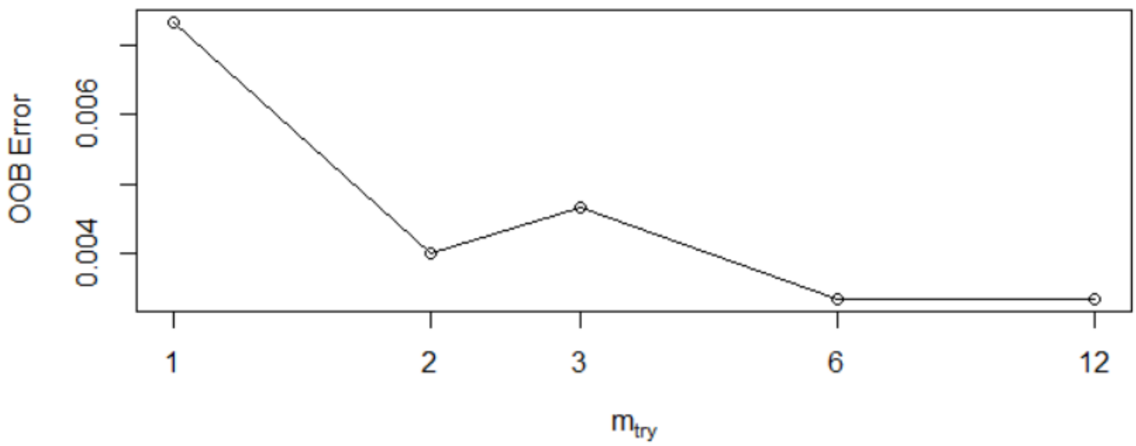
```
> table(bank.test$`Personal Loan`)

   0    1
1369  131
>
```

# TUNED RANDOM FOREST TECHNIQUE

The tuned Random forest technique can be used to find whether the performance can be improved.

```
#tuning the Random Forest algorithm

set.seed(333)

tunedRF = tuneRF(x = bank.test[,-8],
                 y= bank.test$`Personal Loan`,
                 ntreeTry = 501, doBest = T)

print(tunedRF)
```

```
> set.seed(333)
> tunedRF = tuneRF(x = bank.test[,-8],
+                  y= bank.test$`Personal Loan`,
+                  ntreeTry = 501, doBest = T)
mtry = 3  OOB error = 0.47%
Searching left ...
mtry = 2        OOB error = 0.4%
0.1428571 0.05
mtry = 1        OOB error = 0.73%
-0.8333333 0.05
Searching right ...
mtry = 6        OOB error = 0.33%
0.1666667 0.05
mtry = 12       OOB error = 0.33%
0 0.05
> print(tunedRF)

Call:
 randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1])
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 0.4%
Confusion matrix:
     0   1 class.error
0 1364   5 0.003652301
1    1 130 0.007633588
> 
```




## CONCLUSION

On comparing the accuracy of the CART, Random Forest and tuned Random forest models, we can find that the **tuned Random Forest** model has **higher accuracy** which can be used for studying the dataset.

## R SOURCE CODE

Please find attached the R code for reference:

Group_Assignment_DM.R