Thera Bank - Loan Purchase Modeling

Submission Date: Nov 24, 2019

Author: Ayush Jain
Mentor: Deepak Gupta

Table of Content

Table of Contents

1 Project Objective

The objective of this report is to explore the Bank Data “TheraBank.xlsx” and investigate the data to understand the data and to build the best model which can classify the right customers who have a higher probability of purchasing the loan.

The data file consists of 14 variables and 5000 observations and is represented as below:

ID	Customer ID
Age	Customer's age in years
Experience	Years of professional experience
Income	Annual income of the customer ($000)
ZIPCode	Home Address ZIP code.
Family	Family size of the customer
CCAvg	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?
CreditCard	Does the customer use a credit card issued by the bank?

We will further be performing Data Manipulation and Data Cleaning steps to make sure that the data is Accurate and Model Ready

During this Project we will be performing the below Steps on the data.

- Performing Exploratory data analysis on the dataset to visualize and understand the data and identify the outliers and missing values
- Building CART and Random Forest Model on the Dataset and Identifying the best fit model by Validating the Model using various Techniques.
- Checking the performance of the models built using the different Model Performance techniques.

2 Exploratory Data Analysis – Step by step approach

Exploratory Data Analysis is one of the important phases in the data Analysis in understanding the significance and accuracy of the data. It usually consists of setting up the environment to work in R, loading the data and checking the validity of data loaded.

A Typical Data exploration activity consists of the following steps:

- Environment Set up and Data Import.
    - Install Necessary Package in R.
    - Setting Up Working Directory.

- Reading Dataset in R.
- Performing Data Cleaning.

- Variable Identification.

We shall follow these steps in exploring the provided dataset.

2.1 Environment Set up and Data Import

2.1.1 Install necessary Packages.

In this section, we will install and invoke the necessary Packages and Libraries that are going to be the part of our work throughout the project. Having all the packages at the same places increases code readability and Understandability.

```
# Installing and Deploying necessary Packages

install.packages("ROCR")
install.packages("rpart.plot")
install.packages("readxl")
install.packages("randomForest")
install.packages("data.table")
install.packages("ineq")
install.packages("InformationValue")
install.packages("caret")

library(readxl)
library(DataExplorer)
library(corrplot)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(data.table)
library(ROCR)
library(ineq)
library(InformationValue)
library(caret)
```

2.1.2 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. This helps maintain the code readability and avoid unwanted errors.

```
# Setting the Working Directory.
setwd("D:/Great Learning/Project 3")
```

Please refer Appendix A for Source Code.

2.1.3 Import and Read the Dataset

The given dataset is in .xlsx format. Hence, the command ‘read.xslx’ from readxl package is used for importing the file.

```
# Reading the Dataset
theraData <- read_xlsx("TheraBank.xlsx", sheet = 2)
```

Please refer Appendix A for Source Code.

2.1.4 Data Cleaning

Once the Data is imported in R, we will perform the basic operation to understand the viability of the data and check the Accuracy.

- Checking the top six rows of the Data.

```
head(theraData)

## # A tibble: 6 x 14
##      ID `Age (in years)` `Experience (in~ `Income (in K/m~ `ZIP Code`
##   <dbl>            <dbl>            <dbl>            <dbl>      <dbl>
## 1     1               25                1               49      91107
## 2     2               45               19               34      90089
## 3     3               39               15               11      94720
## 4     4               35                9              100      94112
## 5     5               35                8               45      91330
## 6     6               37               13               29      92121
## # ... with 9 more variables: `Family members` <dbl>, CCAvg <dbl>,
## #   Education <dbl>, Mortgage <dbl>, `Personal Loan` <dbl>, `Securities
## #   Account` <dbl>, `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>
```

- Checking for the Extra Variables that can be removed.

```
names(theraData)

##  [1] "ID"                      "Age (in years)"
##  [3] "Experience (in years)" "Income (in K/month)"
##  [5] "ZIP Code"                "Family members"
##  [7] "CCAvg"                   "Education"
##  [9] "Mortgage"                "Personal Loan"
## [11] "Securities Account"      "CD Account"
## [13] "Online"                  "CreditCard"
```

- Removing the ID Column as it is a continuous variable and will not impact the Model.

```
theraData<- theraData[,-1]
```

- Checking the Structure of the Dataset.

```
str(theraData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  13 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 ...
##  $ ZIP Code              : num  91107 90089 94720 94112 91330 ...
##  $ 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 ...
```

- Performing Summary operation to gain better understanding on Data.

```
summary(theraData)

##  Age (in years)  Experience (in years) Income (in K/month)    ZIP Code
##  Min.   :23.00   Min.   :-3.0          Min.   :  8.00      Min.   : 9307
##  1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00      1st Qu.:91911
##  Median :45.00   Median :20.0          Median : 64.00      Median :93437
##  Mean   :45.34   Mean   :20.1          Mean   : 73.77      Mean   :93153
##  3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00      3rd Qu.:94608
##  Max.   :67.00   Max.   :43.0          Max.   :224.00      Max.   :96651
##
##  Family members      CCAvg          Education        Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.397   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  NA's   :18
##  Personal Loan   Securities Account   CD Account          Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
##
```

From Summary, we identified that we have Null Values and the Negative values in our Data which needs to be handled.
We will be dealing with these later in this section.

- Renaming the Variable Names.

```
# Changing the Column names for the dataset

names(theraData) <- make.names(c("Age (in years)","Experience (in
years)","Income (in K/month)","ZIP Code","Family members","CCAvg",
             "Education","Mortgage","Personal Loan","Securities Account","CD
Account","Online","CreditCard"),allow_ = TRUE, unique = FALSE)
```

- Handling Negative Values in the Dataset.

```r
# Removing negative records from the Dataset.
theraData <- subset(theraData, theraData$`Experience (in years)` >= 0)
```

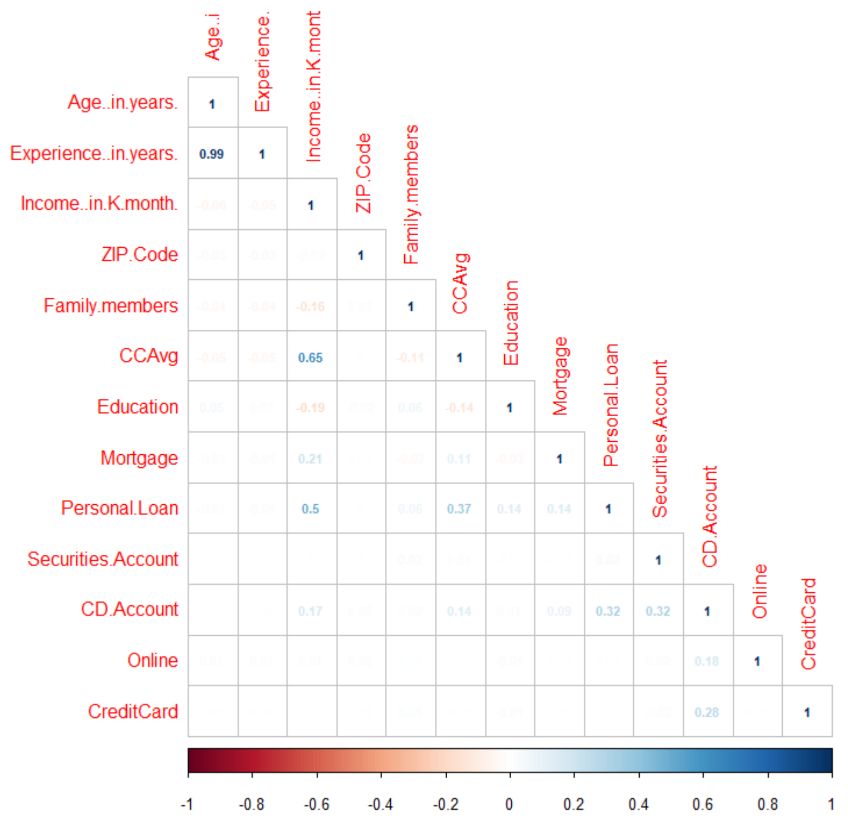
- Removing Null Values in the Dataset.

```r
# Checking for the Null Values.


sum(is.na(theraData))

## [1] 18

colSums(is.na(theraData))

##        Age..in.years. Experience..in.years.   Income..in.K.month.
##                     0                     0                     0
##              ZIP.Code        Family.members                 CCAvg
##                     0                    18                     0
##             Education              Mortgage         Personal.Loan
##                     0                     0                     0
##    Securities.Account            CD.Account                Online
##                     0                     0                     0
##            CreditCard
##                     0
```

```r
# Removing Null Values.

theraData<- na.omit(theraData)
colSums(is.na(theraData))

##        Age..in.years. Experience..in.years.   Income..in.K.month.
##                     0                     0                     0
##              ZIP.Code        Family.members                 CCAvg
##                     0                     0                     0
##             Education              Mortgage         Personal.Loan
##                     0                     0                     0
##    Securities.Account            CD.Account                Online
##                     0                     0                     0
##            CreditCard
##                     0
```

- Checking for the Correlation between the variables and Plotting the Correlation Plot.

```r
# Checking for the Correlation between the Varibles.

matrix <- cor(theraData)
corrplot(matrix, method = "number", type = "lower",number.cex = 0.5)
```



There exist a huge Correlation of 99% between “Age..in.years.” and Experience..in.years.”, hence we will remove one of the Variables from the dataset. Removing “Age..in.years”

```r
# Removing Experience as there exist a huge Correlation between Age and Experience
theraData<- theraData[,-1]
```

- Converting the Dependent Variable and variables that have values of 0 and 1 to Factors.

```r
# Converting the variables to Factors

theraData$Personal.Loan <- as.factor(theraData$Personal.Loan)
theraData$Securities.Account <- as.factor(theraData$Securities.Account)
theraData$CD.Account <- as.factor(theraData$CD.Account)
theraData$Online <- as.factor(theraData$Online)
theraData$CreditCard <- as.factor(theraData$CreditCard)
```

- Checking the Structure of the Dataset again after performing the basic operations.

```
str(theraData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    4930 obs. of  12 variables:
##  $ 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 ...
##  $ ZIP.Code             : num  91107 90089 94720 94112 91330 ...
##  $ 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        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2
...
##  $ Securities.Account   : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1
...
##  $ CD.Account           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ Online               : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1
...
##  $ CreditCard           : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1
...
##  - attr(*, "na.action")=Class 'omit'  Named int [1:18] 21 59 98 161 234
288 484 709 1443 1444 ...
##   .. ..- attr(*, "names")= chr [1:18] "21" "59" "98" "161" ...

nrow(theraData)

## [1] 4930
```

We have removed ID and Age..in.years. from the Dataset and Converted the below variables to Factors.

- Personal.Loan
- Securities.Account
- CD.Account
- Online
- CreditCard

•

2.2 Variable Identification

This section holds the Variables/ Methods that are used during the Analysis of the problem. Below are the Functions that we have used for the Analysis.

- setwd(): setwd(dir) is used to set the working directory to dir.
- read.xlsx(): Reads a file in table format and creates a data frame from it.
- head(): Returns the first parts of a vector, matrix, table, data frame or function.
- str(): Compactly display the internal Structure of an R object.

- summary(): summary is a generic function used to produce result summaries of the results of various model fitting functions.
- names(): Functions to get or set the names of an object.
- Make.names():Make syntactically valid names out of character vectors.
- Sum(): sum returns the sum of all the values present in its arguments.
- Colsums(): Form row and column sums and means for numeric arrays.
-
- Is.null(): NULL is often returned by expressions and functions whose value is undefined. is.null returns TRUE if its argument's value is NULL and FALSE otherwise.
- Boxplot(): It is plotting technique, which is used to identify if there any outliners are present in the data.
- Plot_histogram():Plot histogram for each continuous feature.
- cor(): cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices then the covariances (ororrelations) between the columns of x and the columns of y are computed.
- Corrplot(): This is used to plot the correlation matrix for better visualization and presentation.
- cbind(): This method is used to join variables on the basis of the columns.
- set.seeds(): set.seed is the recommended way to specify seeds.
- Sample.split(): Split data from vector Y into two sets in predefined ratio while preserving relative ratios of different labels in Y. Used to split the data used during classification into train and test subsets.
- Subset(): This method is used to subset the data.
- Rpart(): Fit a rpart model, used to Create a CART model.
- Prp(): Plot an rpart model.
- Rpart.plot(): Plot an rpart model, automatically tailoring the plot for the model's response type
- Predict(): predict is a generic function for predictions from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.
- Table(): table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.
- Prediction(): Every classifier evaluation using ROCR starts with creating a prediction object. This function is used to transform the input data (which can be in vector, matrix, data frame, or list form) into a standardized format.
- Performance():All kinds of predictor evaluations are performed using this function.
- Ineq(): computes the inequality within a vector according to the specified inequality measure. Used to Calculate Gini Gain for the Model.
- Concordance():computes the inequality within a vector according to the specified inequality measure.
- confusionMatrix():Calculate the confusion matrix for the fitted values for a logistic regression model.
- randomForest():randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.
- tuneRF(): Starting with the default value of mtry, search for the optimal value of mtry for randomForest.

3 Univariate Analysis

Univariate analysis is perhaps the simplest form of statistical analysis. Like other forms of statistics, it can be inferential or descriptive. The key fact is that only one variable is involved.

For Numeric variables, default plot is histogram and boxplot while for Categorical variables it is Bar plot.

Histogram: A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.

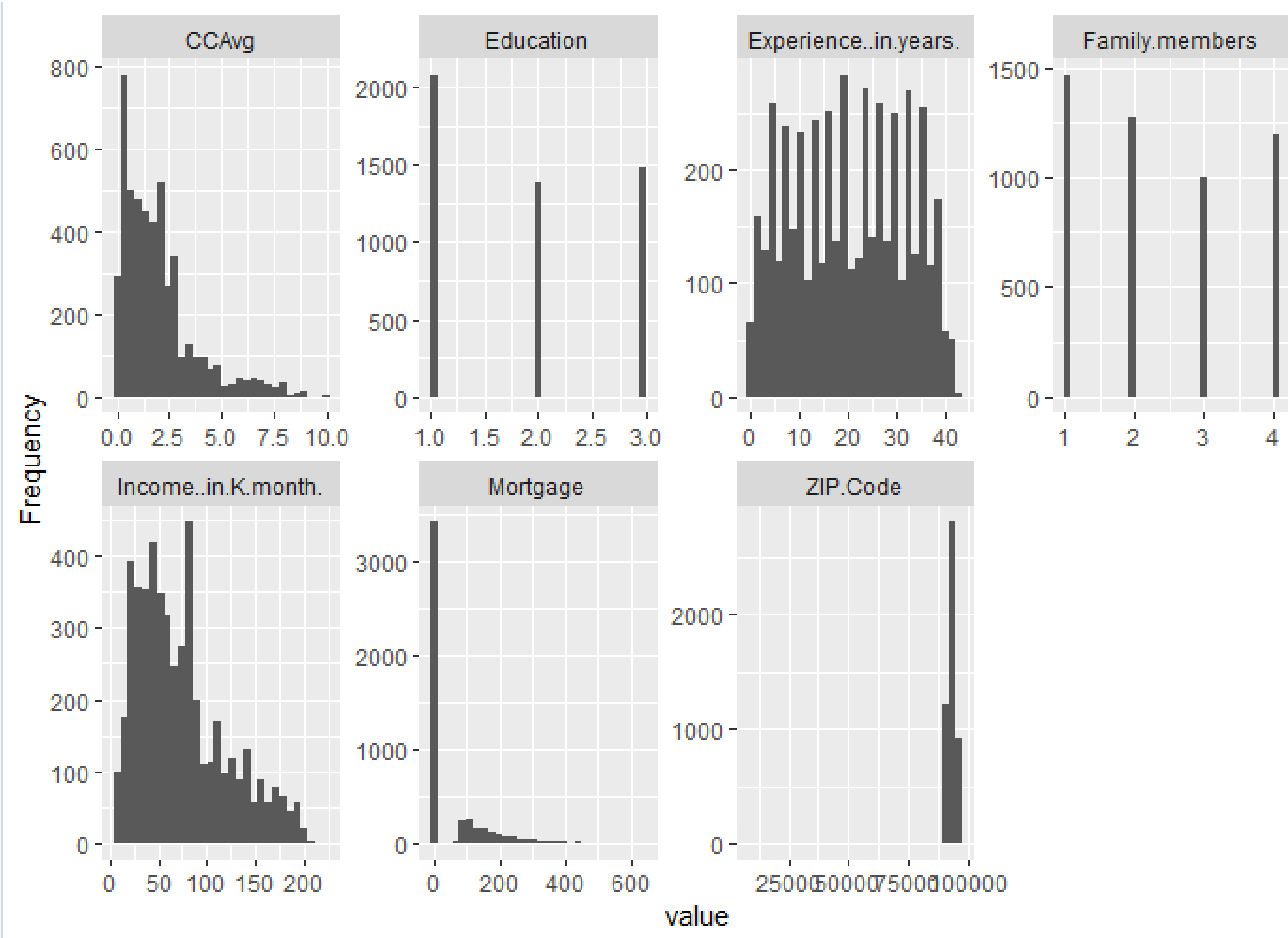
Boxplot: A box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles. Outliers may be plotted as individual points.

In the problem given, we will be using the above two plotting functions to perform the Univariate analysis on the dataset and identify any outliners present in the data.

Plotting the histogram for all the numeric variables in the dataset.

To analyze each variables, we plot the histogram for the variables.

```
# Performing Univariate Analysis.
plot_histogram(theraData)
```



Plotting the Boxplot to identify the Outliers in the data.

We use Boxplot to check if there are any Outliers available in the data, boxplot identify the outliers basis the below formulation.

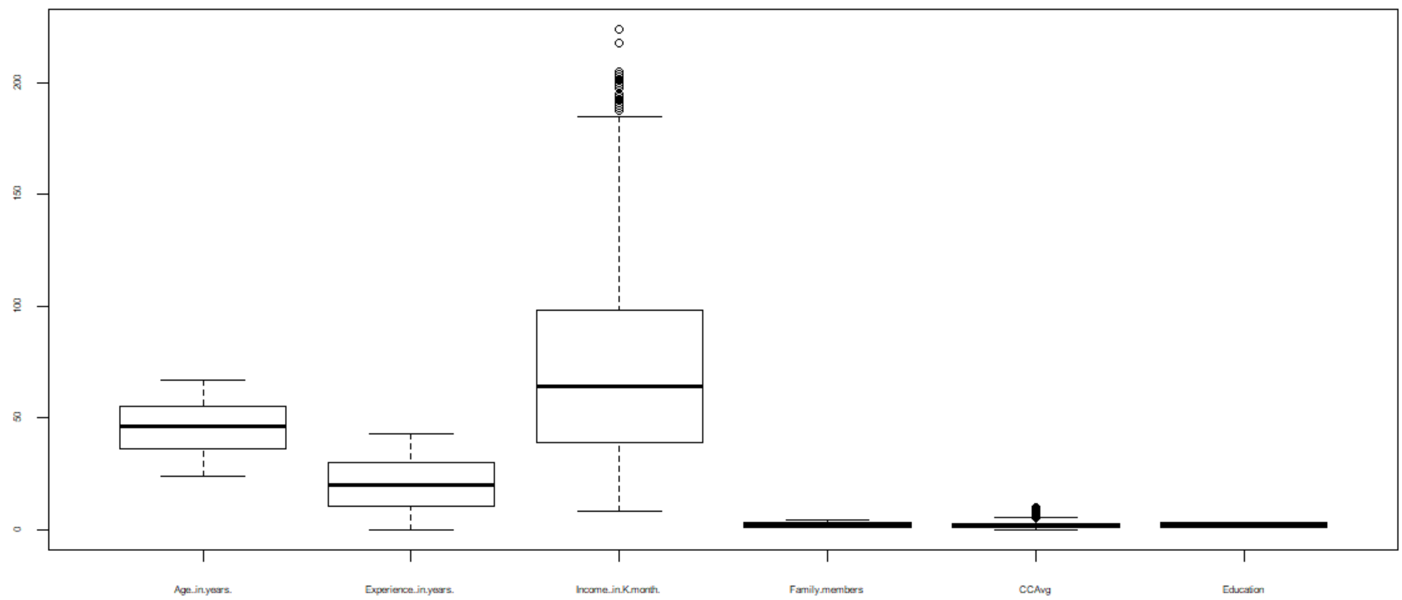
IQR = Q3 -Q1

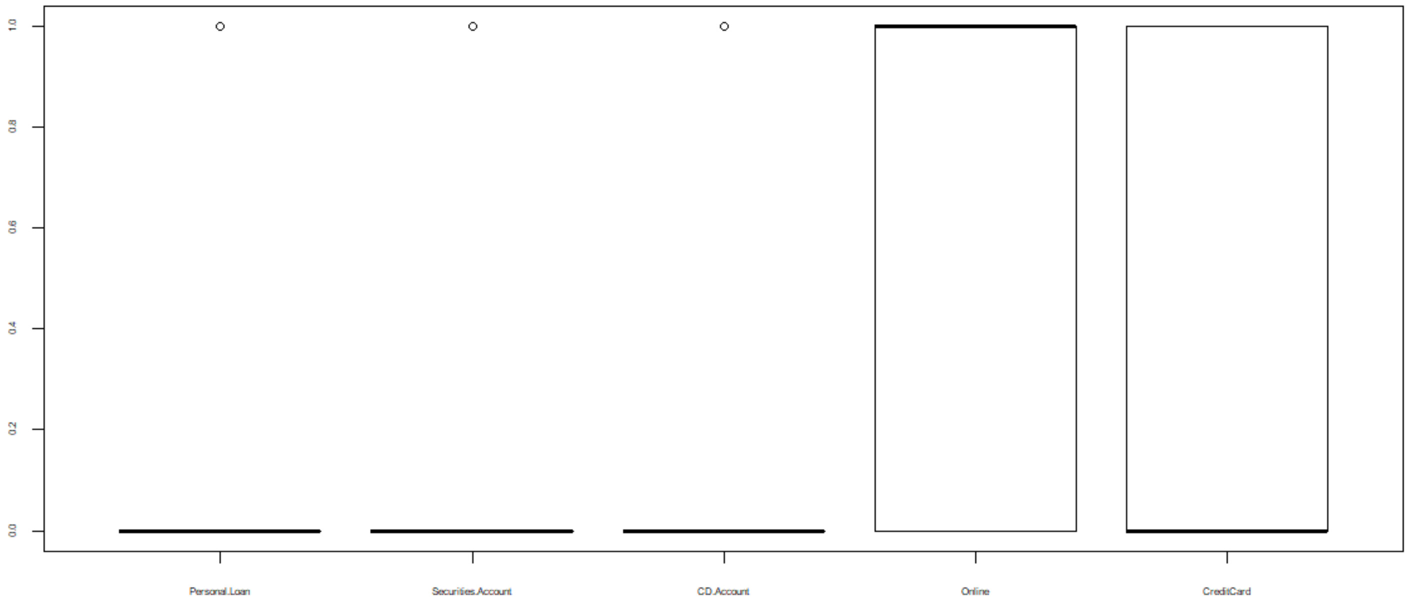
Lower Limit = Q1 – 1.5(IQR)

Upper Limit = Q3 + 1.5(IQR)

Points outside the upper and Lower limits are Outliers.

```
par(mfrow = c(1,1))
boxplot(theraData[,c(1,2,3,5,6,7)],cex.axis = 0.5, horizontal = TRUE)
boxplot(theraData[,c(8,9,10,11,12,13)],cex.axis = 0.5, horizontal = TRUE)
```

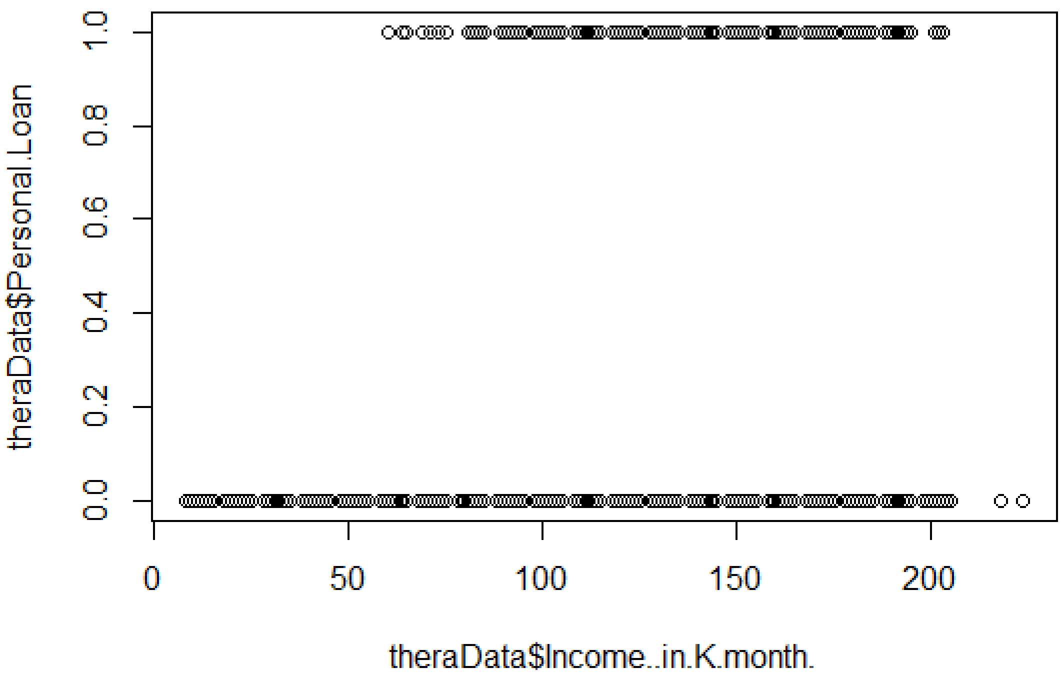




4 Bi-Variate Analysis

Multivariate analysis is a set of techniques used for analysis of data sets that contain more than one variable, and the techniques are especially valuable when working with correlated variables. The techniques provide an empirical method for information extraction, regression, or classification.

For Multivariate analysis, the default plot is the Scatter Plot. We will be plotting the correlation between the different variables with Personal Loan to understand the relation between the dependent variable Personal Loan with the Independent variables.



5 Conclusion

Proceeding on the dataset, we will further be splitting the Dataset into Test and Train, which will be used to Build and Validate the Model. The Model will be built on Train Dataset and will further be Validated using the Train Dataset. The Train Dataset will contain 70% of the data and Test will have 30%

- Splitting the Data into Train and Test

```
# Splitting data into Train and Test with a split of 70, 30 respectively.
set.seed(1000)
index <- sample.split(theraData$Personal.Loan,SplitRatio = 0.7)

Train_Cart <- subset(theraData, index ==TRUE)
Test_Cart <- subset(theraData, index ==F)
```

- Building the CART Model on Train Dataset.

```r
# Building CART Model on Train Data

Model_Train_Cart <- rpart(Personal.Loan~.,data = Train_Cart,method = "class")
Model_Train_Cart

## n= 3451
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3451 335 0 (0.902926688 0.097073312)
##    2) Income..in.K.month.< 113.5 2773  56 0 (0.979805265 0.020194735)
##      4) CCAvg< 2.95 2572  10 0 (0.996111975 0.003888025) *
##      5) CCAvg>=2.95 201  46 0 (0.771144279 0.228855721)
##       10) CD.Account=0 183  31 0 (0.830601093 0.169398907) *
##       11) CD.Account=1 18   3 1 (0.166666667 0.833333333) *
##    3) Income..in.K.month.>=113.5 678 279 0 (0.588495575 0.411504425)
##      6) Education< 1.5 433  45 0 (0.896073903 0.103926097)
##       12) Family.members< 2.5 388   0 0 (1.000000000 0.000000000) *
##       13) Family.members>=2.5 45   0 1 (0.000000000 1.000000000) *
##      7) Education>=1.5 245  11 1 (0.044897959 0.955102041) *
```

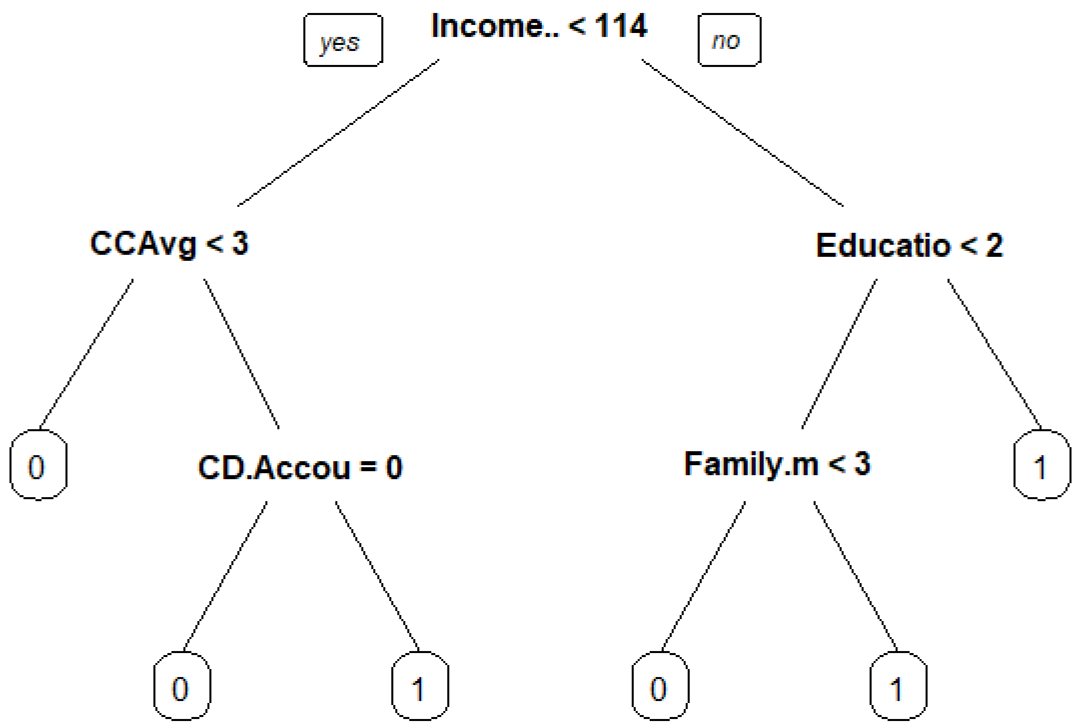
In the Model, Root node have 3451 observations which has 335 observations i.e 9.7% as Ones and rest 90.29% as Zeros.
The first split is on Income less then 113.5 having 2773 total observations and 56 as Ones and rest as Zeros and Income greater then equals 113.5 with 678 as the total observations and contains 279 Ones and rest Zeros.
The next spilt continues on Income < 113.5 and will be splitted on CCAvg >= 2.95 with 201 as the Total number of Observations and 46 Ones and rest as Zeros and so on.

The leaf nodes are built on the below.

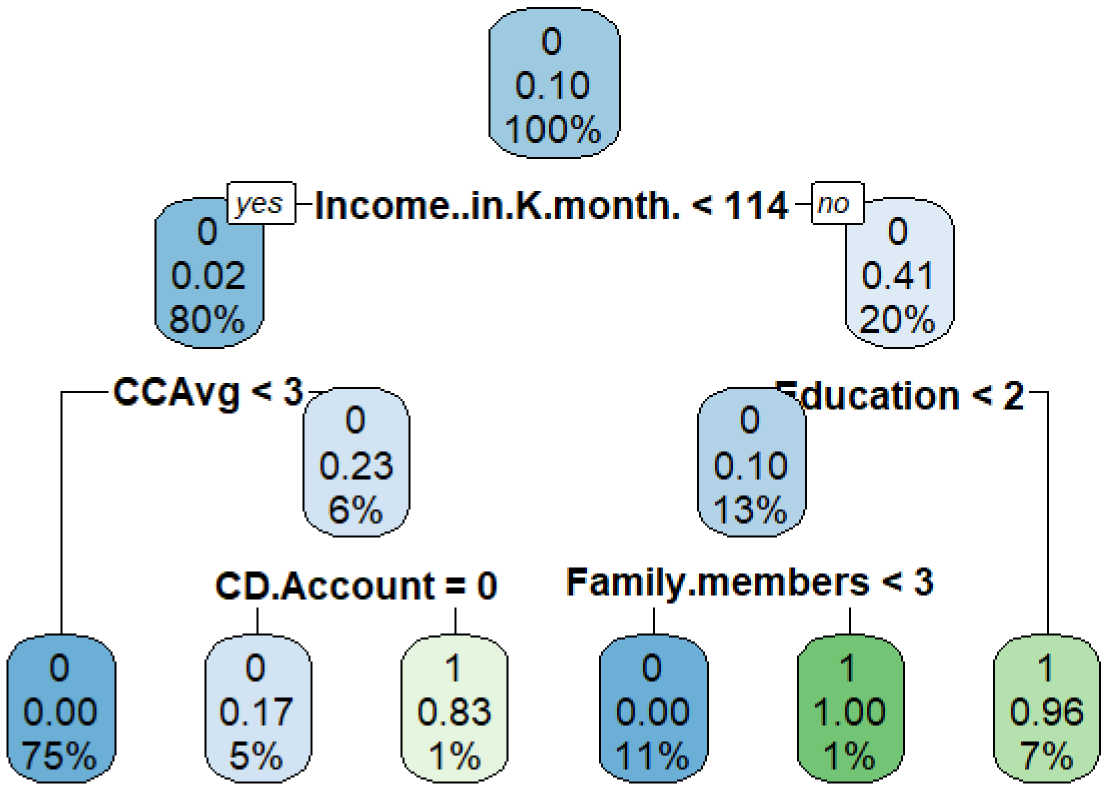
- CCAvg < 2.95 with 2572 as the total observations having 10 Ones and rest as Zeros.
- CD.Account =0 with 183 as the total observations having 31 Ones and rest as Zeros whereas CD.Account =1 with 18 as the total observations having 3 Zeros and rest as Ones
- Family.members < 2.5 with 388 as the total observations having all Zeros whereas Family.members > 2.5 with 45 as the total observations having all Ones.
- Education >= 1.5 with 245 as the total observations having 11 Ones and rest Zeros.

- Plotting the CART Model.

```r
prp(Model_Train_Cart)
```




```
rpart.plot(Model_Train_Cart,tweak = 1.2)
```



- Pruning of the Tree.

```
Model_Train_Cart$cptable

##           CP nsplit rel error    xerror       xstd
## 1 0.33283582      0 1.0000000 1.0000000 0.05191631
## 2 0.13432836      2 0.3343284 0.4089552 0.03423886
## 3 0.01791045      3 0.2000000 0.2567164 0.02733534
## 4 0.01000000      5 0.1641791 0.1970149 0.02401784
```

Since we have received the minimum xerror at the end, this the optimized tree and there is no need for further Pruning.

- Predicting the Values and Probability of gaining One for Train Dataset.

```
########## Predicting the values on Train.

pred_train_cart <- predict(Model_Train_Cart,newdata = Train_Cart, type =
"class")
Train_Cart<- cbind(Train_Cart,pred_train_cart)

# Predicting the probability on Train Data

Train_Cart$probs <- predict(Model_Train_Cart, Train_Cart, type = "prob")[,2]

tbl <- table(Train_Cart$Personal.Loan,Train_Cart$pred_train)
tbl

##
##        0    1
##   0 3102   14
##   1   41  294

print((tbl[1,2]+tbl[2,1])/nrow(Train_Cart))

## [1] 0.01593741
```

We have predicted the values for the Train Data in which we predicted 3102 Zeros and 294 Ones correctly. We got the error rate of 0.015 and the accuracy of (1 - 0.015) which is 0.985.

- Creating the Confusion Matrix and calculating Sensitivity and Specificity on Train Data.

```
## Create Confusion matrix on the above prediction

caret::confusionMatrix(Train_Cart$pred_train_cart,Train_Cart$Personal.Loan)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3102   41
         1   14  294

               Accuracy : 0.9841
                 95% CI : (0.9793, 0.988)
    No Information Rate : 0.9029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9057

 Mcnemar's Test P-Value : 0.0004552

            Sensitivity : 0.9955
            Specificity : 0.8776
         Pos Pred Value : 0.9870
         Neg Pred Value : 0.9545
             Prevalence : 0.9029
         Detection Rate : 0.8989
   Detection Prevalence : 0.9108
      Balanced Accuracy : 0.9366

       'Positive' Class : 0
```

From the above Confusion Matrix, we have achieved the Accuracy of 98.41%, Specificity of 87.76% and Sensitivity of 99.55%

- Preparing the Rank Table

To create the Rank Table, we first decile the data into groups based on Probability.

```
# Preparing the Rank Table on the Train Data.

prob <- seq (0,1, length = 11)
prob

##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

qs_train<- quantile(Train_Cart$probs,prob)
qs_test <- quantile(Test_Cart$probs_rf)

## Warning: Unknown or uninitialised column: 'probs_rf'.

Train_Cart$Decile <- cut(Train_Cart$probs,unique(qs_train), include.lowest =
TRUE, right = FALSE)

table(Train_Cart$Decile)

##
##     [0,0.00389) [0.00389,0.169)       [0.169,1]
##             388            2572             491
```

The data is deciled into 3 groups. Once the Deciles are created, we start calculating the below parameters from the data set

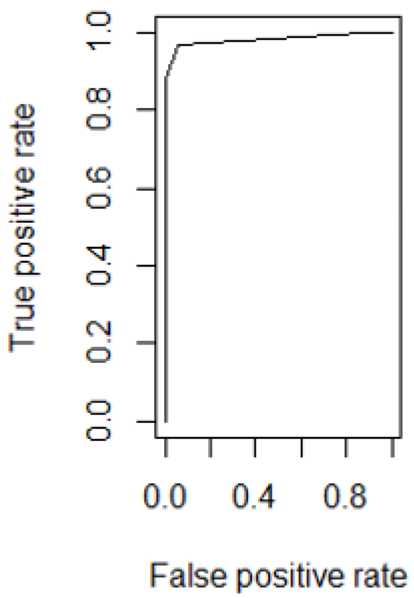
- Count
- Count of Ones
- Count of Zeros
- Response Rate
- Cumulative Response Rate
- Cumulative Non-Response Rate
- Cumulative Relative Response Rate
- Cumulative Relative Non-Response Rate
- KS Value

```
TrainDT_CART<- data.table(Train_Cart)

TrainRankTbl_CART <- TrainDT_CART[,list(
  cnt = length(Personal.Loan),
  cnt_tar1 <- sum(Personal.Loan==1),
  cnt_tar0 <- sum(Personal.Loan==0)
), by = Decile][order(-Decile)]

names(TrainRankTbl_CART) <- c("Decile","Count","Count_One","Count_Zero")
names(TrainRankTbl_CART)

## [1] "Decile"     "Count"      "Count_One"  "Count_Zero"

TrainRankTbl_CART$rrate <-
round(TrainRankTbl_CART$Count_One/TrainRankTbl_CART$Count,4)*100
TrainRankTbl_CART$cum_res<- cumsum(TrainRankTbl_CART$Count_One)
TrainRankTbl_CART$cum_non_res <- cumsum(TrainRankTbl_CART$Count_Zero)
TrainRankTbl_CART$cum_rel_res <-
round(TrainRankTbl_CART$cum_res/sum(TrainRankTbl_CART$Count_One),4)*100
TrainRankTbl_CART$cum_rel_non_res <-
round(TrainRankTbl_CART$cum_non_res/sum(TrainRankTbl_CART$Count_Zero),4)*100
TrainRankTbl_CART$ks <- abs(TrainRankTbl_CART$cum_rel_res -
TrainRankTbl_CART$cum_rel_non_res)

TrainRankTbl_CART

##                Decile Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:        [0.169,1]   491       325        166 66.19     325         166
## 2: [0.00389,0.169)  2572        10       2562  0.39     335        2728
## 3:     [0,0.00389)   388         0        388  0.00     335        3116
##    cum_rel_res cum_rel_non_res    ks
## 1:       97.01            5.33 91.68
## 2:      100.00           87.55 12.45
## 3:      100.00          100.00  0.00
```

KS Value calculated is 91.68

- Plotting ROC Curve.

```
# Plotting the ROC Curve.

predobj_train_cart <- prediction(Train_Cart$probs,Train_Cart$Personal.Loan)
perf_train_cart <- performance(predobj_train_cart,"tpr","fpr")
plot(perf_train_cart)
```



- Calculating the KS Values from ROC Curve.

```
# Calculating the KS value from Prediction and Plot
KS_train_cart<- max(perf_train_cart@y.values[[1]] -
perf_train_cart@x.values[[1]])
KS_train_cart

## [1] 0.9168758
```

- Calculating the AUC value.

```r
# Calculating the AUC value.

auc_train_Cart = performance(predobj_train_cart,"auc")
auc_train_Cart

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9820527
##
##
## Slot "alpha.values":
## list()
```

The AUC Value calculated is: 0.982

- Calculating the Gini Gain on the dataset.

```r
# Calculating the GINI Value.

gini_train_cart = ineq(Train_Cart$probs,type = "Gini")
gini_train_cart

## [1] 0.8705164
```

The Calculated Gini Gain is: .870

- Calculating the Concordance and Discordance Values.

```r
# Calculating the Concordence and Discordence %.

Concordance(actuals = Train_Cart$Personal.Loan, predictedScores =
Train_Cart$probs)

## $Concordance
## [1] 0.9662694
##
## $Discordance
## [1] 0.03373058
##
## $Tied
## [1] 1.387779e-17
##
## $Pairs
## [1] 1043860
```

We have calculated the Concordance of 96.6% and Discordance of 3.3%.

- Predicting the Values and Probability on Test Data.

```r
########## Predicting the Values on Test Data.

pred_test_Cart <- predict(Model_Train_Cart,newdata = Test_Cart, type =
"class")
pred_test_Cart



Test_Cart<- cbind(Test_Cart,pred_test_Cart)

# Predicting the probability on Test Data
Test_Cart$probs <- predict(Model_Train_Cart,Test_Cart,type = "prob")[,2]

tbl_test<- table(Test_Cart$Personal.Loan,Test_Cart$pred_test_Cart)

print((tbl_test[1,2]+tbl_test[2,1])/nrow(Test_Cart))

## [1] 0.02163624
```

```
     0    1
0 1327    9
1   23  120
```

We were able to predict 1327 Zeros and 120 Once correctly from the Test Dataset, having the error rate of 2.16% and Accuracy of (1 – 2.16) i.e. 97.84%

- Creating the Confusion Matrix

```
## Create Confusion matrix on the above prediction

caret::confusionMatrix(Test_Cart$pred_test_Cart,Test_Cart$Personal.Loan)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1327   23
         1    9  120

               Accuracy : 0.9784
                 95% CI : (0.9696, 0.9852)
    No Information Rate : 0.9033
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8705

 Mcnemar's Test P-Value : 0.02156

            Sensitivity : 0.9933
            Specificity : 0.8392
         Pos Pred Value : 0.9830
         Neg Pred Value : 0.9302
             Prevalence : 0.9033
         Detection Rate : 0.8972
   Detection Prevalence : 0.9128
      Balanced Accuracy : 0.9162

       'Positive' Class : 0
```

From the above Confusion Matrix, we have achieved the Accuracy of 97.84%, Specificity of 83.92% and Sensitivity of 99.33%

- Preparing the Rank Table on Test Data.

```r
# Preparing the Rank Table.
Test_Cart$Decile <- cut(Test_Cart$probs,unique(qs_train),include.lowest =
TRUE, right = FALSE)
TestDT_CART <- data.table(Test_Cart)

TestRanktbl_Cart <- TestDT_CART[,list(
  count <- length(Personal.Loan),
  Count_One <- sum(Personal.Loan ==1),
  Count_Zero <- sum(Personal.Loan == 0)
), by = Decile][order(-Decile)]

TestRanktbl_Cart

##              Decile   V1  V2   V3
## 1:        [0.169,1]  218 140   78
## 2: [0.00389,0.169) 1092   3 1089
## 3:      [0,0.00389)  169   0  169

names(TestRanktbl_Cart) <- c("Decile","Count","Count_One","Count_Zero")

TestRanktbl_Cart$rrate <-
round((TestRanktbl_Cart$Count_One/TestRanktbl_Cart$Count),4)*100
TestRanktbl_Cart$cum_res <- cumsum(TestRanktbl_Cart$Count_One)
TestRanktbl_Cart$cum_non_res <- cumsum((TestRanktbl_Cart$Count_Zero))
TestRanktbl_Cart$cum_rel_res <-
round(TestRanktbl_Cart$cum_res/sum(TestRanktbl_Cart$Count_One),4)*100
TestRanktbl_Cart$cum_rel_non_res <-
round(TestRanktbl_Cart$cum_non_res/sum(TestRanktbl_Cart$Count_Zero),4)*100
TestRanktbl_Cart$ks <- abs(TestRanktbl_Cart$cum_rel_res -
TestRanktbl_Cart$cum_rel_non_res)
```

The data is deciled into 3 groups. Once the Deciles are created, we start calculating the below parameters from the data set

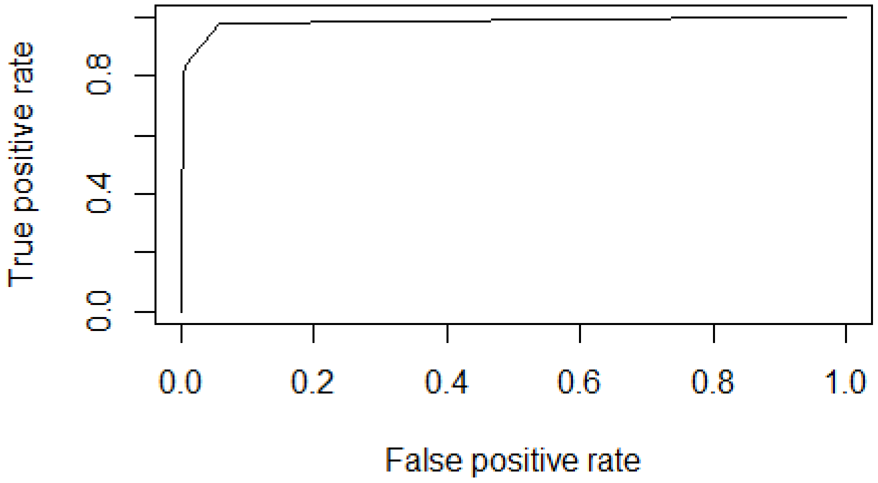
- Count
- Count of Ones
- Count of Zeros
- Response Rate
- Cumulative Response Rate
- Cumulative Non-Response Rate
- Cumulative Relative Response Rate
- Cumulative Relative Non-Response Rate
- KS Value

Decile	Count	Count_One	Count_Zero	rrate	cum_res	cum_non_res	cum_rel_res	cum_rel_non_res	ks
[0.169,1]	218	140	78	64.22	140	78	97.9	5.84	92.06
[0.00389,0.169)	1092	3	1089	0.27	143	1167	100.0	87.35	12.65
[0,0.00389)	169	0	169	0.00	143	1336	100.0	100.00	0.00

From the Rank Table we have achieved the KS value of 92.06%

- Plotting the ROC Curve.

```r
# Plotting the ROC Curve

predobj_test_cart <- prediction(Test_Cart$probs, Test_Cart$Personal.Loan)
perf_test_cart <- performance(predobj_test_cart,"tpr","fpr")
plot(perf_test_cart)
```



- Calculating the KS Value from the Curve.

```r
# Calculating KS from the Plot

KS_Test_cart <- max(perf_test_cart@y.values[[1]] -
perf_test_cart@x.values[[1]])
KS_Test_cart

## [1] 0.9206377
```

- Calculating the AUC Value on Test Data

```r
# Calculating AUC

auc_test_Cart <- performance(predobj_test_cart,"auc")
auc_test_Cart

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9840459
##
##
## Slot "alpha.values":
## list()
```

The Calculated AUC Value is 98.40%

- Calculating the GINI Gain.

```r
#Calculating GINI on Test Cart.

gini_test_cart <- ineq(Test_Cart$probs,"Gini")
gini_test_cart

## [1] 0.8701375
```

Calculated Gini Gain is: 87.01

- Calculating Concordance and Discordance.

```r
# Calculating Concordence on Test Cart

Concordance(actuals = Test_Cart$Personal.Loan, predictedScores =
Test_Cart$probs)

## $Concordance
## [1] 0.9704158
##
## $Discordance
## [1] 0.02958419
##
## $Tied
## [1] -3.469447e-17
##
## $Pairs
## [1] 191048
```

The Calculated Concordance is 97.04% and Discordance is 2.95%

We Built the CART Model on Both Train and Test dataset and performed various performance measures to Validate the Model accuracy. We will now be splitting the dataset again to Train and Test to perform the Random Forest Model.

- Building the RF Model on the Dataset.

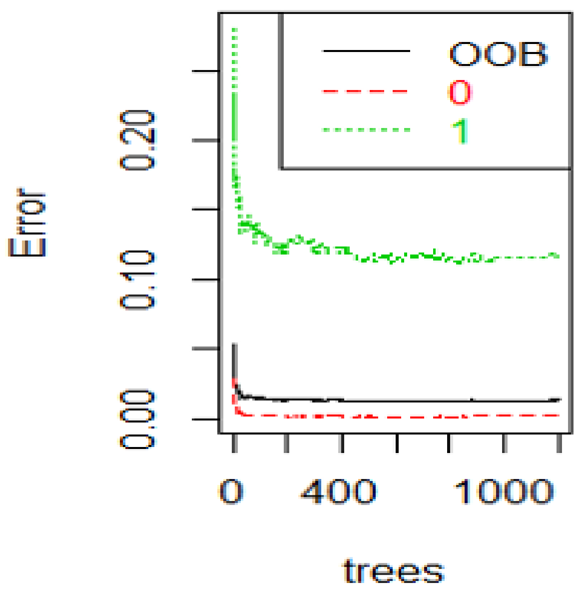
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

```r
###### Building RF Model on the Dataset.

set.seed(1000)
index <- sample.split(theraData$Personal.Loan,SplitRatio = 0.7)

Train_RF <- subset(theraData, index ==TRUE)
Test_RF <- subset(theraData, index ==F)

model_train_rf <- randomForest(Personal.Loan~.,data = Train_RF,type =
"class", mtry = 3,
                         nodesize = 10, ntree= 1200, importance = TRUE)
model_train_rf

##
## Call:
##  randomForest(formula = Personal.Loan ~ ., data = Train_RF, type =
"class",      mtry = 3, nodesize = 10, ntree = 1200, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1200
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.36%
## Confusion matrix:
##      0   1 class.error
## 0 3109   7  0.00224647
## 1   40 295  0.11940299
```

From the Random Forest Model, we have got the Out Of Box error rate of 1.36% and an accuracy of (1-1.36) i.e. 98.6%

- Plotting the RF model.

```r
plot(model_train_rf, main = "")
legend("topright", c("OOB","0","1"),text.col = 1:6,lty = 1:3,col = 1:3)
```



- Tuning the RF Model to find the best mtry and get the optimized tree.

```
trf<- tuneRF(x = Train_RF[,-8],
             y = Train_RF$Personal.Loan,
             mtryStart = 5,
             ntreeTry = 1200,
             stepFactor = 1.5,
             improve = 0.0001,
             trace = TRUE,
             plot = TRUE,
             doBest = FALSE,
             importance = TRUE,
             nodesize = 50)

## mtry = 5  OOB error = 1.54%
## Searching left ...
## mtry = 4     OOB error = 1.77%
## -0.1509434 1e-04
## Searching right ...
## mtry = 7     OOB error = 1.39%
## 0.09433962 1e-04
## mtry = 10    OOB error = 1.65%
## -0.1875 1e-04

trf

##          mtry   OOBError
## 4.OOB       4 0.01767604
## 5.OOB       5 0.01535787
## 7.OOB       7 0.01390901
## 10.OOB     10 0.01651695
```

From the tuning, we recognized that the best fit is on mtry 7 where we have the least OOB error of 0.01390

Hence, we create the Model with ntree = 1200 and mtry = 7.

- Predicting the RF model on Train Dataset.

Once the model is built, we will perform the Model Evaluation techniques by Predicting the Values and Probability on the Model.

```
# Predicting the RF model on Train dataset
pred_train_rf <- predict(model_train_rf,Train_RF,type = "class")

Train_RF <- cbind(Train_RF,pred_train_rf)
Train_RF$probs_rf <- predict(model_train_rf, Train_RF, type = "prob")[,2]
```

- Creating the Confusion Matrix on the above Prediction.

```
## Create Confusion matrix on the above prediction

caret::confusionMatrix(Train_RF$pred_train_rf,Train_RF$Personal.Loan)
```

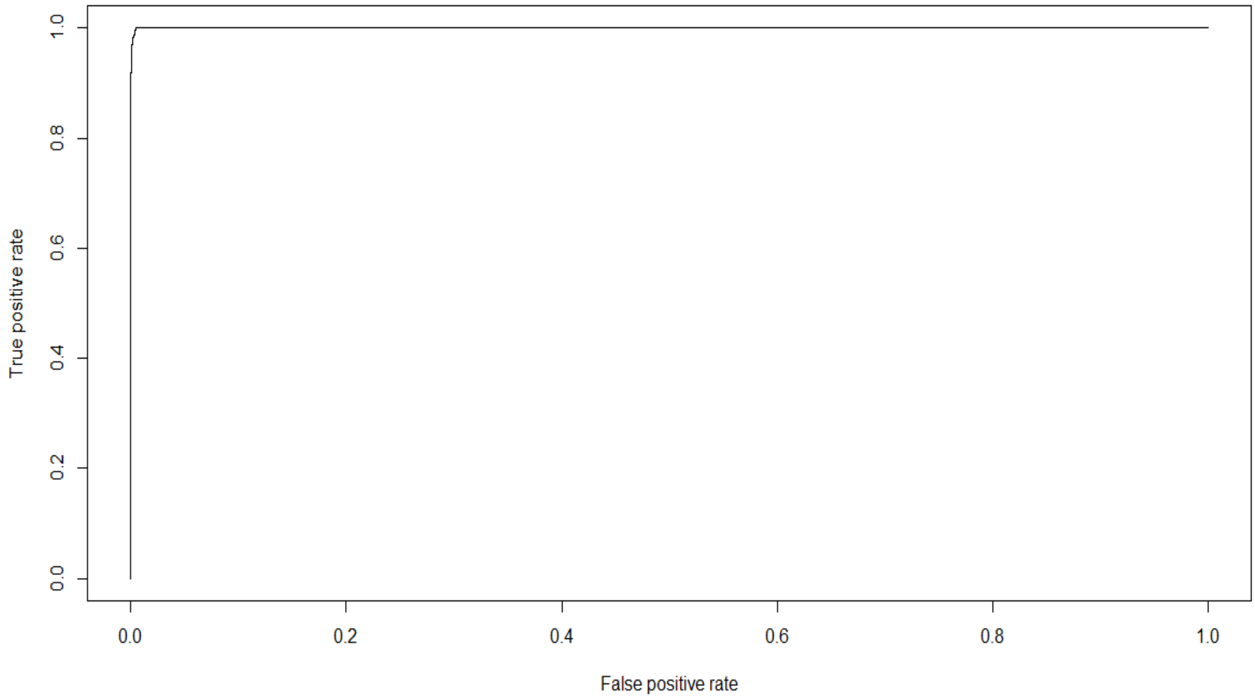
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3114   22
         1    2  313

               Accuracy : 0.993
                 95% CI : (0.9897, 0.9955)
    No Information Rate : 0.9029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9592

 Mcnemar's Test P-Value : 0.0001052

            Sensitivity : 0.9994
            Specificity : 0.9343
         Pos Pred Value : 0.9930
         Neg Pred Value : 0.9937
             Prevalence : 0.9029
         Detection Rate : 0.9023
   Detection Prevalence : 0.9087
      Balanced Accuracy : 0.9668

       'Positive' Class : 0
```

From the above Confusion Matrix, we have achieved the Accuracy of 99.3%, Specificity of 93.43% and Sensitivity of 99.94%.

- Creating the Rank Table on the Train Dataset.

```r
# Creating the Rank Table for RF Model on Train Dataset

Train_RF$Decile_RF <- cut(Train_RF$probs_rf,unique(qs_train),include.lowest =
TRUE,right = FALSE)

table(Train_RF$Decile_RF)

##
##     [0,0.00389) [0.00389,0.169)       [0.169,1]
##            2292             786             373

TrainDT_RF <- data.table(Train_RF)

TrainRanktbl_RF <- TrainDT_RF[,list(
  count <- length(Personal.Loan),
  count_One <- sum(Personal.Loan == 1),
  count_zero <- sum(Personal.Loan == 0)
),by = Decile_RF][order(-Decile_RF)]


names(TrainRanktbl_RF) <- c("Decile_RF", "Count","Count_One","Count_Zero")
```

```r
TrainRanktbl_RF$rrate <-
round((TrainRanktbl_RF$Count_One/TrainRanktbl_RF$Count),4)*100
TrainRanktbl_RF$cum_res <- cumsum(TrainRanktbl_RF$Count_One)
TrainRanktbl_RF$cum_non_res <- cumsum(TrainRanktbl_RF$Count_Zero)
TrainRanktbl_RF$cum_rel_res <-
round((TrainRanktbl_RF$cum_res/sum(TrainRanktbl_RF$cum_res)),4)*100
TrainRanktbl_RF$cum_rel_non_res <-
round((TrainRanktbl_RF$cum_non_res/sum(TrainRanktbl_RF$cum_non_res)),4)*100
TrainRanktbl_RF$ks <- abs(TrainRanktbl_RF$cum_rel_res -
TrainRanktbl_RF$cum_rel_non_res)
TrainRanktbl_RF

##          Decile_RF Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:       [0.169,1]   373       335         38 89.81     335          38
## 2: [0.00389,0.169)   786         0        786  0.00     335         824
## 3:     [0,0.00389)  2292         0       2292  0.00     335        3116
##    cum_rel_res cum_rel_non_res    ks
## 1:       33.33            0.96 32.37
## 2:       33.33           20.71 12.62
## 3:       33.33           78.33 45.00
```



- Calculating the KS Value on the Plot.

```
# Calculating the KS value on Train

KS_Train_RF <- max(perf_Train_RF@y.values[[1]] - perf_Train_RF@x.values[[1]])
KS_Train_RF

## [1] 0.9951861
```

- Calculating the AUC Value on Train Data.

```r
#Calculating the AUC for Train in RF.

auc_train_rf <- performance(predObj_train_RF,"auc")
auc_train_rf

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9998836
##
##
## Slot "alpha.values":
## list()
```

AUC Value is: 99.98%

- Calculating the Gini Gain on Train Data.

```r
# Calculating the GINI

gini_train_rf <- ineq(Train_RF$probs_rf,"Gini")
gini_train_rf

## [1] 0.8886633
```

Calculated Gini Value is: 88.86%

- Calculating the Concordance and Discordance Values.

```
# Calculating Concordence

Concordance(actuals = Train_RF$Personal.Loan, predictedScores =
Train_RF$probs_rf)

## $Concordance
## [1] 0.9998831
##
## $Discordance
## [1] 0.0001168739
##
## $Tied
## [1] -1.568027e-17
##
## $Pairs
## [1] 1043860
```

The Calculated Concordance value is 99.99% and Discordance value is 0.01%.

- Performing the Prediction on Test Dataset.

```
#Validating the RF Model on Test Data
pred_test_RF <- predict(model_train_rf,newdata = Test_RF, type = "class")
pred_test_RF

Test_RF<- cbind(Test_RF,pred_test_RF)

Test_RF$probs_test_rf <- predict(model_train_rf, newdata = Test_RF, type =
"prob")[,2]
```

- Creating Confusion Matrix on Test dataset.

```
## Create Confusion matrix on the above prediction

caret::confusionMatrix(Test_RF$pred_test_RF,Test_RF$Personal.Loan)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1329   19
         1    7  124

               Accuracy : 0.9824
                 95% CI : (0.9743, 0.9885)
    No Information Rate : 0.9033
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8954

 Mcnemar's Test P-Value : 0.03098

            Sensitivity : 0.9948
            Specificity : 0.8671
         Pos Pred Value : 0.9859
         Neg Pred Value : 0.9466
             Prevalence : 0.9033
         Detection Rate : 0.8986
   Detection Prevalence : 0.9114
      Balanced Accuracy : 0.9309

       'Positive' Class : 0
```

From the above Confusion Matrix, we have achieved the Accuracy of 98.24%, Specificity of 86.71% and Sensitivity of 99.48%

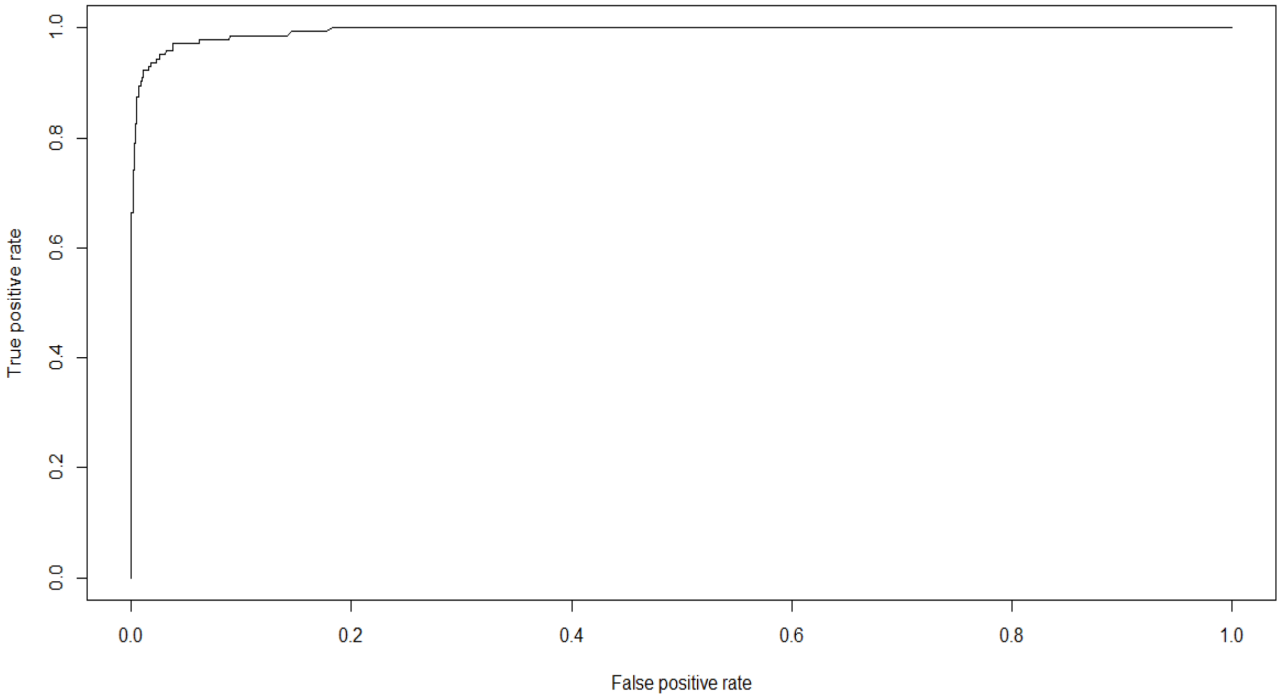
- Preparing the Rank Table on Test data.

```
#Preparing the rank Table

prob <- seq (0,1, length = 11)
prob

##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

qs_test <- quantile(Test_RF$probs_test_rf)
Test_RF$Decile_Test <-
cut(Test_RF$probs_test_rf,unique(qs_test),include.lowest = TRUE,right =
FALSE)
TestDS_RF <- data.table(Test_RF)

TestRanktbl_RF <- TestDS_RF[, list(
  count <- length(Personal.Loan),
  count_one <- sum(Personal.Loan == 1),
  count_zero <- sum(Personal.Loan== 0)
  ),by = Decile_Test][order(-Decile_Test)]


names(TestRanktbl_RF) <-
make.names(c("Decile_Test","Count","Count_One","Count_Zero"))
TestRanktbl_RF$rrate <-
round((TestRanktbl_RF$Count_One/TestRanktbl_RF$Count),4)*100
TestRanktbl_RF$cum_res <- cumsum(TestRanktbl_RF$Count_One)
TestRanktbl_RF$cum_non_res <- cumsum(TestRanktbl_RF$Count_Zero)
TestRanktbl_RF$cum_rel_res <-
round((TestRanktbl_RF$cum_res/sum(TestRanktbl_RF$cum_res)),4)*100
TestRanktbl_RF$cum_rel_non_res <-
round((TestRanktbl_RF$cum_non_res/sum(TestRanktbl_RF$cum_non_res)),4)*100
TestRanktbl_RF$ks <- abs(TestRanktbl_RF$cum_rel_res -
TestRanktbl_RF$cum_rel_non_res)
TestRanktbl_RF
```

```
TestRanktbl_RF

##        Decile_Test Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:   [0.02,0.992]   373       142        231 38.07     142         231
## 2: [0.00167,0.02)   418         1        417  0.24     143         648
## 3:    [0,0.00167)   688         0        688  0.00     143        1336
##    cum_rel_res cum_rel_non_res    ks
## 1:       33.18           10.43 22.75
## 2:       33.41           29.26  4.15
## 3:       33.41           60.32 26.91
```

- Plotting the ROC Curve on Test Data.

```
# Plotting the ROC Curve

predObj_test_RF <- prediction(Test_RF$probs_test_rf,Test_RF$Personal.Loan)
perf_test_RF <- performance(predObj_test_RF,"tpr","fpr")
plot(perf_test_RF)
```



- Calculating the KS from the Plot.

```
# Calculating the KS from ROC Plot  for test RF

KS_Test_RF <- max(perf_test_RF@y.values[[1]] - perf_test_RF@x.values[[1]])
KS_Test_RF

## [1] 0.9346028
```

KS Calculated is: 93.4%

- Calculating the AUC Value for Test.

```r
# Calculating the AUC on RF Model Test Dataset

auc_test_RF <- performance(predObj_test_RF, "auc")
auc_test_RF

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9942946
##
##
## Slot "alpha.values":
## list()
```

AUC Calculated is: 99.42%

- Calculating Gini for Test Data.

```r
# Calculating GINI on RF Model Test dataset

gini_test_rf <- ineq(Test_RF$probs_test_rf,"Gini")
gini_test_rf

## [1] 0.8753027
```

Calculated Gini Gain: 87.53%

- Calculating Concordance and Discordance Values on Test.

```
# Calculating the Concordence on RF Model Test Dataset

Concordance(actuals = Test_RF$Personal.Loan, predictedScores =
Test_RF$probs_test_rf)

## $Concordance
## [1] 0.9942528
##
## $Discordance
## [1] 0.005747247
##
## $Tied
## [1] 4.163336e-17
##
## $Pairs
## [1] 191048
```

Concordance value achieved is: 99.42%
Discordance value achieved is: 0.5%

5.1 Summary

Here is the Summary for both the Models after performing all the Performance measure.

	CART-Train	CART-Test	Random Forest-Train	Random Forest-Test
Accuracy	98.41	97.84	99.3	98.24
Sensitivity	99.55	99.33	99.94	99.48
Specificity	87.76	83.92	93.43	86.71
KS Value	91.68	92.06	99.51	93.4
AUC Value	98.2	98.4	99.98	99.42
Gini Gain	87.05	87.04	88.86	87.53
Concordance	96.6	97.05	99.99	99.42
Discordance	3.4	2.95	0.01	0.6

From the above Comparison Matrix, we can Conclude that the Bank should opt for Random Forest Model as it gives better Accuracy and yields to better performance during all the Performance Measures applied on the Test and Train Dataset. And can classify the right customers who have a higher probability of purchasing the loan.

6 Appendix A – Source Code

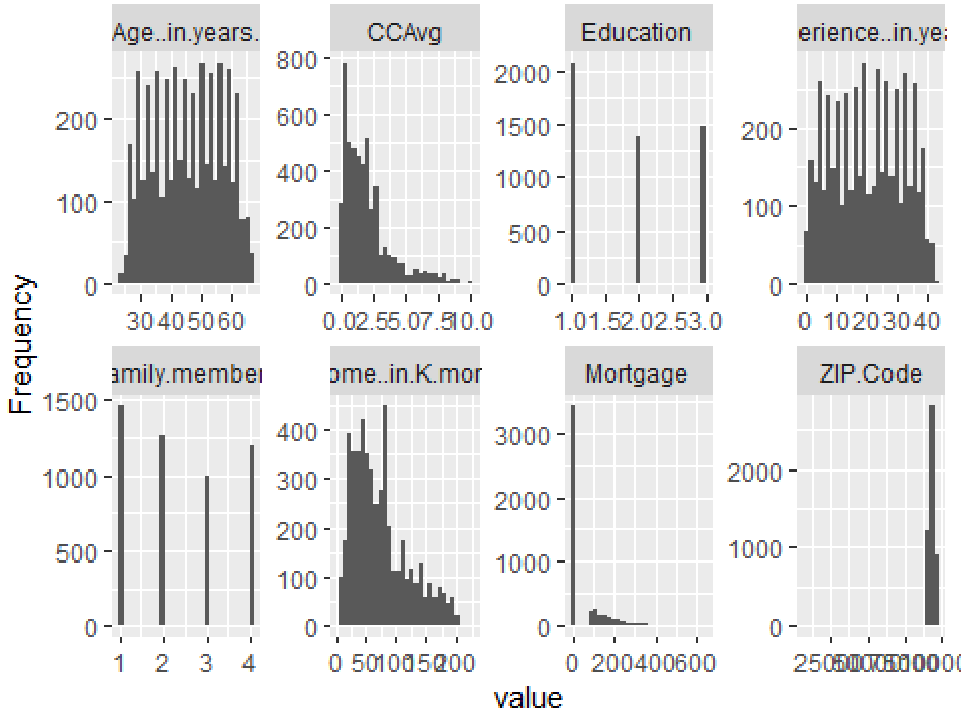
```
#################################################################
#
#Project 3: Thera Bank - Loan Purchase Modeling
#
```

```
##################################################################

#Installing and Deploying required packages

install.packages("e1071")

install.packages("ROCR")

install.packages("rpart.plot")

install.packages("readxl")

install.packages("randomForest")

install.packages("data.table")

install.packages("ineq")

install.packages("InformationValue")

install.packages("caret")

library(readxl)
library(DataExplorer)
library(corrplot)

## corrplot 0.84 loaded

library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(data.table)
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

library(ineq)
library(InformationValue)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'
```

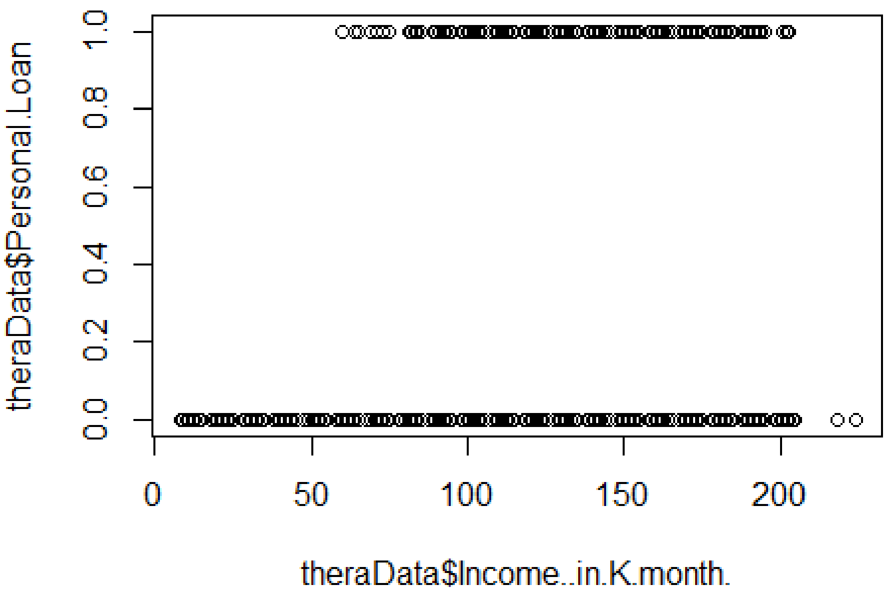
```
## The following object is masked from 'package:randomForest':
##
##     margin

##
## Attaching package: 'caret'

## The following objects are masked from 'package:InformationValue':
##
##     confusionMatrix, precision, sensitivity, specificity

library(e1071)

# Setting the Working Directory.
setwd("D:/Great Learning/Project 3")

# Reading the Dataset
theraData <- read_xlsx("TheraBank.xlsx", sheet = 1)

# Performing Exploratory Data Analysis
head(theraData)

## # A tibble: 6 x 14
##      ID `Age (in years)` `Experience (in~ `Income (in K/m~ `ZIP Code`
##   <dbl>            <dbl>            <dbl>            <dbl>      <dbl>
## 1     1               25                1               49      91107
## 2     2               45               19               34      90089
## 3     3               39               15               11      94720
## 4     4               35                9              100      94112
## 5     5               35                8               45      91330
## 6     6               37               13               29      92121
## # ... with 9 more variables: `Family members` <dbl>, CCAvg <dbl>,
## #   Education <dbl>, Mortgage <dbl>, `Personal Loan` <dbl>, `Securities
## #   Account` <dbl>, `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>

View(theraData)

names(theraData)

##  [1] "ID"                     "Age (in years)"
##  [3] "Experience (in years)" "Income (in K/month)"
##  [5] "ZIP Code"               "Family members"
##  [7] "CCAvg"                  "Education"
##  [9] "Mortgage"               "Personal Loan"
## [11] "Securities Account"     "CD Account"
## [13] "Online"                 "CreditCard"

theraData<- theraData[,-1]

str(theraData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  13 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 ...
##  $ ZIP Code             : num  91107 90089 94720 94112 91330 ...
```

```
##  $ 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(theraData)

##  Age (in years)  Experience (in years) Income (in K/month)    ZIP Code
##  Min.   :23.00   Min.   :-3.0          Min.   :  8.00      Min.   : 9307
##  1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00      1st Qu.:91911
##  Median :45.00   Median :20.0          Median : 64.00      Median :93437
##  Mean   :45.34   Mean   :20.1          Mean   : 73.77      Mean   :93153
##  3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00      3rd Qu.:94608
##  Max.   :67.00   Max.   :43.0          Max.   :224.00      Max.   :96651
##
##  Family members      CCAvg          Education        Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.397   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  NA's   :18
##  Personal Loan   Securities Account   CD Account          Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
##

# Removing negative records from the Dataset.
theraData <- subset(theraData, theraData$`Experience (in years)` >= 0)

# Changing the Column names for the dataset

names(theraData) <- make.names(c("Age (in years)","Experience (in years)","Income (in K/
month)","ZIP Code","Family members","CCAvg",
             "Education","Mortgage","Personal Loan","Securities Account","CD Account","O
nline","CreditCard"),allow_ = TRUE, unique = FALSE)
```

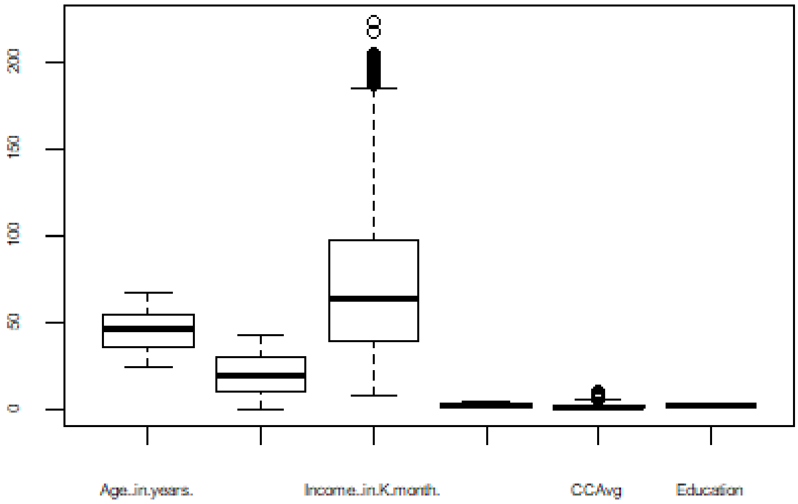
```r
# Performing Univariate Analysis.

plot_histogram(theraData)
```



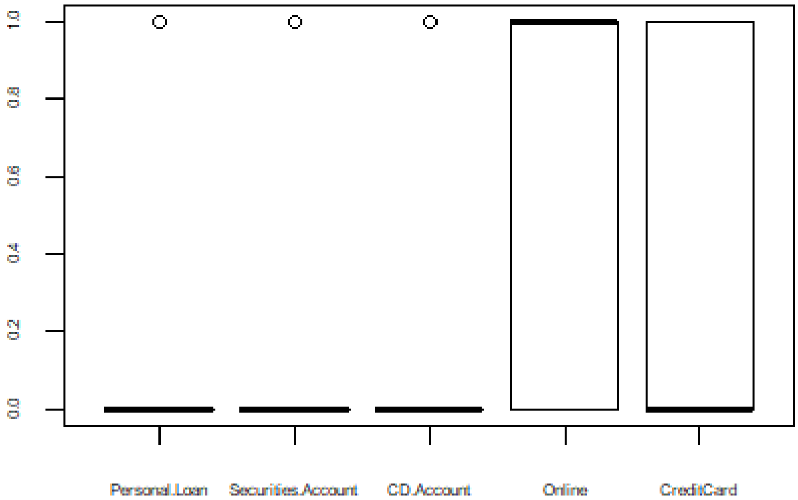

```r
plot(theraData$Income..in.K.month.,theraData$Personal.Loan,)
```




```r
par(mfrow = c(1,1))
boxplot(theraData[,c(1,2,3,5,6,7)],cex.axis = 0.5)
```

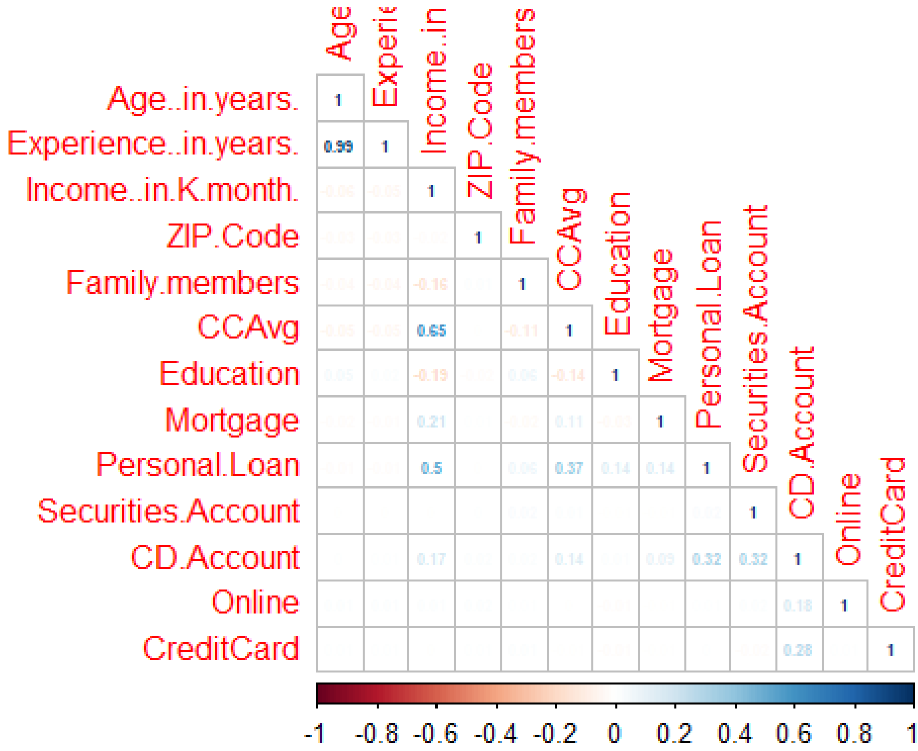



```r
boxplot(theraData[,c(9,10,11,12,13)],cex.axis = 0.5)
```



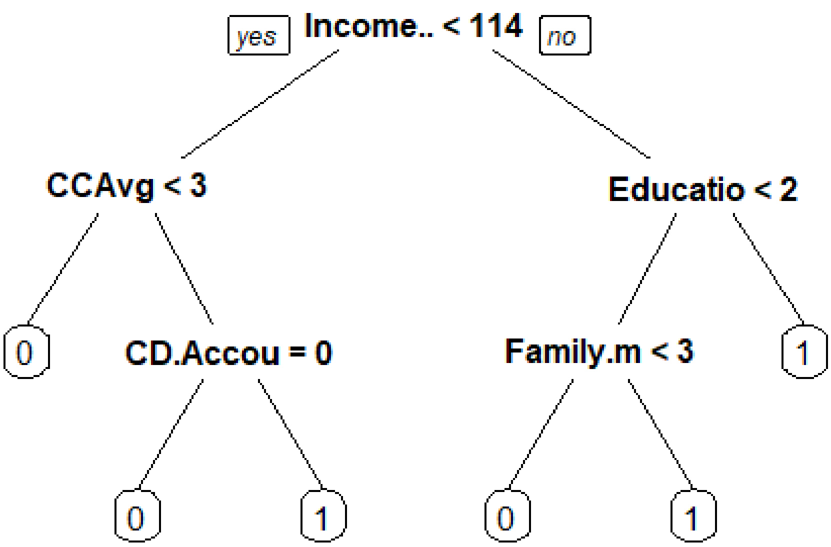

```r
# Checking for the Null Values.

sum(is.na(theraData))

## [1] 18
```

```r
colSums(is.na(theraData))

##        Age..in.years. Experience..in.years.   Income..in.K.month.
##                     0                     0                     0
##              ZIP.Code        Family.members                 CCAvg
##                     0                    18                     0
##             Education              Mortgage         Personal.Loan
##                     0                     0                     0
##    Securities.Account            CD.Account                Online
##                     0                     0                     0
##            CreditCard
##                     0

# Removing Null Values.

theraData<- na.omit(theraData)
colSums(is.na(theraData))

##        Age..in.years. Experience..in.years.   Income..in.K.month.
##                     0                     0                     0
##              ZIP.Code        Family.members                 CCAvg
##                     0                     0                     0
##             Education              Mortgage         Personal.Loan
##                     0                     0                     0
##    Securities.Account            CD.Account                Online
##                     0                     0                     0
##            CreditCard
##                     0

# Checking for the Correlation between the Varibles.

matrix <- cor(theraData)
corrplot(matrix, method = "number", type = "lower",number.cex = 0.5)
```

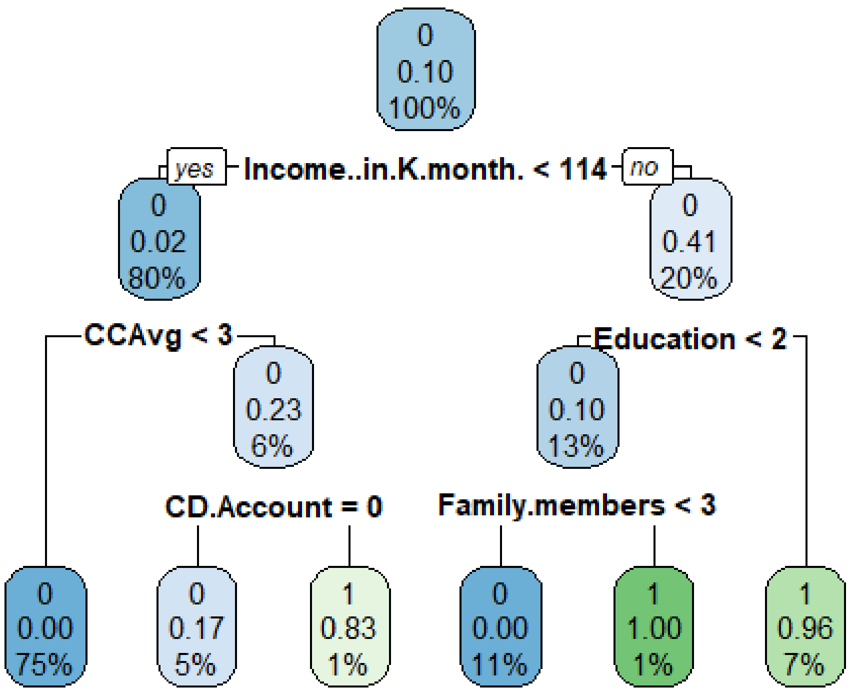



```r
# Removing Experience as there exist a huge Correlation between Age and Experience
theraData<- theraData[,-1]

# Converting the variables to Factors

#theraData$Education <- as.factor(theraData$Education)
theraData$Personal.Loan <- as.factor(theraData$Personal.Loan)
theraData$Securities.Account <- as.factor(theraData$Securities.Account)
theraData$CD.Account <- as.factor(theraData$CD.Account)
theraData$Online <- as.factor(theraData$Online)
theraData$CreditCard <- as.factor(theraData$CreditCard)

str(theraData)

## Classes 'tbl_df', 'tbl' and 'data.frame':    4930 obs. of  12 variables:
##  $ 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 ...
##  $ ZIP.Code             : num  91107 90089 94720 94112 91330 ...
##  $ 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        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Securities.Account   : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ CD.Account           : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Online               : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard           : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int  21 59 98 161 234 288 484 709 1443 1444 ...
##   ..- attr(*, "names")= chr  "21" "59" "98" "161" ...

nrow(theraData)
```

```
## [1] 4930

# Splitting data into Train and Test with a split of 70, 30 respectively.
set.seed(1000)
index <- sample.split(theraData$Personal.Loan,SplitRatio = 0.7)

Train_Cart <- subset(theraData, index ==TRUE)
Test_Cart <- subset(theraData, index ==F)

# Building CART Model on Train Data

Model_Train_Cart <- rpart(Personal.Loan~.,data = Train_Cart,method = "class")
Model_Train_Cart

## n= 3451
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3451 335 0 (0.902926688 0.097073312)
##    2) Income..in.K.month.< 113.5 2773  56 0 (0.979805265 0.020194735)
##      4) CCAvg< 2.95 2572  10 0 (0.996111975 0.003888025) *
##      5) CCAvg>=2.95 201  46 0 (0.771144279 0.228855721)
##       10) CD.Account=0 183  31 0 (0.830601093 0.169398907) *
##       11) CD.Account=1 18   3 1 (0.166666667 0.833333333) *
##    3) Income..in.K.month.>=113.5 678 279 0 (0.588495575 0.411504425)
##      6) Education< 1.5 433  45 0 (0.896073903 0.103926097)
##       12) Family.members< 2.5 388   0 0 (1.000000000 0.000000000) *
##       13) Family.members>=2.5 45   0 1 (0.000000000 1.000000000) *
##      7) Education>=1.5 245  11 1 (0.044897959 0.955102041) *

prp(Model_Train_Cart)
```



```
rpart.plot(Model_Train_Cart,tweak = 1.2)
```




```
Model_Train_Cart$cptable

##           CP nsplit rel error    xerror       xstd
## 1 0.33283582      0 1.0000000 1.0000000 0.05191631
## 2 0.13432836      2 0.3343284 0.3761194 0.03289000
## 3 0.01791045      3 0.2000000 0.2537313 0.02717999
## 4 0.01000000      5 0.1641791 0.1910448 0.02365812

########## Predicting the values on Train.

pred_train_cart <- predict(Model_Train_Cart,newdata = Train_Cart, type = "class")
pred_train_cart

##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
##   0   0   0   0   0   0   0   1   0   0   0   1   0   1   0
##  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30
##   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
##  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
##   1   0   1   0   0   0   1   0   0   0   1   0   1   0   0
##  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
##   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
##   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
## 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
```

```
##    0    0    0    1    0    0    0    0    1    0    0    1    0    0    0
##  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0
##  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180
##    0    0    0    1    0    1    0    1    0    0    1    0    0    0    0
##  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210
##    0    0    0    0    0    0    0    0    1    0    0    1    0    0    0
##  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225
##    0    0    0    0    0    0    0    1    1    0    0    0    1    0    0
##  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255
##    1    1    0    0    0    0    0    0    0    0    0    0    0    0    0
##  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
##  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
##  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315
##    0    1    0    0    0    0    0    0    0    0    0    0    0    0    1
##  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
##  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0
##  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390
##    0    0    0    0    0    0    0    0    0    0    1    0    1    0    0
##  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405
##    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
##  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
##  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435
##    0    0    1    0    0    0    0    0    0    0    0    0    0    1    0
##  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450
##    0    0    1    0    0    0    0    0    0    0    0    0    1    0    0
##  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
##  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525
##    0    0    1    0    1    0    0    0    1    0    1    0    1    1    0
##  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540
```

```
##   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
## 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555
##   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0
## 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570
##   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
## 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585
##   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
## 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600
##   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0
## 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
##   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
## 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0
## 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645
##   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
## 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660
##   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0
## 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675
##   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0
## 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690
##   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
## 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705
##   0   0   1   0   0   0   0   0   0   0   0   0   0   1   1
## 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
##   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1
## 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
## 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750
##   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
## 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765
##   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780
##   0   0   0   0   0   0   1   1   0   0   0   0   0   1   0
## 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795
##   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0
## 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810
##   0   0   1   0   0   0   0   0   0   1   0   0   0   0   0
## 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825
##   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1
## 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840
##   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0
## 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855
##   0   0   0   0   0   0   0   1   0   0   0   1   0   0   0
## 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885
##   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
## 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900
##   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0
## 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915
##   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
## 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930
##   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945
```

```
##    0    0    0    0    0    0    0    0    0    0    1    0    1    0    0
##  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
##    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
##  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975
##    1    0    0    1    0    0    1    0    0    0    0    0    1    0    0
##  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
##  991  992  993  994  995  996  997  998  999 1000 1001 1002 1003 1004 1005
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
## 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035
##    1    0    0    0    0    0    0    1    0    0    1    0    0    0    0
## 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095
##    1    0    1    1    0    0    0    0    0    1    0    0    0    0    0
## 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0
## 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125
##    0    0    0    0    0    0    1    1    0    0    0    1    1    0    0
## 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140
##    0    0    0    0    0    1    1    0    0    0    1    0    0    0    0
## 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155
##    0    0    0    0    0    0    1    0    1    0    0    1    0    0    0
## 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    1
## 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260
##    0    0    0    0    0    1    0    1    1    0    0    0    0    0    0
## 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320
##    0    0    0    0    0    0    1    1    0    0    0    1    0    0    0
## 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
## 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
```

```
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365
##    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380
##    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
## 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410
##    0    0    1    0    0    0    0    0    0    0    0    0    1    0    0
## 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    0    1    0    0    0    0    0    0    0    0    0    1    0    0    0
## 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455
##    0    0    0    0    0    1    0    0    0    0    0    0    1    0    0
## 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470
##    0    1    0    0    0    1    0    0    0    0    0    0    0    0    0
## 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    1
## 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530
##    0    0    0    0    0    0    0    0    0    0    1    1    0    0    0
## 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0
## 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575
##    1    1    0    0    0    1    0    0    0    0    0    1    0    0    0
## 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590
##    0    0    1    0    0    0    0    0    1    0    0    0    0    0    0
## 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605
##    0    0    1    0    0    0    0    0    0    1    0    1    0    0    0
## 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
## 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635
##    1    0    0    0    1    0    0    0    0    0    1    0    0    0    0
## 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
## 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665
##    1    0    0    0    1    0    0    0    0    0    0    0    0    0    0
## 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695
##    0    1    0    0    0    1    0    0    0    0    0    0    0    0    0
## 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710
##    0    1    0    0    0    0    0    1    0    0    0    0    0    1    0
## 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
## 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740
##    0    0    0    0    0    1    0    0    0    0    0    0    1    0    0
## 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755
```

```
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800
##    0    0    0    0    0    0    0    1    1    0    0    1    0    0    0
## 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815
##    0    0    0    0    0    0    0    1    1    0    0    0    0    0    1
## 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860
##    0    1    0    0    0    0    0    1    0    0    0    0    1    0    0
## 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
## 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890
##    0    1    0    0    0    0    0    1    0    0    0    0    1    0    0
## 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935
##    0    0    0    0    1    0    0    1    1    0    0    0    0    0    0
## 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950
##    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
## 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980
##    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995
##    0    0    0    0    0    0    0    0    0    0    1    0    0    1    0
## 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
##    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
## 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025
##    0    0    1    0    0    0    0    0    1    0    0    0    0    0    0
## 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
## 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
## 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0
## 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160
```

```
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
## 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175
##    0    0    0    0    0    0    0    0    1    0    0    1    0    0    1
## 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0
## 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265
##    0    0    0    0    0    0    0    0    1    0    0    0    0    1    0
## 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280
##    0    0    1    0    0    0    0    0    0    0    0    0    0    1    1
## 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310
##    0    0    1    1    0    0    0    0    0    0    0    0    0    0    0
## 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325
##    0    1    0    0    0    0    0    0    0    0    0    1    0    0    0
## 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340
##    0    0    0    0    1    0    0    1    0    0    0    0    0    0    1
## 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
## 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0
## 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400
##    0    0    0    0    0    0    0    1    0    0    0    0    1    0    0
## 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415
##    0    0    0    0    0    0    1    0    0    0    1    0    0    0    0
## 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430
##    0    0    1    0    0    0    0    1    0    0    0    0    0    0    1
## 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0
## 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475
##    0    0    0    0    0    0    0    0    0    1    1    0    0    0    0
## 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
## 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505
##    0    0    0    0    0    0    0    1    0    0    1    0    0    0    0
## 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535
##    0    0    1    0    0    0    0    1    0    0    0    0    1    0    0
## 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0
## 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565
```

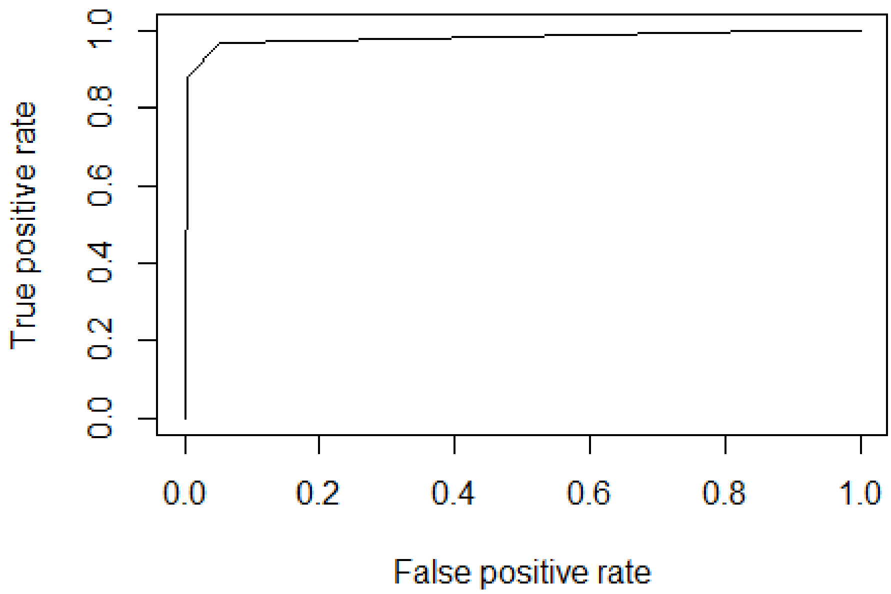
```
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580
##    1    0    1    0    0    0    0    0    0    0    0    0    0    1    0
## 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0
## 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0
## 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
## 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670
##    0    0    0    1    1    0    0    0    0    0    0    1    0    0    0
## 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685
##    0    0    1    0    0    0    0    0    0    0    0    1    0    0    0
## 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730
##    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
## 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756 2757 2758 2759 2760
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774 2775
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0
## 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805
##    1    0    0    1    0    0    0    0    1    0    0    0    0    0    0
## 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815 2816 2817 2818 2819 2820
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830 2831 2832 2833 2834 2835
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
## 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845 2846 2847 2848 2849 2850
##    0    0    0    0    1    0    0    1    0    0    0    0    0    0    0
## 2851 2852 2853 2854 2855 2856 2857 2858 2859 2860 2861 2862 2863 2864 2865
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2866 2867 2868 2869 2870 2871 2872 2873 2874 2875 2876 2877 2878 2879 2880
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2881 2882 2883 2884 2885 2886 2887 2888 2889 2890 2891 2892 2893 2894 2895
##    1    0    0    0    1    0    0    0    0    0    0    0    0    0    1
## 2896 2897 2898 2899 2900 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910
##    0    1    1    0    0    0    0    1    0    0    0    0    0    0    0
## 2911 2912 2913 2914 2915 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 2926 2927 2928 2929 2930 2931 2932 2933 2934 2935 2936 2937 2938 2939 2940
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 2941 2942 2943 2944 2945 2946 2947 2948 2949 2950 2951 2952 2953 2954 2955
##    0    0    0    0    0    0    0    0    0    0    0    0    1    1    0
## 2956 2957 2958 2959 2960 2961 2962 2963 2964 2965 2966 2967 2968 2969 2970
```

```
##    0    1    0    0    0    0    0    0    0    0    0    1    0    0    0
## 2971 2972 2973 2974 2975 2976 2977 2978 2979 2980 2981 2982 2983 2984 2985
##    0    0    0    0    0    1    0    0    1    1    1    0    0    0    0
## 2986 2987 2988 2989 2990 2991 2992 2993 2994 2995 2996 2997 2998 2999 3000
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    1
## 3001 3002 3003 3004 3005 3006 3007 3008 3009 3010 3011 3012 3013 3014 3015
##    0    0    0    0    0    0    0    1    0    0    0    0    0    1    0
## 3016 3017 3018 3019 3020 3021 3022 3023 3024 3025 3026 3027 3028 3029 3030
##    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3031 3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042 3043 3044 3045
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
## 3046 3047 3048 3049 3050 3051 3052 3053 3054 3055 3056 3057 3058 3059 3060
##    0    0    1    0    0    1    0    0    0    0    0    0    0    0    0
## 3061 3062 3063 3064 3065 3066 3067 3068 3069 3070 3071 3072 3073 3074 3075
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3076 3077 3078 3079 3080 3081 3082 3083 3084 3085 3086 3087 3088 3089 3090
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3091 3092 3093 3094 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104 3105
##    1    0    0    0    0    0    0    0    1    0    0    0    0    0    0
## 3106 3107 3108 3109 3110 3111 3112 3113 3114 3115 3116 3117 3118 3119 3120
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3121 3122 3123 3124 3125 3126 3127 3128 3129 3130 3131 3132 3133 3134 3135
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3136 3137 3138 3139 3140 3141 3142 3143 3144 3145 3146 3147 3148 3149 3150
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3151 3152 3153 3154 3155 3156 3157 3158 3159 3160 3161 3162 3163 3164 3165
##    0    0    0    0    0    1    0    0    0    0    0    0    1    0    0
## 3166 3167 3168 3169 3170 3171 3172 3173 3174 3175 3176 3177 3178 3179 3180
##    1    0    0    1    0    0    0    0    0    0    0    0    0    0    0
## 3181 3182 3183 3184 3185 3186 3187 3188 3189 3190 3191 3192 3193 3194 3195
##    0    0    0    1    0    0    0    0    0    0    1    1    0    0    0
## 3196 3197 3198 3199 3200 3201 3202 3203 3204 3205 3206 3207 3208 3209 3210
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
## 3211 3212 3213 3214 3215 3216 3217 3218 3219 3220 3221 3222 3223 3224 3225
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3226 3227 3228 3229 3230 3231 3232 3233 3234 3235 3236 3237 3238 3239 3240
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 3241 3242 3243 3244 3245 3246 3247 3248 3249 3250 3251 3252 3253 3254 3255
##    0    0    1    0    0    0    0    1    0    0    0    0    0    0    0
## 3256 3257 3258 3259 3260 3261 3262 3263 3264 3265 3266 3267 3268 3269 3270
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 3271 3272 3273 3274 3275 3276 3277 3278 3279 3280 3281 3282 3283 3284 3285
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3286 3287 3288 3289 3290 3291 3292 3293 3294 3295 3296 3297 3298 3299 3300
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3301 3302 3303 3304 3305 3306 3307 3308 3309 3310 3311 3312 3313 3314 3315
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3316 3317 3318 3319 3320 3321 3322 3323 3324 3325 3326 3327 3328 3329 3330
##    0    0    0    0    0    0    0    1    1    0    0    0    0    0    1
## 3331 3332 3333 3334 3335 3336 3337 3338 3339 3340 3341 3342 3343 3344 3345
##    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
## 3346 3347 3348 3349 3350 3351 3352 3353 3354 3355 3356 3357 3358 3359 3360
##    1    0    0    0    0    0    0    0    0    0    0    1    1    0    0
## 3361 3362 3363 3364 3365 3366 3367 3368 3369 3370 3371 3372 3373 3374 3375
```

```
##    0    1    0    0    0    0    0    1    0    0    0    0    1    0    0
## 3376 3377 3378 3379 3380 3381 3382 3383 3384 3385 3386 3387 3388 3389 3390
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3391 3392 3393 3394 3395 3396 3397 3398 3399 3400 3401 3402 3403 3404 3405
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
## 3406 3407 3408 3409 3410 3411 3412 3413 3414 3415 3416 3417 3418 3419 3420
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3421 3422 3423 3424 3425 3426 3427 3428 3429 3430 3431 3432 3433 3434 3435
##    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0
## 3436 3437 3438 3439 3440 3441 3442 3443 3444 3445 3446 3447 3448 3449 3450
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 3451
##    0
## Levels: 0 1

Train_Cart<- cbind(Train_Cart,pred_train_cart)

# Predicting the probability on Train Data

Train_Cart$probs <- predict(Model_Train_Cart, Train_Cart, type = "prob")[,2]

head(Train_Cart,n= 5)

##   Experience..in.years. Income..in.K.month. ZIP.Code Family.members CCAvg
## 1                     1                  49    91107              4   1.6
## 2                    15                  11    94720              1   1.0
## 3                     9                 100    94112              1   2.7
## 4                     8                  45    91330              4   1.0
## 5                    13                  29    92121              4   0.4
##   Education Mortgage Personal.Loan Securities.Account CD.Account Online
## 1         1        0             0                  1          0      0
## 2         1        0             0                  0          0      0
## 3         2        0             0                  0          0      0
## 4         2        0             0                  0          0      0
## 5         2      155             0                  0          0      1
##   CreditCard pred_train_cart       probs
## 1          0               0 0.003888025
## 2          0               0 0.003888025
## 3          0               0 0.003888025
## 4          1               0 0.003888025
## 5          0               0 0.003888025

tbl <- table(Train_Cart$Personal.Loan,Train_Cart$pred_train_cart)
tbl

##
##        0    1
##   0 3102   14
##   1   41  294

print((tbl[1,2]+tbl[2,1])/nrow(Train_Cart))

## [1] 0.01593741
```

```r
## Create Confusion matrix on the above prediction

caret::confusionMatrix(Train_Cart$pred_train_cart,Train_Cart$Personal.Loan)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3102   41
##          1   14  294
##
##                Accuracy : 0.9841
##                  95% CI : (0.9793, 0.988)
##     No Information Rate : 0.9029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9057
##
##  Mcnemar's Test P-Value : 0.0004552
##
##             Sensitivity : 0.9955
##             Specificity : 0.8776
##          Pos Pred Value : 0.9870
##          Neg Pred Value : 0.9545
##              Prevalence : 0.9029
##          Detection Rate : 0.8989
##    Detection Prevalence : 0.9108
##       Balanced Accuracy : 0.9366
##
##        'Positive' Class : 0
##

# Preparing the Rank Table on the Train Data.

prob <- seq (0,1, length = 11)
prob

##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

qs_train<- quantile(Train_Cart$probs,prob)

Train_Cart$Decile <- cut(Train_Cart$probs,unique(qs_train), include.lowest = TRUE, right
 = FALSE)

table(Train_Cart$Decile)

##
##     [0,0.00389) [0.00389,0.169)       [0.169,1]
##             388            2572             491

TrainDT_CART<- data.table(Train_Cart)

TrainRankTbl_CART <- TrainDT_CART[,list(
  cnt = length(Personal.Loan),
  cnt_tar1 <- sum(Personal.Loan==1),
  cnt_tar0 <- sum(Personal.Loan==0)
```

```
), by = Decile][order(-Decile)]

names(TrainRankTbl_CART) <- c("Decile","Count","Count_One","Count_Zero")
names(TrainRankTbl_CART)

## [1] "Decile"     "Count"      "Count_One"  "Count_Zero"

TrainRankTbl_CART$rrate <- round(TrainRankTbl_CART$Count_One/TrainRankTbl_CART$Count,4)*
100
TrainRankTbl_CART$cum_res<- cumsum(TrainRankTbl_CART$Count_One)
TrainRankTbl_CART$cum_non_res <- cumsum(TrainRankTbl_CART$Count_Zero)
TrainRankTbl_CART$cum_rel_res <- round(TrainRankTbl_CART$cum_res/sum(TrainRankTbl_CART$C
ount_One),4)*100
TrainRankTbl_CART$cum_rel_non_res <- round(TrainRankTbl_CART$cum_non_res/sum(TrainRankTb
l_CART$Count_Zero),4)*100
TrainRankTbl_CART$ks <- abs(TrainRankTbl_CART$cum_rel_res - TrainRankTbl_CART$cum_rel_no
n_res)

TrainRankTbl_CART

##               Decile Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:       [0.169,1]   491       325        166 66.19     325         166
## 2: [0.00389,0.169)  2572        10       2562  0.39     335        2728
## 3:     [0,0.00389)   388         0        388  0.00     335        3116
##    cum_rel_res cum_rel_non_res    ks
## 1:       97.01            5.33 91.68
## 2:      100.00           87.55 12.45
## 3:      100.00          100.00  0.00

# Plotting the ROC Curve.

predobj_train_cart <- prediction(Train_Cart$probs,Train_Cart$Personal.Loan)
perf_train_cart <- performance(predobj_train_cart,"tpr","fpr")
plot(perf_train_cart)
```




```
# Calculating the KS value from Prediction and Plot
KS_train_cart<- max(perf_train_cart@y.values[[1]] - perf_train_cart@x.values[[1]])
KS_train_cart

## [1] 0.9168758

# Calculating the AUC value.

auc_train_Cart = performance(predobj_train_cart,"auc")
auc_train_Cart

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9820527
##
##
## Slot "alpha.values":
## list()
```

```r
# Calculating the GINI Value.

gini_train_cart = ineq(Train_Cart$probs,type = "Gini")
gini_train_cart

## [1] 0.8705164

# Calculating the Concordence and Discordence %.

Concordance(actuals = Train_Cart$Personal.Loan, predictedScores = Train_Cart$probs)

## $Concordance
## [1] 0.9662694
##
## $Discordance
## [1] 0.03373058
##
## $Tied
## [1] 1.387779e-17
##
## $Pairs
## [1] 1043860

########## Predicting the Values on Test Data.

pred_test_Cart <- predict(Model_Train_Cart,newdata = Test_Cart, type = "class")
pred_test_Cart

##     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
##     0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##    16  17  18  19  20  21  22  23  24  25  26  27  28  29  30
##     0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
##    31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
##     0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
##    46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##     0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
##    61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
##     1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##    76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##     0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##    91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
##     0   1   0   0   0   0   0   0   0   0   0   0   0   0   1
##   106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##     0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
##   121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
##     0   0   0   1   0   0   0   0   1   0   0   0   0   0   0
##   136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
##     0   0   0   1   0   0   1   0   0   0   1   0   0   0   0
##   151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
##     0   0   0   0   0   0   0   0   1   0   0   0   0   0   0
##   166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##     0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
##     0   0   1   0   0   0   0   0   0   0   0   0   1   0   0
##   196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
```

```
##     0    0    0    0    0    0    1    0    0    0    0    0    0    0    0
##   211  212  213  214  215  216  217  218  219  220  221  222  223  224  225
##     0    0    0    0    0    1    0    1    0    0    0    0    1    1    0
##   226  227  228  229  230  231  232  233  234  235  236  237  238  239  240
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    1
##   241  242  243  244  245  246  247  248  249  250  251  252  253  254  255
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   256  257  258  259  260  261  262  263  264  265  266  267  268  269  270
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   271  272  273  274  275  276  277  278  279  280  281  282  283  284  285
##     0    0    0    0    0    1    0    0    0    1    0    0    0    0    0
##   286  287  288  289  290  291  292  293  294  295  296  297  298  299  300
##     0    0    0    0    0    1    0    0    0    1    0    0    0    0    0
##   301  302  303  304  305  306  307  308  309  310  311  312  313  314  315
##     0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
##   316  317  318  319  320  321  322  323  324  325  326  327  328  329  330
##     0    0    0    0    0    0    1    0    0    0    1    0    0    0    0
##   331  332  333  334  335  336  337  338  339  340  341  342  343  344  345
##     0    0    0    0    0    0    1    0    0    0    0    1    0    0    0
##   346  347  348  349  350  351  352  353  354  355  356  357  358  359  360
##     0    0    0    0    0    0    0    0    0    0    1    0    0    0    0
##   361  362  363  364  365  366  367  368  369  370  371  372  373  374  375
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   376  377  378  379  380  381  382  383  384  385  386  387  388  389  390
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   391  392  393  394  395  396  397  398  399  400  401  402  403  404  405
##     1    0    0    0    0    0    0    1    0    0    0    1    0    0    0
##   406  407  408  409  410  411  412  413  414  415  416  417  418  419  420
##     0    0    1    0    0    0    0    0    0    0    0    0    1    0    0
##   421  422  423  424  425  426  427  428  429  430  431  432  433  434  435
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   436  437  438  439  440  441  442  443  444  445  446  447  448  449  450
##     0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
##   451  452  453  454  455  456  457  458  459  460  461  462  463  464  465
##     0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
##   466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
##     0    0    0    0    0    0    0    1    0    0    0    0    0    1    0
##   481  482  483  484  485  486  487  488  489  490  491  492  493  494  495
##     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   496  497  498  499  500  501  502  503  504  505  506  507  508  509  510
##     1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##   511  512  513  514  515  516  517  518  519  520  521  522  523  524  525
##     0    1    0    0    0    0    0    0    0    0    1    0    0    0    0
##   526  527  528  529  530  531  532  533  534  535  536  537  538  539  540
##     0    0    0    0    0    0    0    0    1    0    0    0    0    0    0
##   541  542  543  544  545  546  547  548  549  550  551  552  553  554  555
##     0    0    0    0    1    0    0    0    0    0    0    0    0    0    0
##   556  557  558  559  560  561  562  563  564  565  566  567  568  569  570
##     0    0    0    1    0    1    0    0    0    0    0    0    0    0    0
##   571  572  573  574  575  576  577  578  579  580  581  582  583  584  585
##     0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
##   586  587  588  589  590  591  592  593  594  595  596  597  598  599  600
##     0    1    0    0    0    0    0    0    0    1    0    1    0    0    0
##   601  602  603  604  605  606  607  608  609  610  611  612  613  614  615
```

```
##    0    0    0    0    0    1    0    0    0    0    1    0    0    0    0
##  616  617  618  619  620  621  622  623  624  625  626  627  628  629  630
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  631  632  633  634  635  636  637  638  639  640  641  642  643  644  645
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660
##    1    0    0    0    0    1    0    0    0    1    0    0    0    0    0
##  661  662  663  664  665  666  667  668  669  670  671  672  673  674  675
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    1
##  691  692  693  694  695  696  697  698  699  700  701  702  703  704  705
##    0    0    0    0    0    0    0    1    0    0    0    0    0    1    0
##  706  707  708  709  710  711  712  713  714  715  716  717  718  719  720
##    0    0    0    0    1    1    0    0    0    0    0    0    0    0    0
##  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    0
##  751  752  753  754  755  756  757  758  759  760  761  762  763  764  765
##    0    1    0    0    0    0    0    0    0    0    1    0    0    0    0
##  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  781  782  783  784  785  786  787  788  789  790  791  792  793  794  795
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  796  797  798  799  800  801  802  803  804  805  806  807  808  809  810
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  811  812  813  814  815  816  817  818  819  820  821  822  823  824  825
##    1    0    0    1    0    0    0    0    0    0    0    1    0    1    0
##  826  827  828  829  830  831  832  833  834  835  836  837  838  839  840
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    1
##  841  842  843  844  845  846  847  848  849  850  851  852  853  854  855
##    1    1    0    0    0    0    0    0    0    0    0    0    0    0    0
##  856  857  858  859  860  861  862  863  864  865  866  867  868  869  870
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  871  872  873  874  875  876  877  878  879  880  881  882  883  884  885
##    0    0    0    1    0    0    0    1    1    0    0    0    0    0    1
##  886  887  888  889  890  891  892  893  894  895  896  897  898  899  900
##    0    1    0    1    0    0    1    0    0    0    0    0    0    0    0
##  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    0    0    1    1    0    0    0    0    1    0    0    0    0    0    1
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    1
##  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
##  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0
##  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990
##    0    1    0    1    0    0    0    0    1    0    0    0    0    0    0
##  991  992  993  994  995  996  997  998  999 1000 1001 1002 1003 1004 1005
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
```

```
##    0    0    0    0    0    0    0    0    1    0    1    0    0    0    0
## 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    0    1    0    0    0    0    0    0    0    1    1    0    0    0    0
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065
##    0    0    0    0    0    0    0    0    0    0    0    0    1    1    0
## 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155
##    0    0    1    0    1    1    1    0    0    0    1    0    0    0    0
## 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170
##    0    0    0    0    0    0    1    1    0    0    0    0    1    0    0
## 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185
##    1    0    0    1    0    0    0    0    0    0    0    0    0    0    0
## 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215
##    0    0    0    1    0    0    1    0    0    0    1    0    0    0    0
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0
## 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260
##    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
## 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
## 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290
##    0    0    0    0    0    0    0    0    0    0    0    1    0    0    0
## 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    1
## 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380
##    0    0    0    0    0    1    0    0    0    0    1    1    0    0    0
## 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
```

```
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1471 1472 1473 1474 1475 1476 1477 1478 1479
##    0    0    0    0    0    0    0    0    0
## Levels: 0 1

Test_Cart<- cbind(Test_Cart,pred_test_Cart)

# Predicting the probability on Test Data
Test_Cart$probs <- predict(Model_Train_Cart,Test_Cart,type = "prob")[,2]

tbl_test<- table(Test_Cart$Personal.Loan,Test_Cart$pred_test_Cart)


head(Test_Cart,n=5)

##   Experience..in.years. Income..in.K.month. ZIP.Code Family.members CCAvg
## 1                    19                  34    90089              3   1.5
## 2                    27                  72    91711              2   1.5
## 3                    23                 114    93106              2   3.8
## 4                    41                 112    91741              1   2.0
## 5                    30                  22    95054              1   1.5
##   Education Mortgage Personal.Loan Securities.Account CD.Account Online
## 1         1        0             0                  1          0      0
## 2         2        0             0                  0          0      1
## 3         3        0             0                  1          0      0
## 4         1        0             0                  1          0      0
## 5         3        0             0                  0          0      1
##   CreditCard pred_test_Cart       probs
## 1          0              0 0.003888025
## 2          0              0 0.003888025
## 3          0              1 0.955102041
## 4          0              0 0.003888025
## 5          1              0 0.003888025

print((tbl_test[1,2]+tbl_test[2,1])/nrow(Test_Cart))

## [1] 0.02163624

## Create Confusion matrix on the above prediction

caret::confusionMatrix(Test_Cart$pred_test_Cart,Test_Cart$Personal.Loan)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1327   23
##          1    9  120
##
##                Accuracy : 0.9784
```

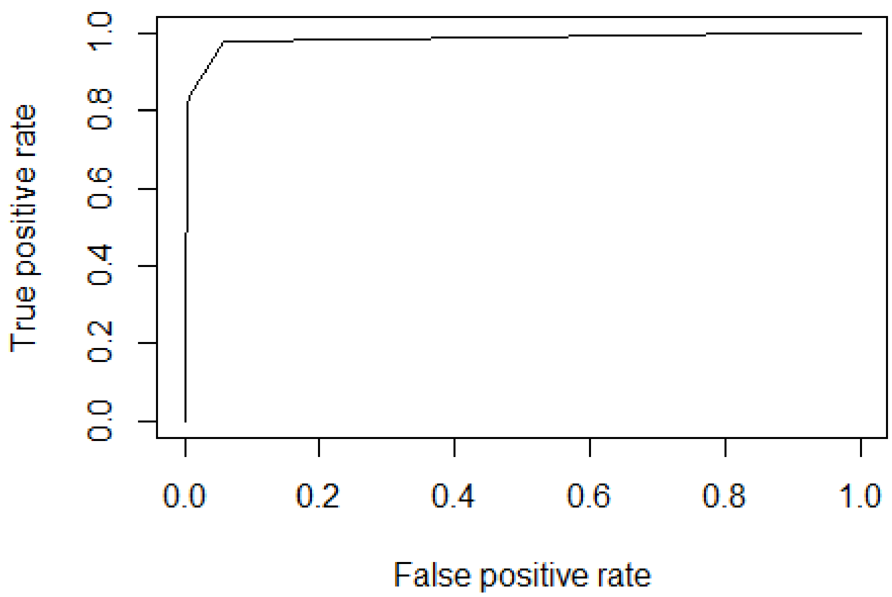
```
##                95% CI : (0.9696, 0.9852)
##     No Information Rate : 0.9033
##     P-Value [Acc > NIR] : < 2e-16
##
##                  Kappa : 0.8705
##
##  Mcnemar's Test P-Value : 0.02156
##
##            Sensitivity : 0.9933
##            Specificity : 0.8392
##         Pos Pred Value : 0.9830
##         Neg Pred Value : 0.9302
##             Prevalence : 0.9033
##         Detection Rate : 0.8972
##   Detection Prevalence : 0.9128
##      Balanced Accuracy : 0.9162
##
##       'Positive' Class : 0
##
```

```
# Preparing the Rank Table.
Test_Cart$Decile <- cut(Test_Cart$probs,unique(qs_train),include.lowest = TRUE, right =
FALSE)
TestDT_CART <- data.table(Test_Cart)

TestRanktbl_Cart <- TestDT_CART[,list(
  count <- length(Personal.Loan),
  Count_One <- sum(Personal.Loan ==1),
  Count_Zero <- sum(Personal.Loan == 0)
), by = Decile][order(-Decile)]

TestRanktbl_Cart
```

```
##              Decile   V1  V2   V3
## 1:       [0.169,1]  218 140   78
## 2: [0.00389,0.169) 1092   3 1089
## 3:     [0,0.00389)  169   0  169
```

```
names(TestRanktbl_Cart) <- c("Decile","Count","Count_One","Count_Zero")

TestRanktbl_Cart$rrate <- round((TestRanktbl_Cart$Count_One/TestRanktbl_Cart$Count),4)*1
00
TestRanktbl_Cart$cum_res <- cumsum(TestRanktbl_Cart$Count_One)
TestRanktbl_Cart$cum_non_res <- cumsum((TestRanktbl_Cart$Count_Zero))
TestRanktbl_Cart$cum_rel_res <- round(TestRanktbl_Cart$cum_res/sum(TestRanktbl_Cart$Coun
t_One),4)*100
TestRanktbl_Cart$cum_rel_non_res <- round(TestRanktbl_Cart$cum_non_res/sum(TestRanktbl_C
art$Count_Zero),4)*100
TestRanktbl_Cart$ks <- abs(TestRanktbl_Cart$cum_rel_res - TestRanktbl_Cart$cum_rel_non_r
es)

# Plotting the ROC Curve

predobj_test_cart <- prediction(Test_Cart$probs, Test_Cart$Personal.Loan)
```

```r
perf_test_cart <- performance(predobj_test_cart,"tpr","fpr")
plot(perf_test_cart)
```




```r
# Calculating KS from the Plot

KS_Test_cart <- max(perf_test_cart@y.values[[1]] - perf_test_cart@x.values[[1]])
KS_Test_cart

## [1] 0.9206377

# Calculating AUC

auc_test_Cart <- performance(predobj_test_cart,"auc")
auc_test_Cart

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9840459
##
```

```
##
## Slot "alpha.values":
## list()
```

```
#Calculating GINI on Test Cart.

gini_test_cart <- ineq(Test_Cart$probs,"Gini")
gini_test_cart
```

```
## [1] 0.8701375
```

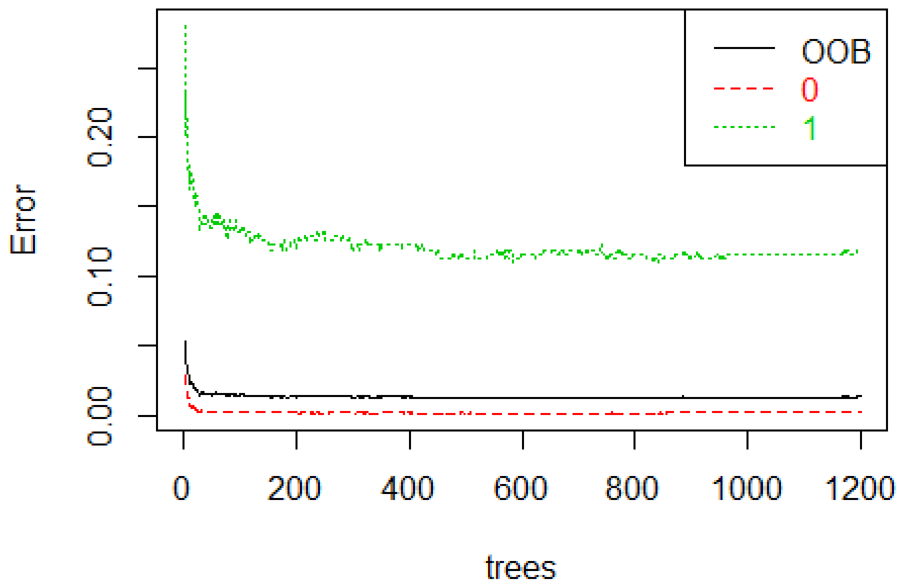
```
# Calculating Concordence on Test Cart

Concordance(actuals = Test_Cart$Personal.Loan, predictedScores = Test_Cart$probs)
```

```
## $Concordance
## [1] 0.9704158
##
## $Discordance
## [1] 0.02958419
##
## $Tied
## [1] -3.469447e-17
##
## $Pairs
## [1] 191048
```

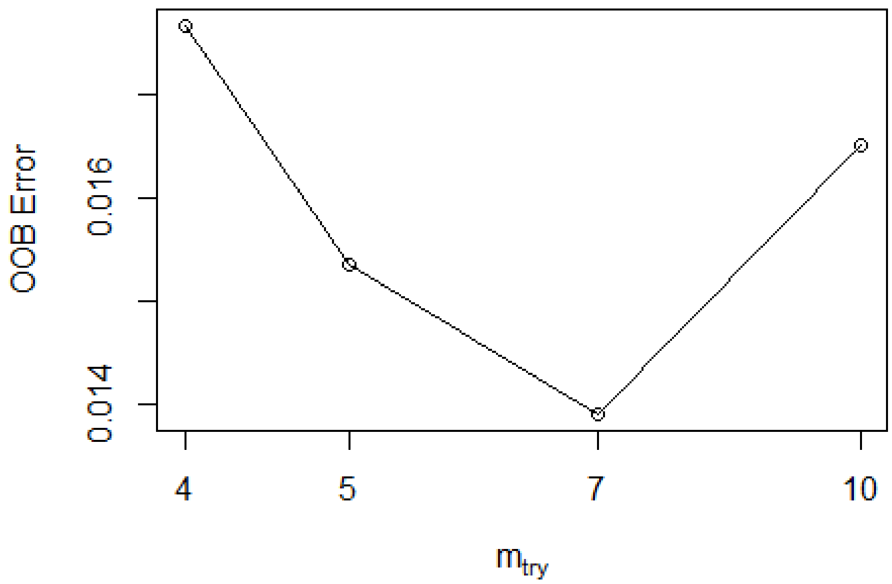
```
###### Building RF Model on the Dataset.

set.seed(1000)
index <- sample.split(theraData$Personal.Loan,SplitRatio = 0.7)

Train_RF <- subset(theraData, index ==TRUE)
Test_RF <- subset(theraData, index ==F)

model_train_rf <- randomForest(Personal.Loan~.,data = Train_RF,type = "class", mtry = 3,
                          nodesize = 10, ntree= 1200, importance = TRUE)
model_train_rf
```

```
##
## Call:
##  randomForest(formula = Personal.Loan ~ ., data = Train_RF, type = "class",      mtry
 = 3, nodesize = 10, ntree = 1200, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1200
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.36%
## Confusion matrix:
##      0   1 class.error
## 0 3109   7  0.00224647
## 1   40 295  0.11940299
```

```
plot(model_train_rf, main = "")
legend("topright", c("OOB","0","1"),text.col = 1:6,lty = 1:3,col = 1:3)
```




```
names(Train_RF)

##  [1] "Experience..in.years." "Income..in.K.month."
##  [3] "ZIP.Code"              "Family.members"
##  [5] "CCAvg"                 "Education"
##  [7] "Mortgage"              "Personal.Loan"
##  [9] "Securities.Account"    "CD.Account"
## [11] "Online"                "CreditCard"

trf<- tuneRF(x = Train_RF[,-8],
             y = Train_RF$Personal.Loan,
             mtryStart = 5,
             ntreeTry = 1200,
             stepFactor = 1.5,
             improve = 0.0001,
             trace = TRUE,
             plot = TRUE,
             doBest = FALSE,
             importance = TRUE,
             nodesize = 50)

## mtry = 5  OOB error = 1.54%
## Searching left ...
## mtry = 4     OOB error = 1.77%
## -0.1509434 1e-04
## Searching right ...
## mtry = 7     OOB error = 1.39%
## 0.09433962 1e-04
## mtry = 10    OOB error = 1.65%
## -0.1875 1e-04
```

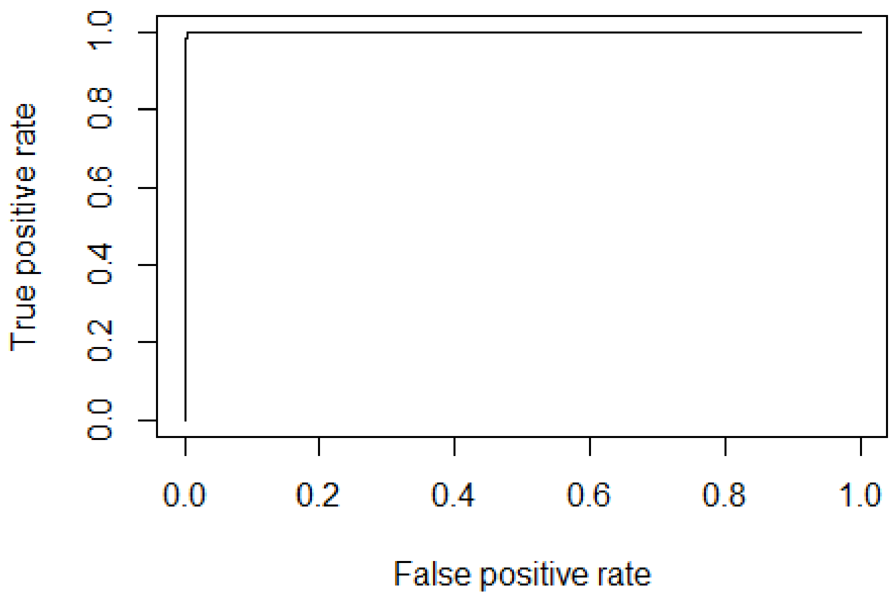



```
trf

##        mtry   OOBError
## 4.OOB     4 0.01767604
## 5.OOB     5 0.01535787
## 7.OOB     7 0.01390901
## 10.OOB   10 0.01651695

model_train_rf1 <- randomForest(Personal.Loan~.,data = Train_RF,type = "class", mtry = 7
,
                               nodesize = 10, ntree= 800, importance = TRUE)

# Predicting the RF model on Train dataset
pred_train_rf <- predict(model_train_rf,Train_RF,type = "class")

Train_RF <- cbind(Train_RF,pred_train_rf)
Train_RF$probs_rf <- predict(model_train_rf, Train_RF, type = "prob")[,2]


## Create Confusion matrix on the above prediction

caret::confusionMatrix(Train_RF$pred_train_rf,Train_RF$Personal.Loan)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3114   22
##          1    2  313
##
##                Accuracy : 0.993
##                  95% CI : (0.9897, 0.9955)
```

```
##     No Information Rate : 0.9029
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9592
##
##  Mcnemar's Test P-Value : 0.0001052
##
##             Sensitivity : 0.9994
##             Specificity : 0.9343
##          Pos Pred Value : 0.9930
##          Neg Pred Value : 0.9937
##              Prevalence : 0.9029
##          Detection Rate : 0.9023
##    Detection Prevalence : 0.9087
##       Balanced Accuracy : 0.9668
##
##        'Positive' Class : 0
##

# Creating the Rank Table for RF Model on Train Dataset

Train_RF$Decile_RF <- cut(Train_RF$probs_rf,unique(qs_train),include.lowest = TRUE,right
 = FALSE)

table(Train_RF$Decile_RF)

##
##     [0,0.00389) [0.00389,0.169)       [0.169,1]
##            2292             786             373

TrainDT_RF <- data.table(Train_RF)

TrainRanktbl_RF <- TrainDT_RF[,list(
  count <- length(Personal.Loan),
  count_One <- sum(Personal.Loan == 1),
  count_zero <- sum(Personal.Loan == 0)
),by = Decile_RF][order(-Decile_RF)]


names(TrainRanktbl_RF) <- c("Decile_RF", "Count","Count_One","Count_Zero")

TrainRanktbl_RF$rrate <- round((TrainRanktbl_RF$Count_One/TrainRanktbl_RF$Count),4)*100
TrainRanktbl_RF$cum_res <- cumsum(TrainRanktbl_RF$Count_One)
TrainRanktbl_RF$cum_non_res <- cumsum(TrainRanktbl_RF$Count_Zero)
TrainRanktbl_RF$cum_rel_res <- round((TrainRanktbl_RF$cum_res/sum(TrainRanktbl_RF$cum_re
s)),4)*100
TrainRanktbl_RF$cum_rel_non_res <- round((TrainRanktbl_RF$cum_non_res/sum(TrainRanktbl_R
F$cum_non_res)),4)*100
TrainRanktbl_RF$ks <- abs(TrainRanktbl_RF$cum_rel_res - TrainRanktbl_RF$cum_rel_non_res)
TrainRanktbl_RF

##          Decile_RF Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:       [0.169,1]   373       335         38 89.81     335          38
## 2: [0.00389,0.169)   786         0        786  0.00     335         824
## 3:     [0,0.00389)  2292         0       2292  0.00     335        3116
```

```
##    cum_rel_res cum_rel_non_res    ks
## 1:       33.33            0.96 32.37
## 2:       33.33           20.71 12.62
## 3:       33.33           78.33 45.00
```

```
# Ploting the ROC Curve
predObj_train_RF <- prediction(Train_RF$probs_rf,Train_RF$Personal.Loan)
perf_Train_RF <- performance(predObj_train_RF, "tpr","fpr")
plot(perf_Train_RF)
```




```
# Calculating the KS value on Train

KS_Train_RF <- max(perf_Train_RF@y.values[[1]] - perf_Train_RF@x.values[[1]])
KS_Train_RF

## [1] 0.9951861

#Calculating the AUC for Train in RF.

auc_train_rf <- performance(predObj_train_RF,"auc")
auc_train_rf

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
```

```
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9998836
##
##
## Slot "alpha.values":
## list()
```

```
# Calculating the GINI

gini_train_rf <- ineq(Train_RF$probs_rf,"Gini")
gini_train_rf
```

```
## [1] 0.8886633
```

```
# Calculating Concordence

Concordance(actuals = Train_RF$Personal.Loan, predictedScores = Train_RF$probs_rf)
```

```
## $Concordance
## [1] 0.9998831
##
## $Discordance
## [1] 0.0001168739
##
## $Tied
## [1] -1.568027e-17
##
## $Pairs
## [1] 1043860
```

```
#Validating the RF Model on Test Data
pred_test_RF <- predict(model_train_rf,newdata = Test_RF, type = "class")
pred_test_RF
```

```
##    1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
##    0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##   16  17  18  19  20  21  22  23  24  25  26  27  28  29  30
##    0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
##   31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
##    0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
##   46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   61  62  63  64  65  66  67  68  69  70  71  72  73  74  75
##    1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##    0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
##   91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
##    0   1   0   0   0   0   0   0   0   0   0   0   0   0   1
##  106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##  121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
##    0   0   0   1   0   0   0   0   1   0   0   0   0   0   0
```

```
##  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150 
##    0    0    0    1    0    0    1    0    0    0    1    0    0    0    0 
##  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165 
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0 
##  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195 
##    0    0    1    0    0    0    0    0    0    0    0    0    1    0    0 
##  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210 
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0 
##  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225 
##    0    0    0    0    0    1    0    1    0    0    0    0    1    1    0 
##  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    1 
##  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285 
##    0    0    0    0    0    1    0    0    0    1    0    0    0    0    0 
##  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300 
##    0    1    0    0    0    1    0    0    0    0    0    0    0    0    0 
##  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315 
##    0    0    0    1    0    0    0    1    0    0    0    0    0    0    0 
##  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330 
##    0    0    0    0    0    0    1    0    0    0    1    0    0    0    0 
##  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345 
##    0    0    0    0    0    0    1    0    0    0    0    1    1    0    0 
##  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360 
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0 
##  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405 
##    1    0    0    0    0    0    0    1    0    0    0    1    0    0    0 
##  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420 
##    0    0    1    0    0    0    0    0    0    0    0    0    1    0    0 
##  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450 
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0 
##  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465 
##    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0 
##  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480 
##    0    0    0    0    0    0    0    1    0    0    0    0    0    1    0 
##  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510 
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525 
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0 
##  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540 
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    0 
```

```
##  541  542  543  544  545  546  547  548  549  550  551  552  553  554  555 
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    1 
##  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570 
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0 
##  571  572  573  574  575  576  577  578  579  580  581  582  583  584  585 
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0 
##  586  587  588  589  590  591  592  593  594  595  596  597  598  599  600 
##    0    0    0    0    0    0    0    0    0    1    0    1    0    0    0 
##  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615 
##    0    0    0    0    0    1    0    0    0    0    1    0    0    0    0 
##  616  617  618  619  620  621  622  623  624  625  626  627  628  629  630 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  631  632  633  634  635  636  637  638  639  640  641  642  643  644  645 
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660 
##    1    0    0    0    0    1    0    0    0    1    0    0    0    0    0 
##  661  662  663  664  665  666  667  668  669  670  671  672  673  674  675 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690 
##    0    0    0    0    0    0    0    0    1    0    0    0    0    0    1 
##  691  692  693  694  695  696  697  698  699  700  701  702  703  704  705 
##    0    0    0    0    0    0    0    1    0    0    0    0    0    1    0 
##  706  707  708  709  710  711  712  713  714  715  716  717  718  719  720 
##    0    0    0    0    1    1    0    0    0    0    0    0    0    0    0 
##  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735 
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750 
##    0    0    0    0    0    0    0    0    1    1    0    0    0    0    0 
##  751  752  753  754  755  756  757  758  759  760  761  762  763  764  765 
##    0    1    0    0    0    0    0    0    0    0    1    0    0    0    0 
##  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780 
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  781  782  783  784  785  786  787  788  789  790  791  792  793  794  795 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  796  797  798  799  800  801  802  803  804  805  806  807  808  809  810 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  811  812  813  814  815  816  817  818  819  820  821  822  823  824  825 
##    1    0    0    1    0    0    0    0    0    0    0    1    0    1    0 
##  826  827  828  829  830  831  832  833  834  835  836  837  838  839  840 
##    0    0    0    0    0    0    0    0    0    1    0    0    0    0    1 
##  841  842  843  844  845  846  847  848  849  850  851  852  853  854  855 
##    1    1    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  856  857  858  859  860  861  862  863  864  865  866  867  868  869  870 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
##  871  872  873  874  875  876  877  878  879  880  881  882  883  884  885 
##    0    0    0    1    0    0    0    1    1    0    0    0    0    0    1 
##  886  887  888  889  890  891  892  893  894  895  896  897  898  899  900 
##    0    0    0    1    0    0    1    0    0    0    0    0    0    0    0 
##  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915 
##    0    0    1    1    0    0    0    0    1    0    0    0    0    0    1 
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930 
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    1 
##  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

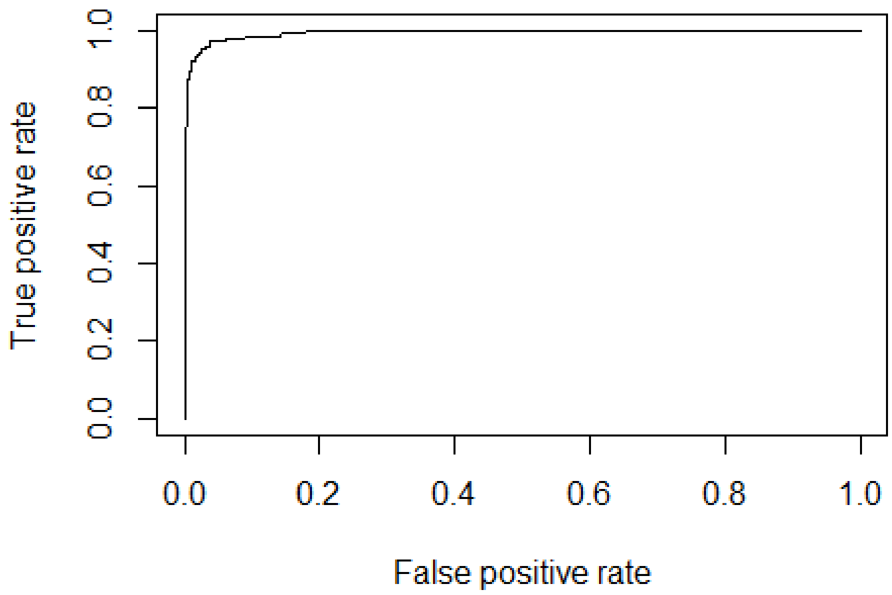
```
##  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960 
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0 
##  961  962  963  964  965  966  967  968  969  970  971  972  973  974  975 
##    0    0    0    0    0    1    0    0    0    0    0    0    0    0    1 
##  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990 
##    0    1    0    1    0    0    0    0    1    0    0    0    0    0    0 
##  991  992  993  994  995  996  997  998  999 1000 1001 1002 1003 1004 1005 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 
##    0    0    0    0    0    0    0    0    1    0    1    0    0    0    0 
## 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 
##    0    1    0    0    0    0    0    0    0    1    1    0    0    0    0 
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0 
## 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 
##    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0 
## 1111 1112 1113 1114 1115 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1141 1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 
##    0    0    1    0    1    1    1    0    0    0    0    0    0    0    0 
## 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170 
##    0    0    0    0    0    0    1    1    0    0    0    0    1    0    0 
## 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 
##    1    0    0    1    0    0    0    0    0    0    0    0    0    0    0 
## 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 
##    0    0    0    1    0    0    1    0    0    0    1    0    0    0    0 
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 
##    0    0    0    0    0    0    0    0    0    0    1    0    0    0    0 
## 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 
##    0    0    0    0    0    0    1    0    0    0    0    0    1    0    0 
## 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 
##    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0 
## 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 
##    0    0    1    0    0    0    0    0    0    0    0    1    0    0    0 
## 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    1    1 
## 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 
##    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0 
## 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
## 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0 
```

```
## 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365
##    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380
##    0    0    0    0    0    1    0    0    0    0    1    1    0    0    0
## 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 1471 1472 1473 1474 1475 1476 1477 1478 1479
##    0    0    0    0    0    0    0    0    0
## Levels: 0 1
```

```
Test_RF<- cbind(Test_RF,pred_test_RF)

Test_RF$probs_test_rf <- predict(model_train_rf, newdata = Test_RF, type = "prob")[,2]


## Create Confusion matrix on the above prediction

caret::confusionMatrix(Test_RF$pred_test_RF,Test_RF$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1329   19
##          1    7  124
##
##                Accuracy : 0.9824
##                  95% CI : (0.9743, 0.9885)
##     No Information Rate : 0.9033
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8954
##
##  Mcnemar's Test P-Value : 0.03098
##
##             Sensitivity : 0.9948
##             Specificity : 0.8671
##          Pos Pred Value : 0.9859
##          Neg Pred Value : 0.9466
##              Prevalence : 0.9033
##          Detection Rate : 0.8986
##    Detection Prevalence : 0.9114
##       Balanced Accuracy : 0.9309
##
```

```
##        'Positive' Class : 0
##

#Preparing the rank Table

prob <- seq (0,1, length = 11)
prob

##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

qs_test <- quantile(Test_RF$probs_test_rf)
Test_RF$Decile_Test <- cut(Test_RF$probs_test_rf,unique(qs_test),include.lowest = TRUE,r
ight = FALSE)
TestDS_RF <- data.table(Test_RF)

TestRanktbl_RF <- TestDS_RF[, list(
  count <- length(Personal.Loan),
  count_one <- sum(Personal.Loan == 1),
  count_zero <- sum(Personal.Loan== 0)
  ),by = Decile_Test][order(-Decile_Test)]


names(TestRanktbl_RF) <- make.names(c("Decile_Test","Count","Count_One","Count_Zero"))
TestRanktbl_RF$rrate <- round((TestRanktbl_RF$Count_One/TestRanktbl_RF$Count),4)*100
TestRanktbl_RF$cum_res <- cumsum(TestRanktbl_RF$Count_One)
TestRanktbl_RF$cum_non_res <- cumsum(TestRanktbl_RF$Count_Zero)
TestRanktbl_RF$cum_rel_res <- round((TestRanktbl_RF$cum_res/sum(TestRanktbl_RF$cum_res))
,4)*100
TestRanktbl_RF$cum_rel_non_res <- round((TestRanktbl_RF$cum_non_res/sum(TestRanktbl_RF$c
um_non_res)),4)*100
TestRanktbl_RF$ks <- abs(TestRanktbl_RF$cum_rel_res - TestRanktbl_RF$cum_rel_non_res)
TestRanktbl_RF

##       Decile_Test Count Count_One Count_Zero rrate cum_res cum_non_res
## 1:   [0.02,0.992]   373       142        231 38.07     142         231
## 2: [0.00167,0.02)   418         1        417  0.24     143         648
## 3:    [0,0.00167)   688         0        688  0.00     143        1336
##    cum_rel_res cum_rel_non_res    ks
## 1:       33.18           10.43 22.75
## 2:       33.41           29.26  4.15
## 3:       33.41           60.32 26.91

# Plotting the ROC Curve

predObj_test_RF <- prediction(Test_RF$probs_test_rf,Test_RF$Personal.Loan)
perf_test_RF <- performance(predObj_test_RF,"tpr","fpr")
plot(perf_test_RF)
```




```
# Calculating the FS from ROC Plot  for test RF

KS_Test_RF <- max(perf_test_RF@y.values[[1]] - perf_test_RF@x.values[[1]])
KS_Test_RF

## [1] 0.9346028

# Calculating the AUC on RF Model Test Dataset

auc_test_RF <- performance(predObj_test_RF, "auc")
auc_test_RF

## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.9942946
##
##
## Slot "alpha.values":
## list()
```

```r
# Calculating GINI on RF Model Test dataset

gini_test_rf <- ineq(Test_RF$probs_test_rf,"Gini")
gini_test_rf

## [1] 0.8753027

# Calculating the Concordence on RF Model Test Dataset

Concordance(actuals = Test_RF$Personal.Loan, predictedScores = Test_RF$probs_test_rf)

## $Concordance
## [1] 0.9942528
##
## $Discordance
## [1] 0.005747247
##
## $Tied
## [1] 4.163336e-17
##
## $Pairs
## [1] 191048
```