PROJECT - 3

Thera Bank Case Study

Thera Bank - Loan Purchase Modelling

Abstract

Build a model to identify potential customers with high probability of purchasing a loan and increase the success rate of marketing team's campaign while reducing overall cost.

Tahmid Bari

tahmid.bari@gmail.com

1. Description

Thera Bank - Loan Purchase Modelling

This case is about a bank (Thera Bank) which has a growing customer base. Majority of these customers are liability customers (depositors) with varying size of deposits. The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process, earn more through the interest on loans. In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors). A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with a minimal budget. The department wants to build a model that will help them identify the potential customers who have a higher probability of purchasing the loan. This will increase the success ratio while at the same time reduce the cost of the campaign. The dataset has data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

Link to the case file:

Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx

You are brought in as a consultant and your job is to build the best model which can classify the right customers who have a higher probability of purchasing the loan. You are expected to do the following:

- EDA of the data available. Showcase the results using appropriate graphs - **(10 Marks)**
- Apply appropriate clustering on the data and interpret the output(Thera Bank wants to understand what kind of customers exist in their database and hence we need to do customer segmentation) - (10 Marks)
- Build appropriate models on both the test and train data (CART & Random Forest). Interpret all the model outputs and do the necessary modifications wherever eligible (such as pruning) - (20 Marks)
- Check the performance of all the models that you have built (test and train). Use all the model performance measures you have learned so far. Share your remarks on which model performs the best. - (20 Marks)

Hint: split <- sample.split(Thera_Bank$Personal Loan, SplitRatio = 0.7)

#we are splitting the data such that we have 70% of the data is Train Data and 30% of the data is my Test Data

train<- subset(Thera_Bank, split == TRUE)

test<- subset(Thera_Bank, split == FALSE)

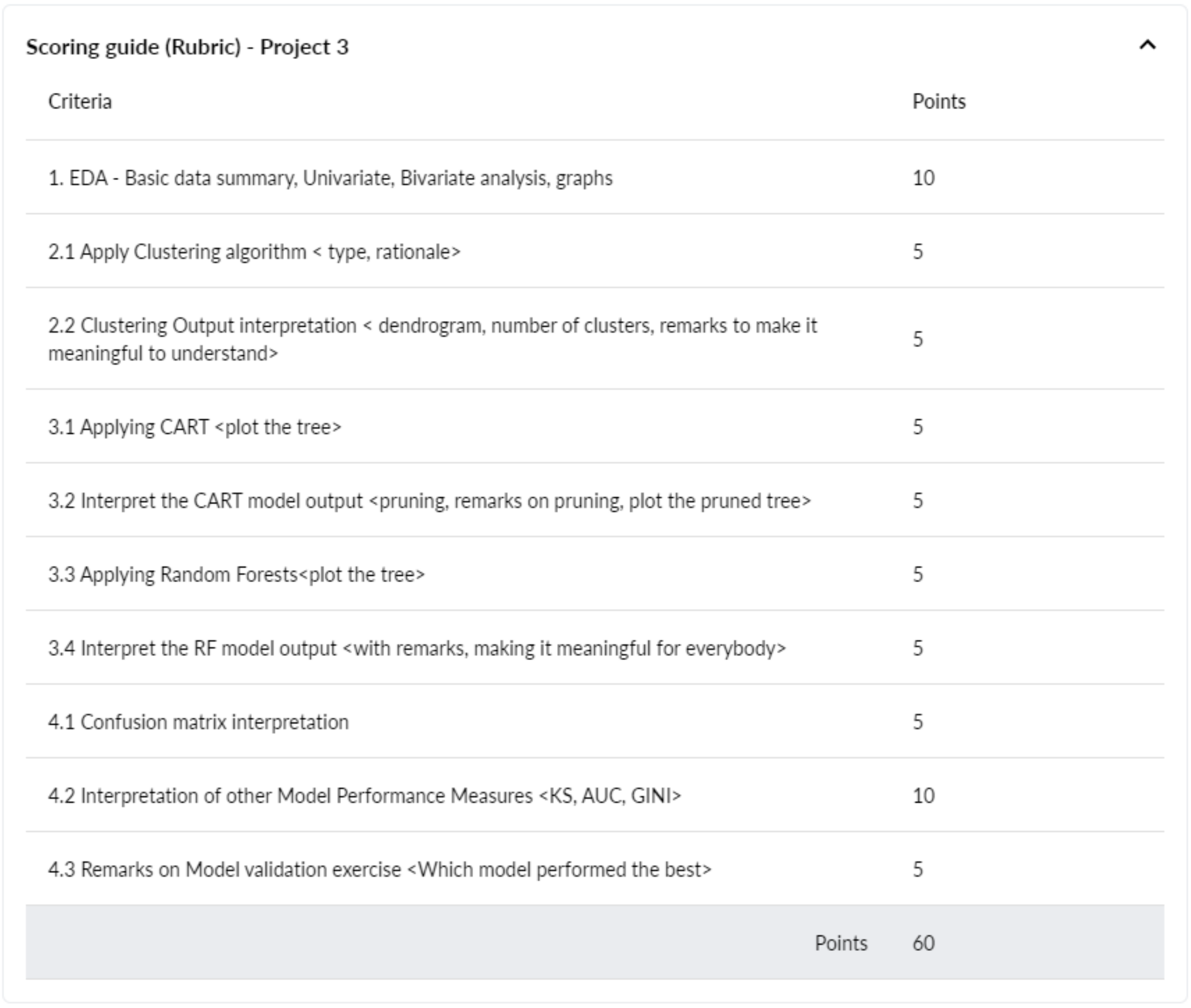
Please note the following:

- Please note the following:

1. There are two parts to the submission:
    1. The output/report in any file format - the key part of the output is the set of observations and insights from the exploration and analysis
    2. Commented R code in .R or .Rmd
2. Please don't share your R code and/or outputs only, we expect some verbiage/story too - a meaningful output that you can share in a business environment
3. Any assignment found copied/ plagiarized with other groups will not be graded and awarded zero marks
4. Please ensure timely submission as post-deadline assignment will not be accepted

Thanks

Program Office



Scoring guide (Rubric) - Project 3

Criteria	Points
1. EDA - Basic data summary, Univariate, Bivariate analysis, graphs	10
2.1 Apply Clustering algorithm < type, rationale>	5
2.2 Clustering Output interpretation < dendrogram, number of clusters, remarks to make it meaningful to understand>	5
3.1 Applying CART <plot the tree>	5
3.2 Interpret the CART model output <pruning, remarks on pruning, plot the pruned tree>	5
3.3 Applying Random Forests<plot the tree>	5
3.4 Interpret the RF model output <with remarks, making it meaningful for everybody>	5
4.1 Confusion matrix interpretation	5
4.2 Interpretation of other Model Performance Measures <KS, AUC, GINI>	10
4.3 Remarks on Model validation exercise <Which model performed the best>	5
Points	60

2. Project Objective

This case study is prepared for Thera Bank for their Personal Loan Campaign so that they can target right customers who have a higher probability of purchasing the loan. The objective is to build the best model using Data Mining techniques which can classify right customers.

We have built two models using CART and Random Forest techniques.

This dataset has been provided by Thera Bank which is interested in increasing its asset base by giving out more loans to potential customers in order to earn Interest Income over a good period of financial years in future

Various variables or predictors have been provided like Income, Age, Mortgage et.al. for us to gauge the Response on Personal Loan(target variable).

We will be performing below steps and will analyse the data using Data Mining techniques to identify such customers:

1. Understanding about the given data and doing EDA with appropriate graphs.
2. Applying appropriate clustering on the data.
3. Build appropriate models on both the test and train data using CART & Random Forest method.
4. Check the performance of all the models that you have built (test and train).
5. Use all the model performance measures to evaluate the model which is built.
6. Sharing remarks on which model performs the best.

Complete case study is performed on given dataset to build suitable model for campaign using Data Mining Techniques like;

a. Clustering
b. CART
c. Random Forest
d. and finally perform Model Performance Measures by various metrics.
    i. Confusion Matrix,
    ii. AUC – ROC,
    iii. Gini Coefficient,
    iv. Concordant – Discordant Ratio and
    v. Kolmogorov Smirnov Chart

PROBLEM STATEMENT:

Bank would like to know which customers are likely to accept a personal loan. What characteristics would forecast this? If the bank were to consider expending advertising efforts to contact customers who would be likely to consider a personal loan, which customers should the bank contact first?
By answering this question correctly, the bank will be able to optimize its advertising effort by directing its attention to the highest-yield customers. This is a classification problem. There will be two classes in this case:

1. Customers with high probability accepting a personal loan, and
2. Customers with a low probability of accepting a personal loan.

We will be unable to classify customers with certainty whether they will accept or reject loan offer, but we may be able to classify the customers into mutually exclusive categories.

3. Exploratory Data Analysis

Set up working Directory:

Setting a working directory on starting of the R session makes importing and exporting data files and code files easier. Basically, working directory is the location/ folder on the PC where you have the data, codes etc. related to the project.

```
##Set working directory
setwd("C:/Users/Tahmid Bari/Desktop/Great Learning/Course Work/Machine Learning/Project-3")

##Check working directory
getwd()
```

Import and Read the Data Set:

The given dataset is in excel format. Hence, the command 'read_excel' is used for importing the file.

```
TBdata <-read_excel("Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx")
file.exists("C:\\Users\\Tahmid Bari\\Desktop\\Great Learning\\Course Work\\Machine Learning\\Project-3\\Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx")
```

Install necessary Packages and Invoke Libraries:

Use this section to install necessary packages and invoke associated libraries. Having all the packages at the same places increases code readability

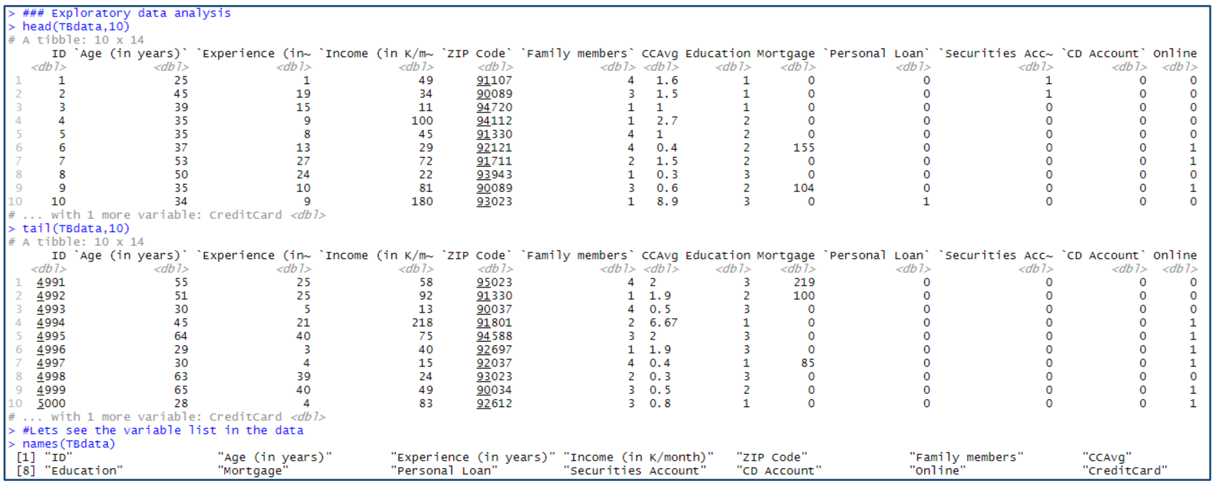
```
## Loading Library ##
library(readxl)
library("readr")
library(dplyr)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(lattice)
library(DataExplorer)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(ROCR)
library(ineq)
library(InformationValue)
library(data.table)
library(scales)
library(Metrics)
library(grDevices)
library(factoextra)
library(ROCit)
library(kableExtra)
```

Variable Identification:

Different commands are used to find out if the dataset has been read properly like head and tail. For finding the number of rows and columns dim command is used. To check the variable and structure commands like str and summary are used.

### Exploratory Data Analysis

```
head(TBdata,10)
tail(TBdata,10)
#Lets see the variable list in the data
names(TBdata)
```



```
> ### Exploratory data analysis
> head(TBdata,10)
# A tibble: 10 x 14
      ID `Age (in years)` `Experience (in~ `Income (in K/m~ `ZIP Code` `Family members` CCAvg Education Mortgage `Personal Loan` `Securities Acc~ `CD Account` Online
   <dbl>            <dbl>            <dbl>            <dbl>      <dbl>            <dbl> <dbl>     <dbl>    <dbl>           <dbl>            <dbl>        <dbl>  <dbl>
 1     1               25                1               49      91107                4   1.6         1        0               0                1            0      0
 2     2               45               19               34      90089                3   1.5         1        0               0                1            0      0
 3     3               39               15               11      94720                1   1           1        0               0                0            0      0
 4     4               35                9              100      94112                1   2.7         2        0               0                0            0      0
 5     5               35                8               45      91330                4   1           2        0               0                0            0      0
 6     6               37               13               29      92121                4   0.4         2      155               0                0            0      1
 7     7               53               27               72      91711                2   1.5         2        0               0                0            0      1
 8     8               50               24               22      93943                1   0.3         3        0               0                0            0      0
 9     9               35               10               81      90089                3   0.6         2      104               0                0            0      1
10    10               34                9              180      93023                1   8.9         3        0               1                0            0      0
# ... with 1 more variable: CreditCard <dbl>
> tail(TBdata,10)
# A tibble: 10 x 14
      ID `Age (in years)` `Experience (in~ `Income (in K/m~ `ZIP Code` `Family members` CCAvg Education Mortgage `Personal Loan` `Securities Acc~ `CD Account` Online
   <dbl>            <dbl>            <dbl>            <dbl>      <dbl>            <dbl> <dbl>     <dbl>    <dbl>           <dbl>            <dbl>        <dbl>  <dbl>
 1  4991               55               25               58      95023                4  2            3      219               0                0            0      0
 2  4992               51               25               92      91330                1  1.9          2      100               0                0            0      0
 3  4993               30                5               13      90037                4  0.5          3        0               0                0            0      0
 4  4994               45               21              218      91801                2  6.67         1        0               0                0            0      1
 5  4995               64               40               75      94588                3  2            3        0               0                0            0      1
 6  4996               29                3               40      92697                1  1.9          3        0               0                0            0      1
 7  4997               30                4               15      92037                4  0.4          1       85               0                0            0      1
 8  4998               63               39               24      93023                2  0.3          3        0               0                0            0      0
 9  4999               65               40               49      90034                3  0.5          2        0               0                0            0      1
10  5000               28                4               83      92612                3  0.8          1        0               0                0            0      1
# ... with 1 more variable: CreditCard <dbl>
> #Lets see the variable list in the data
> names(TBdata)
 [1] "ID"                   "Age (in years)"       "Experience (in years)" "Income (in K/month)"  "ZIP Code"             "Family members"       "CCAvg"
 [8] "Education"            "Mortgage"             "Personal Loan"         "Securities Account"   "CD Account"           "Online"               "CreditCard"
```

Data Set:

The dataset has data of 5,000 customers which include customer demographic information (age, income, etc.), and their relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign.

Variable Name	Explanation
ID	Customer ID
Age	Customer's age in completed years
Experience	No. of years of professional experience
Income	Annual income of the customer ($000)
ZIP Code	Home address, ZIP code
Family	Family size of the customer
CC Avg.	Avg. spending on credit cards per month ($000)
Education	Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage	Value of house mortgage if any ($000)
Personal Loan	Did this customer accept the personal loan offered in the last campaign?
Securities Account	Does the customer have a securities account with the bank?

CD Account	Does the customer have a certificate of deposit (CD) account with the bank?
Online	Does the customer use Internet banking facilities?
Credit Card	Does the customer use a credit card issued by Universal Bank?

Removing the columns which are not required for study and Rearranging the columns for ease of study
#Reordering the position of target variable/dependent variable Personal Loan to last

```
TBdata<- TBdata [c(1,2,3,4,5,6,7,8,9,11,12,13,14,10)]
```

## The variable ID is a unique number/ID does not have any explanatory power for explaining. Zipcode is just add-on information of customer in this case So we can safely drop ID and zipcode from the dataset.

# Removing column Id and zip code as they will not help much in analysis since they are basically
## addon - para information

```
TBdata = TBdata[,-c(1,5)]
```

Missing Value Identification:

In R the missing values are coded by the symbol NA. To identify missing in your dataset the function is na() . When you import dataset from other statistical applications the missing values might be coded with a number.

```
##Check Missing values
colSums(is.na(TBdata))
#Family members has missing values
```

Looking at the Summary of the Data:

```
##Lets see the datatype of the data set
str(TBdata)
summary(TBdata)
#Experience (in years) has negative values which is incorrect value
##Check the dimension or shape of the data
dim(TBdata)
```

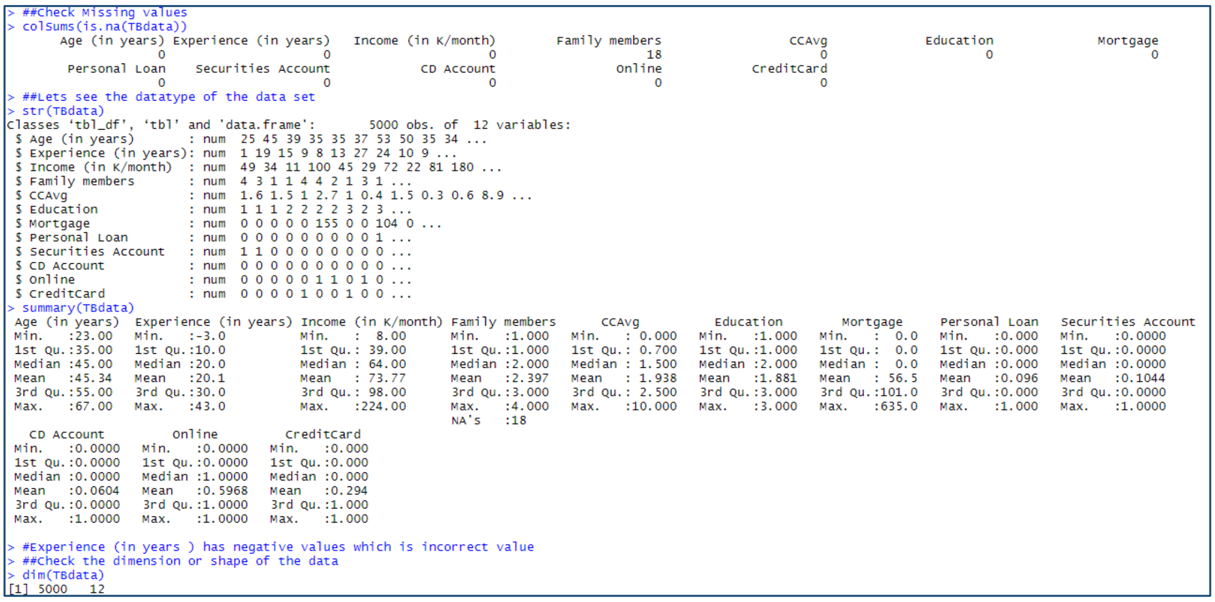
Analysis of Dataset:

Dataset has 5000 rows of observations and 14 variables
Family Members have 18 observations missing
Since 18 obs rows having "NA" as family members are also having vital other predictors we might as well replace NA with median value to factor them instead of discarding them

Fixing the missing values and negative Values in the data set

```
#Fixing the Missing Values in Family members and negative values in Experience by algorithms
TBdata$`Experience (in years)`[TBdata$`Experience (in years)` < 0] <- 0
TBdata$`Family members`[is.na(TBdata$`Family members`)] <- median(TBdata$`Family members`,
na.rm=TRUE)
```

We use zero as the least absolute value to fix the negative values in Experience and the median as it an integer value rather mean



```
> ##Check Missing values
> colSums(is.na(TBdata))
      Age (in years) Experience (in years)   Income (in K/month)        Family members                 CCAvg             Education              Mortgage 
                   0                     0                     0                    18                     0                     0                     0 
       Personal Loan    Securities Account            CD Account                Online            CreditCard 
                   0                     0                     0                     0                     0 
> ##Lets see the datatype of the data set
> str(TBdata)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       5000 obs. of  12 variables:
 $ Age (in years)       : num  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience (in years): num  1 19 15 9 8 13 27 24 10 9 ...
 $ Income (in K/month)  : num  49 34 11 100 45 29 72 22 81 180 ...
 $ Family members       : num  4 3 1 1 4 4 2 1 3 1 ...
 $ CCAvg                : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
 $ Education            : num  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage             : num  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal Loan        : num  0 0 0 0 0 0 0 0 0 1 ...
 $ Securities Account   : num  1 1 0 0 0 0 0 0 0 0 ...
 $ CD Account           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Online               : num  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard           : num  0 0 0 0 1 0 0 1 0 0 ...
> summary(TBdata)
 Age (in years)  Experience (in years) Income (in K/month) Family members      CCAvg            Education        Mortgage      Personal Loan   Securities Account
 Min.   :23.00   Min.   :-3.0          Min.   :  8.00      Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0   Min.   :0.000   Min.   :0.0000    
 1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00      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 :45.00   Median :20.0          Median : 64.00      Median :2.000   Median : 1.500   Median :2.000   Median :  0.0   Median :0.000   Median :0.0000    
 Mean   :45.34   Mean   :20.1          Mean   : 73.77      Mean   :2.397   Mean   : 1.938   Mean   :1.881   Mean   : 56.5   Mean   :0.096   Mean   :0.1044    
 3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00      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.   :67.00   Max.   :43.0          Max.   :224.00      Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0   Max.   :1.000   Max.   :1.0000    
                                                           NA's   :18                                                                                       
   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  

> #Experience (in years ) has negative values which is incorrect value
> ##Check the dimension or shape of the data
> dim(TBdata)
[1] 5000   12
```

Converting the Variables as Factor:

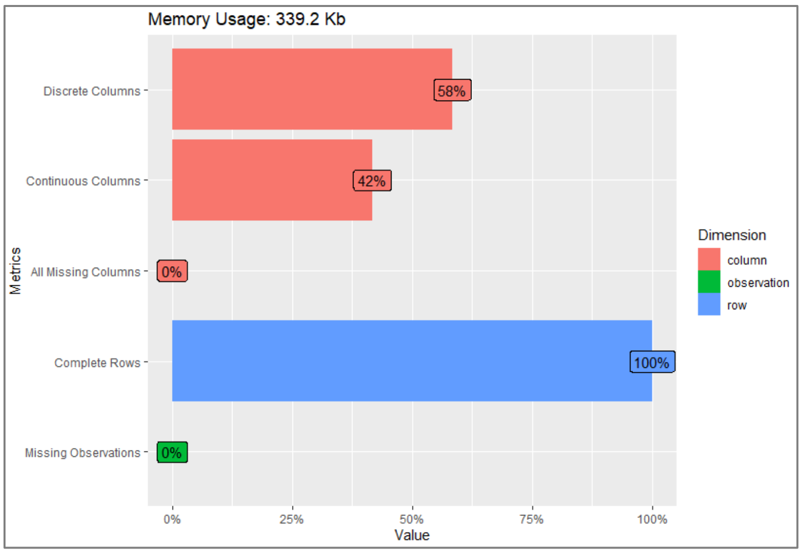
Columns like Personal Loan, CD Account, Online et.al are factor values with levels “0” and “1” Save Education which is ordered factor with 3 levels 1 < 2 < 3

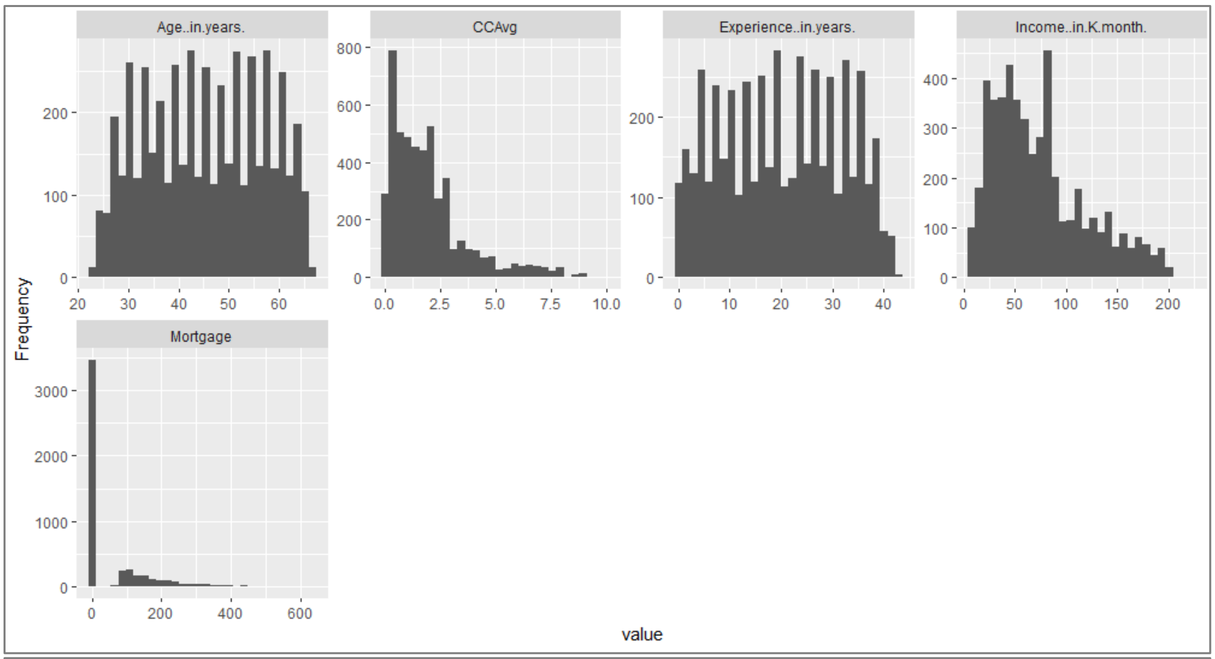
```
##Converting target variable into factor
TBdata$`Personal Loan` = as.factor(TBdata $`Personal Loan`)
#Converting categorical variables into factor
TBdata$`Family members` = as.factor(TBdata $`Family members`)
TBdata$Education = as.factor(TBdata $Education)
TBdata$`Securities Account`= as.factor(TBdata $`Securities Account`)
TBdata$`CD Account` = as.factor(TBdata $`CD Account`)
TBdata$Online = as.factor(TBdata $Online)
TBdata$CreditCard = as.factor(TBdata$CreditCard)
```

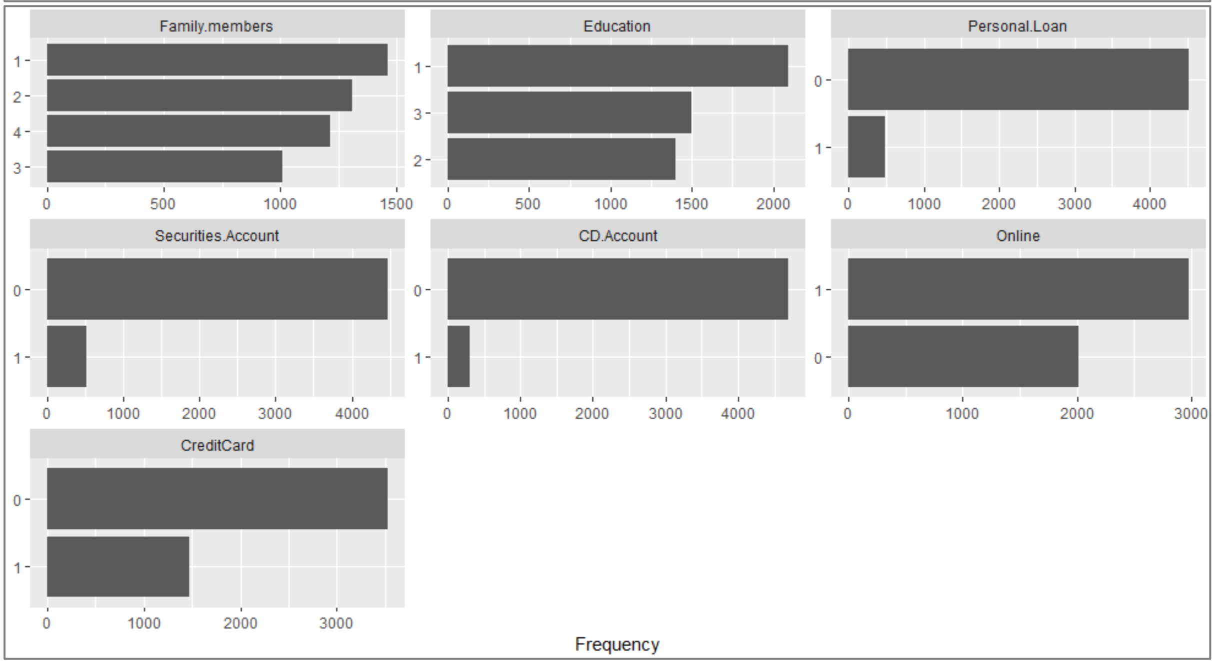
Univariate Analysis:

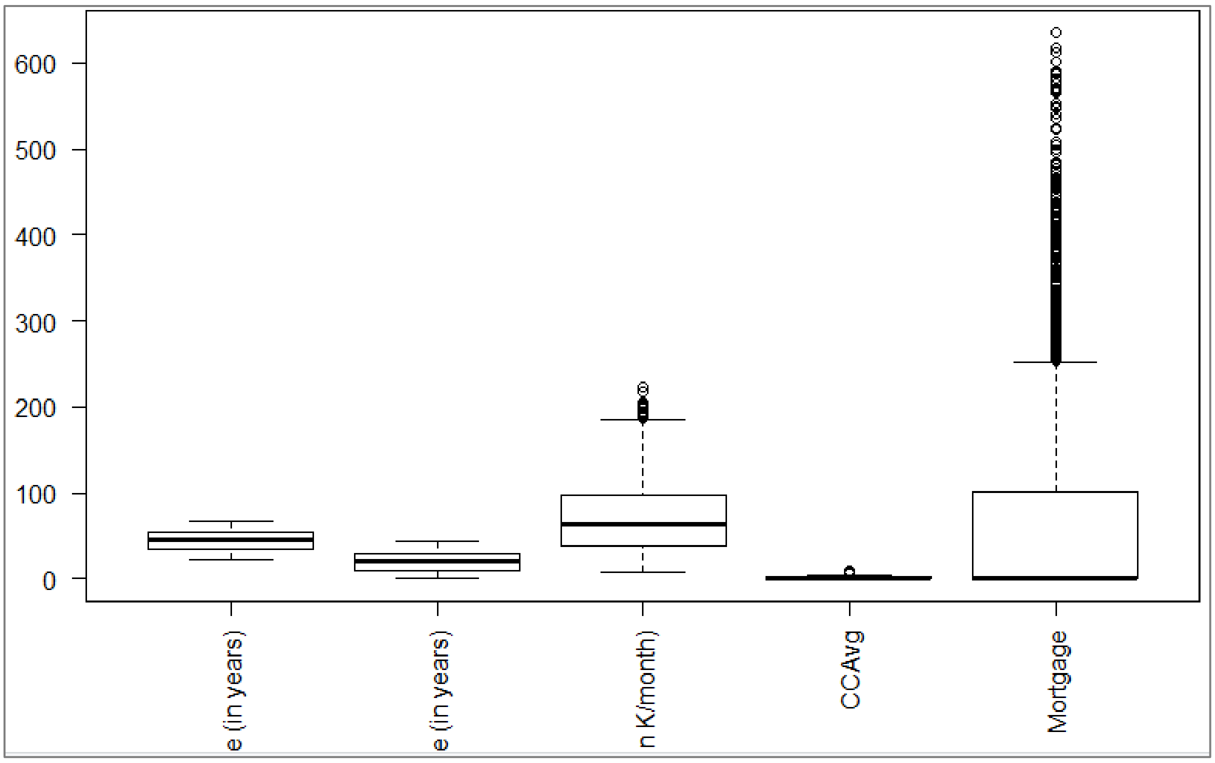
Introduction Plot of Data

```
## Introducing the Dataset
plot_intro(TBdata)
```




Age..in.years.
CCAvg
Experience..in.years.
Income..in.K.month.
Mortgage
Frequency
value


Family.members
Education
Personal.Loan
Securities.Account
CD.Account
Online
CreditCard
Frequency



Analysis:

Age feature is normally distributed with majority of customers falling between 30 years and 60 years of age. We can confirm this by looking at the describe statement above, which shows mean is almost equal to median

Experience is normally distributed with more customer having experience starting from 8 years. Here the **mean** is equal to **median**.

Income is positively skewed. Majority of the customers have income between 45K and 55K. We can see that **mean** is greater than the **median**

CCAvg is also a positively skewed variable and average spending is between 0K to 10K

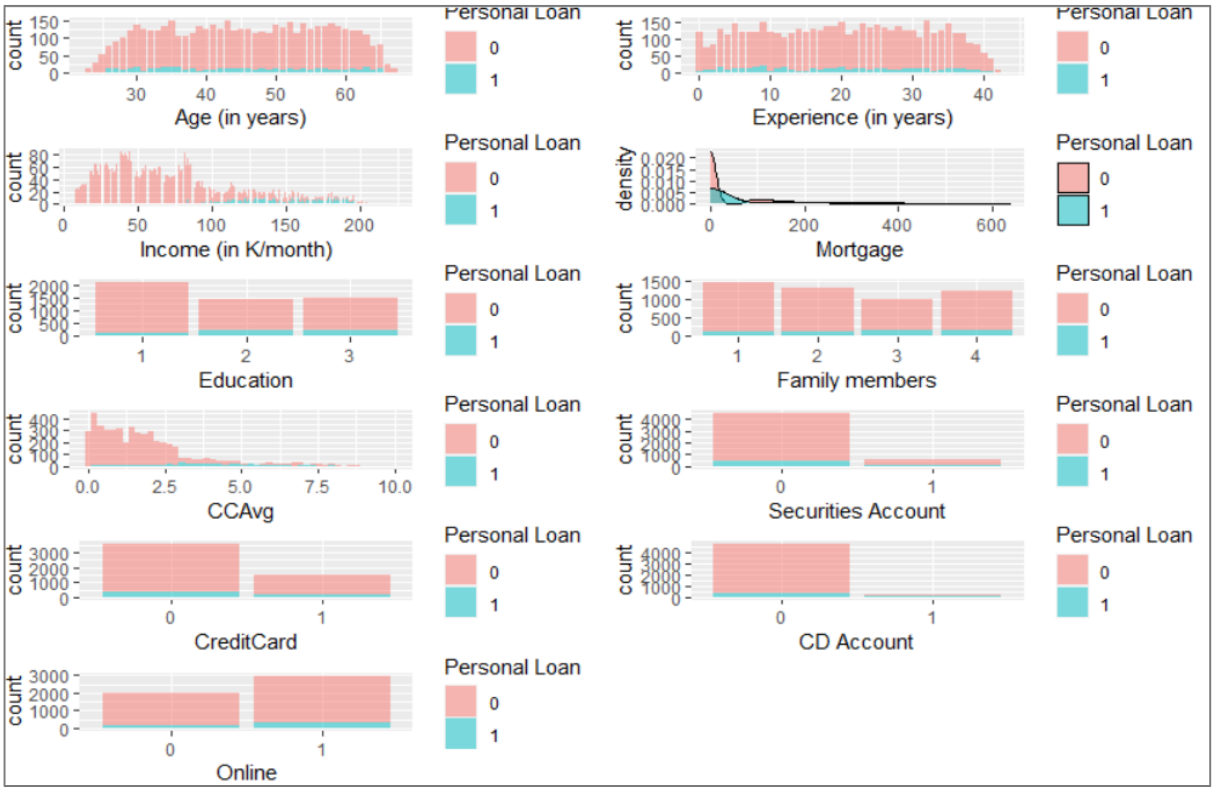
Mortgage 70% of the individuals have a mortgage of less than 40K.

The variables family and education are ordinal variables. The distribution of families is evenly distributed

Income, Credit Card and Mortgage predictors have outliers

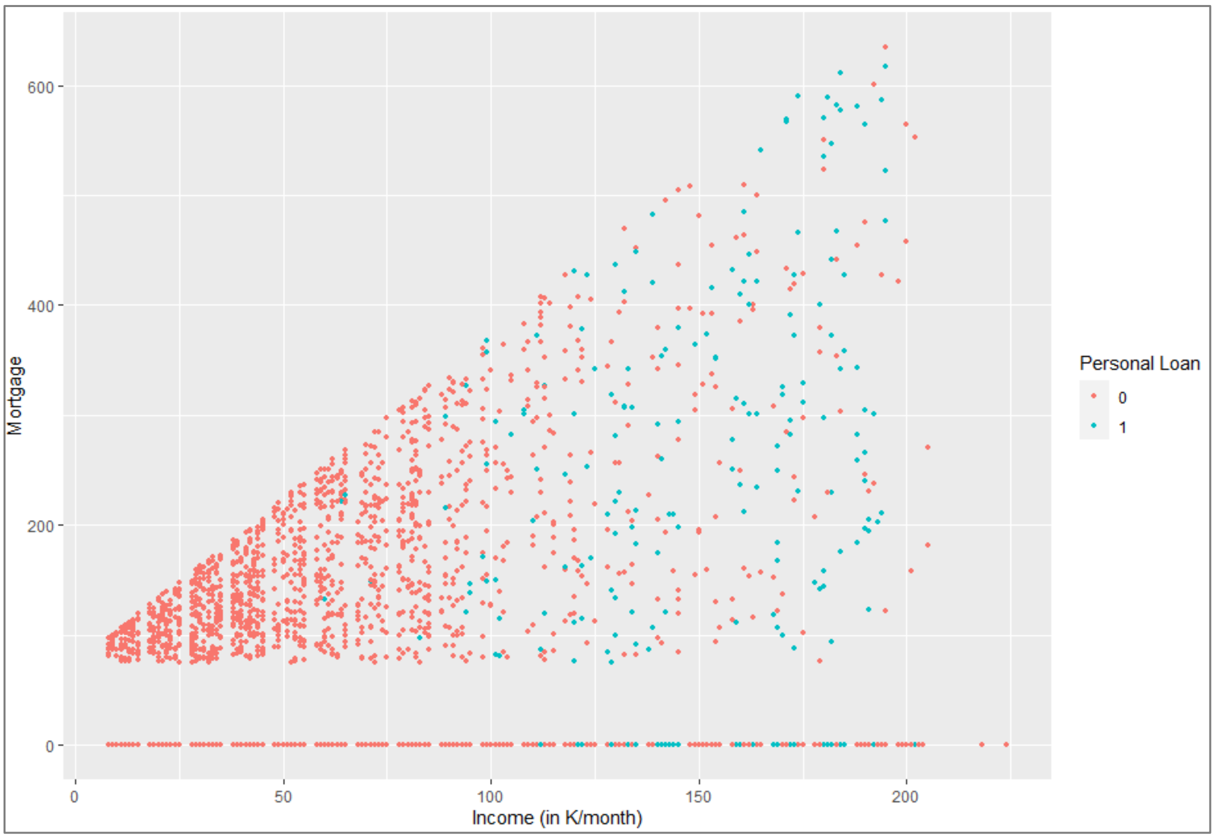
People excepted the loans in last campaign are very few.

Bivariate Analysis:

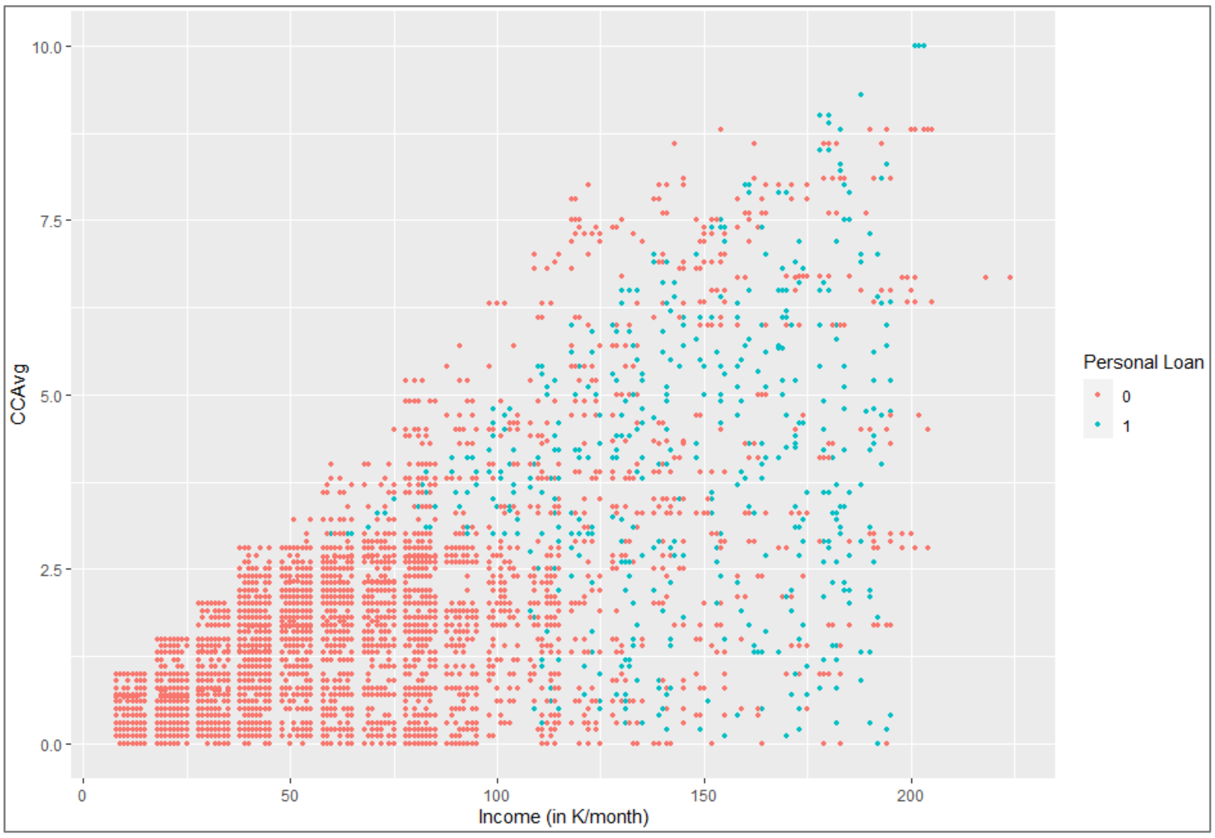


Analysis:

- Proportion of no-loan takers is very high across all categories
- People having income in 1st quartile i.e. between 38 K to 90K have no Personal loans. People with high income have personal loans
- Data is almost skewed towards No-Personal Loans
- Most of the loan applicants are Professionals
- Very few loan applicants have a securities account.



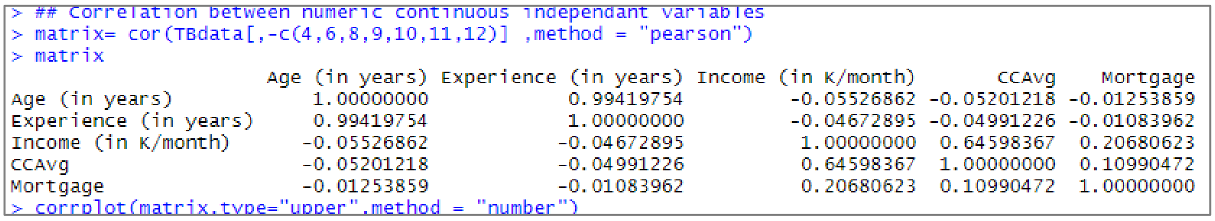
Mortgage
0
200
400
600
0
50
100
150
200
Income (in K/month)
Personal Loan
0
1



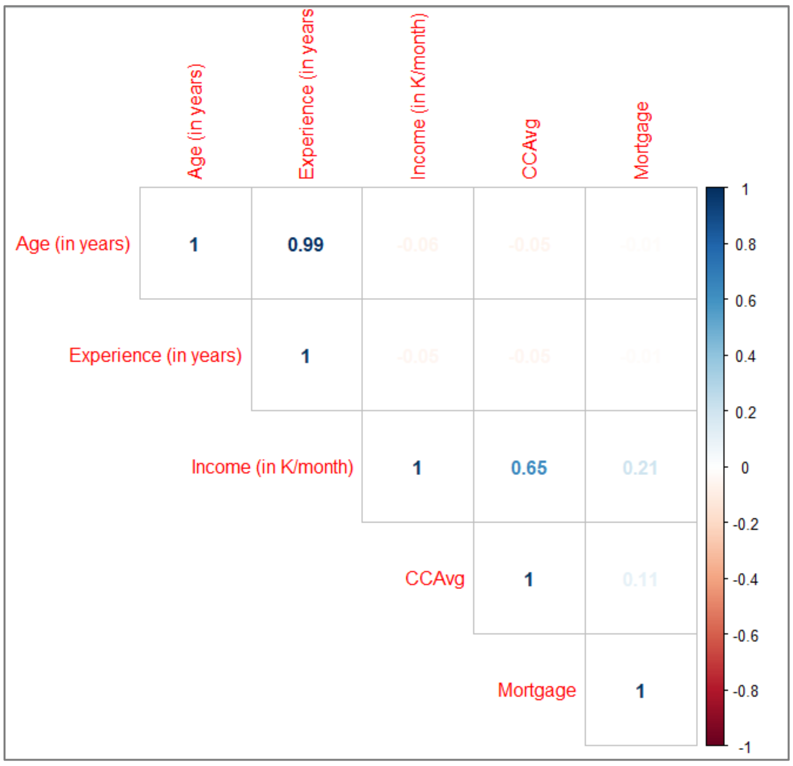
CCAvg
0.0
2.5
5.0
7.5
10.0
0
50
100
150
200
Income (in K/month)
Personal Loan
0
1

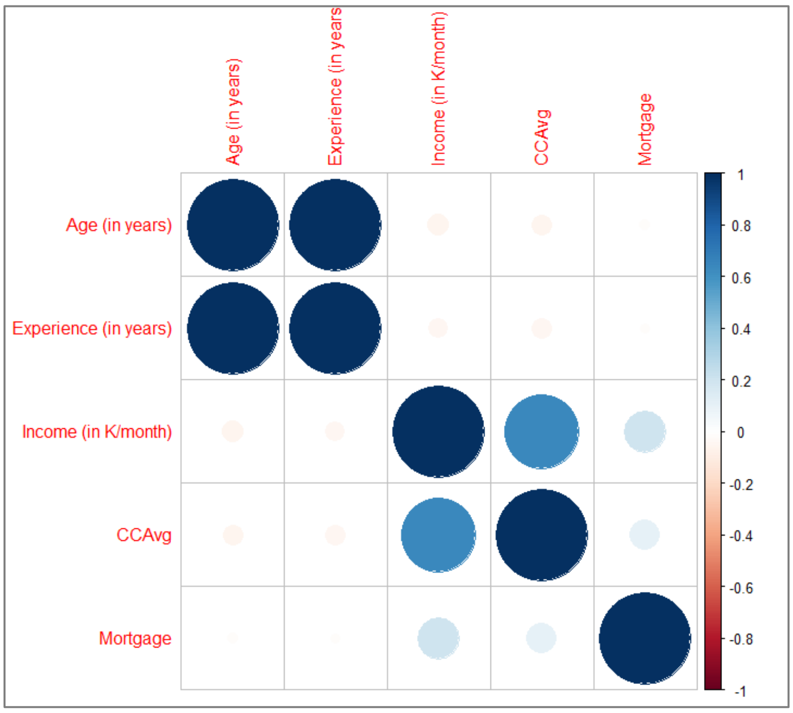
Correlation Matrix

The below is the correlation matrix:



```
> ## Correlation between numeric continuous independant variables
> matrix= cor(TBdata[,-c(4,6,8,9,10,11,12)] ,method = "pearson")
> matrix
                     Age (in years) Experience (in years) Income (in K/month)       CCAvg    Mortgage
Age (in years)           1.00000000            0.99419754         -0.05526862 -0.05201218 -0.01253859
Experience (in years)    0.99419754            1.00000000         -0.04672895 -0.04991226 -0.01083962
Income (in K/month)     -0.05526862           -0.04672895          1.00000000  0.64598367  0.20680623
CCAvg                   -0.05201218           -0.04991226          0.64598367  1.00000000  0.10990472
Mortgage                -0.01253859           -0.01083962          0.20680623  0.10990472  1.00000000
> corrplot(matrix,type="upper",method = "number")
```





Looking at the above correlation plot helps us to understand:
Age and Experience are highly correlated
Income and CCAvg are moderately correlated

4. Clustering:

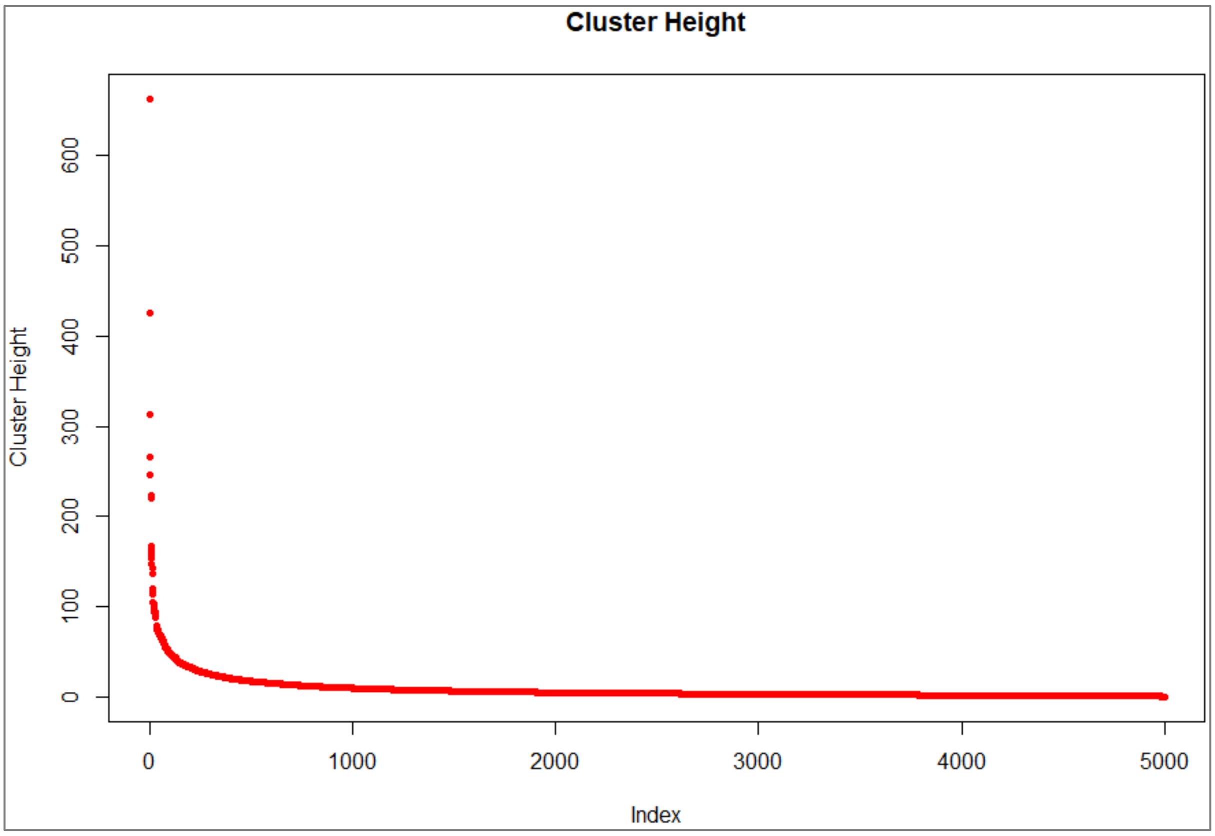
We will start with Clustering Analysis and checking which Clustering (Hierarchal / k-mean) method is best for given datasets as Clustering analysis helps is grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups (clusters).

I. Hierarchal Clustering:

Hierarchal Clustering can be performed by using Chebyshev & Euclidian distance Method and then the results of hierarchical clustering will be shown using dendrogram from top to bottom by executing following R code:

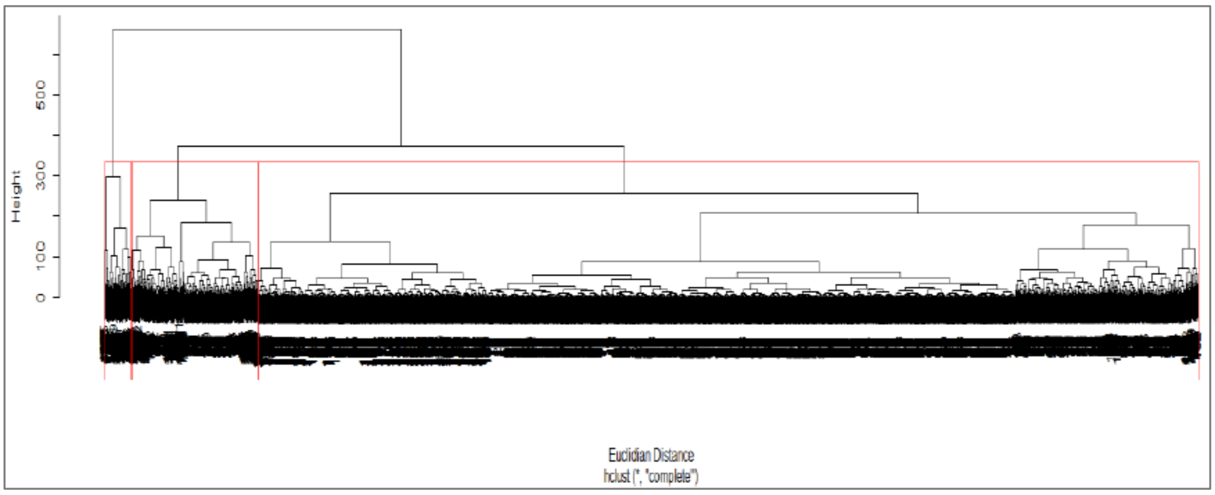
```
## Distance Calculation
Distchebyshev = dist(x=TBdata, method = "maximum")
Disteuc = dist(x=TBdata, method = "euclidean")
HCclusteuc = hclust(Disteuc, method = "complete")
HCclustch = hclust(Distchebyshev, method = "complete")

## Cluster Height, Sorting and Plotting
clusterheight = HCclusteuc$height
clusterheight = sort(clusterheight, decreasing = TRUE)
plot(clusterheight, pch =20, col="red", main="Cluster Height", ylab="Cluster Height")
lines(clusterheight, lty=2, lwd=2, col="blue")

## Cluster Plotting and Comparison
par(mfrow=c(2,1))
plot(HCclusteuc, labels = as.character(TBdata[,2]), main = "HClust using Euclidian method", xlab = "Euclidian
distance", ylab = "Height")rect.hclust(HCclusteuc, k=3, border = "red")
plot(HCclusteuc, labels = as.character(TBdata[,2]), main = "HClust using Chebychev method", xlab = "Chebychev
distance", ylab = "Height")rect.hclust(HCclusteuc, k=3, border = "red")
```

Cluster Height v/s index Plot

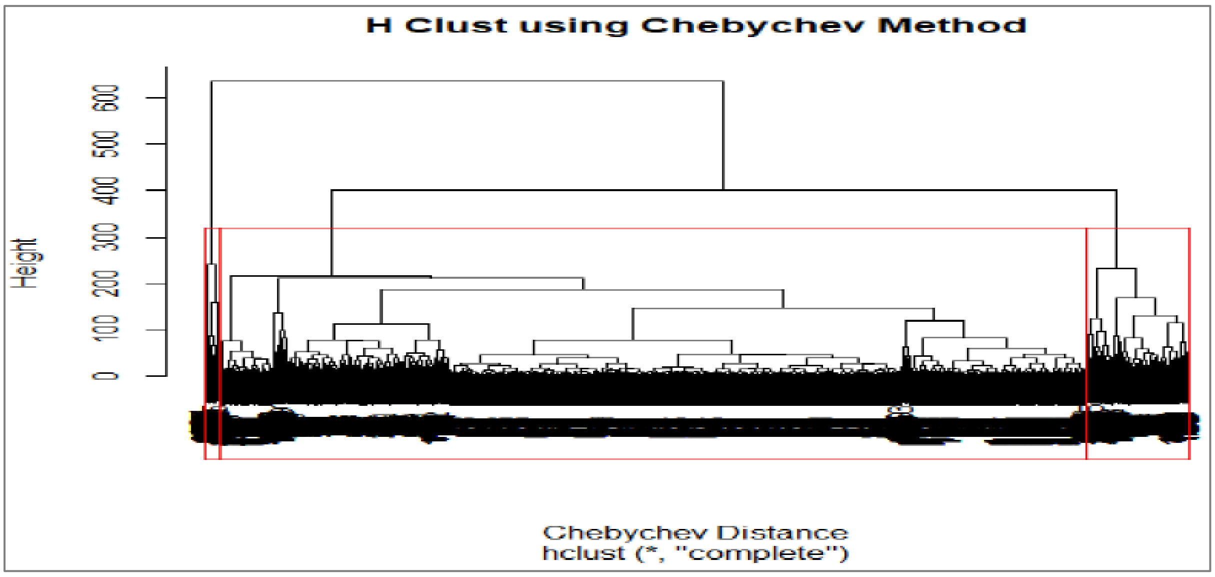


Note that after cluster height of 100 all vertical distance between two distances are not much so there is a possibility of 4-5 clusters where we can clearly cover maximum vertical distance which can be seen in above cluster plot as we will be plotting clusters based upon both the above defined model by executing following R Codes:- Output of above codes can be seen below:

H Clust using Euclidian Method and dividing in 3 random clusters

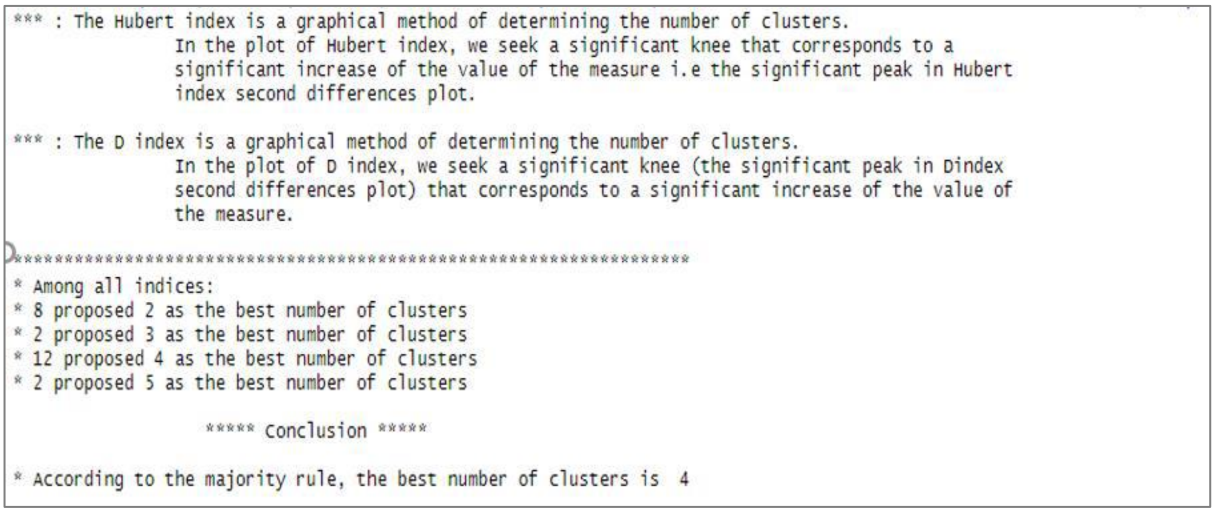


H Clust using Chebychev Method dividing in 3 random clusters



II. K-Means Clustering

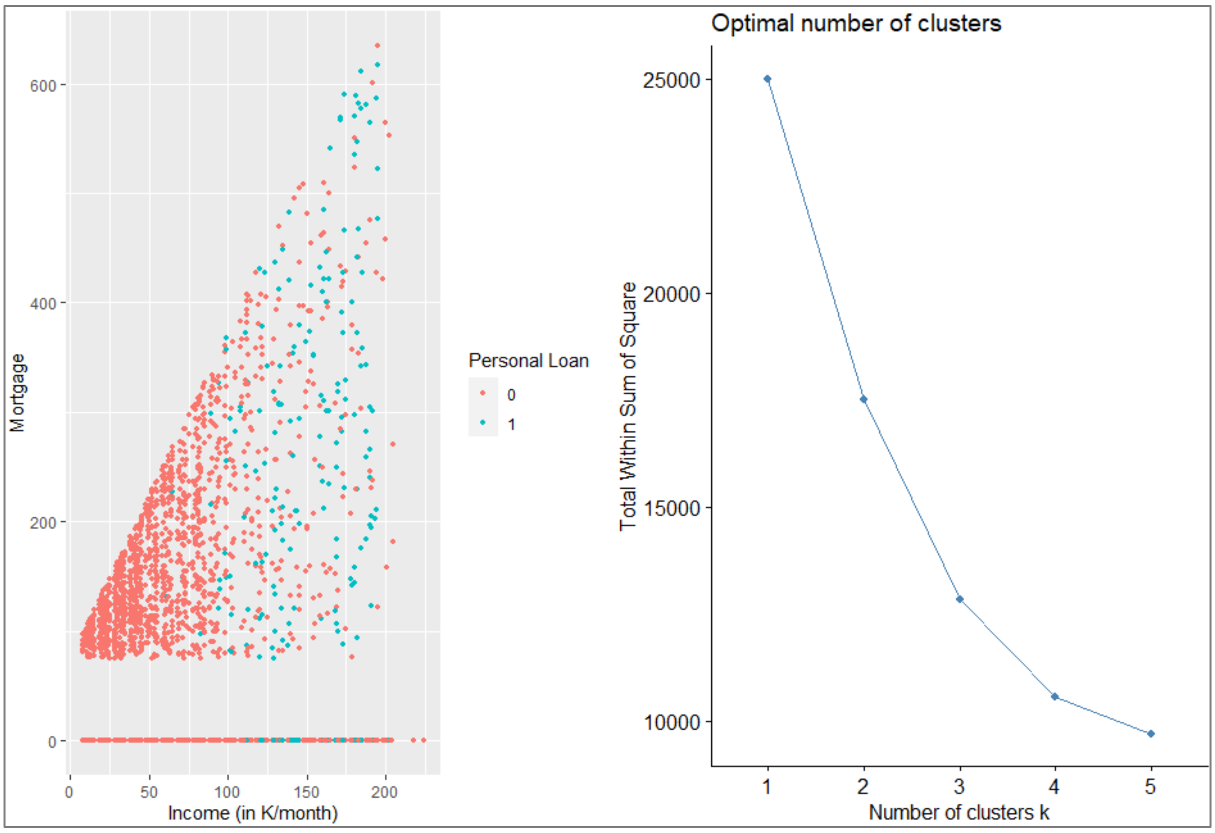
To analyse our data set lets us start with scaling our data to control the variability of the dataset, it converts data into specific range using a linear transformation which generate good quality clusters and improve the accuracy of clustering algorithms.



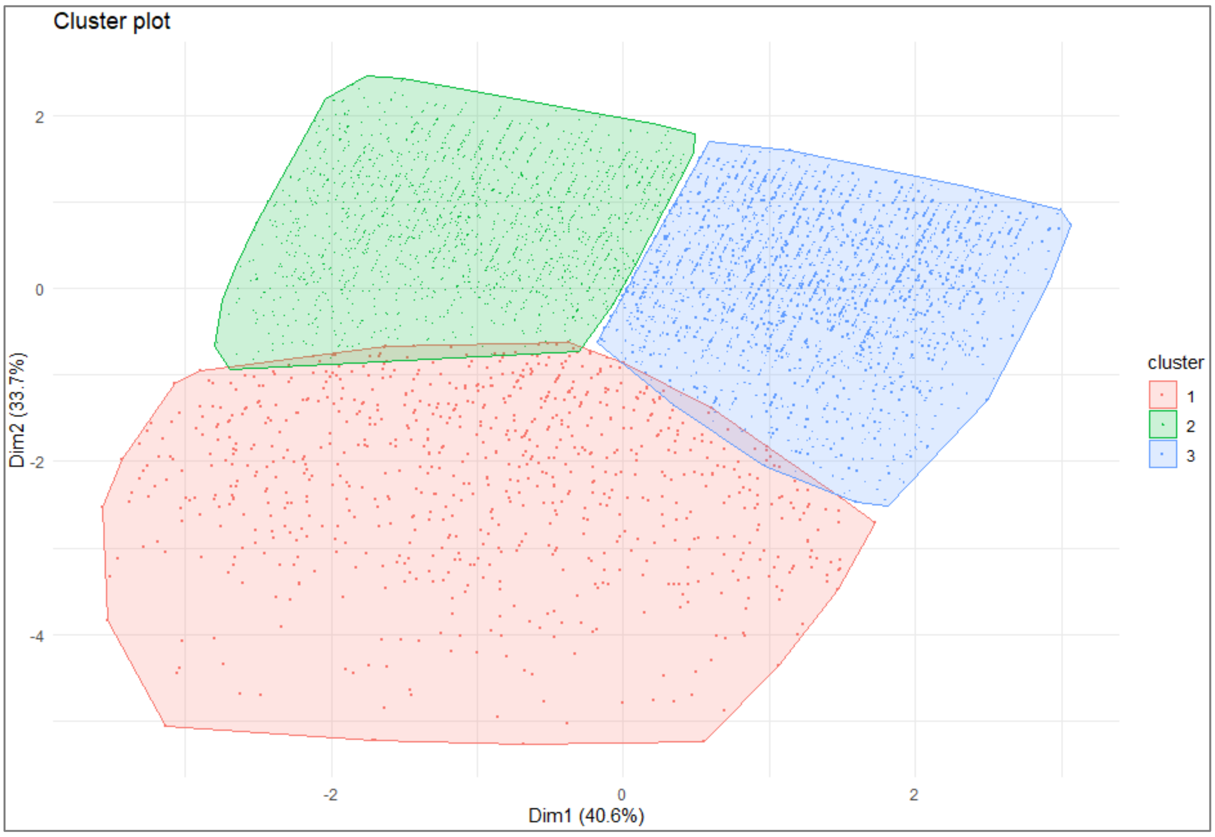
```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 8 proposed 2 as the best number of clusters
* 2 proposed 3 as the best number of clusters
* 12 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  4
```

Checking the optimal number of Clusters

```
## checking optimal number of clusters to categorize dataset
P20 = fviz_nbclust(TBdata.scaled, kmeans, method = "silhouette", k.max = 5)
p21 = fviz_nbclust(TBdata.scaled, kmeans, method = "wss", k.max = 5)
grid.arrange(p12, p21, ncol=2)
set.seed(1234)
TBdata.clusters = kmeans(TBdata.scaled, 3, nstart = 10)
fviz_cluster(TBdata.clusters, TBdata.scaled, geom = "point",
        ellipse = TRUE, pointsize = 0.2, ) + theme_minimal()
```



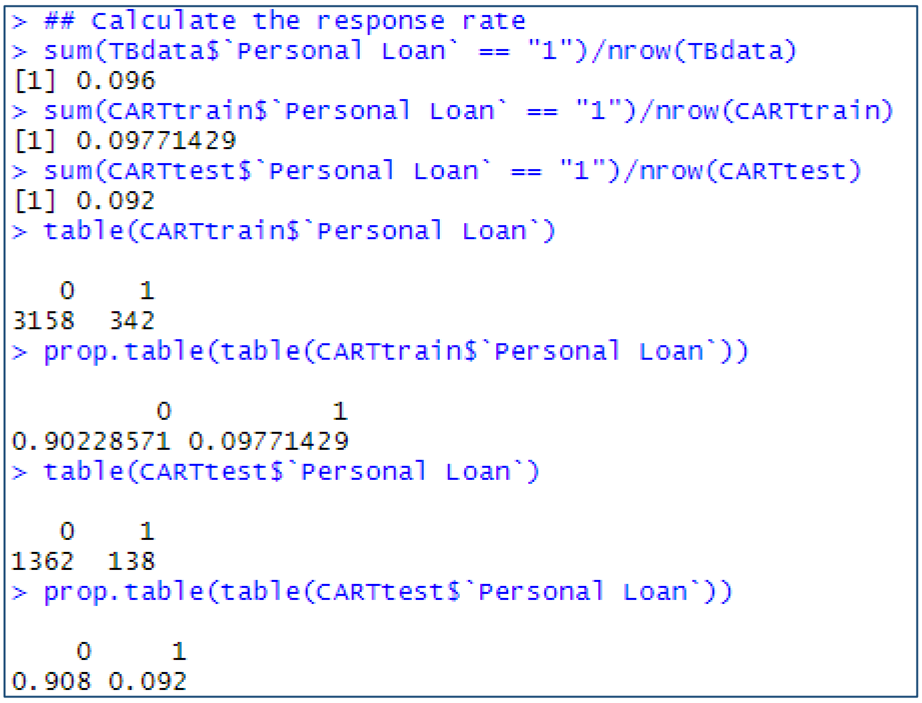
K-Means suits this type of data categorization K means divides the dataset into 3 clusters:



Splitting of Dataset into Train - Test Set

Next, we will split the dataset into training and test datasets. We will build the model based on the training dataset and test the model performance using the test dataset:

```
## Splitting the dataset into train and test for development and out of sample testing respectively
set.seed(100)
P_Loan_TRAIN_INDEX <- sample(1:nrow(TBdata),0.70*nrow(TBdata))
CARTtrain <- TBdata[P_Loan_TRAIN_INDEX,]
CARTtest <- TBdata[-P_Loan_TRAIN_INDEX,]
```



```
> ## Calculate the response rate
> sum(TBdata$`Personal Loan` == "1")/nrow(TBdata)
[1] 0.096
> sum(CARTtrain$`Personal Loan` == "1")/nrow(CARTtrain)
[1] 0.09771429
> sum(CARTtest$`Personal Loan` == "1")/nrow(CARTtest)
[1] 0.092
> table(CARTtrain$`Personal Loan`)

   0    1 
3158  342 
> prop.table(table(CARTtrain$`Personal Loan`))

         0          1 
0.90228571 0.09771429 
> table(CARTtest$`Personal Loan`)

   0    1 
1362  138 
> prop.table(table(CARTtest$`Personal Loan`))

    0     1 
0.908 0.092 
```

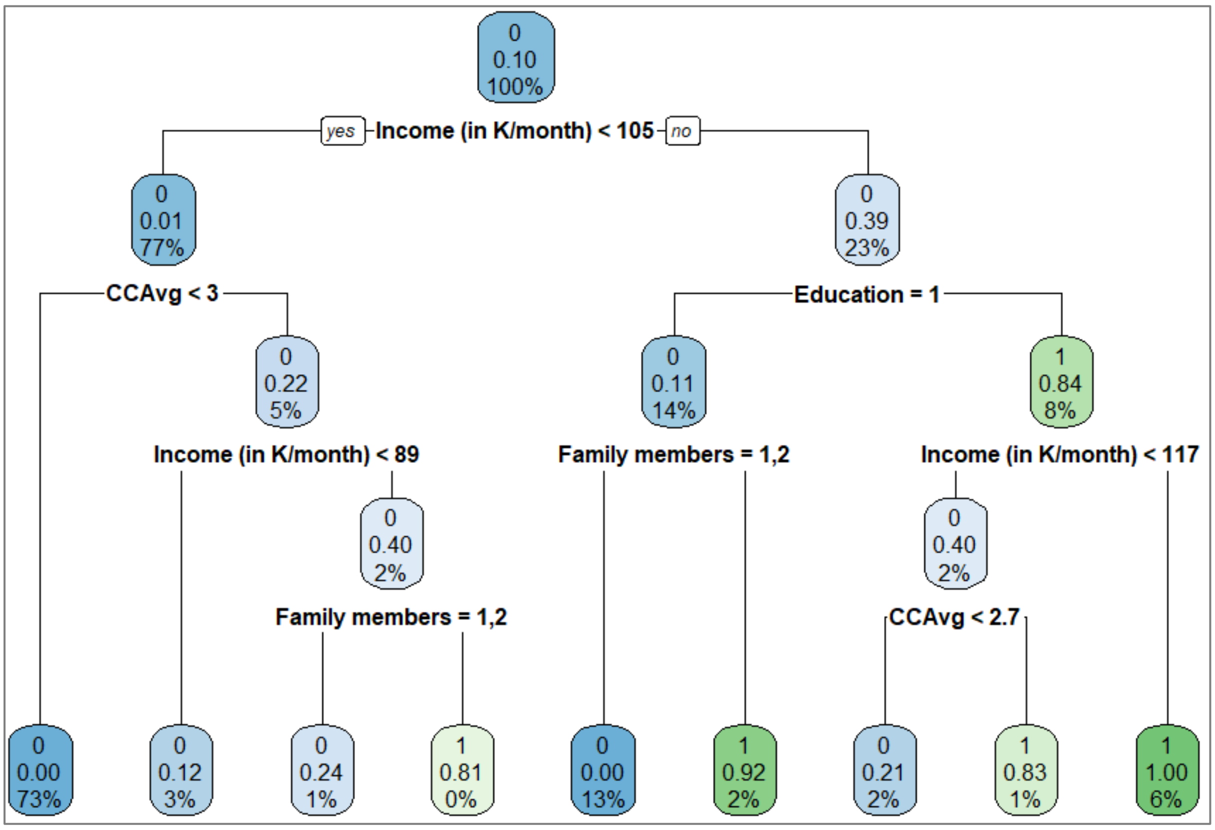
Output Analysis

The data is well distributed in the training and validation sets almost in the same proportion as they were in proportion earlier before split Now as we had successfully partitioned our data, we can proceed further with building of CART and Random Forest.

5. CART Model

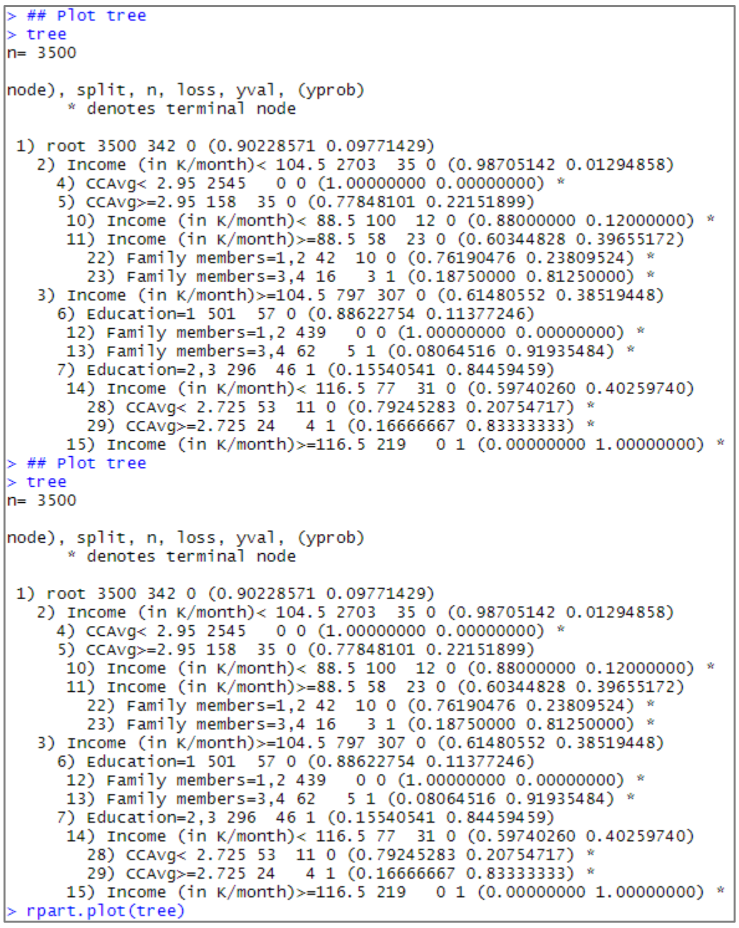
Classification trees use recursive partitioning algorithms to learn and grow on data:

```
## Build first CART model
## Setting the control parameters for rpart
#minsplit: if the number of records in a given node falls below a threshold, the node will not be split further.
#minbucket: minimum records in a terminal node. if the records are less, that bucket will not be created.
#minsplit = 3(minbucket)
#cp = cost complexity parameter
#We begin by building a very complex classification tree, by setting the "cost complexity" threshold
#to "0" and the minimum bucket size to be 10. Then we plot the tree using rpart.
tree = rpart(formula = `Personal Loan` ~ .,data=CARTtrain,method="class",minbucket = 10,cp=0)
```



Analysis:

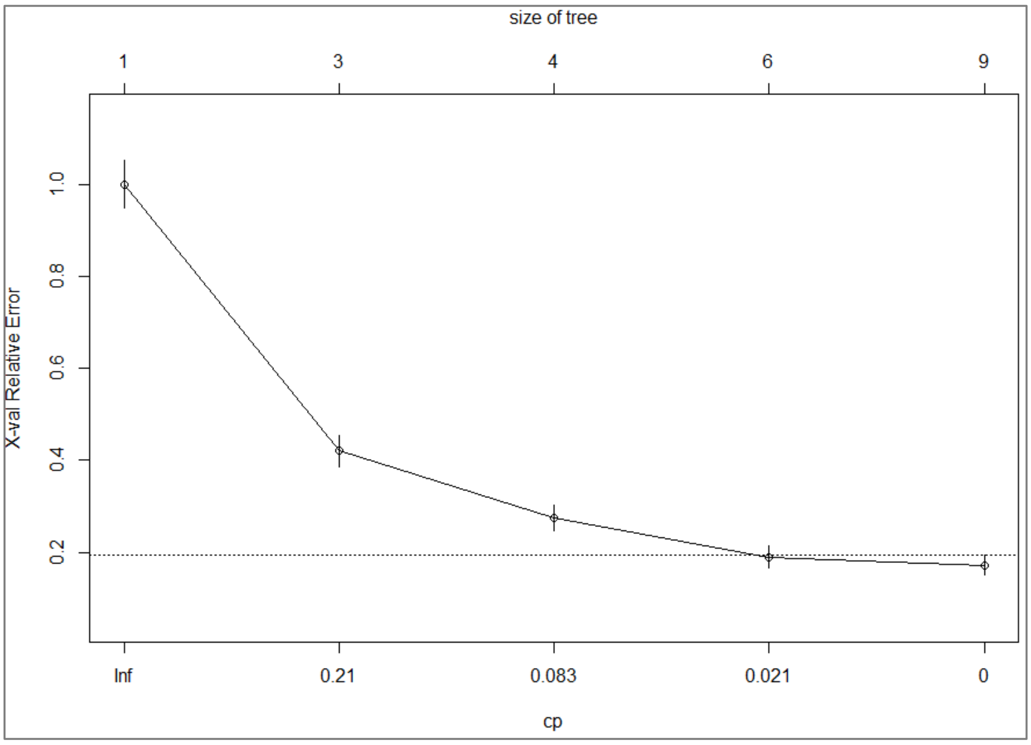
The above CART tree helps us to understand the first split of root node is by the income
Further the tree is split into sub nodes by CCAvg, Education and Family members
Terminal nodes help us make the decision



```
> ## Plot tree
> tree
n= 3500

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 3500 342 0 (0.90228571 0.09771429)
   2) Income (in K/month)< 104.5 2703  35 0 (0.98705142 0.01294858)
     4) CCAvg< 2.95 2545   0 0 (1.00000000 0.00000000) *
     5) CCAvg>=2.95 158  35 0 (0.77848101 0.22151899)
      10) Income (in K/month)< 88.5 100  12 0 (0.88000000 0.12000000) *
      11) Income (in K/month)>=88.5 58  23 0 (0.60344828 0.39655172)
        22) Family members=1,2 42  10 0 (0.76190476 0.23809524) *
        23) Family members=3,4 16   3 1 (0.18750000 0.81250000) *
   3) Income (in K/month)>=104.5 797 307 0 (0.61480552 0.38519448)
     6) Education=1 501  57 0 (0.88622754 0.11377246)
      12) Family members=1,2 439   0 0 (1.00000000 0.00000000) *
      13) Family members=3,4 62   5 1 (0.08064516 0.91935484) *
     7) Education=2,3 296  46 1 (0.15540541 0.84459459)
      14) Income (in K/month)< 116.5 77  31 0 (0.59740260 0.40259740)
        28) CCAvg< 2.725 53  11 0 (0.79245283 0.20754717) *
        29) CCAvg>=2.725 24   4 1 (0.16666667 0.83333333) *
      15) Income (in K/month)>=116.5 219   0 1 (0.00000000 1.00000000) *
```

```
> ## Plot tree
> tree
n= 3500

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 3500 342 0 (0.90228571 0.09771429)
   2) Income (in K/month)< 104.5 2703  35 0 (0.98705142 0.01294858)
     4) CCAvg< 2.95 2545   0 0 (1.00000000 0.00000000) *
     5) CCAvg>=2.95 158  35 0 (0.77848101 0.22151899)
      10) Income (in K/month)< 88.5 100  12 0 (0.88000000 0.12000000) *
      11) Income (in K/month)>=88.5 58  23 0 (0.60344828 0.39655172)
        22) Family members=1,2 42  10 0 (0.76190476 0.23809524) *
        23) Family members=3,4 16   3 1 (0.18750000 0.81250000) *
   3) Income (in K/month)>=104.5 797 307 0 (0.61480552 0.38519448)
     6) Education=1 501  57 0 (0.88622754 0.11377246)
      12) Family members=1,2 439   0 0 (1.00000000 0.00000000) *
      13) Family members=3,4 62   5 1 (0.08064516 0.91935484) *
     7) Education=2,3 296  46 1 (0.15540541 0.84459459)
      14) Income (in K/month)< 116.5 77  31 0 (0.59740260 0.40259740)
        28) CCAvg< 2.725 53  11 0 (0.79245283 0.20754717) *
        29) CCAvg>=2.725 24   4 1 (0.16666667 0.83333333) *
      15) Income (in K/month)>=116.5 219   0 1 (0.00000000 1.00000000) *
```

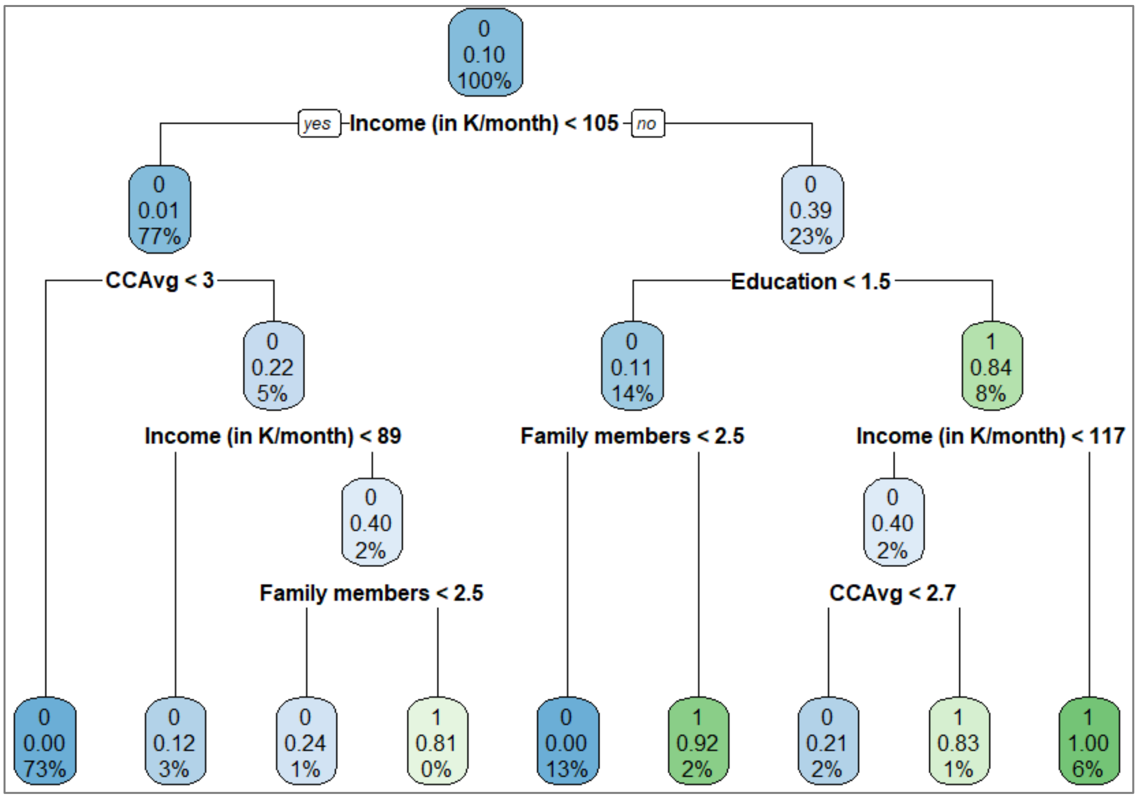
```
> rpart.plot(tree)
```

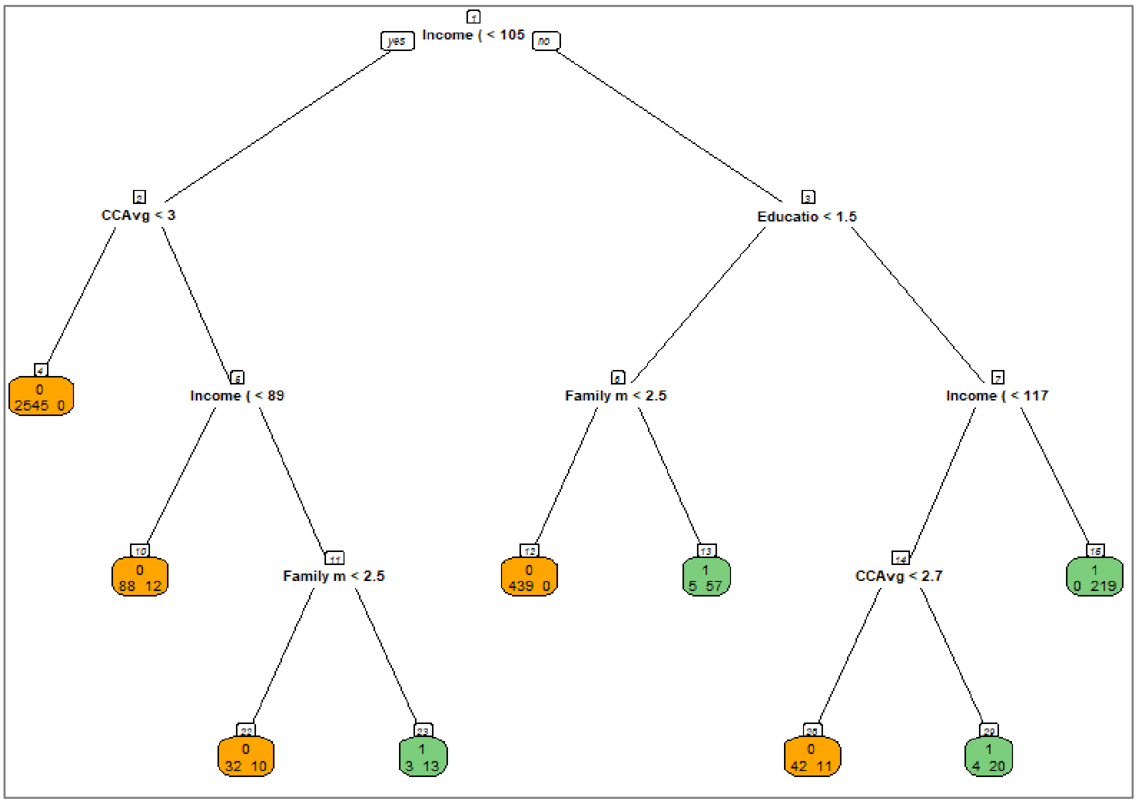


From above we can find the cp value for Pruning the tree

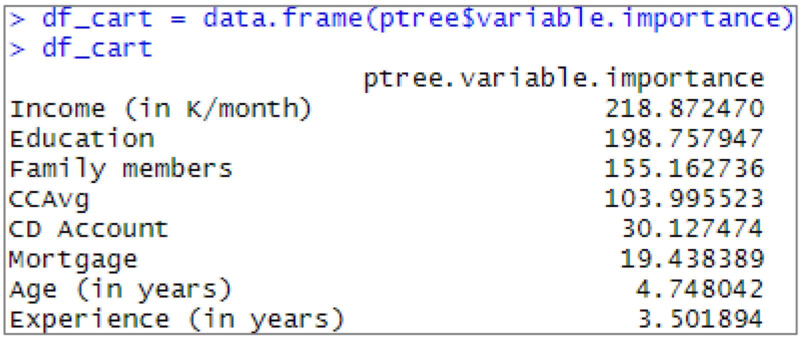
Pruning the Tree:

#The unnecessarily complex tree above can be pruned using a cost complexity threshold.
#Using a complexity threshold of 0.003 gives us a much simpler tree.





```
## Variable Importance
library(caret)
summary(ptree)
ptree$variable.importance
df_cart = data.frame(ptree$variable.importance)
df_cart
```



```
> df_cart = data.frame(ptree$variable.importance)
> df_cart
                           ptree.variable.importance
Income (in K/month)                       218.872470
Education                                 198.757947
Family members                            155.162736
CCAvg                                     103.995523
CD Account                                 30.127474
Mortgage                                   19.438389
Age (in years)                              4.748042
Experience (in years)                       3.501894
```

Predicting the Target variable using the 'Pruned Tree'. On test dataset

Since this is a loan prediction and we want to be more careful to weed out possible defaulters rather than deny the disbursal to deserving prospects We will set the threshold for probability as high as 0.70

All the predicted probabilities >=0.7 will be considered as class "1" and rest class "0"

Model Performance Measures

After predicting the values, we carry out model performance measures to compare various models to find the best model required.

Confusion Matrix for Cart model

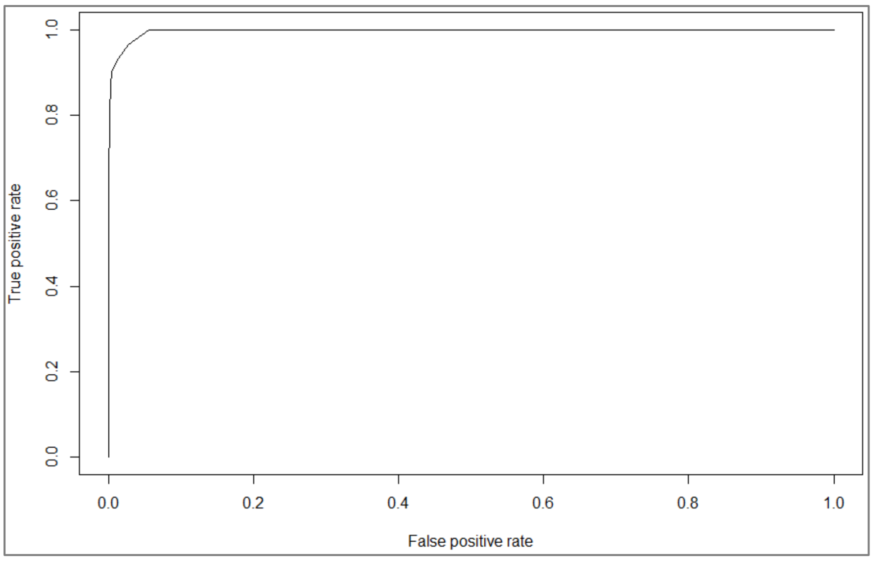
CARTTrain	Predicted	CARTTest	Predicted
0 1		0 1	
actual	0 3146 33	actual	0 1352 18
1 12 309		1 10 120	

Above is the table which shows the values of predictions and actual and will help to check the performance of model by finding accuracy, specificity sensitivity etc. parameters.

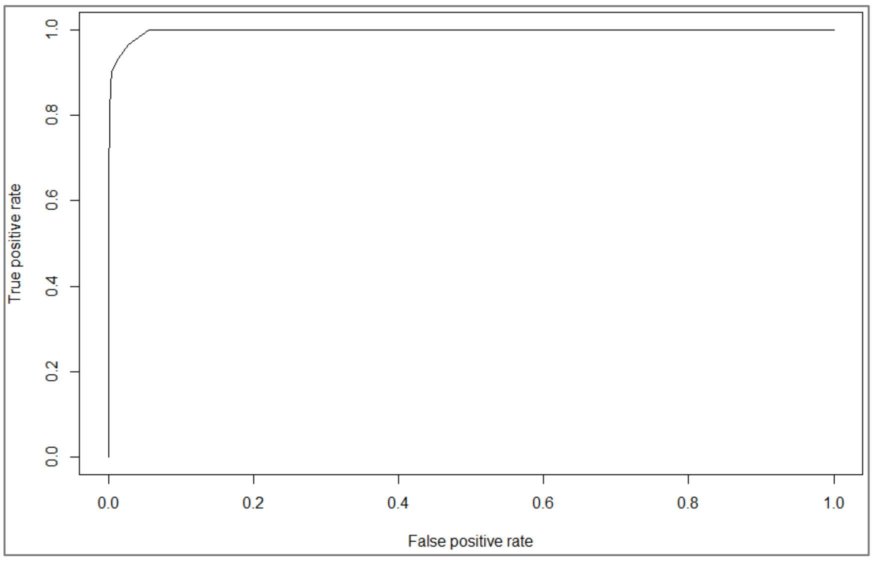
Model performance Measures by CART Model:

CART									
	Accuracy	Sensitivity	Specificity	Lift	KS	AUC	GINI Coefficient	GINI	Concordance
Train Data	98.7%	90.3%	99.6%	8.99	94.5%	99.7%	1.98	89.73%	99.62%
Test Data	98.1%	87.1%	99.3%	9.09	92.9%	99.5%	1.98	89.77%	99.36%

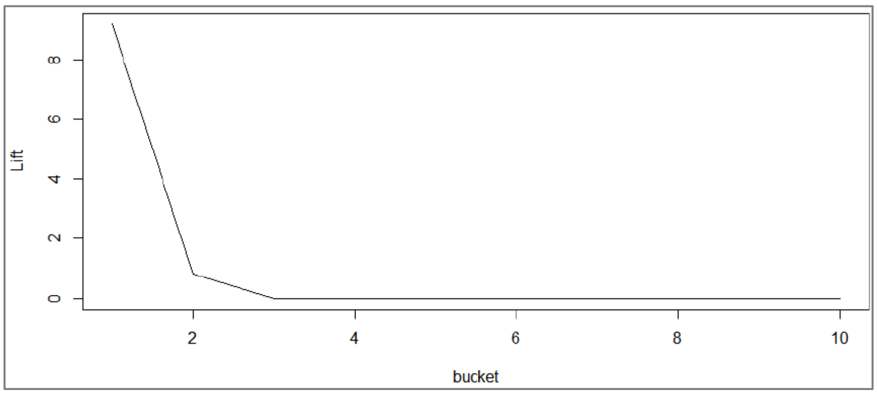
CART Train Dataset ROC Curve



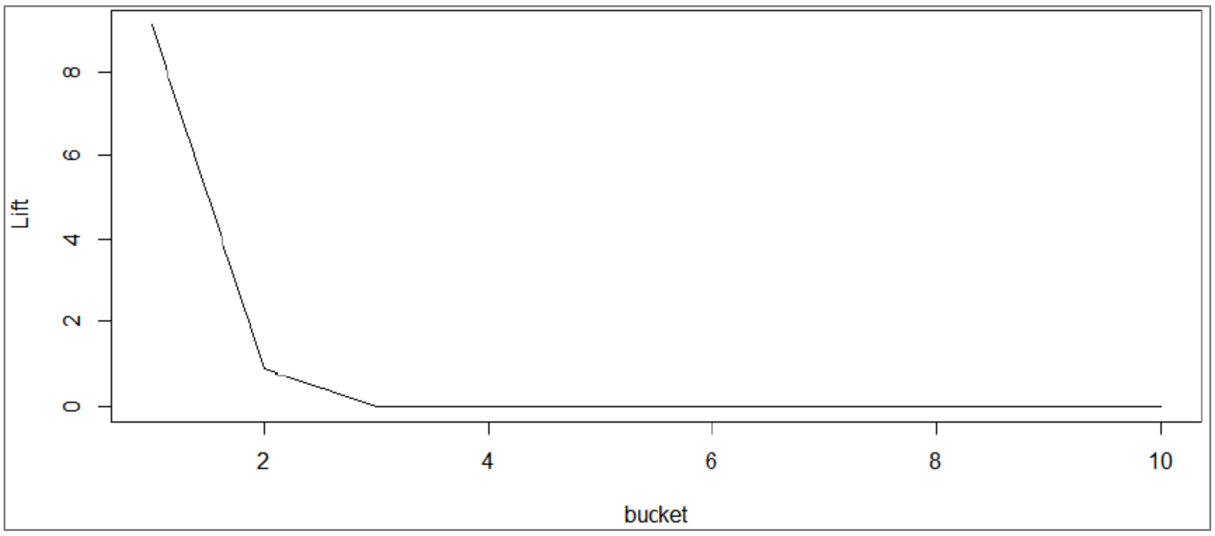
CART Test Dataset ROC Curve



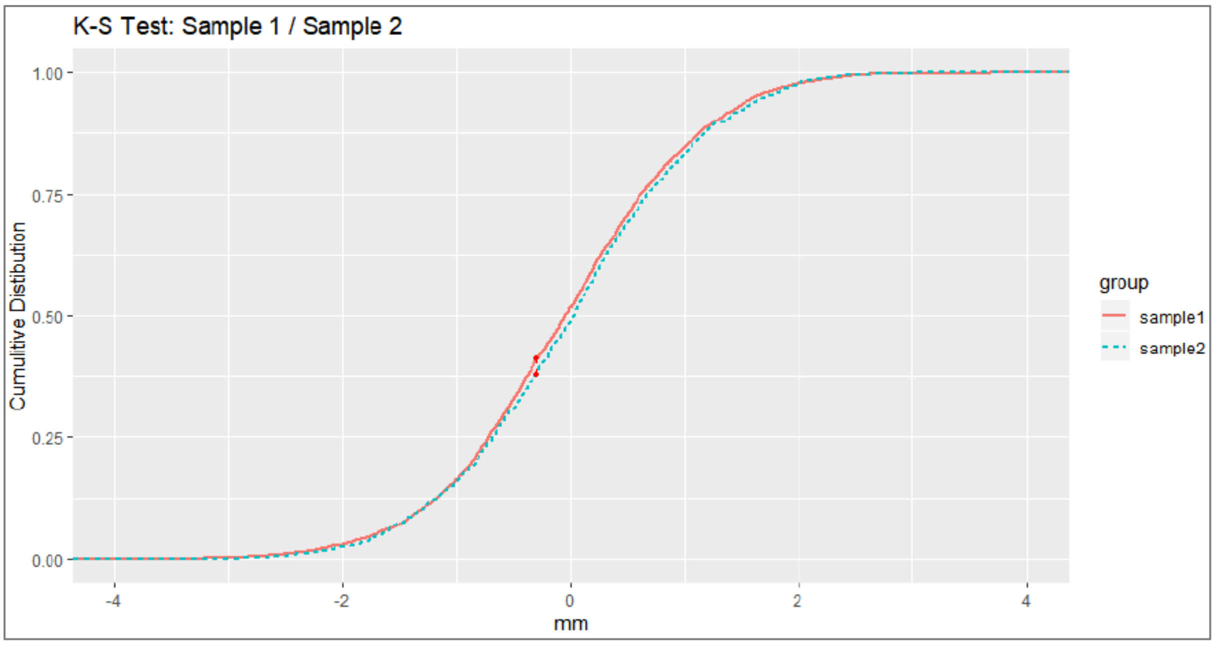
Lift Graph for Cart Train Model



Lift Graph for Cart Test Model



KS Graph for CART Model



Rank Ordering Table for Cart Train Model

deciles	cnt	cnt_P_Loan	cnt_no_P_Loan	rrate	cum_P_Loan	cum_no_P_Loan	cum_rel_P_Loan	cum_rel_no_P_L	rrate_perc	cum_rel_P_Loan	cum_rel_no_P_L	cum_cnt	cum_P_Loan_ra	ks	lift
10	363	319	44	0.8788	319	44	0.9327	0.0139	87.90%	93.27%	1.40%	363	0.8788	0.9188	8.99
9	3137	23	3114	0.0073	342	3158	1	1	0.70%	100.00%	100.00%	3500	0.0977		1

Rank Order Table for Cart Test Model

Deciles	cnt	cnt_P_L	cnt_no_	rrate	cum_P_ Loan	cum_no	cum_rel	cum_rel	rrate_p	cum_rel P_Loan	cum_rel no_P_L	cum_cn	cum_P_	ks	lift
10	153	128	25	0.8366	128	25	0.9275	0.0184	83.70%	92.75%	1.80%	153	0.8366	0.9091	9.09
9	1347	10	1337	0.0074	138	1362	1	1	0.70%	100.00 %	100.00 %	1500	0.092	0	0

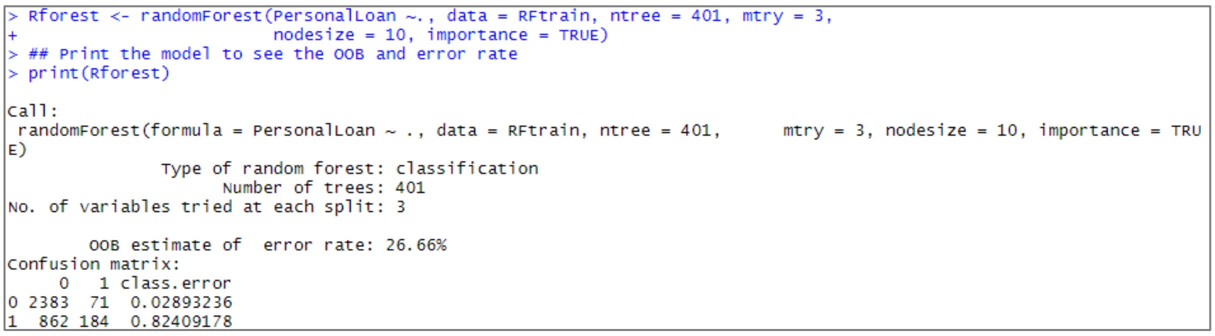
Inference from CART Model:

The model is accurate 98% of times and the various performance and graphs can be studied above.
Income is the variable which is playing a vital role in distribution of Personal Loan to prospective customers.

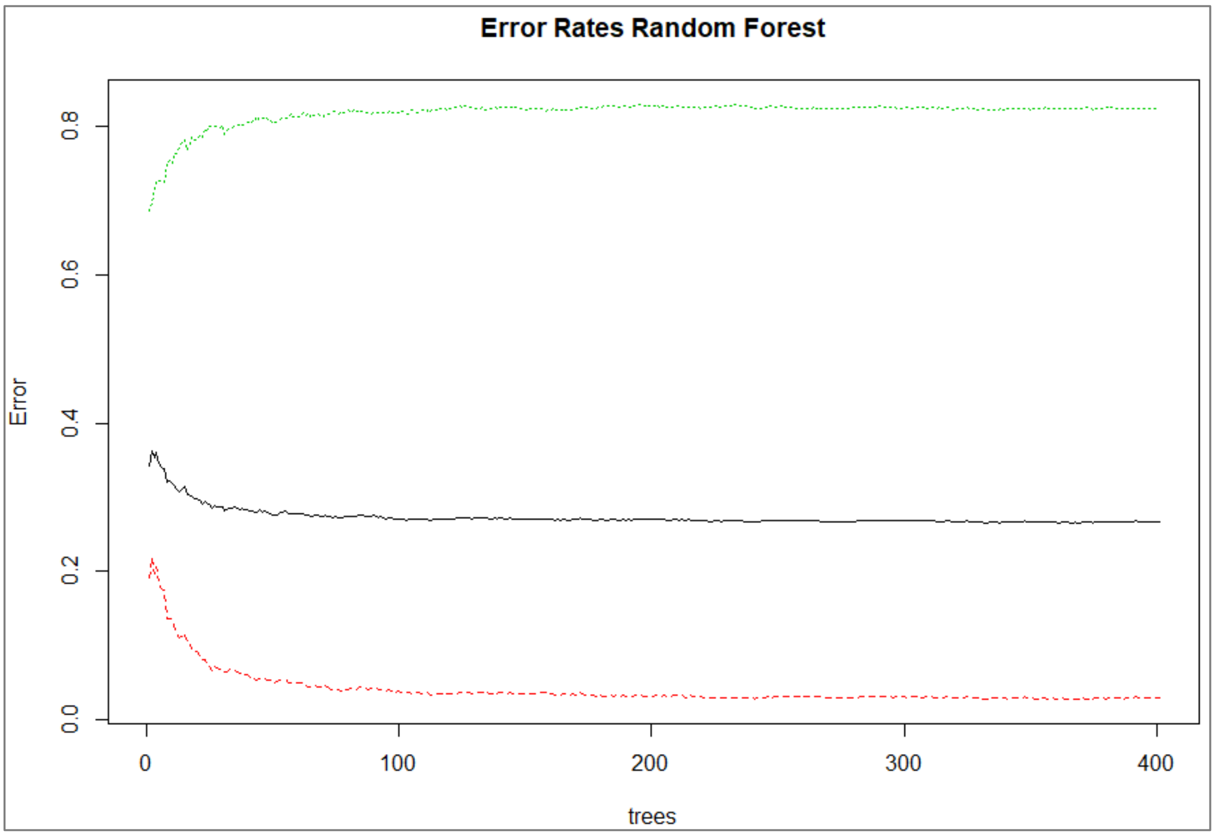
6. Random Forest Model:

Random Forests is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration. Random forest is an ensemble method used by combining weak and strong learners to give a better accuracy or output. It's a combination of multiple trees each chosen randomly to grow on dataset Uses averaging in the sense that weak and strong learners combined produce better results rather than a single CART tree.

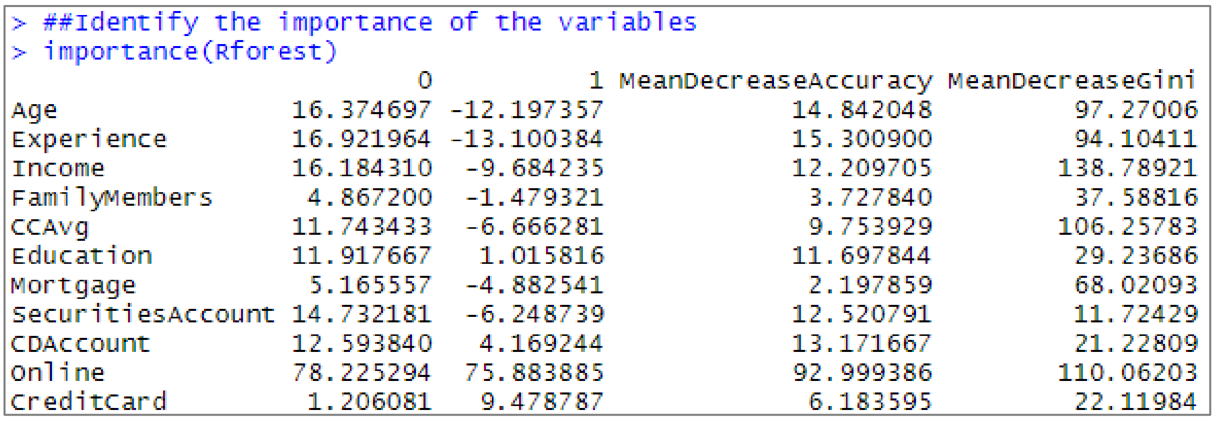
```
#ntree: number of trees to grow
#mtry: number of variables to be considered for split
#nodesize: minimum size (number of records) of terminal nodes
#should importance of predictors be assessed
mtry = floor(sqrt(ncol(RFtrain)))
mtry
Rforest <- randomForest(PersonalLoan ~., data = RFtrain, ntree = 401, mtry = 3,
           nodesize = 10, importance = TRUE)
```



```
> Rforest <- randomForest(PersonalLoan ~., data = RFtrain, ntree = 401, mtry = 3,
+                         nodesize = 10, importance = TRUE)
> ## Print the model to see the OOB and error rate
> print(Rforest)

Call:
 randomForest(formula = PersonalLoan ~ ., data = RFtrain, ntree = 401,      mtry = 3, nodesize = 10, importance = TRU
E)
               Type of random forest: classification
                     Number of trees: 401
No. of variables tried at each split: 3

        OOB estimate of  error rate: 26.66%
Confusion matrix:
     0   1 class.error
0 2383  71  0.02893236
1  862 184  0.82409178
```



#The error rate plot w.r.t number of trees reveals that anything more than, say 181, trees are #really not that valuable.

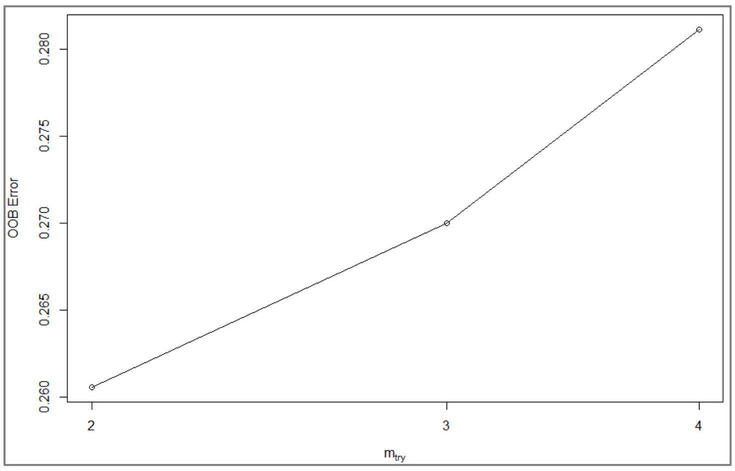


```
> ##Identify the importance of the variables
> importance(Rforest)
                          0          1 MeanDecreaseAccuracy MeanDecreaseGini
Age               16.374697 -12.197357            14.842048         97.27006
Experience        16.921964 -13.100384            15.300900         94.10411
Income            16.184310  -9.684235            12.209705        138.78921
FamilyMembers      4.867200  -1.479321             3.727840         37.58816
CCAvg             11.743433  -6.666281             9.753929        106.25783
Education         11.917667   1.015816            11.697844         29.23686
Mortgage           5.165557  -4.882541             2.197859         68.02093
SecuritiesAccount 14.732181  -6.248739            12.520791         11.72429
CDAccount         12.593840   4.169244            13.171667         21.22809
Online            78.225294  75.883885            92.999386        110.06203
CreditCard         1.206081   9.478787             6.183595         22.11984
```

#Larger the MeanDecrease values, the more important the variable

Tune up the RF model to find out the best mtry

```
## Tune up the RF model to find out the best mtry
# Now we will "tune" the Random Forest by trying different m values. We will stick with
# 201 trees (odd number of trees are preferable). The returned forest, "tRforest" is the
# one corresponding to the best m
# Starting with the default value of mtry, search for the optimal value (with respect to Out-of-Bag error
# estimate) of mtry for randomForest.
# Parameter Explanation
# x - Predictor variables
# y - Target Variable (factor for classification, numeric for regression)
# mtryStart - starting value of mtry; default is the same as in randomForest
# ntreeTry - number of trees used at the tuning step
# stepFactor - at each iteration, mtry is inflated (or deflated) by this value
# improve - the (relative) improvement in OOB error must be by this much for the search to continue
# trace - whether to print the progress of the search
# plot - whether to plot the OOB error as function of mtry
# doBest - whether to run a forest using the optimal mtry found
# If doBest=FALSE (default), it returns a matrix whose first column contains the mtry values searched, and the
second column the corresponding OOB error.
# If doBest=TRUE, it returns the randomForest object produced with the optimal mtry.
# nodesize - min terminal node size
# importance - compute variable importance or not
```



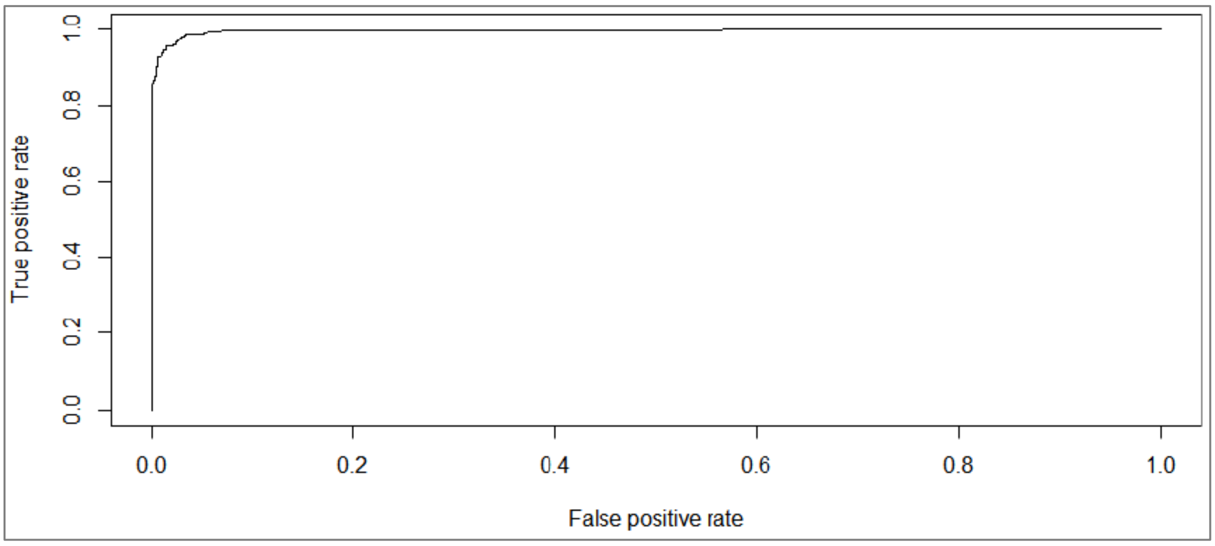
Model Performance Measures:

RF Train		Predicted		RF Test		Predicted	
		0	1			0	1
actual	0	3155	55	actual	0	1361	22
	1	3	287		1	1	116

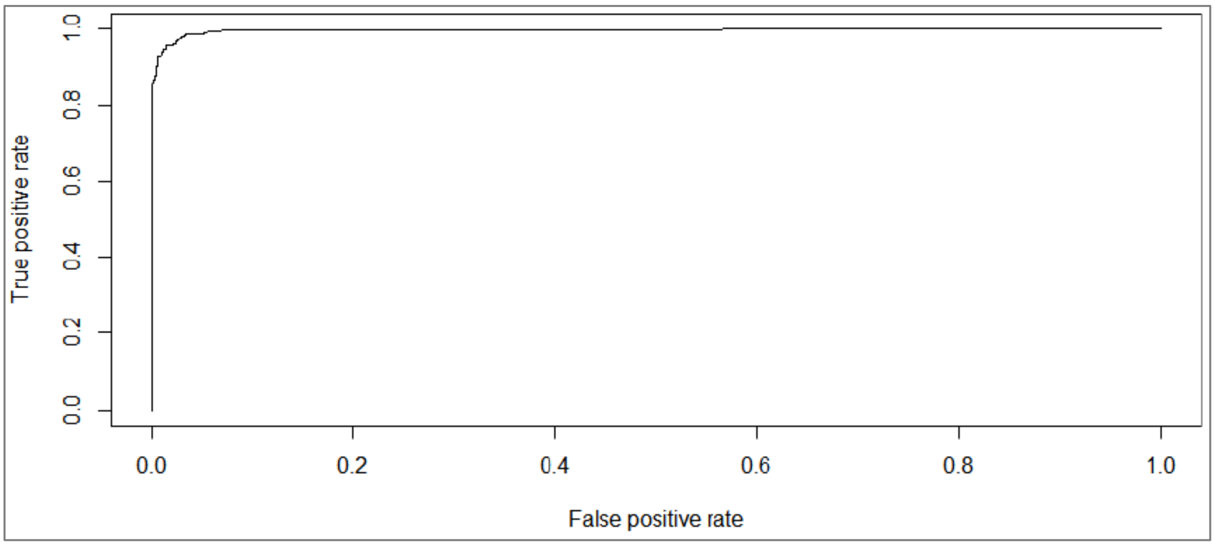
Above is the table which shows the values of predictions and actual and will help to check the performance of model by finding accuracy, specificity sensitivity etc. parameters.

Random Forest									
Accuracy		Sensitivity	Specificity	Lift	KS (%)	AUC (%)	GINI Coefficient	GINI (%)	Concordance (%)
Train Data	98.3%	83.9%	99.9%	9.33	95.1	99.6	0.99	89.6	99.5
Test Data	98.5%	84.1%	99.9%	9.29	94.6	99.7	0.99	89.8	99.7

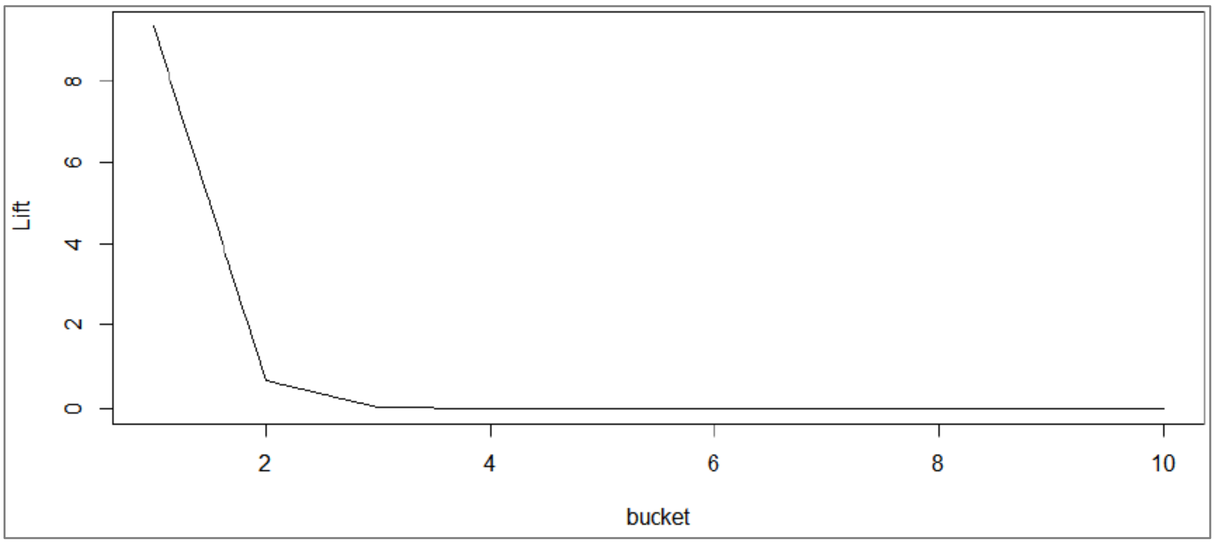
ROC Curve for RF Train Model



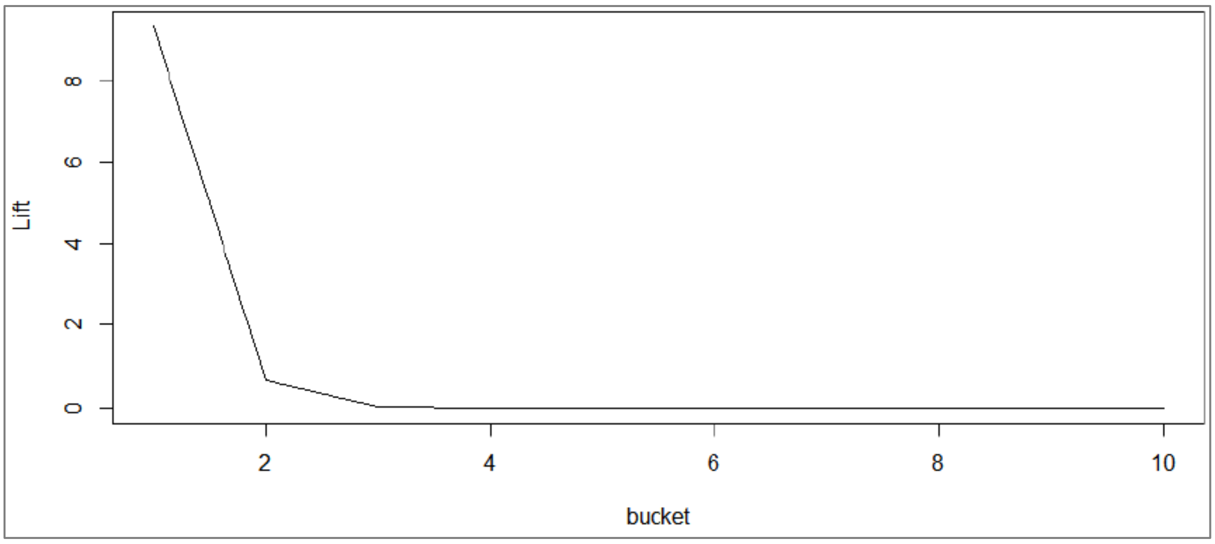
ROC Curve for RF Test Model



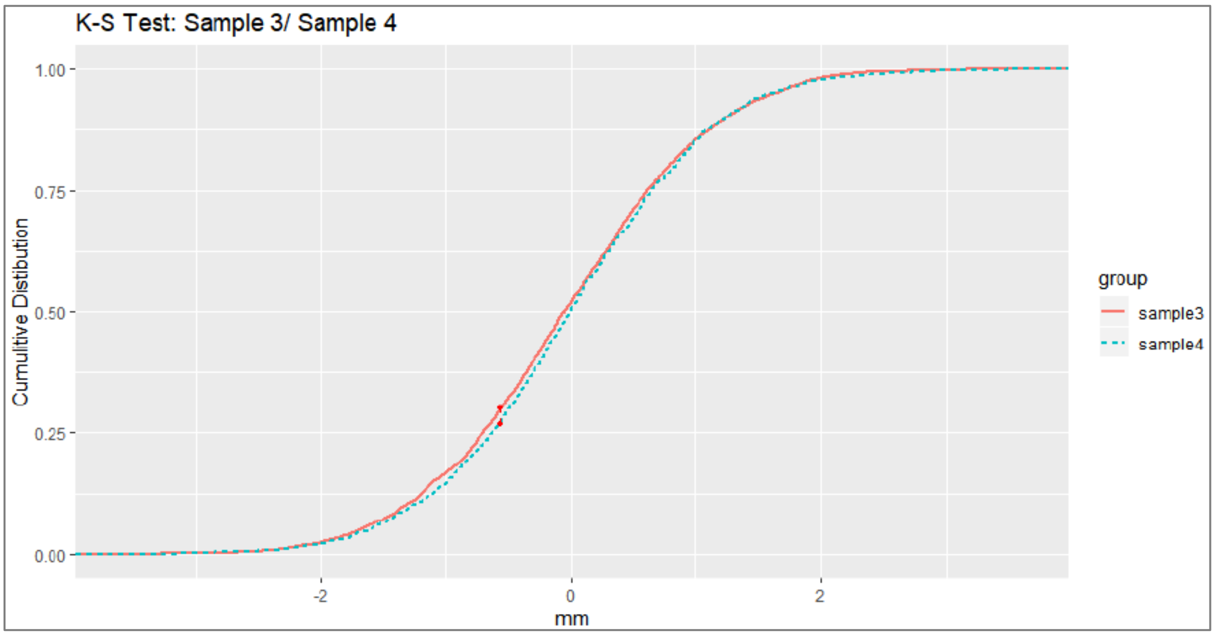
Lift Chart for RF Train Model



Lift Graph for RF Test Model



KS Graph for RF Model



Rank Ordering Table for RF Train Model

cnt	cnt_P_L	cnt_no_P_L	rrate	cum_P_L	cum_no_P_L	cum_rel_P_L	cum_rel_no_P_L	rrate_perc	cum_rel_P_L_	cum_rel_no_P	cum_cnt	cum_P_L_rate	ks	lift	cnt
10	350	319	31	0.9114	319	31	0.9327	0.0098	91.10%	93.27%	1.00%	350	0.9114	0.9229	9.33
9	359	22	337	0.0613	341	368	0.9971	0.1165	6.10%	99.71%	11.60%	709	0.481	0.8806	4.92
8	2791	1	2790	4.00E-04	342	3158	1	1	0.00%	100.00%	100.00%	3500	0.0977	0	1

Rank Order Table for RF Test Model

Deciles	cnt	cnt_P_L oan	cnt_no_ P_Loan	rrate	cum_P_ Loan	cum_no P_Loan	cum_rel	cum_rel	rrate_p erc	cum_rel P_Loan	cum_rel no_P_L	cum_cn	cum_P_ Loan_r	ks	lift
10	151	129	22	0.8543	129	22	0.9348	0.0162	85.40%	93.48%	1.60%	151	0.8543	0.9186	9.29
9	177	9	168	0.0508	138	190	1	0.1395	5.10%	100.00%	14.00%	328	0.4207	0.8605	4.57
8	1172	0	1172	0	138	1362	1	1	0.00%	100.00%	100.00%	1500	0.092	0	1

Inferences from RF Model

The Accuracy of the RF model is 98.5%

Education plays an important role in this model followed by Income and mortgage etc.

Inferences from the Case Study

Education, Income, Age, Mortgage are playing important role in the determination of prospective customers for Personal Loans.

There needs to be a campaign targeting the people with Low income. Also, zero mortgage value customers should be targeted to accept Personal Loans., as they haven’t accepted any loans in the last campaign. Both the models CART and Random Forest are almost giving good results in classifying the right customers who have a higher probability of purchasing the loan.

Our objective to find out Thera bank approach to their customers for campaigning of personal loan. Before launching new marketing campaign, they wanted to analyse the customer profile based on the available historical data by studying the variables. Overall, we used two models CART and Random Forest for studying the customer profiles. On studying we found that Decision Tree algorithm have the high accuracy to decide as our final model for ref and use.

The Random Forest Model is better predictive than the CART Model.

```
#=========================================================================#
#Data Analysis  -   Thera Bank - Loan Purchase Modeling
#Developer      -  Tahmid Bari
#Date          -   May 23, 2020
#=========================================================================#

#install.packages("ggplot2")
#install.packages("corrplot")
#install.packages("xlsx")
#install.packages("dplyr")
#install.packages("devtools")
#install.packages("NbClust")
#install.packages("caret, repos = http://cran.us.r-project.org")
#install.packages("rpart, repos = http://cran.us.r-project.org")
#install.packages("rpart.plot, repos = http://cran.us.r-project.org")
#install.packages("randomForest, repos = http://cran.us.r-project.org")


## Loading Library ##
library(xlsx)
library(readxl)
library("readr")
library(dplyr)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(lattice)
library(DataExplorer)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(ROCR)
library(ineq)
library(InformationValue)
library(data.table)
library(scales)
library(Metrics)
library(grDevices)
library(factoextra)
library(ROCit)
library(kableExtra)

## Set working directory
setwd("C:/Users/Tahmid Bari/Desktop/Great Learning/Course Work/Machine Learning/Project-3")

## Check working directory
getwd()
```

```
TBdata <- read_excel("Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx")
file.exists("C:\\Users\\Tahmid Bari\\Desktop\\Great Learning\\Course Work\\Machine Learning\\Project-3\\Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx")

### Exploratory Data Analysis
head(TBdata,10)
tail(TBdata,10)

## Lets see the variable list in the data
names(TBdata)

## Reordering the position of target variable/dependent variable Personal Loan to last Data
TBdata[c(1,2,3,4,5,6,7,8,9,11,12,13,14,10)]

## Removing column Id and zip code as they will not help much in analysis since they are basically
## addon - para information
TBdata = TBdata[,-c(1,5)]

## Check Missing values
colSums(is.na(TBdata))
## Family members has missing values

## Lets see the datatype of the data set
str(TBdata)
summary(TBdata)

## Experience (in years ) has negative values which is incorrect value
## Check the dimension or shape of the data
dim(TBdata)

## Fixing the Missing Values in Family members and negative values in Experience by algorithms
TBdata$`Experience (in years)`[TBdata$`Experience (in years)` < 0] <- 0
TBdata$`Family members`[is.na(TBdata$`Family members`)] <- median(TBdata$`Family members`, na.rm=TRUE)

## Converting target variable into factor
TBdata$`Personal Loan` = as.factor(TBdata$`Personal Loan`)

## Converting categorical variables into factor
TBdata$`Family members` = as.factor(TBdata $`Family members`)
TBdata$Education = as.factor(TBdata $Education)
TBdata$`Securities Account`= as.factor(TBdata $`Securities Account`)
TBdata$`CD Account` = as.factor(TBdata $`CD Account`)
TBdata$Online = as.factor(TBdata $Online)
TBdata$CreditCard = as.factor(TBdata$CreditCard)

## Introducing the Dataset
plot_intro(TBdata)
```

```
## Plotting the histogram for all numerical variables
plot_histogram(TBdata)

##Frequency distribution for categorical variable (Univariate Analysis)
table(TBdata[,c(4)])
table(TBdata[,c(6)])
table(TBdata[,c(8)])
table(TBdata[,c(9)])
table(TBdata[,c(10)])
table(TBdata[,c(11)])
table(TBdata[,c(12)])

## Plotting bar plot for all factor variables
plot_bar(TBdata[c(4,6,8,9,10,11,12)])

## Box plot for all variables
boxplot(TBdata[-c(4,6,8,9,10,11,12)],las=2)



## Bivariate analyisis
## Plotting boxplot for Personal Loan (Response variable) for all variables
p1 = ggplot(TBdata, aes(`Age (in years)`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p2 = ggplot(TBdata, aes(`Experience (in years)`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p3 = ggplot(TBdata, aes(`Income (in K/month)`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p4 = ggplot(TBdata, aes(`Mortgage`, fill= `Personal Loan`)) + geom_density(alpha=0.5)
p5 = ggplot(TBdata, aes(`Education`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p6 = ggplot(TBdata, aes(`Family members`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p7= ggplot(TBdata, aes(`CCAvg`, fill= `Personal Loan`)) + geom_histogram(alpha=0.5, bins=50)
p8 = ggplot(TBdata, aes(`Securities Account`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p9 =ggplot(TBdata, aes(`CreditCard`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p10 = ggplot(TBdata, aes(`CD Account`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
p11 = ggplot(TBdata, aes(`Online`, fill= `Personal Loan`)) + geom_bar(alpha=0.5)
grid.arrange(p1, p2, p3, p4,p5,p6,p7,p8,p9,p10,p11, ncol = 2, nrow = 6)
p12 = ggplot(TBdata, aes(`Income (in K/month)`,y=`Mortgage`, color= `Personal Loan`)) + geom_point(size=1)
p12
p14 = ggplot(TBdata, aes(`Income (in K/month)`, y=`CCAvg`, color= `Personal Loan`)) + geom_point(size=1)
p14

## Correlation between numeric continuous independant variables
matrix= cor(TBdata[,-c(4,6,8,9,10,11,12)] ,method = "pearson")
matrix
corrplot(matrix,type="upper",method = "number")
corrplot(matrix, method = "circle")

## Distance Calculation
Distchebyshev = dist(x=TBdata, method = "maximum")
Disteuc = dist(x=TBdata, method = "euclidean")
HCclusteuc = hclust(Disteuc, method = "complete")
HCclustch = hclust(Distchebyshev, method = "complete")
```

```
## Cluster Height, Sorting and Plotting
clusterheight = HCclusteuc$height
clusterheight = sort(clusterheight, decreasing = TRUE)
plot(clusterheight, pch =20, col="red", main="Cluster Height", ylab="Cluster Height")
lines(clusterheight, lty=2, lwd=2, col="blue")

## Cluster Plotting and Comparison
par(mfrow=c(2,1))
plot(HCclusteuc, labels = as.character(TBdata[,2]), main = "HClust using Euclidian method", xlab = "Euclidian
distance", ylab = "Height")rect.hclust(HCclusteuc, k=3, border = "red")
plot(HCclusteuc, labels = as.character(TBdata[,2]), main = "HClust using Chebychev method", xlab = "Chebychev
distance", ylab = "Height")rect.hclust(HCclusteuc, k=3, border = "red")


###### Clustering Using K Means
TBdata.clus = TBdata%>% select_if(is.numeric)
TBdata.scaled = scale(TBdata.clus, center = TRUE)
TBdata.dist = dist(TBdata.scaled, method = "euclidean")

## checking optimal number of clusters to categorize dataset
P20 = fviz_nbclust(TBdata.scaled, kmeans, method = "silhouette", k.max = 5)
p21 = fviz_nbclust(TBdata.scaled, kmeans, method = "wss", k.max = 5)
grid.arrange(p12, p21, ncol=2)
set.seed(1234)
TBdata.clusters = kmeans(TBdata.scaled, 3, nstart = 10)
fviz_cluster(TBdata.clusters, TBdata.scaled, geom = "point",
        ellipse = TRUE, pointsize = 0.2, ) + theme_minimal()

## Splitting the dataset into train and test for development and out of sample testing respectively
set.seed(100)
P_Loan_TRAIN_INDEX <- sample(1:nrow(TBdata),0.70*nrow(TBdata))
CARTtrain <- TBdata[P_Loan_TRAIN_INDEX,]
CARTtest <- TBdata[-P_Loan_TRAIN_INDEX,]

## Calculate the response rate
sum(TBdata$`Personal Loan` == "1")/nrow(TBdata)
sum(CARTtrain$`Personal Loan` == "1")/nrow(CARTtrain)
sum(CARTtest$`Personal Loan` == "1")/nrow(CARTtest)
table(CARTtrain$`Personal Loan`)
prop.table(table(CARTtrain$`Personal Loan`))
table(CARTtest$`Personal Loan`)
prop.table(table(CARTtest$`Personal Loan`))


## Check top 6 observation of train dataset
head(CARTtrain)

## CART Model
## Import rpart and rpart.plot library for creating CART model
#library(rpart)
#library(rpart.plot)
## Build first CART model
## Setting the control parameters for rpart
#minsplit: if the number of records in a given node falls below a threshold, the node will not be split further.
#minbucket: minimum records in a terminal node. if the records are less, that bucket will not be created.
#minsplit = 3(minbucket)
#cp = cost complexity parameter
#We begin by building a very complex classification tree, by setting the "cost complexity" threshold
#to "0" and the minimum bucket size to be 10. Then we plot the tree using rpart.
tree = rpart(formula = `Personal Loan` ~ .,data=CARTtrain,method="class",minbucket = 10,cp=0)

## Plot tree
tree
rpart.plot(tree)

## The cost complexity table can be obtained using the printcp or plotcp functions
## Print cp value
printcp(tree)

## Plot cp value
plotcp(tree)

#Pruning the tree
#The unncessarily complex tree above can be pruned using a cost complexity threshold.
#Using a complexity threshold of 0.003 gives us a much simpler tree.
ptree = prune(tree,cp=0.003,"CP")

## Print cp value
printcp(ptree)

## Check the updated tree
## Plot tree
ptree
rpart.plot(ptree)

## Visualize the CART tree
boxcols <- c("orange", "palegreen3")[ptree$frame$yval]

# Ptree$frame$yval
par(xpd=TRUE)
prp(ptree, faclen=0, cex=0.6, extra=1, box.col=boxcols, nn=TRUE, uniform=TRUE)

## Variable Importance
library(caret)
summary(ptree)
ptree$variable.importance
df_cart = data.frame(ptree$variable.importance)
```

```
df_cart

## Use this tree to do the prediction on train as well as test data set
CARTtrain$CART.Pred = predict(ptree,data=CARTtrain,type="class")
CARTtrain$CART.Score = predict(ptree,data=CARTtrain,type="prob")[,"1"]
CARTtest$CART.Pred = predict(ptree,CARTtest,type="class")
CARTtest$CART.Score = predict(ptree,CARTtest,type="prob")[,"1"]

##Confusion Metrix
#train dataset
CART_CM_train = confusionMatrix(CARTtrain$`Personal Loan`, CARTtrain$CART.Score,threshold = 0.7)
CART_CM_train

## Test dataset
CART_CM_test = confusionMatrix(CARTtest$`Personal Loan`, CARTtest$CART.Score, threshold = 0.7)
CART_CM_test

## Error Rate
#train
(CART_CM_train[1,2]+CART_CM_train[2,1])/nrow(CARTtrain)
#test
(CART_CM_test[1,2]+CART_CM_test[2,1])/nrow(CARTtest)

## Accuracy
#train
(CART_CM_train[1,1]+CART_CM_train[2,2])/nrow(CARTtrain)
#test
(CART_CM_test[1,1]+CART_CM_test[2,2])/nrow(CARTtest)

##Specificity
#train
(CART_CM_train[1,1])/(CART_CM_train[1,1]+CART_CM_train[2,1])
#test
(CART_CM_test[1,1])/(CART_CM_test[1,1]+CART_CM_test[2,1])

##Sensitivity
#train
(CART_CM_train[2,2])/(CART_CM_train[2,2]+CART_CM_train[1,2])
#test
(CART_CM_test[2,2])/(CART_CM_test[2,2]+CART_CM_test[1,2])

### ROCR and ineq packages to compute AUC, KS and gini
predobjtrain = prediction(CARTtrain$CART.Score,CARTtrain$`Personal Loan`)
## View(predobjtrain)
preftrain = performance(predobjtrain,"tpr","fpr")
plot(preftrain)

## View(preftrain)
predobjtest = prediction(CARTtest$CART.Score,CARTtest$`Personal Loan`)
```

```
preftest = performance(predobjtest,"tpr","fpr")
plot(preftest)

## KS
max(preftrain@y.values[[1]]-preftrain@x.values[[1]])
# preftrain@y.values[[1]]
# preftrain@x.values[[1]]
max(preftest@y.values[[1]]-preftest@x.values[[1]])

## AUC
auctrain=performance(predobjtrain,"auc")
as.numeric(auctrain@y.values)

auctest=performance(predobjtest,"auc")
as.numeric(auctest@y.values)

## Gini
ineq(CARTtrain$CART.Score,"gini")
ineq(CARTtest$CART.Score,"gini")

## Concordance
Concordance(actuals=CARTtrain$`Personal Loan`,predictedScores = CARTtrain$CART.Score)
Concordance(actuals=CARTtest$`Personal Loan`,predictedScores = CARTtest$CART.Score)

## Define Decile function for rank ordering
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
          ifelse(x<deciles[2], 2,
                ifelse(x<deciles[3], 3,
                      ifelse(x<deciles[4], 4,
                            ifelse(x<deciles[5], 5,
                                  ifelse(x<deciles[6], 6,
                                        ifelse(x<deciles[7], 7,
                                              ifelse(x<deciles[8], 8,
                                                    ifelse(x<deciles[9], 9, 10
                                                    ))))))))))
}

## Rank order table for train dataset
CARTtrain$deciles <- decile(CARTtrain$CART.Score)

## Ranking code
inter_datatable_train_cart = data.table(CARTtrain)
rank <- inter_datatable_train_cart[, list(
 cnt = length(as.integer(as.character(`Personal Loan`))),
 cnt_P_Loan = sum(as.integer(as.character(`Personal Loan`))==1),
 cnt_no_P_Loan = sum(as.integer(as.character(`Personal Loan`)) == 0)) ,
 by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_P_Loan / rank$cnt,4);
rank$cum_P_Loan <- cumsum(rank$cnt_P_Loan)
rank$cum_no_P_Loan <- cumsum(rank$cnt_no_P_Loan)
rank$cum_rel_P_Loan <- round(rank$cum_P_Loan / sum(rank$cnt_P_Loan),4);
rank$cum_rel_no_P_Loan <- round(rank$cum_no_P_Loan / sum(rank$cnt_no_P_Loan),4);
rank$rrate_perc <- percent(rank$rrate)
rank$cum_rel_P_Loan_perc <- percent(rank$cum_rel_P_Loan)
rank$cum_rel_no_P_Loan_perc <- percent(rank$cum_rel_no_P_Loan)
rank$cum_cnt<-cumsum(rank$cnt)
rank$cum_P_Loan_rate<-round(rank$cum_P_Loan / rank$cum_cnt,4)
overall_P_Loan_rate<-sum(as.integer(as.character(CARTtrain$`Personal Loan`)))/nrow(CARTtrain)
rank$ks = abs(rank$cum_rel_P_Loan - rank$cum_rel_no_P_Loan);

## Derive Lift
rank$lift<-round(rank$cum_P_Loan_rate/overall_P_Loan_rate,2)
View(rank)
write.csv(rank, "RankOrderingCM1.csv")

### Rank Order table for test dataset

CARTtest$deciles <- decile(CARTtest$CART.Score)

## Ranking code
inter_datatable_test_cart = data.table(CARTtest)
rank_test_cart <- inter_datatable_test_cart[, list(
 cnt = length(as.integer(as.character(`Personal Loan`))),
 cnt_P_Loan = sum(as.integer(as.character(`Personal Loan`))==1),
 cnt_no_P_Loan = sum(as.integer(as.character(`Personal Loan`)) == 0)) ,
 by=deciles][order(-deciles)]
rank_test_cart$rrate <- round(rank_test_cart$cnt_P_Loan / rank_test_cart$cnt,4);
rank_test_cart$cum_P_Loan <- cumsum(rank_test_cart$cnt_P_Loan)
rank_test_cart$cum_no_P_Loan <- cumsum(rank_test_cart$cnt_no_P_Loan)
rank_test_cart$cum_rel_P_Loan <- round(rank_test_cart$cum_P_Loan / sum(rank_test_cart$cnt_P_Loan),4);
rank_test_cart$cum_rel_no_P_Loan <- round(rank_test_cart$cum_no_P_Loan /
sum(rank_test_cart$cnt_no_P_Loan),4);
rank_test_cart$rrate_perc <- percent(rank_test_cart$rrate)
rank_test_cart$cum_rel_P_Loan_perc <- percent(rank_test_cart$cum_rel_P_Loan)
rank_test_cart$cum_rel_no_P_Loan_perc <- percent(rank_test_cart$cum_rel_no_P_Loan)
rank_test_cart$cum_cnt<-cumsum(rank_test_cart$cnt)
rank_test_cart$cum_P_Loan_rate<-round(rank_test_cart$cum_P_Loan / rank_test_cart$cum_cnt,4)
overall_P_Loan_rate_test<-sum(as.integer(as.character(CARTtest$`Personal Loan`)))/nrow(CARTtest)
rank_test_cart$ks = abs(rank_test_cart$cum_rel_P_Loan - rank_test_cart$cum_rel_no_P_Loan);

## Get Lift
rank_test_cart$lift<-round(rank_test_cart$cum_P_Loan_rate/overall_P_Loan_rate_test,2)
```

```
View(rank_test_cart)
write.csv(rank_test_cart, "RankOrderingCM2.csv")

## Lift graph for CART model
library(lift)
plotLift(CARTtrain$CART.Score,CARTtrain$`Personal Loan`,cumulative = FALSE,n.buckets = 10)
plotLift(CARTtest$CART.Score,CARTtest$`Personal Loan`,cumulative = FALSE,n.buckets = 10)


### KS Graph for CART Model
sample1<-rnorm(CARTtrain$CART.Pred)
sample2<-rnorm(CARTtest$CART.Pred)
group <- c(rep("sample1", length(sample1)), rep("sample2", length(sample2)))
dat <- TBdata.frame(KSD = c(sample1,sample2), group = group)
cdf1 <- ecdf(sample1)
cdf2 <- ecdf(sample2)
minMax <- seq(min(sample1, sample2), max(sample1, sample2), length.out=length(sample1))
x0 <- minMax[which( abs(cdf1(minMax) - cdf2(minMax)) == max(abs(cdf1(minMax) - cdf2(minMax))) )]
y0 <- cdf1(x0)
y1 <- cdf2(x0)
ggplot(dat, aes(x = KSD, group = group, colour = group, linetype=group))+
  stat_ecdf(size=1) +
  xlab("mm") +
  ylab("Cumulitive Distibution") +
  geom_segment(aes(x = x0[1], y = y0[1], xend = x0[1], yend = y1[1]),
               linetype = "dashed", color = "red") +
  geom_point(aes(x = x0[1] , y= y0[1]), color="red", size=1) +
  geom_point(aes(x = x0[1] , y= y1[1]), color="red", size=1) +
  ggtitle("K-S Test: Sample 1 / Sample 2")

## Buildig Random Forest model
names(TBdata)=c("Age","Experience","Income","FamilyMembers"
             ,"CCAvg","Education","Mortgage","SecuritiesAccount"
             ,"CDAccount","Online","CreditCard","PersonalLoan")

## Spliting the dataset into train and test for development and out of sample testing respectively
#dim(HRData)
set.seed(100)
P_Loan_TRAIN_INDEX <- sample(1:nrow(TBdata),0.70*nrow(TBdata))
RFtrain <- TBdata[P_Loan_TRAIN_INDEX,]
RFtest <- TBdata[-P_Loan_TRAIN_INDEX,]
RFtrain

## Import randomForest library for building random forest model
library(randomForest)

## Set a seed for the randomness
seed = 1000
set.seed(seed)
```

```
##Build the first RF model
#ntree: number of trees to grow
#mtry: number of variables to be considered for split
#nodesize: minimum size (number of records) of terminal nodes
#should importance of predictors be assessed
mtry = floor(sqrt(ncol(RFtrain)))
mtry
Rforest <- randomForest(PersonalLoan ~., data = RFtrain, ntree = 401, mtry = 3,
                    nodesize = 10, importance = TRUE)

## Print the model to see the OOB and error rate
print(Rforest)

## Plot the RF to know the optimum number of trees
Rforest$err.rate
plot(Rforest, main="Error Rates Random Forest")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)

#The error rate plot w.r.t number of trees reveals that anything more than, say 181, trees are
#really not that valuable.
##Identify the importance of the variables
importance(Rforest)

#Larger the MeanDecrease values, the more important the variable.Look at the help files to get
#a better sense of how these are computed.

## Tune up the RF model to find out the best mtry
# Now we will "tune" the Random Forest by trying different m values. We will stick with
# 201 trees (odd number of trees are preferable). The returned forest, "tRforest" is the
# one corresponding to the best m
# Starting with the default value of mtry, search for the optimal value (with respect to Out-of-Bag error
# estimate) of mtry for randomForest.
# Parameter Explanation
# x - Predictor variables
# y - Target Variable (factor for classification, numeric for regression)
# mtryStart - starting value of mtry; default is the same as in randomForest
# ntreeTry - number of trees used at the tuning step
# stepFactor - at each iteration, mtry is inflated (or deflated) by this value
# improve - the (relative) improvement in OOB error must be by this much for the search to continue
# trace - whether to print the progress of the search
# plot - whether to plot the OOB error as function of mtry
# doBest - whether to run a forest using the optimal mtry found
# If doBest=FALSE (default), it returns a matrix whose first column contains the mtry values searched, and the
second column the corresponding OOB error.
# If doBest=TRUE, it returns the randomForest object produced with the optimal mtry.
# nodesize - min terminal node size
# importance - compute variable importance or not
set.seed(seed)
```

```
tRforest = tuneRF(x=RFtrain[,-c(12)], y=RFtrain$PersonalLoan, ntreeTry = 181, mtryStart = 3,
            stepFactor = 1.5,
            improve = 0.0001, nodesize = 10, trace = TRUE, plot = TRUE,
            doBest = TRUE, importance = TRUE)
#names(RFtrain)
## Build the refined RF model
Rforest1 <- randomForest(PersonalLoan~., data = RFtrain, ntree = 181, mtry = 6, nodesize = 10,
               importance = TRUE)

## Use this tree to do the prediction on train as well as test data set
RFtrain$RF.Pred = predict(Rforest1,data=RFtrain,type="class")
RFtrain$RF.Score = predict(Rforest1,data=RFtrain,type="prob")[,"1"]
RFtest$RF.Pred = predict(Rforest1,RFtest,type="class")
RFtest$RF.Score = predict(Rforest1,RFtest,type="prob")[,"1"]

## Performance Measure Parameters
RF_CM_train = confusionMatrix(RFtrain$PersonalLoan, RFtrain$RF.Score,threshold = 0.7)
RF_CM_train
#test dataset
RF_CM_test = confusionMatrix(RFtest$PersonalLoan, RFtest$RF.Score, threshold = 0.7)
RF_CM_test

## Error Rate
(RF_CM_train[1,2]+RF_CM_train[2,1])/nrow(RFtrain)
(RF_CM_test[1,2]+RF_CM_test[2,1])/nrow(RFtest)

## Accuracy
(RF_CM_train[1,1]+RF_CM_train[2,2])/nrow(RFtrain)
(RF_CM_test[1,1]+RF_CM_test[2,2])/nrow(RFtest)

## Specificity ##
#train
(RF_CM_train[1,1])/(RF_CM_train[1,1]+RF_CM_train[2,1])
#test
(RF_CM_test[1,1])/(RF_CM_test[1,1]+RF_CM_test[2,1])

## Sensitivity ##
#train
(RF_CM_train[2,2])/(RF_CM_train[2,2]+RF_CM_train[1,2])
#test
(RF_CM_test[2,2])/(RF_CM_test[2,2]+RF_CM_test[1,2])

## Probablity Related Parameters like KS, ROC, AUC, Concordance, Discordance and Gini
predobjtrain = prediction(RFtrain$RF.Score,RFtrain$PersonalLoan)
preftrain = performance(predobjtrain,"tpr","fpr")
plot(preftrain)
predobjtest = prediction(RFtest$RF.Score,RFtest$PersonalLoan)
preftest = performance(predobjtest,"tpr","fpr")
plot(preftest)
```

```
## KS
max(preftrain@y.values[[1]]-preftrain@x.values[[1]])
max(preftest@y.values[[1]]-preftest@x.values[[1]])

## AUC
auctrain=performance(predobjtrain,"auc")
as.numeric(auctrain@y.values)
auctest=performance(predobjtest,"auc")
as.numeric(auctest@y.values)

## Gini
ineq(RFtrain$RF.Score,"gini")
ineq(RFtest$RF.Score,"gini")

## Concordance
Concordance(actuals=RFtrain$PersonalLoan,predictedScores = RFtrain$RF.Score)
Concordance(actuals=RFtest$PersonalLoan,predictedScores = RFtest$RF.Score)

## Rank Order Table
RFtrain$deciles <- decile(RFtrain$RF.Score)

## Ranking Code
inter_datatable_train_RF = data.table(RFtrain)
rank_FM <- inter_datatable_train_RF[, list(
  cnt = length(as.integer(as.character(PersonalLoan))),
  cnt_P_L = sum(as.integer(as.character(PersonalLoan))==1),
  cnt_no_P_L = sum(as.integer(as.character(PersonalLoan)) == 0)) ,
  by=deciles][order(-deciles)]
rank_FM$rrate <- round(rank_FM$cnt_P_L / rank_FM$cnt,4);
rank_FM$cum_P_L <- cumsum(rank_FM$cnt_P_L)
rank_FM$cum_no_P_L <- cumsum(rank_FM$cnt_no_P_L)
rank_FM$cum_rel_P_L <- round(rank_FM$cum_P_L / sum(rank_FM$cnt_P_L),4);
rank_FM$cum_rel_no_P_L <- round(rank_FM$cum_no_P_L / sum(rank_FM$cnt_no_P_L),4);
rank_FM$rrate_perc <- percent(rank_FM$rrate)
rank_FM$cum_rel_P_L_perc <- percent(rank_FM$cum_rel_P_L)
rank_FM$cum_rel_no_P_L_perc <- percent(rank_FM$cum_rel_no_P_L)
rank_FM$cum_cnt<-cumsum(rank_FM$cnt)
rank_FM$cum_P_L_rate<-round(rank_FM$cum_P_L / rank_FM$cum_cnt,4)
overall_P_L_rate<-sum(as.integer(as.character(RFtrain$PersonalLoan)))/nrow(RFtrain)
rank_FM$ks = abs(rank_FM$cum_rel_P_L - rank_FM$cum_rel_no_P_L);

# Derive Lift
rank_FM$lift<-round(rank_FM$cum_P_L_rate/overall_P_L_rate,2)
View(rank_FM)
write.csv(rank_FM, "RankOrdering_RF.1.csv")
RFtest$deciles <- decile(RFtest$RF.Score)

## Ranking Code
```

```
inter_datatable_test_RF = data.table(RFtest)
rank_test_RF <- inter_datatable_test_RF[, list(
  cnt = length(as.integer(as.character(PersonalLoan))),
  cnt_P_L = sum(as.integer(as.character(PersonalLoan))),
  cnt_no_P_L = sum(as.integer(as.character(PersonalLoan)) == 0)) ,
  by=deciles][order(-deciles)]
rank_test_RF$rrate <- round(rank_test_RF$cnt_P_L / rank_test_RF$cnt,4);
rank_test_RF$cum_P_L <- cumsum(rank_test_RF$cnt_P_L)
rank_test_RF$cum_no_P_L <- cumsum(rank_test_RF$cnt_no_P_L)
rank_test_RF$cum_rel_P_L <- round(rank_test_RF$cum_P_L / sum(rank_test_RF$cnt_P_L),4);
rank_test_RF$cum_rel_no_P_L <- round(rank_test_RF$cum_no_P_L / sum(rank_test_RF$cnt_no_P_L),4);
rank_test_RF$rrate_perc <- percent(rank_test_RF$rrate)
rank_test_RF$cum_rel_P_L_perc <- percent(rank_test_RF$cum_rel_P_L)
rank_test_RF$cum_rel_no_P_L_perc <- percent(rank_test_RF$cum_rel_no_P_L)
rank_test_RF$cum_cnt<-cumsum(rank_test_RF$cnt)
rank_test_RF$cum_P_L_rate<-round(rank_test_RF$cum_P_L / rank_test_RF$cum_cnt,4)
overall_P_L_rate_test<-sum(as.integer(as.character(RFtest$PersonalLoan)))/nrow(RFtest)
rank_test_RF$ks = abs(rank_test_RF$cum_rel_P_L - rank_test_RF$cum_rel_no_P_L);

# Get Lift
rank_test_RF$lift<-round(rank_test_RF$cum_P_L_rate/overall_P_L_rate_test,2)
View(rank_test_RF)
write.csv(rank_test_RF, "RankOrdering_RF2.csv")

## Lift Graph for Rf Model
library(lift)
plotLift(RFtrain$RF.Score,RFtrain$PersonalLoan,cumulative = FALSE,n.buckets = 10)
plotLift(RFtest$RF.Score,RFtest$PersonalLoan,cumulative = FALSE,n.buckets = 10)

### KS Graph for RF Model
sample3<-rnorm(RFtrain$RF.Pred)
sample4<-rnorm(RFtest$RF.Pred)
group <- c(rep("sample3", length(sample3)), rep("sample4", length(sample4)))
dat <- data.frame(KSD = c(sample3,sample4), group = group)
cdf1 <- ecdf(sample3)
cdf2 <- ecdf(sample4)
minMax <- seq(min(sample3, sample4), max(sample3, sample4), length.out=length(sample3))
x0 <- minMax[which( abs(cdf1(minMax) - cdf2(minMax)) == max(abs(cdf1(minMax) - cdf2(minMax))) )]
y0 <- cdf1(x0)
y1 <- cdf2(x0)
ggplot(dat, aes(x = KSD, group = group, colour = group, linetype=group))+
  stat_ecdf(size=1) +
  xlab("mm") +
  ylab("Cumulitive Distibution") +
  geom_segment(aes(x = x0[1], y = y0[1], xend = x0[1], yend = y1[1]),
            linetype = "dashed", color = "red") +
  geom_point(aes(x = x0[1] , y= y0[1]), color="red", size=1) +
  geom_point(aes(x = x0[1] , y= y1[1]), color="red", size=1) +
  ggtitle("K-S Test: Sample 3/ Sample 4")
```