Mini Project – **Thera Bank - Loan Purchase Modeling**

Project Report

Jyosmitha M

Contents

1. Project Objective

Thera Bank which has a growing customer base. Majority of them are liable customers who has deposits with the bank. There are very few customers who are borrowers of a loan (asset customers). Bank is planning to earn more through interest rate and **focusing on providing loans to customers who are also depositors.**
The objective of the report is to explore the data set (“TheraBank_dataset.csv”) which was collected last year in a campaign conducted by bank in R and to build a best model to build a model that will help bank to identify the potential customers who have a higher probability of purchasing the loan. This report will consist of the following:

- Importing the dataset in R
- Understanding the structure of dataset
- Graphical exploration
- Descriptive statistics
- Insights from the dataset
- Building predictive models (CART and RF)
- Validating the models with various performance measures (Confusion matrix, ROC/AUC)
- Conclusion

2. Assumptions

Assumptions are as below:

- In ML we don’t assume that data is normally distributed. It can be random.
- There will be a few variables which needed to be converted into factors
- A few columns which are not relevant to building the model might also present
- As this data is collected from a survey, there maybe null values and outliers in the data set
- Data collected is not biased to any variable or columns

3. Environment Set up and Data Import

3.1 Install necessary Packages and Invoke Libraries

- library(dplyr)
- library("readxl")
- library("mice")
- library(e1071)
- library(dplyr)
- library(plyr)
- library(DataExplorer)
- library(caTools)
- library(rpart)
- library(rpart.plot)
- library(corrplot)
- library(caret)

- library(randomForest)

3.2 Functions used in R-code:

- md.pattern
- complete.cases
- rpart
- rpart.control
- cbind
- apply
- filter
- with
- randomForest
- tuneRF
- round
- subset, predict,prediction

3.3 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

```
> setwd("C:/Users/ammu/Desktop/Great Lakes/4. Data Mining/Project")
> getwd()
[1] "C:/Users/ammu/Desktop/Great Lakes/4. Data Mining/Project"
```

3.3 Import and Read the Dataset

The given dataset is in .xlsx format in 2nd tab of the sheet. Hence, the command 'read.xlsx,2' is used for importing the file.

```
TheraBank=read_excel("Thera_Bank_dataset.xlsx",2)
```

4. Meta Data:

Column Name	Description
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?

5. Exploratory Data Analysis

5.1 Basic Data Summary:

5.1.1 Dimensions and View:

There are **5000 rows** and **14 columns** in the dataset

```
> View(TheraBank)
> dim(TheraBank)
[1] 5000   14
```

5.1.2 Head and Tail:

First 6 and last 6 records in the dataset

```
> head(TheraBank)
# A tibble: 6 x 14
     ID `Age (in years)` `Experience (in~ `Income (in K/y~ `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
> tail(TheraBank)
# A tibble: 6 x 14
     ID `Age (in years)` `Experience (in~ `Income (in K/y~ `ZIP Code`
  <dbl>            <dbl>            <dbl>            <dbl>      <dbl>
1  4995               64               40               75      94588
2  4996               29                3               40      92697
3  4997               30                4               15      92037
4  4998               63               39               24      93023
5  4999               65               40               49      90034
6  5000               28                4               83      92612
```

5.1.3 Five point Summary:

- **Experience** column has **negative values** which needs to be corrected by taking abs value
- **Possible outliers** are there in Income, CCAVG, Mortgage
- **Null values** are present in Family Members columns
- Family Members, Education, Personal Loan, securities Account, CD Account, Online, Credit Card seems to be **factor variables**

- ID and Zip Code seems to **be irrelevant for analysis** as they won't play a role in classifying the customer group

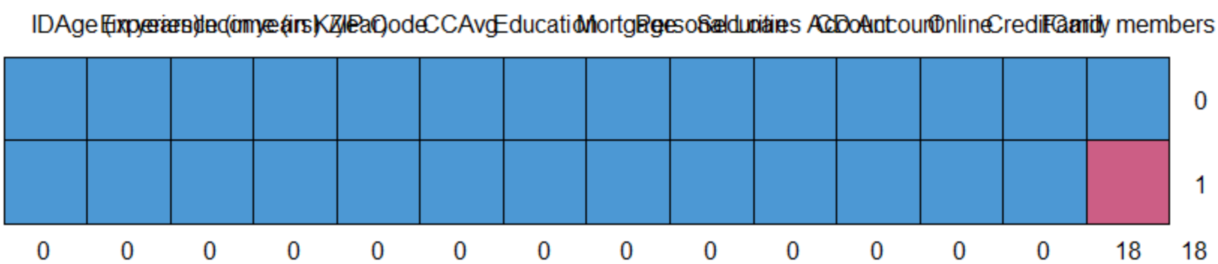
```
> summary(TheraBank)
       ID        Age (in years)   Experience (in years) Income (in K/year)
 Min.   :   1   Min.   :23.00   Min.   :-3.0          Min.   :  8.00    
 1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00    
 Median :2500   Median :45.00   Median :20.0          Median : 64.00    
 Mean   :2500   Mean   :45.34   Mean   :20.1          Mean   : 73.77    
 3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00    
 Max.   :5000   Max.   :67.00   Max.   :43.0          Max.   :224.00    
                                                                        
    ZIP Code      Family members       CCAvg          Education        Mortgage    
 Min.   : 9307   Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0
.0  
 1st Qu.:91911   1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0
.0  
 Median :93437   Median :2.000   Median : 1.500   Median :2.000   Median :  0
.0  
 Mean   :93153   Mean   :2.397   Mean   : 1.938   Mean   :1.881   Mean   : 56
.5  
 3rd Qu.:94608   3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101
.0  
 Max.   :96651   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  
```

5.1.4 Find out nulls

18 Nulls values are present in family members column. The same can be found with function md.pattern pictorially.

```
> anyNA(TheraBank)
[1] TRUE
> sum(is.na(TheraBank))
[1] 18
> sum(rowSums(is.na(TheraBank)))
[1] 18
> sum(colSums(is.na(TheraBank)))
```

```
[1] 18
> md.pattern(TheraBank)
     ID Age (in years) Experience (in years) Income (in K/year) ZIP Code CCAv
g
4982  1              1                     1                  1        1
1
18    1              1                     1                  1        1
1
      0              0                     0                  0        0
0
     Education Mortgage Personal Loan Securities Account CD Account Online
4982         1        1             1                  1          1      1
18           1        1             1                  1          1      1
             0        0             0                  0          0      0
     CreditCard Family members
4982          1              1  0
18            1              0  1
              0             18 18
```



5.1.5 Remove nulls

Complete.cases() function can be used to remove the null values. 18 rows have been removed

```
> #remove nulls
> TheraBankData= TheraBank[complete.cases(TheraBank),]
> anyNA(TheraBankData)
[1] FALSE
```

5.1.6 Column names and re-naming

Column Names have spaces. They must be renamed into syntactically correct.

```
> names(TheraBank)
 [1] "ID"                    "Age (in years)"        "Experience (in years)"
 [4] "Income (in K/year)"    "ZIP Code"              "Family members"
 [7] "CCAvg"                 "Education"             "Mortgage"
[10] "Personal Loan"         "Securities Account"    "CD Account"
[13] "Online"                "CreditCard"
> #rename variables
> names(TheraBankData)=c("ID","Age","Experience","Income","ZipCode","FamilyMe
mbers",
+                          "CCAvg","Education","Mortagage","PersonalLoan","Secu
ritiesAccounts",
+                          "CDAccount","Online","CreditCard")
> names(TheraBankData)
```

```
 [1] "ID"                  "Age"                 "Experience"
 [4] "Income"              "ZipCode"             "FamilyMembers"
 [7] "CCAvg"               "Education"           "Mortagage"
[10] "PersonalLoan"        "SecuritiesAccounts"  "CDAccount"
[13] "Online"              "CreditCard"
```

5.1.7 Attaching the new column names to dataframe

```
> attach(TheraBankData)
```

5.1.8 Correcting the negative values in Experience column

There are 14 negative values in Experience column and all of them are corrected by taking **absolute** values

```
#change the experience negative values
> TheraBankData %>% filter (TheraBankData$Experience <0)
# A tibble: 52 x 14
      ID   Age Experience Income ZipCode FamilyMembers CCAvg Education Mortag
age
   <dbl> <dbl>      <dbl>  <dbl>   <dbl>         <dbl> <dbl>     <dbl>     <d
bl>
 1    90    25         -1    113   94303             4  2.3          3
0
 2   227    24         -1     39   94085             2  1.7          2
0
 3   316    24         -2     51   90630             3  0.3          3
0
 4   452    28         -2     48   94132             2  1.75         3
89
 5   525    24         -1     75   93014             4  0.2          1
0
 6   537    25         -1     43   92173             3  2.4          2
176
 7   541    25         -1    109   94010             4  2.3          3
314
 8   577    25         -1     48   92870             3  0.3          3
0
 9   584    24         -1     38   95045             2  1.7          2
0
10   598    24         -2    125   92835             2  7.2          1
0
# ... with 42 more rows, and 5 more variables: PersonalLoan <dbl>,
#   SecuritiesAccounts <dbl>, CDAccount <dbl>, Online <dbl>, CreditCard <dbl>
> TheraBankData$Experience =abs(TheraBankData$Experience)
> TheraBankData %>% filter (TheraBankData$Experience <0)
# A tibble: 0 x 14
# ... with 14 variables: ID <dbl>, Age <dbl>, Experience <dbl>, Income <dbl>,
#   ZipCode <dbl>, FamilyMembers <dbl>, CCAvg <dbl>, Education <dbl>,
#   Mortagage <dbl>, PersonalLoan <dbl>, SecuritiesAccounts <dbl>,
#   CDAccount <dbl>, Online <dbl>, CreditCard <dbl>
```

5.1.9 Structure of the dataset:

- All the variables in the dataset are of **numeric** datatype.
- Below variables are **not required** for classification
  1. ID
  2. ZIP Code
- Below variables can be **converted into factors**:
  1. FamilyMembers
  2. Education
  3. PersonalLoan
  4. SecuritiesAccount
  5. CDAccount
  6. Online
  7. CreditCard
- As per metadata, education levels are ordered factors given as below:
  Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professiona
  l

```
> str(TheraBank)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':     5000 obs. of  14 variables:
 $ ID                   : num  1 2 3 4 5 6 7 8 9 10 ...
 $ 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/year)   : 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 ...
```

5.1.10 Converting necessary variables into factors:

Converted education in ordered factor and other variables (necessary as factors) and checked the structure and attaching again.

```
#Converting Education into factors
TheraBankData$Education=as.factor(Education)
TheraBankData$Education=revalue(TheraBankData$Education,
            c("1"="Undergrad", "2"="Graduate","3"="Advanced/Professional"))
TheraBankData$Education=factor(TheraBankData$Education,
            levels = c("Undergrad","Graduate","Advanced/Professional")
            ,order= TRUE)
#converting Other variables into factors
TheraBankData$FamilyMembers =as.factor(TheraBankData$FamilyMembers)
TheraBankData$Education=as.factor(TheraBankData$Education)
TheraBankData$PersonalLoan =as.factor(TheraBankData$PersonalLoan )
TheraBankData$SecuritiesAccount =as.factor(TheraBankData$SecuritiesAccount)
TheraBankData$CDAccount =as.factor(TheraBankData$CDAccount)
TheraBankData$Online=as.factor(TheraBankData$Online)
TheraBankData$CreditCard=as.factor(TheraBankData$CreditCard)
```

```
> str(TheraBankData)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':     4982 obs. of  14 variables:
 $ ID                : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Age               : num  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience        : num  1 19 15 9 8 13 27 24 10 9 ...
 $ Income            : num  49 34 11 100 45 29 72 22 81 180 ...
 $ ZipCode           : num  91107 90089 94720 94112 91330 ...
 $ FamilyMembers     : Factor w/ 4 levels "1","2","3","4": 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         : Ord.factor w/ 3 levels "Undergrad"<"Graduate"<..: 1 1 1
2 2 2 2 3 2 3 ...
 $ Mortagage         : num  0 0 0 0 0 155 0 0 104 0 ...
 $ PersonalLoan      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
 $ SecuritiesAccount: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
 $ CDAccount         : 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 ...
> attach(TheraBankData)
```

5.1.11 Scaling the features:

Below features are not on same scale. Hence it is ideal to scale the dataset so that model is build without biasing on any column.

- Age
- Experience
- Income
- CCAvg
- Mortagage

```
#feature sclaing
z=TheraBankData[c(2:4,7,9)]
m=apply(z ,2,mean)
s=apply(z,2,sd)
TheraBankDataScaling=scale(z,m,s)
TheraBankDataScaled=TheraBankData[-c(2:4,7,9)]
TheraBankDataScaled=cbind(TheraBankDataScaled,TheraBankDataScaling)
head(TheraBankDataScaled)
summary(TheraBankDataScaled)
attach(TheraBankDataScaled)
View(TheraBankDataScaled)
```

```
  ID ZipCode FamilyMembers Education PersonalLoan SecuritiesAccount CDAccount
Online CreditCard
1  1   91107             4 Undergrad            0                 1         0
0          0
2  2   90089             3 Undergrad            0                 1         0
0          0
3  3   94720             1 Undergrad            0                 0         0
0          0
4  4   94112             1  Graduate            0                 0         0
0          0
```

```
5  5   91330             4  Graduate            0                  0         0
0          1
6  6   92121             4  Graduate            0                  0         0
1          0
          Age Experience     Income      CCAvg  Mortagage
1 -1.77207692 -1.6744020 -0.5371972 -0.1944232 -0.5557035
2 -0.02852262 -0.0985976 -0.8629999 -0.2516129 -0.5557035
3 -0.55158891 -0.4487763 -1.3625639 -0.5375615 -0.5557035
4 -0.90029977 -0.9740445  0.5705317  0.4346636 -0.5557035
5 -0.90029977 -1.0615892 -0.6240779 -0.5375615 -0.5557035
6 -0.72594434 -0.6238657 -0.9716007 -0.8806998  0.9675427
```

Summary of the scaled data

```
summary(TheraBankData)
       ID          ZipCode      FamilyMembers                Education
 Min.   :   1   Min.   : 9307   1:1464       Undergrad            :2088
 1st Qu.:1254   1st Qu.:91911   2:1292       Graduate             :1399
 Median :2502   Median :93437   3:1009       Advanced/Professional:1495
 Mean   :2502   Mean   :93153   4:1217
 3rd Qu.:3750   3rd Qu.:94608
 Max.   :5000   Max.   :96651
 PersonalLoan SecuritiesAccount CDAccount Online   CreditCard      Age
 0:4504       0:4463            0:4682    0:2013   0:3517     Min.   :-1.94643
 1: 478       1: 519            1: 300    1:2969   1:1465     1st Qu.:-0.90030
                                                              Median :-0.02852
                                                              Mean   : 0.00000
                                                              3rd Qu.: 0.84325
                                                              Max.   : 1.88939
   Experience          Income             CCAvg            Mortagage
 Min.   :-1.76195   Min.   :-1.4277   Min.   :-1.1095   Min.   :-0.5557
 1st Qu.:-0.88650   1st Qu.:-0.7544   1st Qu.:-0.7091   1st Qu.:-0.5557
 Median :-0.01105   Median :-0.2114   Median :-0.2516   Median :-0.5557
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.86439   3rd Qu.: 0.5271   3rd Qu.: 0.3203   3rd Qu.: 0.4369
 Max.   : 2.00247   Max.   : 3.2638   Max.   : 4.6095   Max.   : 5.6847

>
```

5.1.11 Removal of unnecessary variables

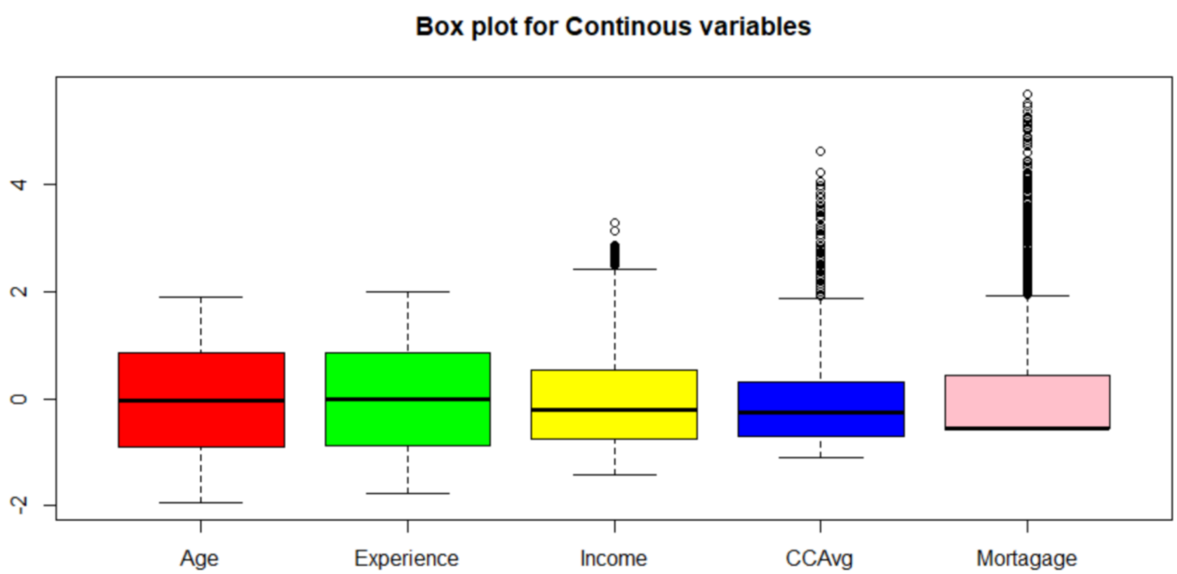
ID and Zip code are not useful for classification. Hence variables are removed from the dataframe

```
> #remove unecessary variables(ID, ZipCode)
> names(TheraBankDataScaled)
 [1] "ID"                "ZipCode"           "FamilyMembers"     "Education"
 [5] "PersonalLoan"      "SecuritiesAccount" "CDAccount"         "Online"
 [9] "CreditCard"        "Age"               "Experience"        "Income"
[13] "CCAvg"             "Mortagage"
> TheraBankDataScaled=TheraBankDataScaled[,-c(1,2)]
> names(TheraBankDataScaled)
 [1] "FamilyMembers"     "Education"         "PersonalLoan"      "SecuritiesA
ccount"
 [5] "CDAccount"         "Online"            "CreditCard"        "Age"
 [9] "Experience"        "Income"            "CCAvg"             "Mortagage"
```

5.1.12 Outliers:

All the other variables are categorical/factor except below variables shown in graph. In these variables we have outliers in **Income, CCAvg, Mortgage.** These outliers **should not be removed/imputed**. As data is collected from general population across all categories, this is expected

```
boxplot(TheraBankDataScaling
        ,main="Box plot for Continous variables"
        ,col=c("Red","Green","Yellow","Blue","Pink"))
```



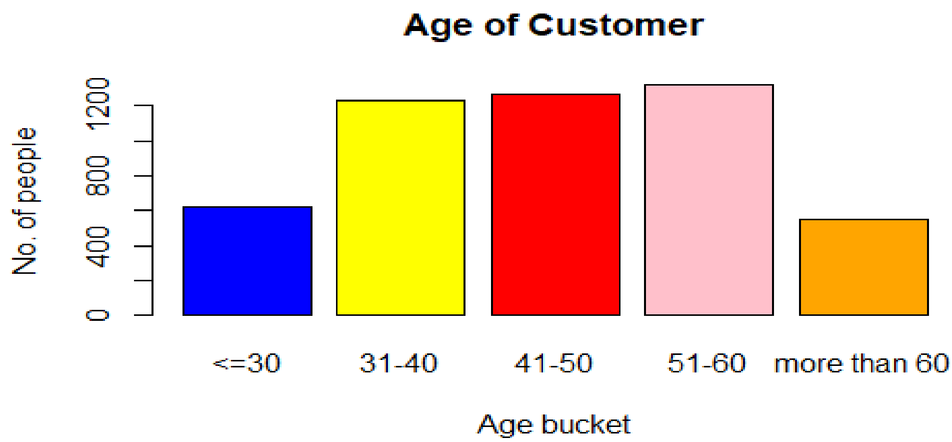
6. Univariate Analysis:

This helps us in understanding the variables characteristics.

6.1 Age:

- max customers are of age 35.
- Majority of the customers are falling in 51-60 age bucket followed by 41-50 and then 31-40.
- Senior citizens are relatively low when compared to other age buckets

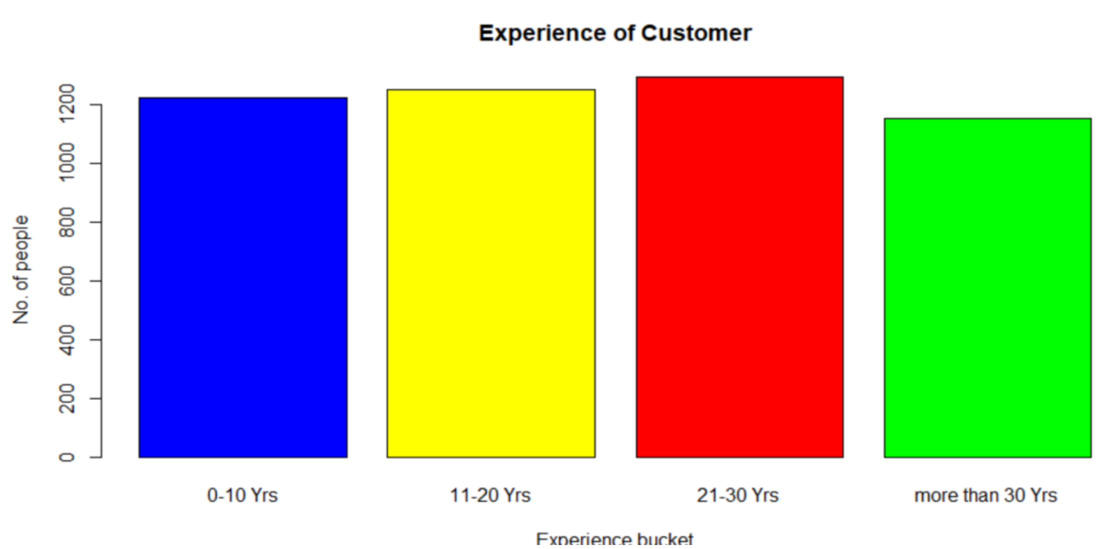
```
> uniqv[which.max(tabulate(match(Age, uniqv)))]
[1] 35
> min(Age)
[1] 23
> max(Age)
[1] 67
```



6.2 Professional Experience:

- 21-30 years' experience bucket has most customer followed by 11-20 and then less than 10 years' experience. More than 30 years of experience bucket is having less customers
- 66 Customers have 0 professional experience

```
> zeroexp=TheraBankData %>% select (Experience)%>%filter (TheraBankData$Exper
ience ==0) %>% count()
> zeroexp
  Experience freq
1          0   66
```

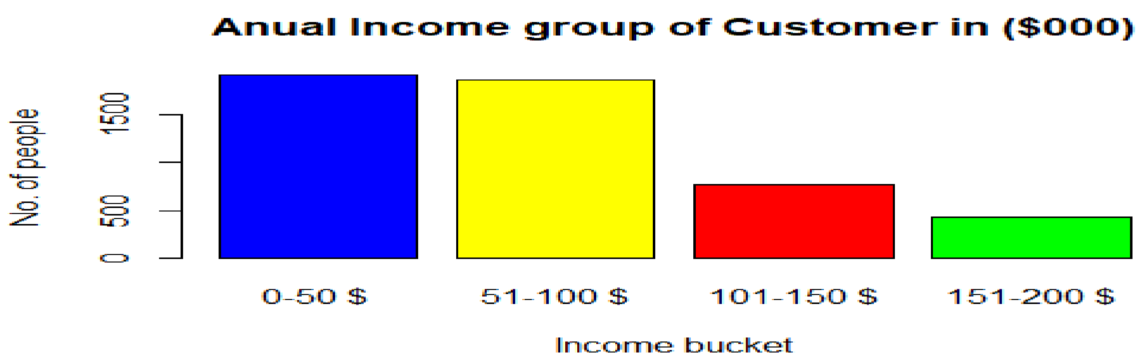


6.3 Average income:

- Most customers are earning an average income of less than 50k dollars per annum. This is followed by next bucker 51000-10000$, and then 101000-150000$. Customers earning more than 100k dollars are relatively low when compared to other buckets.

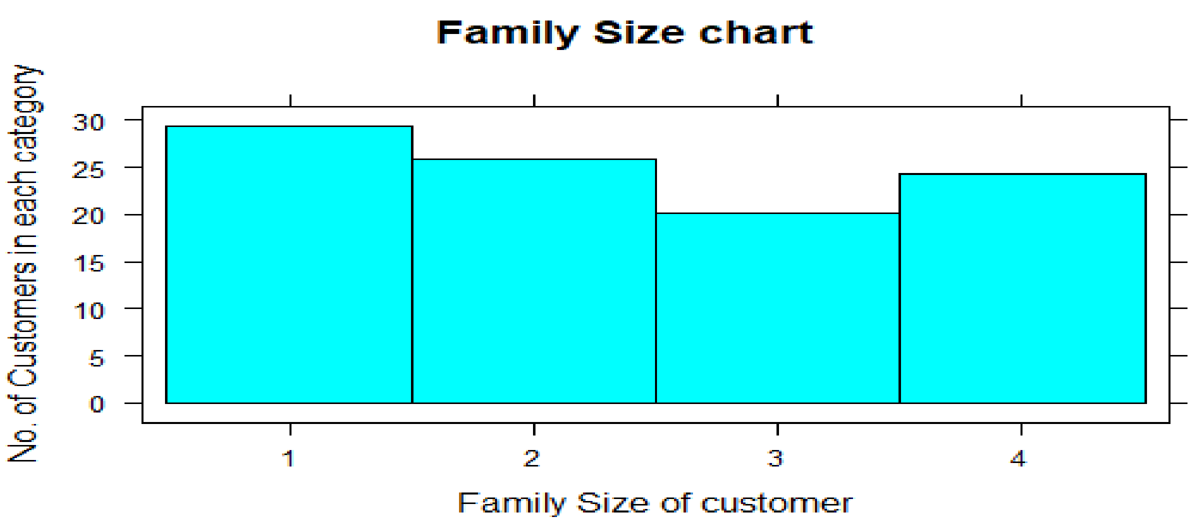
- There are no customers without income

```
> zeroinc=TheraBankData %>% select (Income)%>%filter (TheraBankData$Income ==
0) %>% count()
> zeroinc
[1] Income freq
<0 rows> (or 0-length row.names)
```



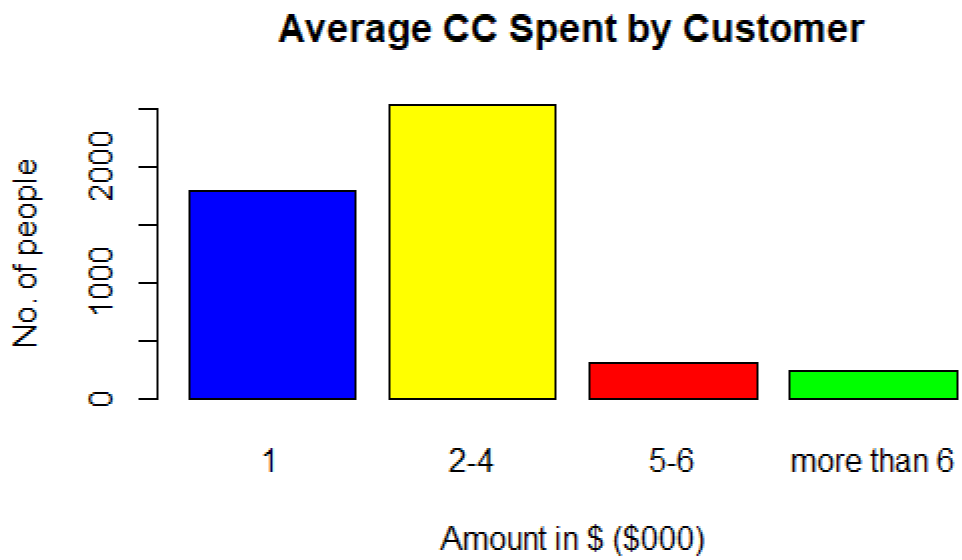
6.4 Number of Family members:

Most of the customers are singles, followed by a family size of 2 and then 4. Family size of 3 is relatively low.



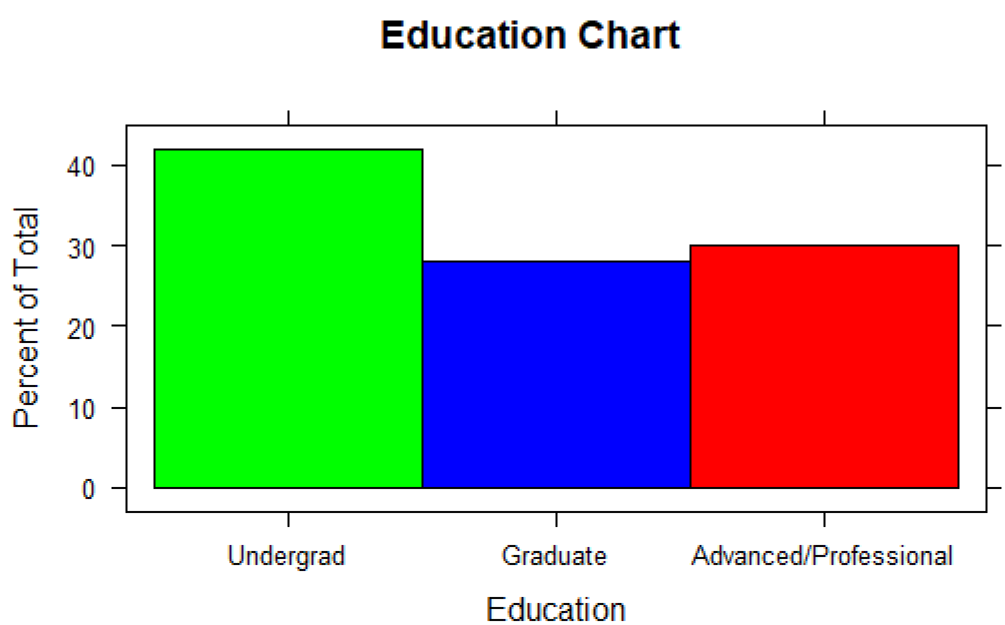
6.5 Average spent of credit card:

Average CC spend by customer in a month is more in 2k-4k bucket. Number of customers spending less 1 is relatively low when compared with top bucker. Customers who spend more than 5k are very low.



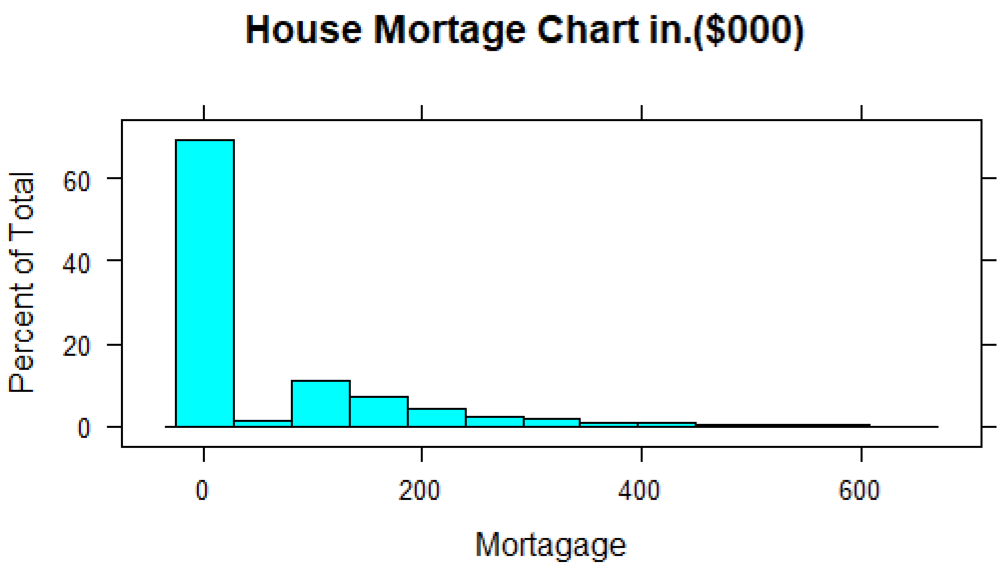
6.6 Education details:

Most of the customers are undergraduates, followed by advanced/Professional and then Graduates.



6.7 House Mortgage:

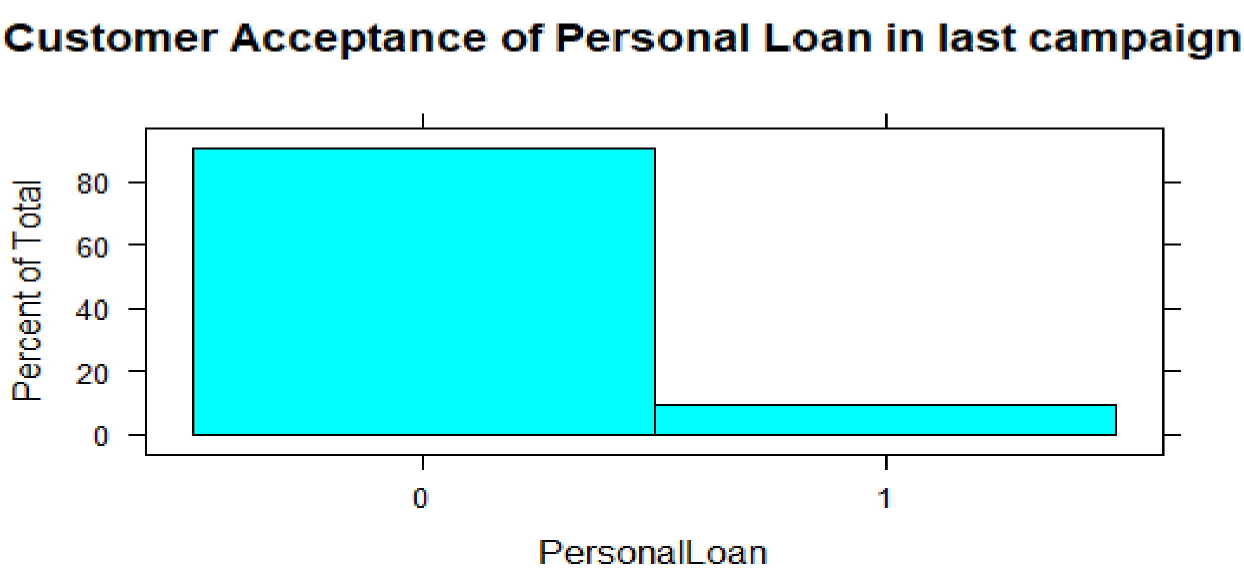
Majority of the customers are not having any mortgage.



6.8 Personal Loan acceptance in the last campaign

4504 customers have not accepted the personal loan offer and 478 customers accepted the personal loan offer

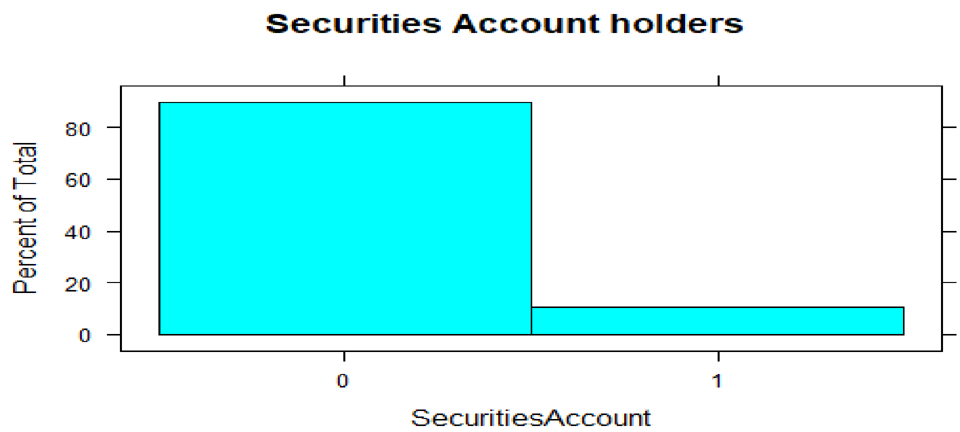
```
> table(PersonalLoan)
PersonalLoan
   0    1
4504  478
```



6.9 Securities Account holders:

Around 4463 customers are not holders of securities account and 519 are holders of the same

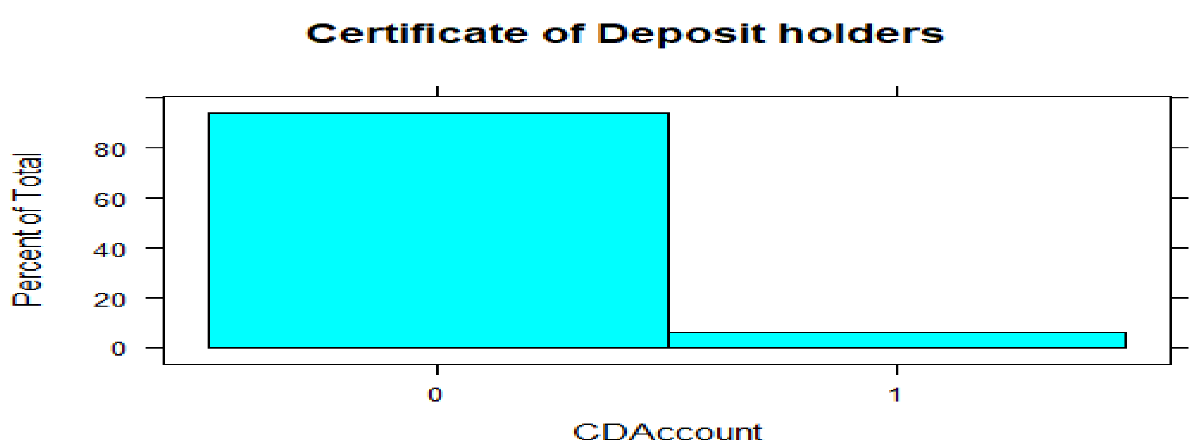
```
> table(SecuritiesAccount)
SecuritiesAccount
   0    1
4463  519
```



6.10 Certificate of deposit holders:

Around 4682 customers are not holders of securities account and 300 are holders of the same

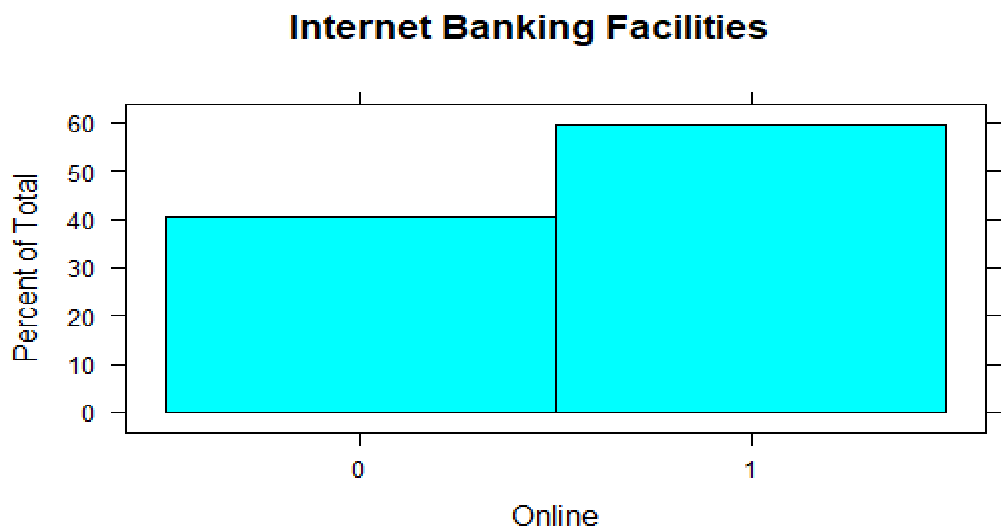
```
table(CDAccount)
CDAccount
   0    1
4682  300
```



6.11 Internet Banking facilities:

Most of the customers opted for internet banking facilities

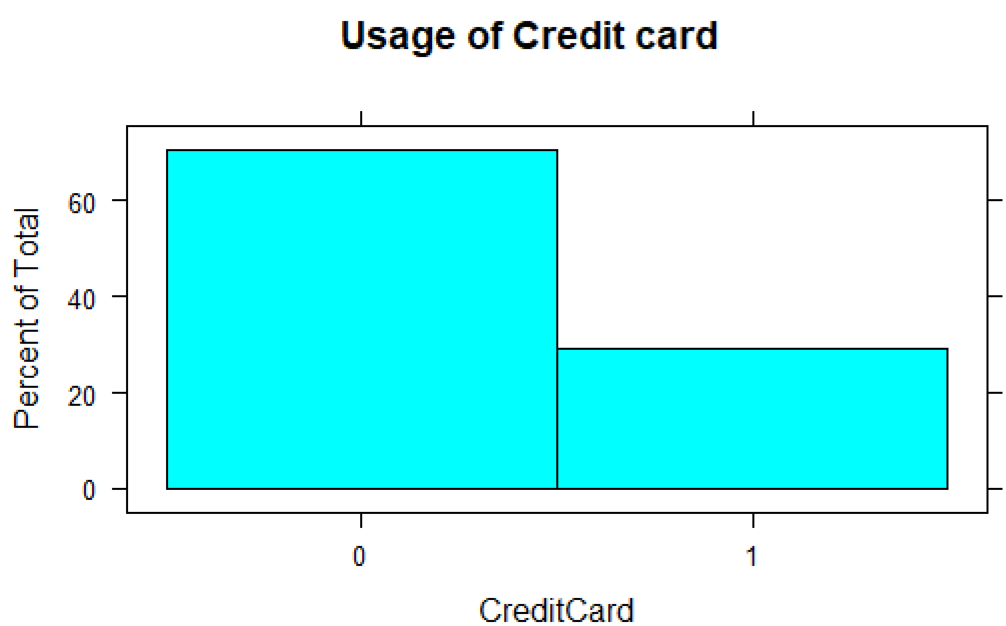
```
> table(Online)
Online
   0    1
2013 2969
```



6.12 Usage of Credit Card:

Less number of customers use the credit cards from the bank. Around 3517 of the don't use and 1465 use the credit cards

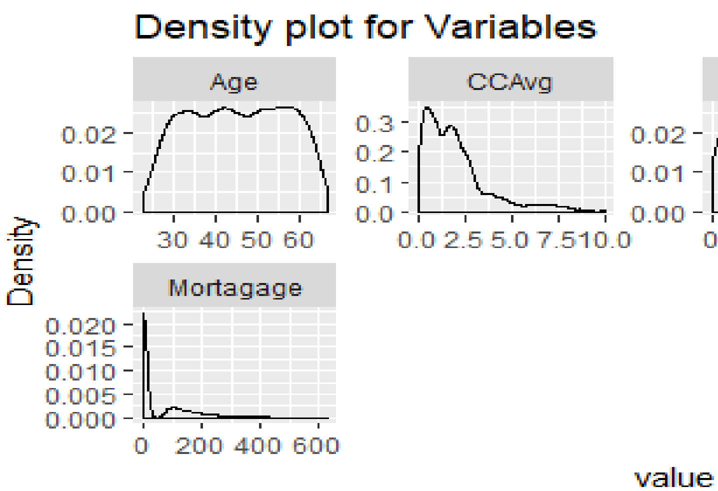
```
> table(CreditCard)
CreditCard
   0    1
3517 1465
```

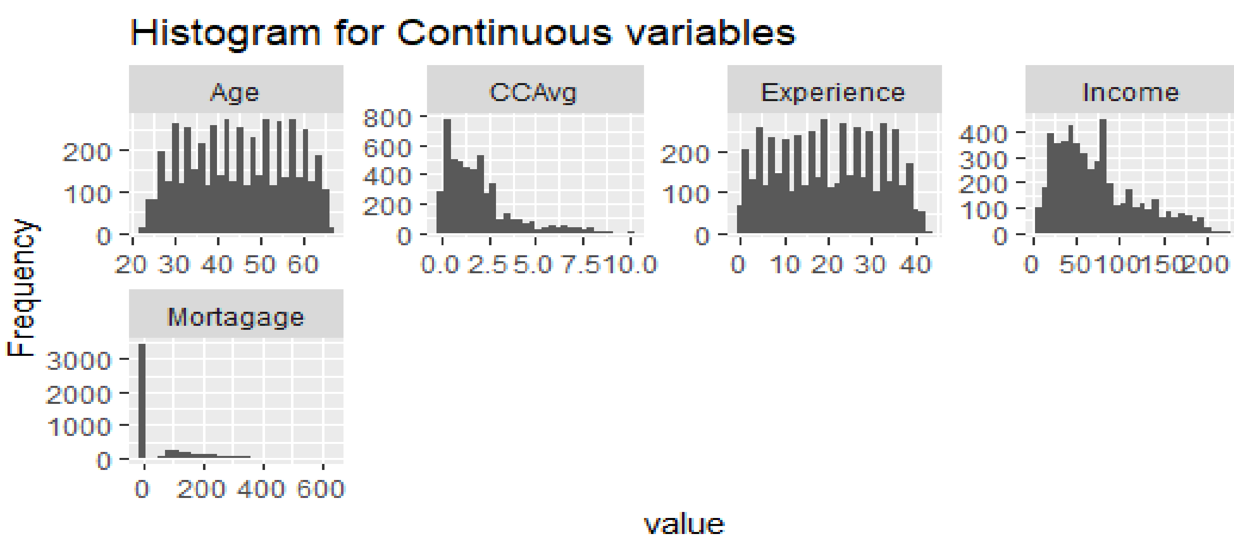


6.13 Density and Histogram plot for Continuous variables:

Normal distribution: Age, Experience

Right Skewed: CCAvg, Mortgage, Income





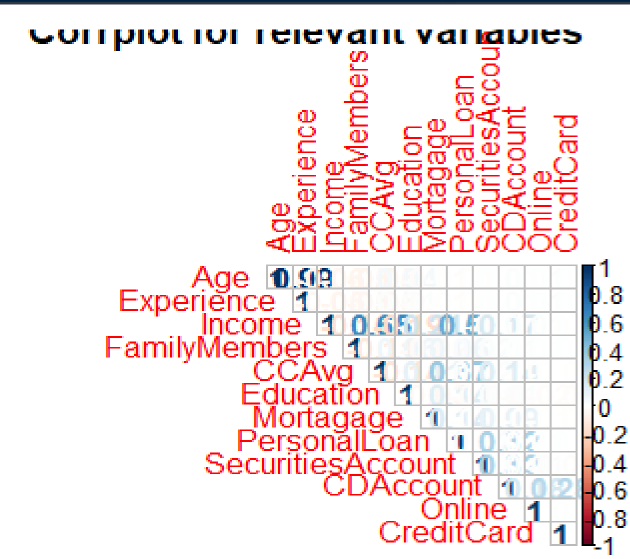
7. Bivariate Analysis:

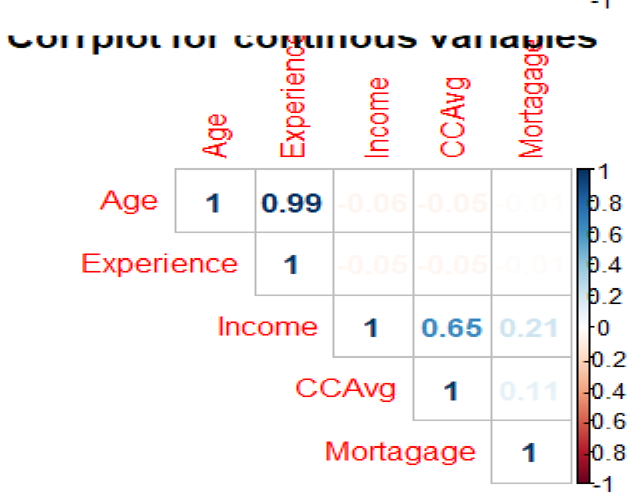
7.1 Corrplot for overall and continuous variables:

There is a good correlation between below variables:

1. Age and Experience
2. Income and CCAvg

```r
#Bivariate Analysis
TheraBankCor= TheraBank[complete.cases(TheraBank),]
names(TheraBankCor)=c("ID","Age","Experience","Income","ZipCode"
                      ,"FamilyMembers","CCAvg","Education","Mortagage"
                      ,"PersonalLoan","SecuritiesAccount",
                      "CDAccount","Online","CreditCard")
TheraBankCor %>% filter(TheraBankCor$Experience <0)
TheraBankCor$Experience =abs(TheraBankCor$Experience)
#remove unecessary variables
TheraBankCor=TheraBankCor[,-c(1,5)]
#corrplot
dev.off()
corrplot(cor(TheraBankCor),method = "number",type = "upper")
#continous variables corrploe
TheraBankCon=TheraBankCor[ ,c(1:3,5,7)]
corrplot(cor(TheraBankCon),method = "number"
         ,type = "upper"
         ,main="Corrplot for continous variables")
```

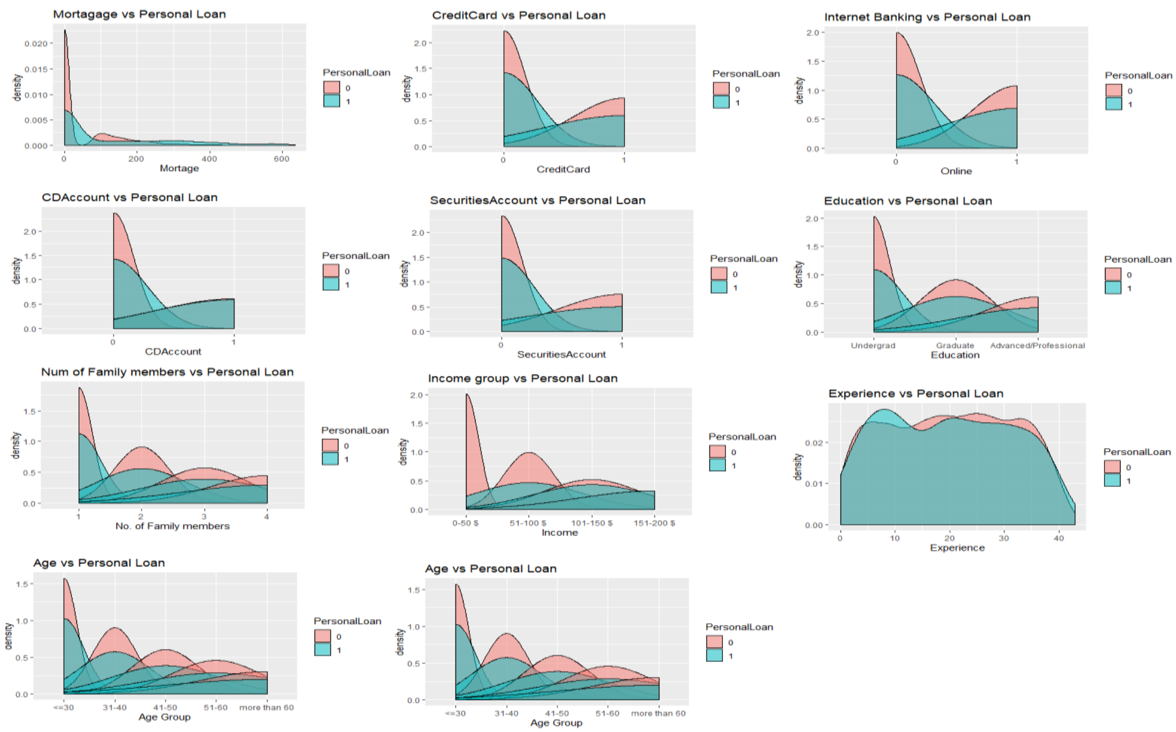


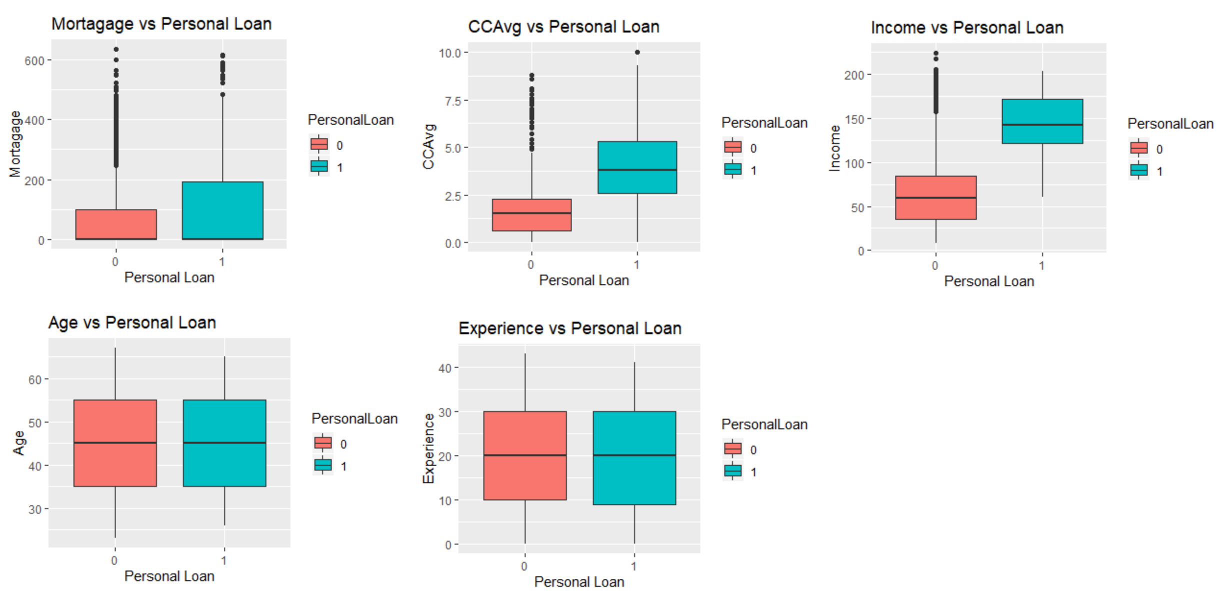


7.2 Personal loan vs Categorical Variables: (Barplot and Density plots, box plots)

From the bar plots of Personal Loan vs Categorical variables in the data set, there are the findings on the bi-variate analysis:

- Customers who opted for **Online internet banking**, have **credit card and have securities account** are having a relation in taking personal loans than the ones who did not opt for the above options.
- We see outliers in **CCAvg** and **mortgage** yet not opted for Personal loan. These Customers can be targeted by providing exciting offers.
- The **Advanced/ Professionals** are interested in taking personal loan for higher studies. Irrespective of educational background, we find few customers still opting for personal loans
- Those customers who have **Annual income** group <50k$~90k$ are not showing any interest in personal loan. However, customers who has Annual income of >100k$ are showing more interest in personal loan and a very few customers between 51k$-100k$ as well. These customers can be targeted as well
- Irrespective of **family size**, and **monthly income** we find customers opting for personal loan. However, family size of 3 and 4 are showing more interest in opting for personal loan.


Mortagage vs Personal Loan
density
Mortage
PersonalLoan
0
1
CreditCard vs Personal Loan
CreditCard
Internet Banking vs Personal Loan
Online
CDAccount vs Personal Loan
CDAccount
SecuritiesAccount vs Personal Loan
SecuritiesAccount
Education vs Personal Loan
Undergrad
Graduate
Advanced/Professional
Education
Num of Family members vs Personal Loan
No. of Family members
Income group vs Personal Loan
0-50 $
51-100 $
101-150 $
151-200 $
Income
Experience vs Personal Loan
Experience
Age vs Personal Loan
<=30
31-40
41-50
51-60
more than 60
Age Group
Age vs Personal Loan


Mortagage vs Personal Loan
Mortagage
Personal Loan
PersonalLoan
0
1
CCAvg vs Personal Loan
CCAvg
Income vs Personal Loan
Income
Age vs Personal Loan
Age
Experience vs Personal Loan
Experience

8. CART Model:

CART stands for classification and regression

8.1 Splitting the data and checking its proportion and base line conversion:

Set the seed to ensure same randomness whenever model runs. Split the data set into train and test 70%-30%. Check the proportion of train and test data is splitting is done on approximately equal proportion

```
#splitting data
set.seed(1234)
splitCart=sample.split(TheraBankDataScaled$PersonalLoan ,SplitRatio=0.7)
trainDataCart=subset(TheraBankDataScaled,splitCart=="TRUE")
testDataCart=subset(TheraBankDataScaled,splitCart=="FALSE")
View(splitCart)
#checking how the many and how much proportion opted personal loan and how many didn't
table(TheraBankDataScaled$PersonalLoan )
round(prop.table(table(TheraBankDataScaled$PersonalLoan)),3)
#checking the proportion of data in the train and test dataset
round(prop.table(table(trainDataCart$PersonalLoan)),3)
round(prop.table(table(testDataCart$PersonalLoan)),3)
```

8.2 Building CART model:

a. Rpart:

Rpart- function for building a CART model. Here we took train data as input and method as “class” because it’s a classification problem.

Rpart.control – hyper tuning parameters

Minbucket-number of items in the bucket. This is usually 1/3 of minsplit

Minsplit- there should not be a any split after this number of splits

Cp- **Cost/complexity parameter**. Lower the CP value, more complex the tree. This controls the size of the tree on how far it can grow. It tells about the minimum improvement that a tree should make when splitting further

```
treeCart=rpart(formula = PersonalLoan~.
           ,data=trainDataCart,method="class"
           ,control=rpart.control(minsplit = 20, minbucket = 4,cp=0))
```

b. Tree construction and splits proportions Interpretation:

Number of observations in root node is 3488. This node has 335 1’s (people who took personal loan) and 0 signifies that there are more number of 0’s in the node. The 2 proportions of 0’s and 1’s is given in the brackets respectively.

The 1st split of root node happened based on Income variable (there must be a significant gain when split was made with this variable) into 2 nodes. Incomes <0.88 and income >=0.88. Rest of the explanation is same as above para.

```
> treeCart
n= 3488

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

  1) root 3488 335 0 (0.903956422 0.096043578)
    2) Income< 0.8854743 2807  63 0 (0.977556110 0.022443890)
      4) CCAvg< 0.5776379 2591   8 0 (0.996912389 0.003087611)
        8) Income< 0.7117129 2509   0 0 (1.000000000 0.000000000) *
        9) Income>=0.7117129 82   8 0 (0.902439024 0.097560976)
         18) Education=Undergrad 32   0 0 (1.000000000 0.000000000) *
         19) Education=Graduate,Advanced/Professional 50   8 0 (0.840000000 0
.160000000)
           38) Age< -1.031066 16   0 0 (1.000000000 0.000000000) *
           39) Age>=-1.031066 34   8 0 (0.764705882 0.235294118)
             78) Experience>=0.995711 12   0 0 (1.000000000 0.000000000) *
             79) Experience< 0.995711 22   8 0 (0.636363636 0.363636364)
              158) CCAvg< -0.05144893 16   4 0 (0.750000000 0.250000000) *
              159) CCAvg>=-0.05144893 6   2 1 (0.333333333 0.666666667) *
      5) CCAvg>=0.5776379 216  55 0 (0.745370370 0.254629630)
       10) CDAccount=0 201  43 0 (0.786069652 0.213930348)
         20) Income< 0.4076304 122  14 0 (0.885245902 0.114754098) *
         21) Income>=0.4076304 79  29 0 (0.632911392 0.367088608)
           42) Education=Undergrad 42   5 0 (0.880952381 0.119047619) *
           43) Education=Graduate,Advanced/Professional 37  13 1 (0.351351351
0.648648649)
             86) FamilyMembers=1,2,4 29  13 1 (0.448275862 0.551724138)
              172) CCAvg< 1.521268 24  11 0 (0.541666667 0.458333333)
                344) CCAvg>=1.378294 4   0 0 (1.000000000 0.000000000) *
                345) CCAvg< 1.378294 20   9 1 (0.450000000 0.550000000)
                  690) Age>=-0.4644112 15   6 0 (0.600000000 0.400000000) *
                  691) Age< -0.4644112 5   0 1 (0.000000000 1.000000000) *
              173) CCAvg>=1.521268 5   0 1 (0.000000000 1.000000000) *
             87) FamilyMembers=3 8   0 1 (0.000000000 1.000000000) *
       11) CDAccount=1 15   3 1 (0.200000000 0.800000000) *
    3) Income>=0.8854743 681 272 0 (0.600587372 0.399412628)
      6) Education=Undergrad 449  45 0 (0.899777283 0.100222717)
       12) FamilyMembers=1,2 404   0 0 (1.000000000 0.000000000) *
       13) FamilyMembers=3,4 45   0 1 (0.000000000 1.000000000) *
      7) Education=Graduate,Advanced/Professional 232   5 1 (0.021551724 0.97
8448276) *
```

c. Cost complexity output Interpretation:

Variables used in tree construction: Below 7 variables qualified the criteria given in the rpart function(minbucket and minsplit) and splits are done basing on these nodes

Root node error: total no.of 1's (who took personal loans)/total number of observations (in train dataset here) in the 1st node before split

N: total number of observations in the 1st node in train dataset (here)

CP Table: Our aim is to reduce the cross-validation error as maximum as possible. Hence, we take cp value against the lowest xerror. Above tree is overfitting and the complete tree is drawn to see how long the tree can grow. It gives the number of splits made and the relative error is decreased (when compared with parent node) and the cross-validation error at each split and the respective standard deviation.

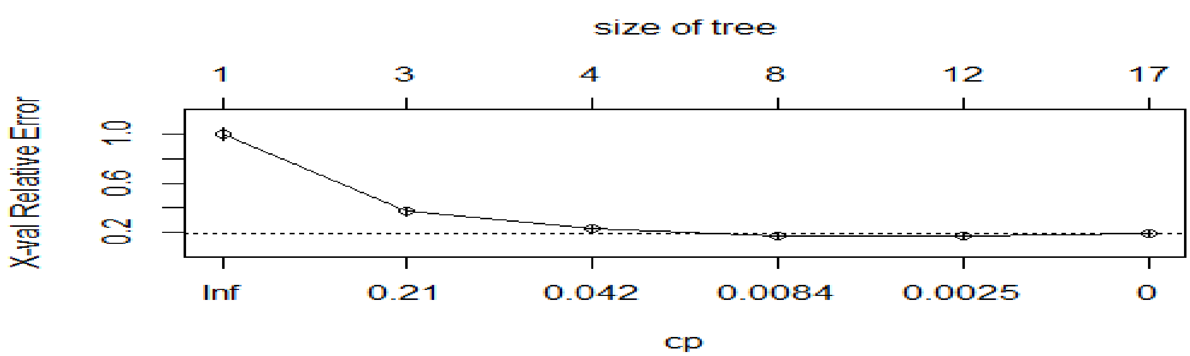
Not all variables are used while constructing the tree.

```
> printcp(treeCart)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = trainDataCart, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 4, cp = 0))

Variables actually used in tree construction:
[1] Age          CCAvg        CDAccount    Education    Experience
[6] FamilyMembers Income

Root node error: 335/3488 = 0.096044

n= 3488

         CP nsplit rel error  xerror     xstd
1 0.3313433      0   1.00000 1.00000 0.051946
2 0.1343284      2   0.33731 0.37015 0.032644
3 0.0134328      3   0.20299 0.23582 0.026230
4 0.0052239      7   0.14328 0.17313 0.022544
5 0.0011940     11   0.12239 0.17612 0.022734
6 0.0000000     16   0.11642 0.19104 0.023660
```

d. **Plotting Cost Complexity:**

```
> plotcp(treeCart)
```

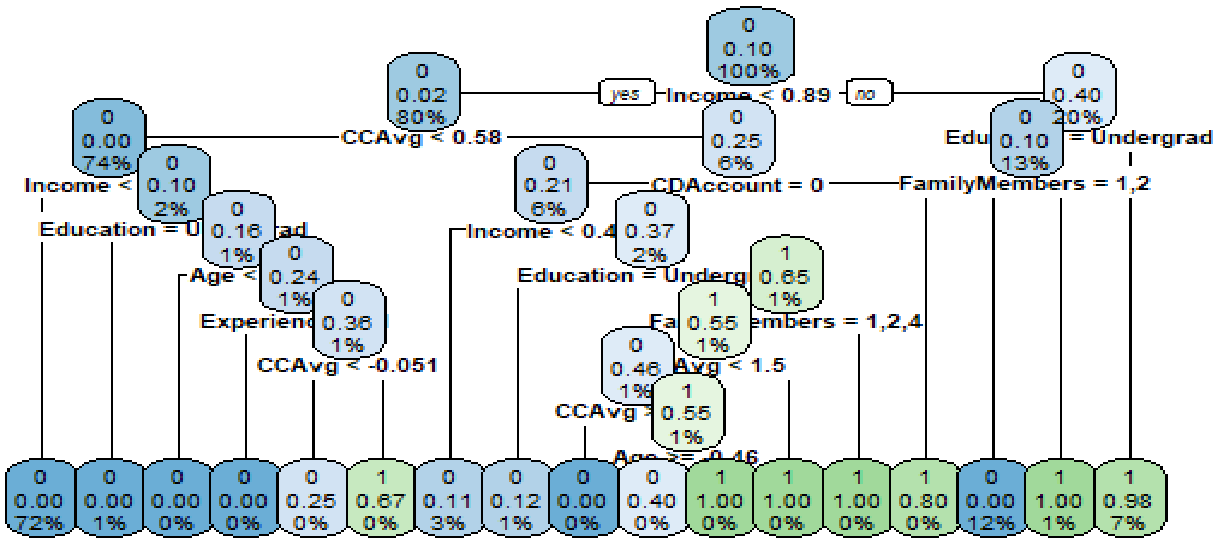
Below plot show the CP value on X-axis and relative error on Y-axis. We see the graph was decreasing almost but there is a slight increase in the xerror at the end.

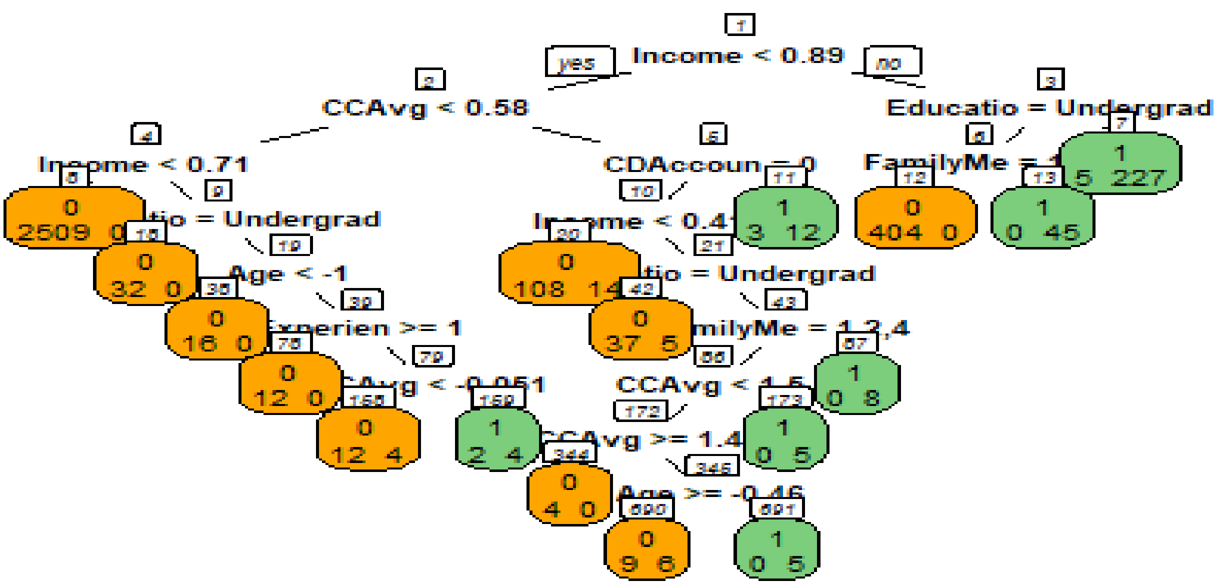


e. **Plotting the tree (unpruned) and plot with prp:**

```
> rpart.plot(treeCart, cex=0.6)
```

The tree was made to grow completely with CP=0. This is overfitting model and captured noise. Since multiple nodes are made, it does not make any sense business wise as it will get difficult to target more number of customer groups.





9. Pruning the tree:

Since the model built with CP=0 is complex and doesn't serve our business purpose, we cut down the tree (prune) it with CP which has minimum xerror. Out objective here is to reduce the error that we make.

```
#pruning cart model
print(treeCart$cptable)
treeCart$cptable[,"xerror"]
min(treeCart$cptable[,"xerror"])
bestcp=treeCart$cptable[which.min(treeCart$cptable[,"xerror"]), "CP"]
bestcp
ptree=prune(treeCart, cp=bestcp)
print(ptree)
rpart.plot(ptree,cex=0.6)
summary(ptree)
path.rpart()
ptree$variable.importance
barplot(ptree$variable.importance)
predict.class=predict(ptree, trainDataCart, type="class")
#predict.class
```

9.2 Taking the best CP value:

From the below table, we see, the **minimum cross-validation error is 0.1731343** and associated cp value is 0.005. After this, the xerror started increasing. Hence **0.005** can be taken as optimum CP value

```
treeCart$cptable
           CP nsplit rel error    xerror       xstd
1 0.331343284      0 1.0000000 1.0000000 0.05194591
2 0.134328358      2 0.3373134 0.3701493 0.03264418
3 0.013432836      3 0.2029851 0.2358209 0.02622974
4 0.005223881      7 0.1432836 0.1731343 0.02254385
5 0.001194030     11 0.1223881 0.1761194 0.02273404
6 0.000000000     16 0.1164179 0.1910448 0.02366049

> treeCart$cptable[,"xerror"]
        1         2         3         4         5         6 
1.0000000 0.3701493 0.2358209 0.1731343 0.1761194 0.1910448 
> min(treeCart$cptable[,"xerror"])
[1] 0.1731343
> bestcp=treeCart$cptable[which.min(treeCart$cptable[,"xerror"]), "CP"]
> bestcp
[1] 0.005223881
```

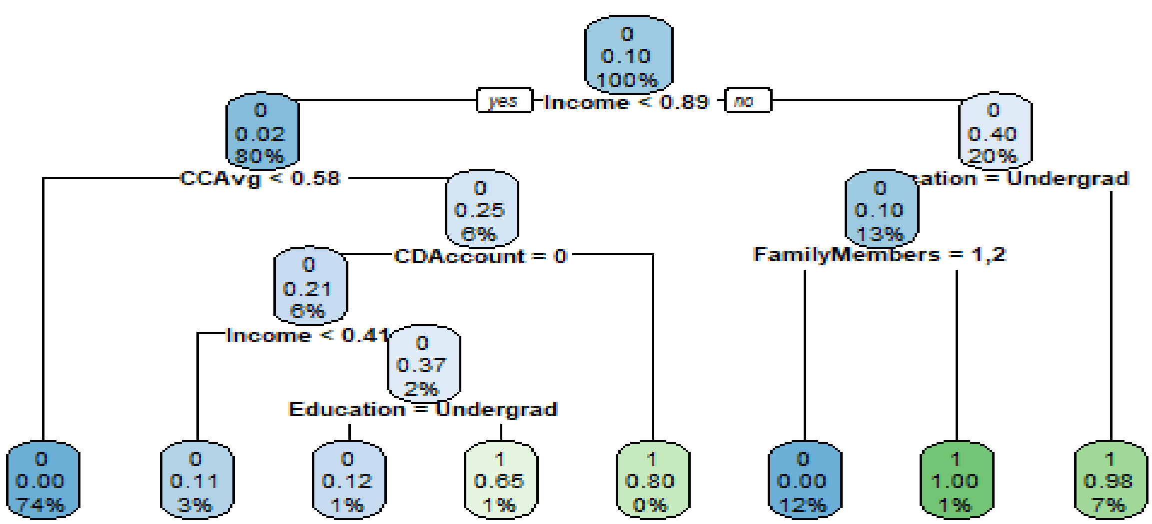
9.3 Textual representation of Pruned tree:

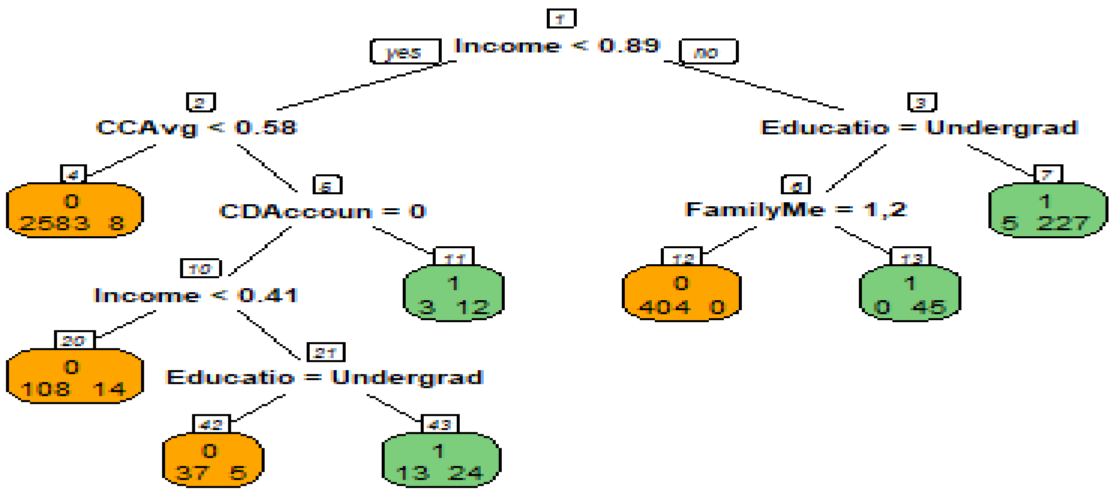
We build the cart model with the new CP which we got from the above step. We see the number of node in this model are less than the unpruned tree.

```
> ptree=prune(treeCart, cp=bestcp)
> print(ptree)
n= 3488

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

 1) root 3488 335 0 (0.903956422 0.096043578)
   2) Income< 0.8854743 2807  63 0 (0.977556110 0.022443890)
     4) CCAvg< 0.5776379 2591   8 0 (0.996912389 0.003087611) *
     5) CCAvg>=0.5776379 216  55 0 (0.745370370 0.254629630)
      10) CDAccount=0 201  43 0 (0.786069652 0.213930348)
        20) Income< 0.4076304 122  14 0 (0.885245902 0.114754098) *
        21) Income>=0.4076304 79  29 0 (0.632911392 0.367088608)
          42) Education=Undergrad 42   5 0 (0.880952381 0.119047619) *
          43) Education=Graduate,Advanced/Professional 37  13 1 (0.351351351 0.6486486
49) *
      11) CDAccount=1 15   3 1 (0.200000000 0.800000000) *
   3) Income>=0.8854743 681 272 0 (0.600587372 0.399412628)
     6) Education=Undergrad 449  45 0 (0.899777283 0.100222717)
      12) FamilyMembers=1,2 404   0 0 (1.000000000 0.000000000) *
      13) FamilyMembers=3,4 45   0 1 (0.000000000 1.000000000) *
     7) Education=Graduate,Advanced/Professional 232   5 1 (0.021551724 0.978448276) *

>
```

9.4 Plotting pruned tree(with prp as well):

```
> rpart.plot(ptree,cex=0.6)
```





9.5 Summary of Pruned Tree:

```
> summary(ptree)
Call:
rpart(formula = PersonalLoan ~ ., data = trainDataCart, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 4, cp = 0))
  n= 3488

           CP nsplit rel error    xerror       xstd
1 0.331343284      0 1.0000000 1.0000000 0.05194591
2 0.134328358      2 0.3373134 0.3701493 0.03264418
3 0.013432836      3 0.2029851 0.2358209 0.02622974
4 0.005223881      7 0.1432836 0.1731343 0.02254385

Variable importance
    Education        Income FamilyMembers         CCAvg      CDAccount     Mor
tagage
           34            23            20            12              8
3

Node number 1: 3488 observations,    complexity param=0.3313433
  predicted class=0  expected loss=0.09604358  P(node) =1
    class counts:  3153   335
   probabilities: 0.904 0.096
  left son=2 (2807 obs) right son=3 (681 obs)
  Primary splits:
      Income    < 0.8854743  to the left,  improve=155.75920, (0 missing)
      CCAvg     < 0.5776379  to the left,  improve=112.08900, (0 missing)
      CDAccount splits as  LR, improve= 58.99615, (0 missing)
      Mortagage < 2.161571   to the left,  improve= 25.79838, (0 missing)
      Education splits as  LRR, improve= 13.92162, (0 missing)
  Surrogate splits:
      CCAvg     < 1.206725   to the left,  agree=0.876, adj=0.363, (0 split)
      Mortagage < 2.800352   to the left,  agree=0.823, adj=0.095, (0 split)

Node number 2: 2807 observations,    complexity param=0.01343284
```

```
  predicted class=0  expected loss=0.02244389  P(node) =0.8047592
    class counts:  2744    63
   probabilities: 0.978 0.022 
  left son=4 (2591 obs) right son=5 (216 obs)
  Primary splits:
      CCAvg     < 0.5776379  to the left,  improve=25.2307300, (0 missing)
      Income    < 0.4076304  to the left,  improve=10.4531200, (0 missing)
      CDAccount splits as  LR, improve= 1.6300600, (0 missing)
      Mortagage < 1.906059   to the left,  improve= 1.1185550, (0 missing)
      Age       < 1.671443   to the left,  improve= 0.3038027, (0 missing)

Node number 3: 681 observations,    complexity param=0.3313433
  predicted class=0  expected loss=0.3994126  P(node) =0.1952408
    class counts:   409   272
   probabilities: 0.601 0.399 
  left son=6 (449 obs) right son=7 (232 obs)
  Primary splits:
      Education     splits as  LRR, improve=235.955100, (0 missing)
      FamilyMembers splits as  LLRR, improve=134.537500, (0 missing)
      CDAccount     splits as  LR, improve= 50.130770, (0 missing)
      Income        < 1.797722   to the left,  improve= 11.550570, (0 missing
)
      CCAvg         < 2.685079   to the right, improve=  5.846686, (0 missing
)
  Surrogate splits:
      FamilyMembers splits as  LLRR, agree=0.746, adj=0.254, (0 split)
      CDAccount     splits as  LR, agree=0.721, adj=0.181, (0 split)
      CCAvg         < 3.951831   to the left,  agree=0.668, adj=0.026, (0 spl
it)
      Mortagage     < 4.510318   to the left,  agree=0.665, adj=0.017, (0 spl
it)
      Income        < 0.9289146  to the right, agree=0.662, adj=0.009, (0 spl
it)

Node number 4: 2591 observations
  predicted class=0  expected loss=0.003087611  P(node) =0.7428326
    class counts:  2583     8
   probabilities: 0.997 0.003 

Node number 5: 216 observations,    complexity param=0.01343284
  predicted class=0  expected loss=0.2546296  P(node) =0.06192661
    class counts:   161    55
   probabilities: 0.745 0.255 
  left son=10 (201 obs) right son=11 (15 obs)
  Primary splits:
      CDAccount     splits as  LR, improve=9.588751, (0 missing)
      Income        < 0.4076304  to the left,  improve=6.799376, (0 missing)
      Age           < 1.671443   to the left,  improve=3.044769, (0 missing)
      Education     splits as  LRR, improve=2.667939, (0 missing)
      FamilyMembers splits as  LLRR, improve=2.421215, (0 missing)
  Surrogate splits:
      Age < 1.671443   to the left,  agree=0.935, adj=0.067, (0 split)

Node number 6: 449 observations,    complexity param=0.1343284
  predicted class=0  expected loss=0.1002227  P(node) =0.1287271
```

```
    class counts:   404    45
   probabilities: 0.900 0.100
  left son=12 (404 obs) right son=13 (45 obs)
  Primary splits:
      FamilyMembers splits as  LLRR, improve=80.979960, (0 missing)
      CDAccount     splits as  LR, improve=11.326920, (0 missing)
      Mortagage     < 1.237796   to the left,  improve= 1.980615, (0 missing)
      CCAvg         < 2.685079   to the right, improve= 1.510494, (0 missing)
      Experience    < 1.783613   to the left,  improve= 1.290068, (0 missing)
  Surrogate splits:
      CDAccount splits as  LR, agree=0.904, adj=0.044, (0 split)
      Mortagage < 5.006602   to the left,  agree=0.902, adj=0.022, (0 split)

Node number 7: 232 observations
  predicted class=1  expected loss=0.02155172  P(node) =0.06651376
    class counts:     5   227
   probabilities: 0.022 0.978

Node number 10: 201 observations,    complexity param=0.01343284
  predicted class=0  expected loss=0.2139303  P(node) =0.05762615
    class counts:   158    43
   probabilities: 0.786 0.214
  left son=20 (122 obs) right son=21 (79 obs)
  Primary splits:
      Income        < 0.4076304  to the left,  improve=6.106244, (0 missing)
      Education     splits as  LRR, improve=2.906752, (0 missing)
      FamilyMembers splits as  LLRR, improve=2.241140, (0 missing)
      Online        splits as  RL, improve=2.212395, (0 missing)
      CCAvg         < 1.321104   to the right, improve=1.516528, (0 missing)
  Surrogate splits:
      CCAvg      < 1.092345   to the left,  agree=0.667, adj=0.152, (0 split)
      Mortagage  < 1.886404   to the left,  agree=0.652, adj=0.114, (0 split)
      Experience < -0.6676381 to the right, agree=0.647, adj=0.101, (0 split)
      Age        < -0.8567109 to the right, agree=0.637, adj=0.076, (0 split)

Node number 11: 15 observations
  predicted class=1  expected loss=0.2  P(node) =0.004300459
    class counts:     3    12
   probabilities: 0.200 0.800

Node number 12: 404 observations
  predicted class=0  expected loss=0  P(node) =0.1158257
    class counts:   404     0
   probabilities: 1.000 0.000

Node number 13: 45 observations
  predicted class=1  expected loss=0  P(node) =0.01290138
    class counts:     0    45
   probabilities: 0.000 1.000

Node number 20: 122 observations
  predicted class=0  expected loss=0.1147541  P(node) =0.03497706
    class counts:   108    14
   probabilities: 0.885 0.115
```

```
Node number 21: 79 observations,    complexity param=0.01343284
  predicted class=0  expected loss=0.3670886  P(node) =0.02264908
    class counts:    50    29
   probabilities: 0.633 0.367
  left son=42 (42 obs) right son=43 (37 obs)
  Primary splits:
      Education     splits as  LRR, improve=11.0344700, (0 missing)
      FamilyMembers splits as  LLRR, improve= 9.2957550, (0 missing)
      CCAvg         < 1.329683   to the right, improve= 3.1023250, (0 missing
)
      Income        < 0.6682725  to the right, improve= 2.8627070, (0 missing
)
      Online        splits as  RL, improve= 0.8821599, (0 missing)
  Surrogate splits:
      FamilyMembers splits as  LLRR, agree=0.696, adj=0.351, (0 split)
      Income        < 0.6248322  to the right, agree=0.671, adj=0.297, (0 spl
it)
      CCAvg         < 1.149535   to the right, agree=0.646, adj=0.243, (0 spl
it)
      Experience    < 1.258345   to the left,  agree=0.633, adj=0.216, (0 spl
it)
      Online        splits as  RL, agree=0.620, adj=0.189, (0 split)

Node number 42: 42 observations
  predicted class=0  expected loss=0.1190476  P(node) =0.01204128
    class counts:    37     5
   probabilities: 0.881 0.119

Node number 43: 37 observations
  predicted class=1  expected loss=0.3513514  P(node) =0.0106078
    class counts:    13    24
   probabilities: 0.351 0.649
```

Path for accessing a node:

```
> path.rpart(ptree,nodes = 2:4)

 node number: 2
   root
   Income< 0.8855

 node number: 3
   root
   Income>=0.8855

 node number: 4
   root
   Income< 0.8855
   CCAvg< 0.5776
```

Variable Importance of Unpruned and Pruned tree:

```
> treeCart$variable.importance
    Education        Income FamilyMembers         CCAvg     CDAccount      Mor
tagage
```

```
   247.988588    170.474843    148.021037     97.905607     55.903868     22.
476681
   Experience           Age        Online
     9.391240      7.394241      2.087603
> ptree$variable.importance
    Education        Income FamilyMembers         CCAvg     CDAccount     Mor
tagage
   246.989564    167.180061    144.862753     91.438769     55.903868     21.
430278
   Experience        Online           Age
     3.004186      2.087603      1.103015
```

__Variables used in building the Unpruned and Pruned tree:__

Unpruned tree: Age, CCAvg, CDAccount, Education, Experience, FamilyMembers, Income

Pruned tree: CCAvg, CDAccount, Education, FamilyMembers, Income

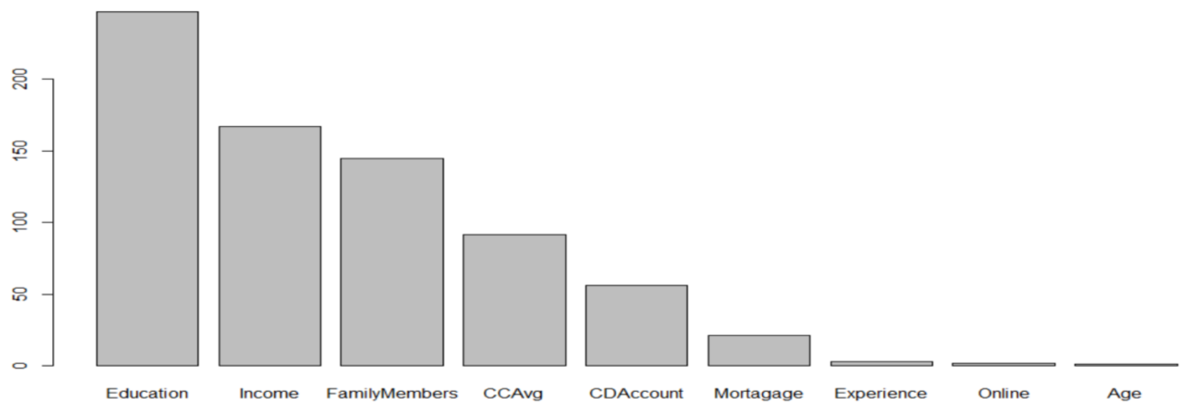
Experience and **Age** are not considered while pruning the tree. They are considered as unimportant variables

```
> printcp(treeCart)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = trainDataCart, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 4, cp = 0))

Variables actually used in tree construction:
[1] Age           CCAvg         CDAccount     Education     Experience    FamilyMembers
[7] Income

Root node error: 335/3488 = 0.096044

n= 3488

         CP nsplit rel error  xerror     xstd
1 0.3313433      0   1.00000 1.00000 0.051946
2 0.1343284      2   0.33731 0.37015 0.032644
3 0.0134328      3   0.20299 0.23582 0.026230
4 0.0052239      7   0.14328 0.17313 0.022544
5 0.0011940     11   0.12239 0.17612 0.022734
6 0.0000000     16   0.11642 0.19104 0.023660
> printcp(ptree)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = trainDataCart, method = "class",
    control = rpart.control(minsplit = 20, minbucket = 4, cp = 0))

Variables actually used in tree construction:
[1] CCAvg         CDAccount     Education     FamilyMembers Income

Root node error: 335/3488 = 0.096044

n= 3488
```

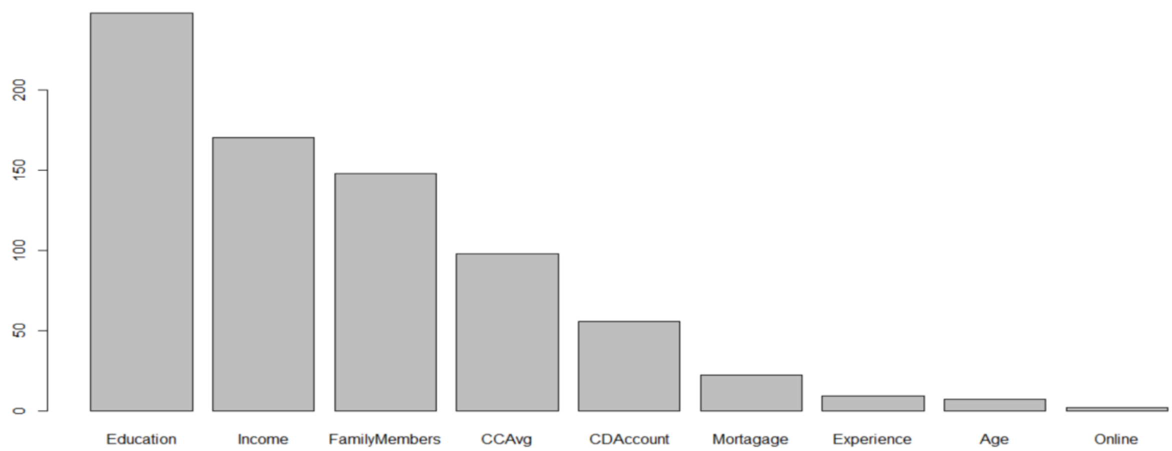
```
          CP nsplit rel error  xerror     xstd
1 0.3313433      0   1.00000 1.00000 0.051946
2 0.1343284      2   0.33731 0.37015 0.032644
3 0.0134328      3   0.20299 0.23582 0.026230
4 0.0052239      7   0.14328 0.17313 0.022544

>
```

Plotting variable importance Pruned and unpruned tree:

```
> barplot(ptree$variable.importance)
```




```
> barplot(treeCart$variable.importance)
```



9.6 Remarks on Pruning:

1. **Experience** and **Age** are not considered while pruning the tree. They are considered as unimportant variables

2. Best Cost/complexity parameter is **0.0052239** and **7 splits** were made in the pruned tree with CCAvg, CDAccount, Education, FamilyMembers, Income variables into consideration. The cross-validation error (xerror) associated is **0.17313** and the relative error is **0.14328**
3. Bank must focus more on the **income** of the customer as it is playing important role with priority. If the income is >0.85, then next focus should be made on **Credit Card spent** by the customer. If the income is <0.85, then they must focus on the **educational background** of the customer.
4. We are arrived with **8 terminal** nodes in pruned tree where as in unpruned tree, we had ~ 18 nodes. Pruning helped us in cutting down the tree and gave us descent number of classification groups and primary variables that we need to focus upon.
5. The model built is performing good on both train and test data. Here bank is trying to focus on those members who are interested to take loans. Hence we need to consider **specificity** as a measure and as per confusion matrix of CART model (below), it is showing 99% which is a good number.

10.Building Random Forest model:

Multiple cart models are created in RF model. Works with boot strapping mechanism

10.1 Splitting data for RF model and checking the mtry value:

Data is split into train and test with 70-30 ratio. Mtry is the number of independent variables to consider while multiple cart models in RF is built. Picking low m value will lead to loosing important columns while building the decision tree and high m value will lead to collinearity between the decision trees (cart models)

```
#RF
set.seed(1234)
splitRF=sample.split(TheraBankDataScaled$PersonalLoan ,SplitRatio=0.7)
trainDataRF=subset(TheraBankDataScaled,splitRF=="TRUE")
testDataRF=subset(TheraBankDataScaled,splitRF=="FALSE")
mtry1 = floor(sqrt(ncol(trainDataRF)))
mtry1
#View(trainData)
names(trainDataRF)
```

10.2 Separating dependent and independent variables.

Here Personal loan is the y variable/dependent variable and all other variables except ID, zip code are x variables/independent variables

```
names(trainDataRF)
#separating Y/predictor/dependent variable
x=trainDataRF[ ,-3]
#View(x)
#Viewing the independent variables
names(x)
y=trainDataRF$PersonalLoan
```

10.3 Picking optimum mtry value:

Internally RF will picking up square root of the number of columns as mtry value and try other mtry values with inflating stepFactor until the improvement is achieved. It will stop as soon as improvement is not qualifying the given threshold. Our tree starts with 3 mtry columns where OOB error is 1.43%. When it search left with mtry=2, OOB error is 1.63% and showed increase. Hence RF will not pursue in that direction any further. Same process will continue till improvement is reached and it will stop where the OOB error is low. **Here at mtry=6, we got lowest OOB as 1.18%**

Keywords:

tuneRF: Used to obtain optimum mtry value starting with default value for RF. Default value is square root of the independent variables.

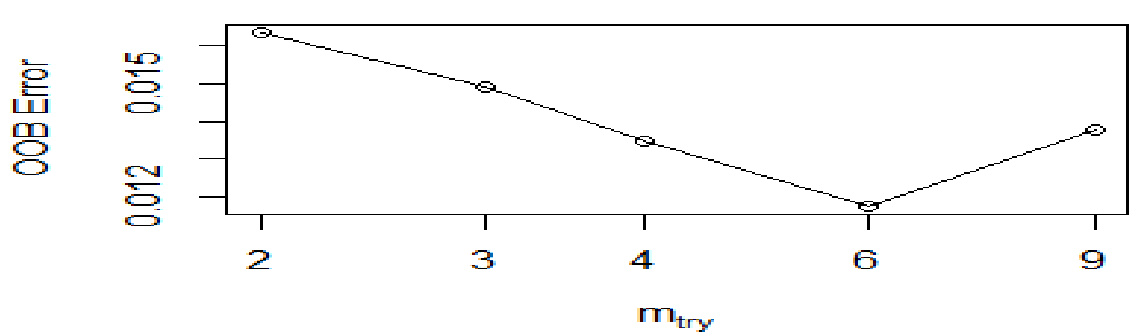
stepFactor: at each iteration, mtry value is inflated or deflated by this value

improve: the (relative) improvement in OOB error must be by this much for the search to continue

ntree: No.of trees used in the tuning step

```
> bestmtry = tuneRF(x, y, stepFactor = 1.5, improve = 1e-5, ntree=500)
mtry = 3  OOB error = 1.49%
Searching left ...
mtry = 2        OOB error = 1.63%
-0.09615385 1e-05
Searching right ...
mtry = 4        OOB error = 1.35%
0.09615385 1e-05
mtry = 6        OOB error = 1.18%
0.1276596 1e-05
mtry = 9        OOB error = 1.38%
-0.1707317 1e-05
```

Plotting the OOB Error Graph:



10.4 Tuning RF with the optimum mtry obtained above:

Ahere best mtry value we got is mtry=4 with least OOB error as 1.89% Number pf splits made =4.

```
> TRF = tuneRF(x, y,
+              mtryStart = 6,ntreeTry = 500,stepFactor = 1.5,improve = 0.0001
,
+              trace=TRUE,
+              plot=TRUE,
+              doBest= TRUE,
+              nodesize=100,
+              importance=TRUE)
mtry = 6  OOB error = 2.04%
Searching left ...
mtry = 4       OOB error = 1.89%
0.07042254 1e-04
mtry = 3       OOB error = 2.32%
-0.2272727 1e-04
Searching right ...
mtry = 9       OOB error = 2.01%
-0.06060606 1e-04
> print(TRF)

Call:
 randomForest(x = x, y = y, mtry = res[which.min(res[, 2]), 1],      nodesize
= 100, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 1.98%
Confusion matrix:
     0   1 class.error
0 3149   4 0.001268633
1   65 270 0.194029851
```

10.5 Building RF Model:

randomForest is the function used to build RF mode.

Y variable/Dependent variable= personalLoan

X variable as train data input for RF

Ntree= Number of trees to grow. This should not be too small or too large relative to dataset

Mtry=number of variables randomly sampled as candidate at each split

Nodesize=minimum size of terminal nodes

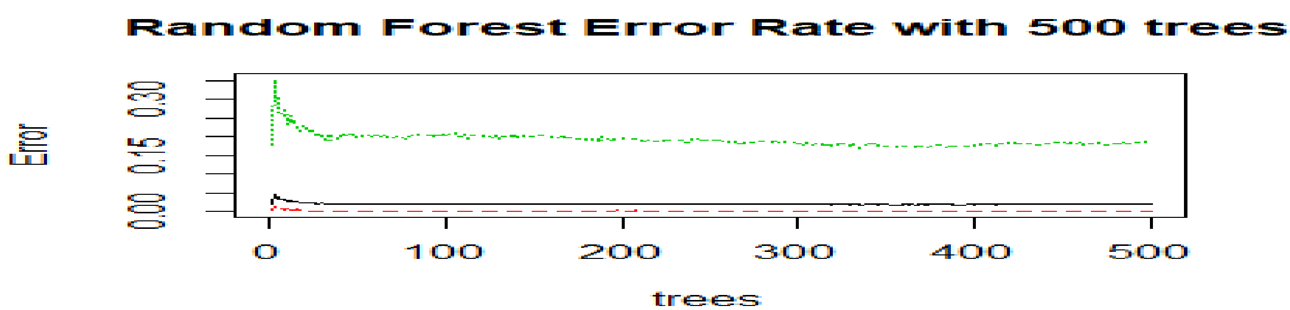
Importance=to assess what variables are having importance in the data.

Here 500 splits are made and 4 variables are considered at each split. OOB error is 2.01%.

```
> TheraRF= randomForest(PersonalLoan~.,data = trainDataRF,ntree=500,mtry=4
+                       ,nodesize=100,
+                       importance=TRUE)
> print(TheraRF)

Call:
 randomForest(formula = PersonalLoan ~ ., data = trainDataRF,      ntree = 50
0, mtry = 4, nodesize = 100, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 2.01%
Confusion matrix:
     0   1 class.error
0 3145   8 0.002537266
1   62 273 0.185074627
```

Plotting RF error rate:

From the below graph, it looks like the OOB error is stagnate after ntee=300. As RF is resource consuming, we can reduce number of trees in out model for better performance.

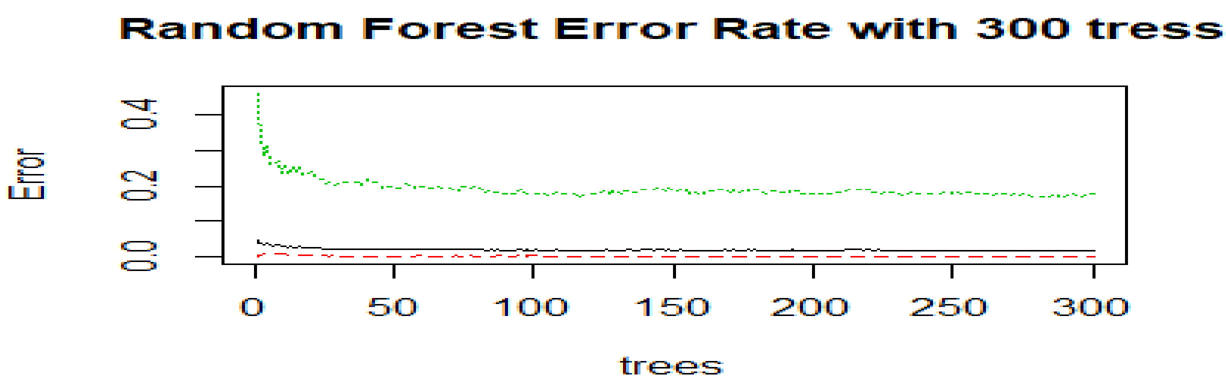


Tuning RF with optimum mtry and ntree values:

Here the tree size is reduced in the RF mode and the OOB error estimate is 1.89 % 4 splits are made with mtry=4 and ntree=300

```
> TheraRF= randomForest(PersonalLoan~.,data = trainDataRF,ntree=300,mtry=4
+                       ,nodesize=100,
+                       importance=TRUE)
> print(TheraRF)

Call:
 randomForest(formula = PersonalLoan ~ ., data = trainDataRF,      ntree = 30
0, mtry = 4, nodesize = 100, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 4
```

```
        OOB estimate of  error rate: 1.89%
Confusion matrix:
     0   1 class.error
0 3146   7 0.002220108
1   59 276 0.176119403
```



10.6 Variable importance:

Mean Decrease Accuracy: How much accuracy would be decreased if a variable is removed while making prediction. This is mostly useful in considering which variables are important in the dataset while making predictions. The top most variable is important and importance decreases as we go down

Mean Decrease Gini: This tells us how much decrease in the impurity would be there if a variable is removed. For classifying the data, we need groups pure as much as possible without overfitting.

10.7 Interpretation of Random Forest:

Family Members, Education, Income, CCAvg are playing major role in deciding whether a person would take personal loan.

We can target below groups:

a. **Income range >100**$: These customers can be offered personal loan with exciting offers

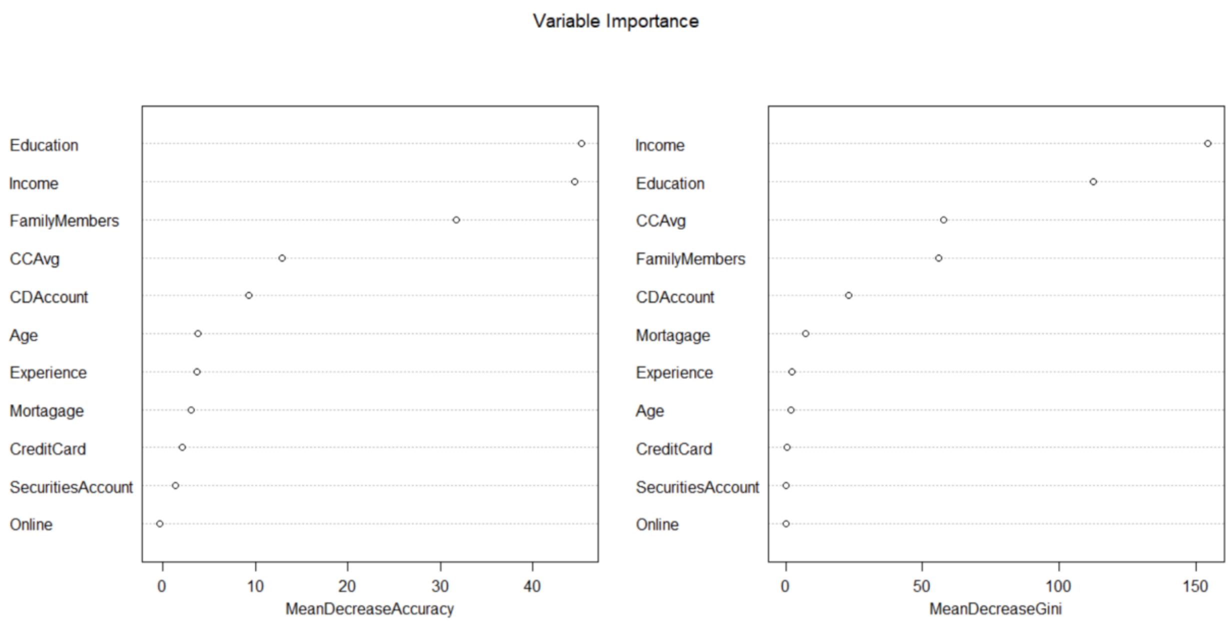
b. **Family members 3-4 in size**: These customers can be offered loans as they would be interested in improving the quality of life since they have a family.

c. **Educational background**: Customers who have advanced/Professional background can be offered loans

d. **Credit card spent**: Customers whose average credit card spent rate is more can be offered personal loan with less interest of credit card bill

```
> importance(TheraRF)
                          0           1 MeanDecreaseAccuracy MeanDecreaseGini
FamilyMembers     31.656310 21.80842666            31.704905       56.1181053
Education         45.487190 34.03102277            45.269229      112.6110950
SecuritiesAccount  1.797272 -1.07990661             1.360692        0.3216192
CDAccount          7.256877  6.52512384             9.318518       22.9168782
Online            -0.608409 -0.10658358            -0.407919        0.3153094
CreditCard         1.699720 -0.03989862             2.097637        0.4477171
Age                3.656581 -0.04173125             3.806450        1.9375065
Experience         3.920266 -0.82053756             3.673082        2.1512986
Income            41.983832 35.85947557            44.487812      154.4476637
CCAvg             11.491052 11.47559515            12.851488       57.6365117
Mortagage          6.418462 -4.88866888             3.019735        7.4866397
```

Variable importance plot:



Making Prediction using RF Model:

Prediction on train and test data using RF Model:

```
> predict.classRF=predict(TheraRF, trainDataRF, type="class")
> #predict.classRF
> tabdevRF=table(trainDataRF$PersonalLoan ,predict.classRF)
> tabdevRF
   predict.classRF
       0    1
  0 3146    7
  1   57  278
> predict.classRFTest=predict(TheraRF, testDataRF, type="class")
> #predict.classRF
> tabdevRFTest=table(testDataRF$PersonalLoan ,predict.classRFTest)
> tabdevRFTest
```

```
  predict.classRFTest
        0    1
  0 1344    7
  1   21  122
```

11. Confusion Matrix of CART and Random Forest models:

Confusion Matrix is one of the model performances measures to check how well our model is fitting the test or new data.

Important Measures of Confusion Matrix: Sensitivity, Specificity, Accuracy

Sensitivity: Also called as True positive rate or Recall. This is proportion of actual positive cases which are correctly identified. TP/(TP+FN)

Specificity: Also called as True Negative rate or False Positive rate. This is proportion of negatives that were correctly identified. TN/(TN+FP)

Accuracy: 1-error rate. This is how many correct predictions are done in both classes. Error rate: FP+FN/(TP+TN+FP+FN)

11.1 Tabular Representation of confusion matrices for comparison

Measure	CART Model		Random Forest Model	
	Train	Test	Train	Test
Sensitivity	0.9915	0.9919	0.9822	0.9846
Specificity	0.9362	0.9565	0.9745	0.9457
Accuracy	0.9862	0.9886	0.9817	0.9813

The confusion matrices which we made considered positive rate as ‘0’. From the data we collected from the campaign, customers who did not took personal loan are marked as ‘0’ and who took personal loan are marked as ‘1’. Since we are focusing on customers who are interested to take personal loan basin on past trend of data, we are looking at “**True Negativity**” or “**Specificity**” as our measure of interest. Though accuracy in RF is reduced, it shouldn’t case issue.

Upon comparing the output of confusion matrix from both models, we notice that **randomForest performed better than CART model** due to following reason:

a) CART does overfitting of the data whereas RF does not overfit the data.

b) RF uses bagging, bootstrapping and aggregation which improves model performance

c) RF has one level of testing within the algorithm on train dataset (with 70-30 split) when model is built. CART has no such kind of internal testing in the algorithm.

Train Data Confusion Matrix for CART:

```
> confusionMatrix(tabCartTrain)
Confusion Matrix and Statistics

            predict.class
PersonalLoan    0    1
           0 3132   21
           1   27  308

               Accuracy : 0.9862
                 95% CI : (0.9818, 0.9898)
    No Information Rate : 0.9057
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9201

 Mcnemar's Test P-Value : 0.4705

            Sensitivity : 0.9915
            Specificity : 0.9362
         Pos Pred Value : 0.9933
         Neg Pred Value : 0.9194
             Prevalence : 0.9057
         Detection Rate : 0.8979
   Detection Prevalence : 0.9040
      Balanced Accuracy : 0.9638

       'Positive' Class : 0
```

Test Data Confusion Matrix CART:

```
> confusionMatrix(tabCartTest)
Confusion Matrix and Statistics

            predict.class
PersonalLoan    0    1
           0 1345    6
           1   11  132

               Accuracy : 0.9886
                 95% CI : (0.9818, 0.9934)
    No Information Rate : 0.9076
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9332

 Mcnemar's Test P-Value : 0.332

            Sensitivity : 0.9919
            Specificity : 0.9565
         Pos Pred Value : 0.9956
```

```
         Neg Pred Value : 0.9231
             Prevalence : 0.9076
         Detection Rate : 0.9003
   Detection Prevalence : 0.9043
      Balanced Accuracy : 0.9742

       'Positive' Class : 0
```

<u>Train Data Confusion Matrix for RandomForest Model:</u>

```
> confusionMatrix(tabdevRF)
Confusion Matrix and Statistics

   predict.classRF
       0    1
  0 3146    7
  1   57  278

               Accuracy : 0.9817
                 95% CI : (0.9766, 0.9858)
    No Information Rate : 0.9183
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8868

 Mcnemar's Test P-Value : 9.068e-10

            Sensitivity : 0.9822
            Specificity : 0.9754
         Pos Pred Value : 0.9978
         Neg Pred Value : 0.8299
             Prevalence : 0.9183
         Detection Rate : 0.9019
   Detection Prevalence : 0.9040
      Balanced Accuracy : 0.9788

       'Positive' Class : 0
```

<u>Test Data Confusion Matrix for RandomForest Model:</u>

```
> confusionMatrix(tabdevRFTest)
Confusion Matrix and Statistics

   predict.classRFTest
       0    1
  0 1344    7
  1   21  122

               Accuracy : 0.9813
                 95% CI : (0.973, 0.9875)
    No Information Rate : 0.9137
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8868
```

```
Mcnemar's Test P-Value : 0.01402

            Sensitivity : 0.9846
            Specificity : 0.9457
         Pos Pred Value : 0.9948
         Neg Pred Value : 0.8531
             Prevalence : 0.9137
         Detection Rate : 0.8996
   Detection Prevalence : 0.9043
      Balanced Accuracy : 0.9652

       'Positive' Class : 0
```

12.Model Performance Measures:

ROC and AUC stands for Receiver Operating Characteristics and Area under the curve.

```
?prediction

predobjtrain = prediction(trainDataCart$predict.score, trainDataCart$PersonalLoan )
preftrain = performance(predobjtrain, "tpr", "fpr")
plot(preftrain,main="Train Data ROC")

predobjtest = prediction(testDataCart$predict.score, testDataCart$PersonalLoan)
preftest = performance(predobjtest, "tpr", "fpr")
plot(preftest,main="Test Data ROC")

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

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

Area under curve for train data is 0.986 and test data is 0.9755. ROC is plotted axis are below:

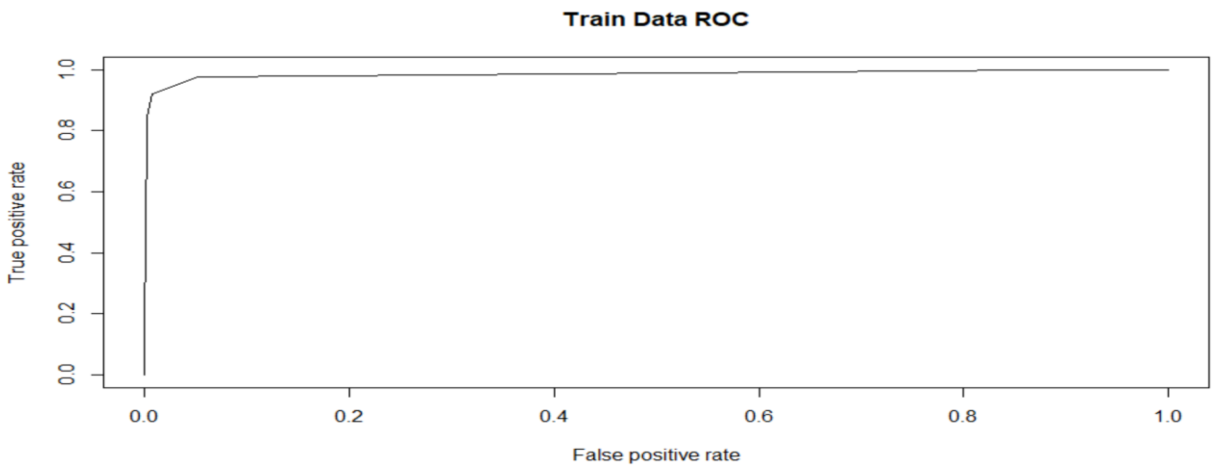
X axis: False positive rate/ Specificity=FP/(FP+TN)

Y Axis: True Positive rate/ Sensitivity/Recall=TP/(TP+FN)

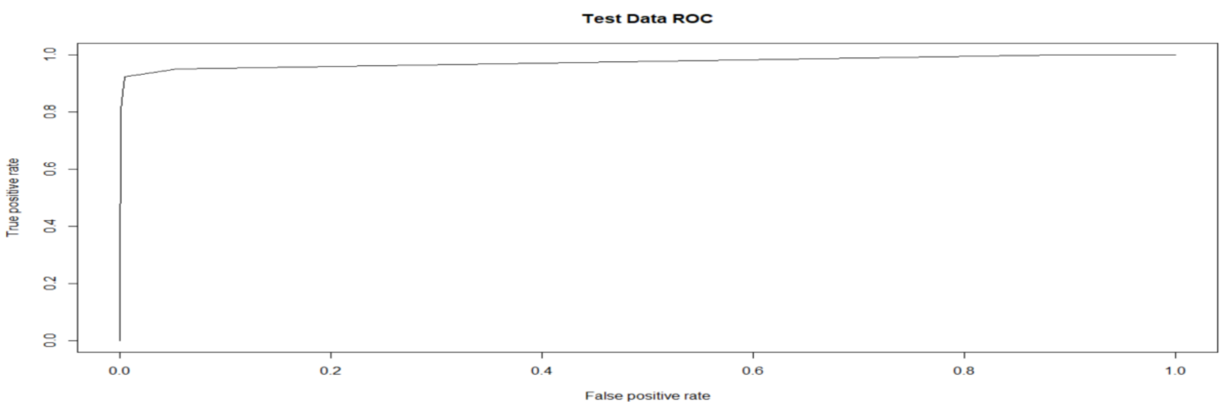
Usually the area under test data curve is relatively less than train data curve. TPR will increase and becomes stagnant eventually. Higher the area under the curve, better is the performance of the model.

```
> auctrain
[1] 0.9863489
> auctest
[1] 0.9755426
```

ROC Curve for Train Data:



ROC Curve for Test Data:



Gini Coefficient:

```
> #Gini Coefficient train
> Ginitrain= (2*auctrain) - 1
> Ginitrain
[1] 0.9726979
>
> Ginitrainnew = ineq(trainDataCart$predict.score , "gini")
> Ginitrainnew
[1] 0.8792765
>
> Ginitest= (2*auctest) - 1
> Ginitest
[1] 0.9510852
>
> Ginitestnew = ineq(trainDataCart$predict.score, "gini")
> Ginitestnew
[1] 0.8792765
```

KS Value:

```
> #KS value
> KStrain=max(preftrain@y.values[[1]]- preftrain@x.values[[1]])
> KStrain
[1] 0.9234711
>
> KStest=max(preftest@y.values[[1]]- preftest@x.values[[1]])
> KStest
[1] 0.9186358
```

13. Remarks on Model validation exercise

As per the Problem dataset, we were asked to find those group of customers who are potential loan takers as well. Of these models, **confusion matrix is good option** for model evaluation as it has various measure to determine the model with 3 categories. As we want to predict "**True Negative Rate**"/Specificity (the customer who is not likely going to take a loan and we also predicted the same, thereby reducing the cost of campaign), confusion matrix gives the % appropriately.

Validation Confusion Matrix:

Measure	CART Model		Random Forest Model	
	Train	Test	Train	Test
Sensitivity	0.9915	0.9919	0.9822	0.9846
Specificity	0.9362	0.9565	0.9745	0.9457
Accuracy	0.9862	0.9886	0.9817	0.9813

Validation with ROC/AUC:

This validation although gave a good % of validation to the built model, it can't help us achieve to a larger extent on what we are focusing on (TPR) to solve the business issue

Measure	Train	Test
AUC	0.986349	0.975543
Gini (with formula)	0.972698	0.879277
Gini (with package)	0.951085	0.879277
KS-Value	0.923471	0.918636

****************THE END****************