Mini Project- Thera Bank: Loan Purchase Modeling

Data Mining

Table of Contents

1. Project Objective

Our job is to build the best model which can classify the right customers who have a higher probability of purchasing the loan for Thera Bank

This exploration report will consist of the following:

- Understanding the structure of dataset
- Graphical exploration
- Descriptive statistics
- General Insights from the dataset
- Build appropriate models CART & Random Forest
- Validate the Model
- Check the performance of all the models

2. Assumptions

Following assumption we made for this analysis

- The Data Provided to us was not tempered.
- Linearity - Linearity assumes a straight line relationship between each of the two variables.
- Homoscedasticity - Homoscedasticity assumes that data is equally distributed about the regression line.

3. Exploratory Data Analysis Step by Step approach

A Typical Data exploration activity consists of the following steps:

1. Environment Set up and Data Import
2. Data Cleaning
3. Variable Identification
4. Univariate Analysis
5. Bi-Variate Analysis
6. Correlation Analysis

We shall follow these steps in exploring the provided dataset.

3.1 Environment Set up and Data Import

3.1.1 Install necessary Packages and Invoke Libraries

Following are the Libraries are used in the analysis

Code for loading library

```
> library(tidyverse)
> library(dplyr)
> library(ggplot2)
> library(DataExplorer)
> library(rpart)
> library(rpart.plot)
> library(rattle)
> library(RColorBrewer)
> library(caTools)
```

```
> library(caret)
> library(randomForest)
> library(data.table)
> library(ROCR)
> library(ineq)
> library(corrplot)
> library(InformationValue)
```

3.1.2 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.

Code for setting working directory

```
#Setting the Working Directory
> setwd ("E:/000GL/000 0Projects/004/Project/Final")
> getwd()
```

Please refer to Appendix A for Source Code.

3.1.3 Import and Read the Dataset

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

Code for Read the Dataset

```
# Importing Data
## Import the Cold_Storage_Temp_Data.csv
TheraData <- read.csv("Thera Bank_Personal_Loan_Modelling-data.csv")
```

Please refer to Appendix A for Source Code.

3.2 Variable Identification

3.2.1 Variable Identification – Inferences

Our Data contain 5000 obs. of 14 variables.
Column name of our Data are:

- ID
- Age
- Experience
- Income
- ZIPCode
- FamilyMembers
- CCAvg
- Education
- Mortgage
- PersonalLoan
- SecuritiesAccount
- CDAccount

- Online
- CreditCard

Size of Data 5000*14

We also checked the summary of data in which we found.

- 18(i.e 0.36%) NA in the **Family Member**
- 52 Negative values in the **Experience** which is not possible.
- ID, FamilyMembers, Education, PersonalLoan, SecuritiesAccount, CDAccount, Online, CreditCard are numeric which needs to be turn in factors

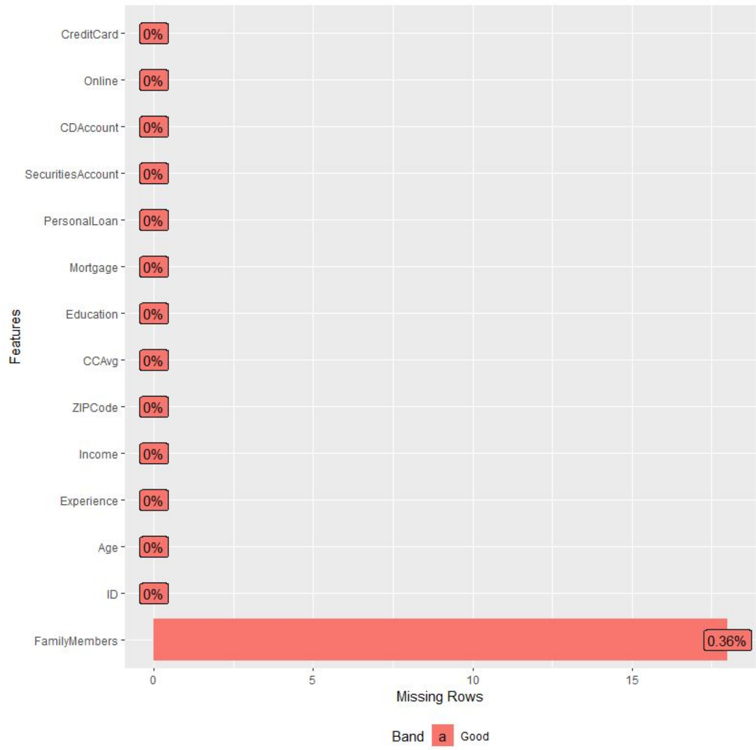
***PersonalLoan** is our Dependent Variable, **ID** needs to be removed (because just being a serial no.) and rest all are Independent Variable*

Command for variable identifications and Output

```
> summary(TheraData)
       ID             Age          Experience       Income          ZIPCode      FamilyMembers
 Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307   Min.   :1.000
 1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911   1st Qu.:1.000
 Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437   Median :2.000
 Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93153   Mean   :2.397
 3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608   3rd Qu.:3.000
 Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651   Max.   :4.000
                                                                                NA's   :18
     CCAvg          Education        Mortgage      PersonalLoan   SecuritiesAccount
 Min.   : 0.000   Min.   :1.000   Min.   :  0.0   Min.   :0.000   Min.   :0.0000
 1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000
 Median : 1.500   Median :2.000   Median :  0.0   Median :0.000   Median :0.0000
 Mean   : 1.938   Mean   :1.881   Mean   : 56.5   Mean   :0.096   Mean   :0.1044
 3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000
 Max.   :10.000   Max.   :3.000   Max.   :635.0   Max.   :1.000   Max.   :1.0000

   CDAccount          Online         CreditCard
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
 Median :0.0000   Median :1.0000   Median :0.000
 Mean   :0.0604   Mean   :0.5968   Mean   :0.294
 3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.000

> #EDA
> myData = TheraData
> str(myData)
'data.frame':	5000 obs. of  14 variables:
 $ ID               : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Age              : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience       : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIPCode          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
 $ FamilyMembers    : int  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        : int  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : int  0 0 0 0 0 155 0 0 104 0 ...
 $ PersonalLoan     : int  0 0 0 0 0 0 0 0 0 1 ...
```

```
 $ SecuritiesAccount: int  1 1 0 0 0 0 0 0 0 0 ...
 $ CDAccount        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Online           : int  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard       : int  0 0 0 0 1 0 0 1 0 0 ...
> dim(myData)
[1] 5000   14
```

Please refer to Appendix A for Source Code.



(Missing Variable Plot)

3.3 Data Cleaning

In Data cleaning process we has done following thing

- Remove Missing Variable
- Remove Negative Experience
- Update the Datatype to factors of certain variable

Updated summary of Data is as follows

Dimension of Data - 4930 * 14

```
> myData = TheraData
> myData$ID = as.factor(myData$ID)
> myData$FamilyMembers = as.factor(myData$FamilyMembers)
> myData$Education = as.factor(myData$Education)
> myData$PersonalLoan = as.factor(myData$PersonalLoan)
> myData$SecuritiesAccount = as.factor(myData$SecuritiesAccount)
```

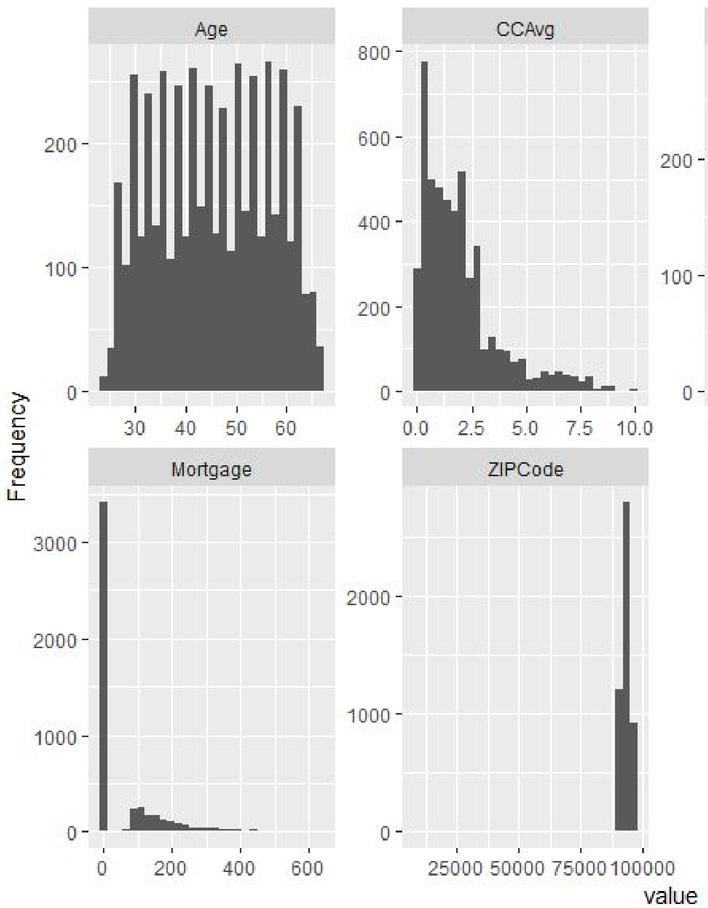
```
> myData$CDAccount = as.factor(myData$CDAccount)
> myData$Online = as.factor(myData$Online)
> myData$CreditCard = as.factor(myData$CreditCard)
> ##Also treating the negetive work experience by removing them
> myData$Experience[myData$Experience < 0] = NA
> sum(is.na(myData))
[1] 70
> ##70 values are missing which is less than 3% of the data so we can remove the NA Data
> MainData = na.omit(myData)
> dim(MainData)
[1] 4930   14
> summary(MainData)
       ID            Age          Experience        Income          ZIPCode      FamilyMembers
 1      :   1   Min.   :24.00   Min.   : 0.00   Min.   :  8.00   Min.   : 9307   1:1462
 2      :   1   1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:91910   2:1270
 3      :   1   Median :46.00   Median :20.00   Median : 64.00   Median :93437   3:1000
 4      :   1   Mean   :45.55   Mean   :20.32   Mean   : 73.77   Mean   :93152   4:1198
 5      :   1   3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   3rd Qu.:94608
 6      :   1   Max.   :67.00   Max.   :43.00   Max.   :224.00   Max.   :96651
 (Other):4924
     CCAvg        Education    Mortgage      PersonalLoan SecuritiesAccount CDAccount Online
 Min.   : 0.000   1:2072    Min.   :  0.00   0:4452       0:4417            0:4630    0:1991
 1st Qu.: 0.700   2:1383    1st Qu.:  0.00   1: 478       1: 513            1: 300    1:2939
 Median : 1.500   3:1475    Median :  0.00
 Mean   : 1.938             Mean   : 56.68
 3rd Qu.: 2.600             3rd Qu.:101.00
 Max.   :10.000             Max.   :635.00

 CreditCard
 0:3480
 1:1450

> class(MainData)
[1] "data.frame"
> str(MainData)
'data.frame':   4930 obs. of  14 variables:
 $ ID               : Factor w/ 5000 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Age              : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience       : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIPCode          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
 $ 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        : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : int  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 ...
 - attr(*, "na.action")= 'omit' Named int  21 59 90 99 162 227 236 290 316 452 ...
  ..- attr(*, "names")= chr  "21" "59" "90" "99" ...
> names(MainData)
 [1] "ID"                "Age"               "Experience"        "Income"
 [5] "ZIPCode"           "FamilyMembers"     "CCAvg"             "Education"
 [9] "Mortgage"          "PersonalLoan"      "SecuritiesAccount" "CDAccount"
[13] "Online"            "CreditCard"
> head(MainData)
  ID Age Experience Income ZIPCode FamilyMembers CCAvg Education Mortgage PersonalLoan
```

```
1  1  25          1     49   91107            4   1.6         1        0            0
2  2  45         19     34   90089            3   1.5         1        0            0
3  3  39         15     11   94720            1   1.0         1        0            0
4  4  35          9    100   94112            1   2.7         2        0            0
5  5  35          8     45   91330            4   1.0         2        0            0
6  6  37         13     29   92121            4   0.4         2      155            0
  SecuritiesAccount CDAccount Online CreditCard
1                 1         0      0          0
2                 1         0      0          0
3                 0         0      0          0
4                 0         0      0          0
5                 0         0      0          1
6                 0         0      1          0
> tail(MainData)
       ID Age Experience Income ZIPCode FamilyMembers CCAvg Education Mortgage PersonalLoan
4995 4995  64         40     75   94588             3   2.0         3        0            0
4996 4996  29          3     40   92697             1   1.9         3        0            0
4997 4997  30          4     15   92037             4   0.4         1       85            0
4998 4998  63         39     24   93023             2   0.3         3        0            0
4999 4999  65         40     49   90034             3   0.5         2        0            0
5000 5000  28          4     83   92612             3   0.8         1        0            0
     SecuritiesAccount CDAccount Online CreditCard
4995                 0         0      1          0
4996                 0         0      1          0
4997                 0         0      1          0
4998                 0         0      0          0
4999                 0         0      1          0
5000                 0         0      1          1
```

3.4 Univariate Analysis

Histogram of the Data

"hist" is used to plot the histogram of numeric variable



3.4.1 Age

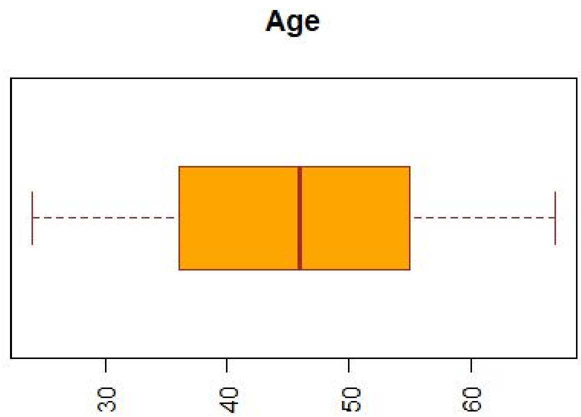
Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
24.00	36.00	46.00	45.55	55.00	67.00

Standard Deviation

11.32826

Box Plot



3.4.2 Experience

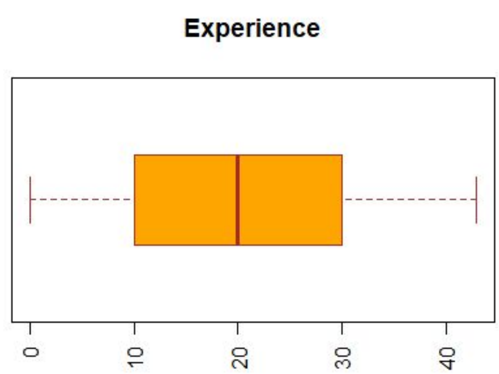
Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	10.00	20.00	20.32	30.00	43.00

Standard Deviation

11.31943

Box Plot



3.4.3 Income (in K)

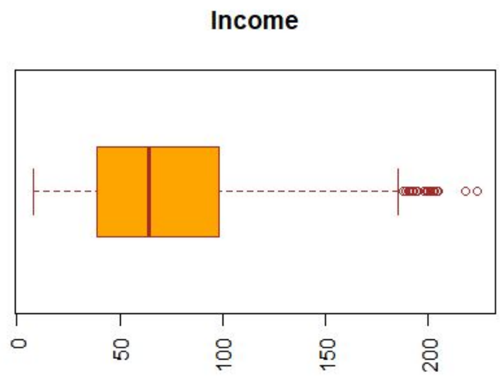
Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
8.00	39.00	64.00	73.77	98.00	224.00

Standard Deviation

46.11939

Box Plot



3.4.4 CCAvg

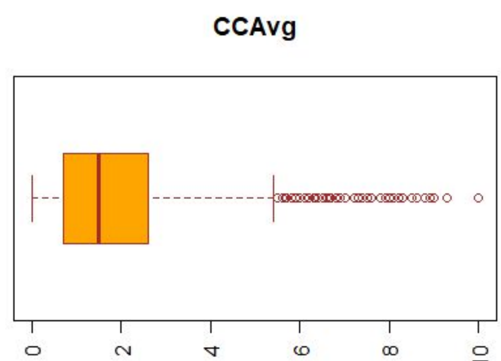
Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	0.700	1.500	1.938	2.600	10.000

Standard Deviation

1.748613

Box Plot



3.4.5 Mortgage

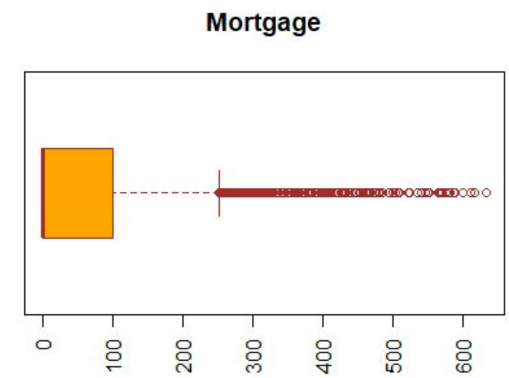
Summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	0.00	0.00	56.68	101.00	635.00

Standard Deviation

101.8722

Box Plot



3.4.6 Education

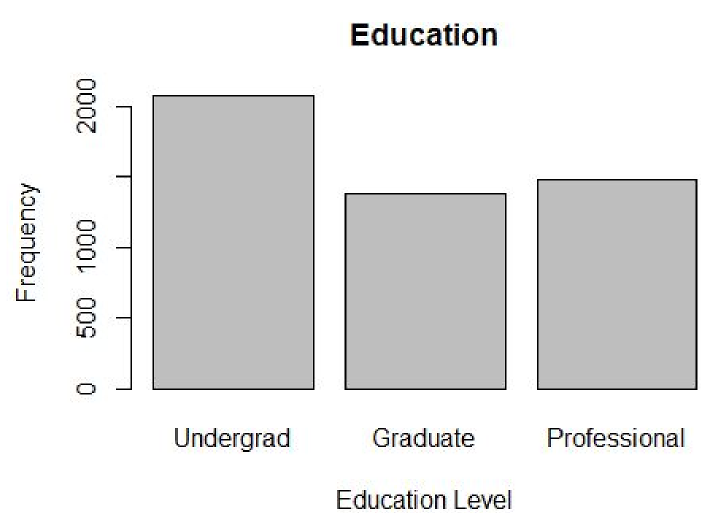
Summary:

1 (UnderGrads) - 2072

2 (Graduates) - 1383

3 (Professional) - 1475

Bar Plot



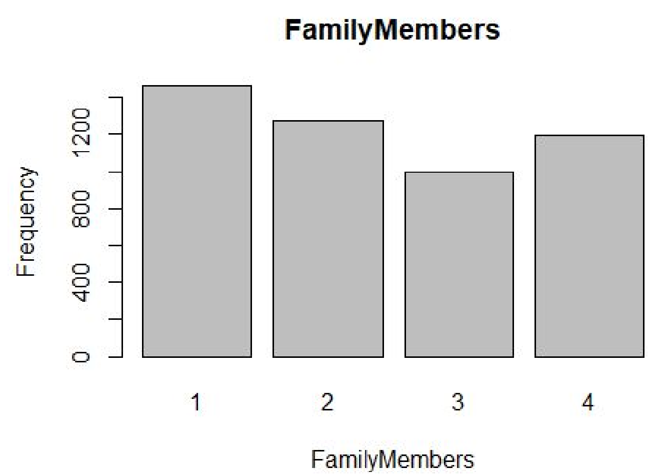
3.4.7 Family

Summary:

Family Member Count

1	2	3	4
1462	1270	1000	1198

Bar Plot



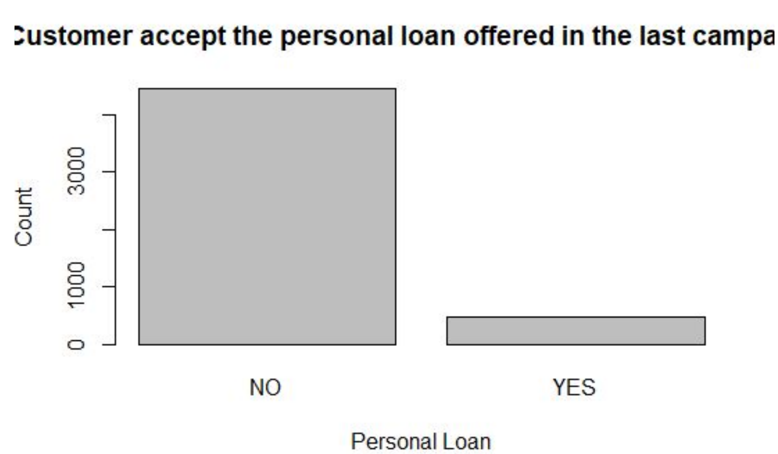
3.4.8 Personal Loan

Summary:

Personal Loan Acceptance count

No	Yes
4452	478

Bar Plot



3.4.9 SecuritiesAccount

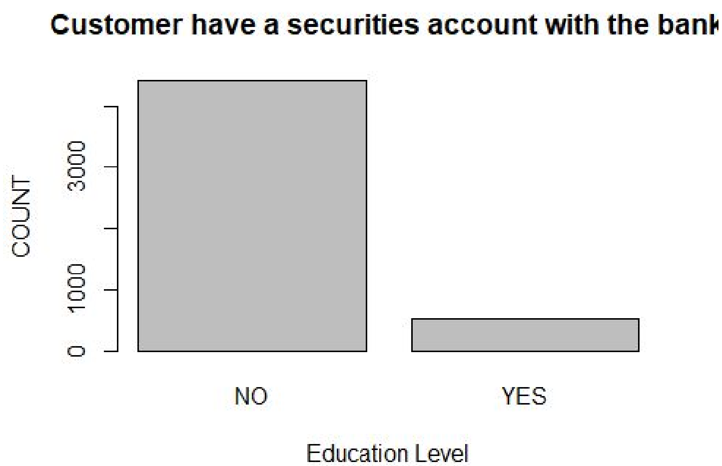
Summary:

Count of customer have a securities account with the bank.

No Yes

4417 513

Bar Plot



3.4.10 CDAccount

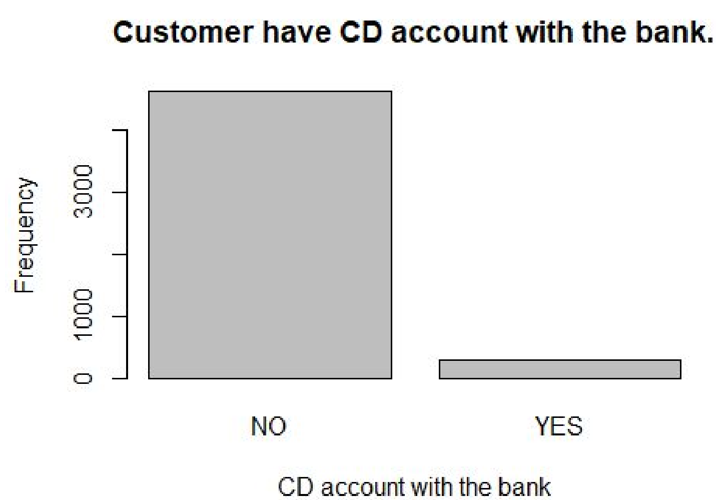
Summary:

Count of customer that have a certificate of deposit (CD) account with the bank.

No Yes

4630 300

Bar Plot



3.4.11 CreditCard

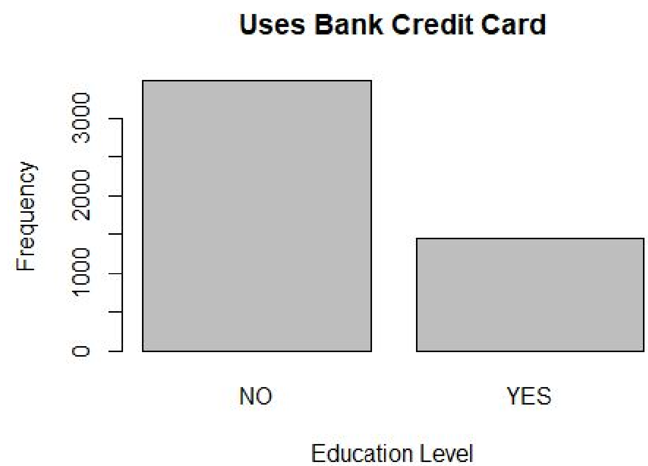
Summary:

Count of customers use internet banking facilities.

No Yes

3480 1450

Bar Plot



3.4.12 Online

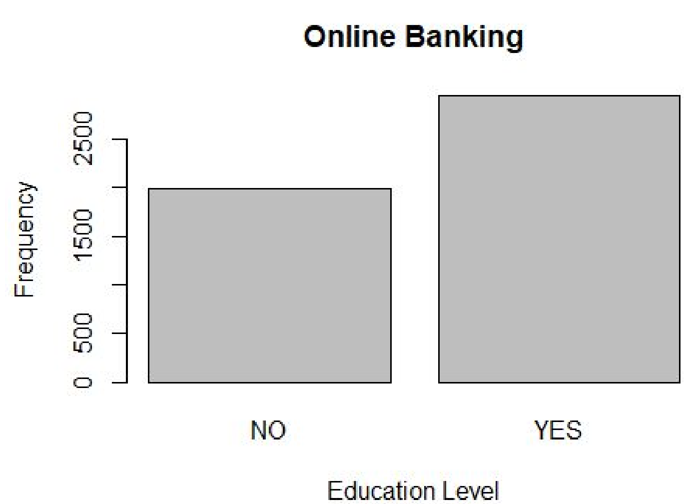
Summary:

Count of customer use internet banking facilities.

No Yes

1991 2939

Bar Plot



3.4.13 ZipCode

Summary

We have Total 4253 pincodes and pincode 94720 have maximum users

94720 : 163

94305 : 125

95616 : 115

90095 : 70

93106 : 56

92037 : 54

(Other):4347

```
> ##Univariant Analysis
> ##Histogram of Continious Variable
> plot_histogram(MainData)
> names(MainData)
 [1] "ID"                 "Age"                "Experience"        "Income"
 [5] "ZIPCode"            "FamilyMembers"      "CCAvg"             "Education"
 [9] "Mortgage"           "PersonalLoan"       "SecuritiesAccount" "CDAccount"
[13] "Online"             "CreditCard"
> summary(MainData$Age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  24.00   36.00   46.00   45.55   55.00   67.00
> sd(MainData$Age)
[1] 11.32826
> boxplot(MainData$Age
+         ,horizontal = TRUE
+         ,las =2
+         ,main = "Age"
+         ,col = "orange"
+         ,border = "brown")
> summary(MainData$Experience)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   10.00   20.00   20.32   30.00   43.00
> sd(MainData$Experience)
[1] 11.31943
> boxplot(MainData$Experience
+         ,horizontal = TRUE
+         ,las =2
+         ,main = "Experience"
+         ,col = "orange"
+         ,border = "brown")
> summary(MainData$Income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8.00   39.00   64.00   73.77   98.00  224.00
> sd(MainData$Income)
[1] 46.11939
> boxplot(MainData$Income
+         ,horizontal = TRUE
+         ,las =2
+         ,main = "Income"
+         ,col = "orange"
+         ,border = "brown")
> summary(MainData$CCAvg)
```

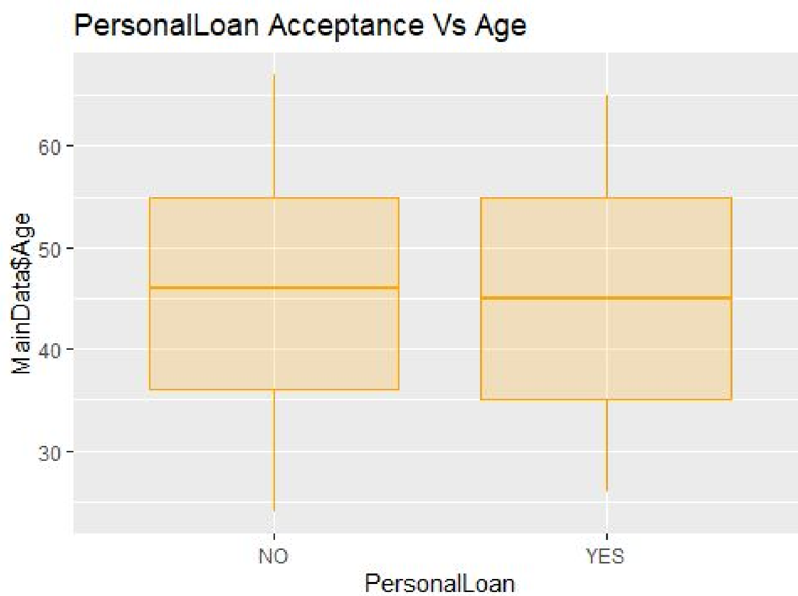
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   0.700   1.500   1.938   2.600  10.000 
> sd(MainData$CCAvg)
[1] 1.748613
> boxplot(MainData$CCAvg
+         ,horizontal = TRUE
+         ,las =2
+         ,main = "CCAvg"
+         ,col = "orange"
+         ,border = "brown")
> summary(MainData$Mortgage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00    0.00    0.00   56.68  101.00  635.00 
> sd(MainData$Mortgage)
[1] 101.8722
> boxplot(MainData$Mortgage
+         ,horizontal = TRUE
+         ,las =2
+         ,main = "Mortgage"
+         ,col = "orange"
+         ,border = "brown")
> summary(MainData$Education)
   1    2    3 
2072 1383 1475 
> barplot(table(MainData$Education), main="Education",
+         xlab="Education Level",
+         names.arg=c("Undergrad","Graduate", "Professional"),
+         ylab = "Frequency")
> summary(MainData$FamilyMembers)
   1    2    3    4 
1462 1270 1000 1198 
> barplot(table(MainData$FamilyMembers), main="FamilyMembers",
+         xlab="FamilyMembers",
+         ylab = "Frequency")
> summary(MainData$PersonalLoan)
   0    1 
4452  478 
> barplot(table(MainData$PersonalLoan), main="Customer accept the personal loan offered in the last 
campaign?",
+         xlab="Personal Loan",
+         names.arg=c("NO","YES"),
+         ylab = "Count")
> summary(MainData$SecuritiesAccount)
   0    1 
4417  513 
> barplot(table(MainData$SecuritiesAccount), main="Customer have a securities account with the bank",
+         xlab="Education Level",
+         names.arg=c("NO","YES"),
+         ylab = "COUNT")
> summary(MainData$CDAccount)
   0    1 
4630  300 
> barplot(table(MainData$CDAccount), main="Customer have CD account with the bank.",
+         xlab="CD account with the bank",
+         names.arg=c("NO","YES"),
+         ylab = "Frequency")
> summary(MainData$Online)
   0    1 
1991 2939 
```

```
> barplot(table(MainData$Online), main="Online Banking",
+         xlab="Education Level",
+         names.arg=c("NO","YES"),
+         ylab = "Frequency")
> summary(MainData$CreditCard)
   0    1 
3480 1450 
> barplot(table(MainData$CreditCard), main="Uses Bank Credit Card",
+         xlab="Education Level",
+         names.arg=c("NO","YES"),
+         ylab = "Frequency")
> ZipTemp = MainData
> ZipTemp$ZIPCode = as.factor(ZipTemp$ZIPCode)
> summary(ZipTemp)
       ID            Age          Experience        Income          ZIPCode     FamilyMembers
 1      :   1   Min.   :24.00   Min.   : 0.00   Min.   :  8.00   94720  : 163   1:1462
 2      :   1   1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   94305  : 125   2:1270
 3      :   1   Median :46.00   Median :20.00   Median : 64.00   95616  : 115   3:1000
 4      :   1   Mean   :45.55   Mean   :20.32   Mean   : 73.77   90095  :  70   4:1198
 5      :   1   3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   93106  :  56
 6      :   1   Max.   :67.00   Max.   :43.00   Max.   :224.00   92037  :  54
 (Other):4924                                                    (Other):4347
     CCAvg        Education    Mortgage      PersonalLoan SecuritiesAccount CDAccount Online
 Min.   : 0.000   1:2072    Min.   :  0.00   0:4452       0:4417            0:4630    0:1991
 1st Qu.: 0.700   2:1383    1st Qu.:  0.00   1: 478       1: 513            1: 300    1:2939
 Median : 1.500   3:1475    Median :  0.00
 Mean   : 1.938             Mean   : 56.68
 3rd Qu.: 2.600             3rd Qu.:101.00
 Max.   :10.000             Max.   :635.00

 CreditCard
 0:3480
 1:1450
```

3.5 Bi-Variate Analysis

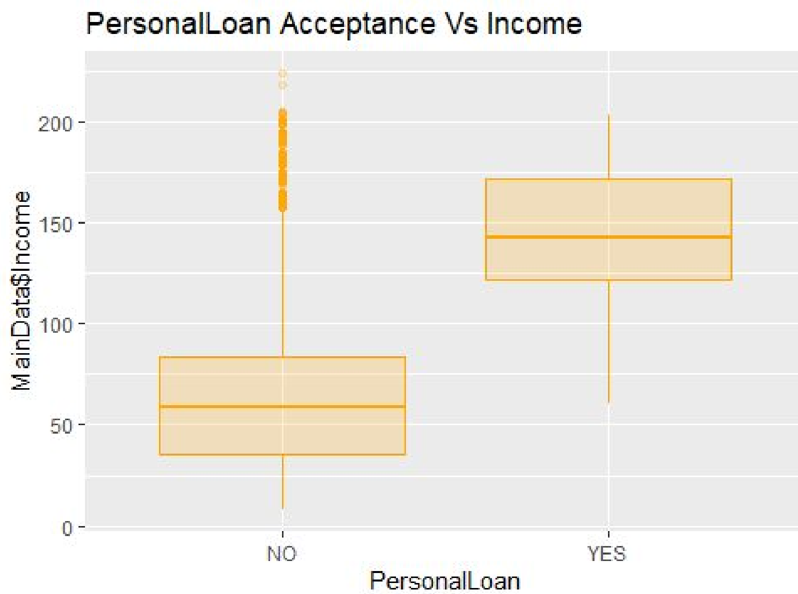
We will do Bivariate Analysis for all the Dependent Variable with Independent Variable.

3.5.1 Personal Loan vs Age

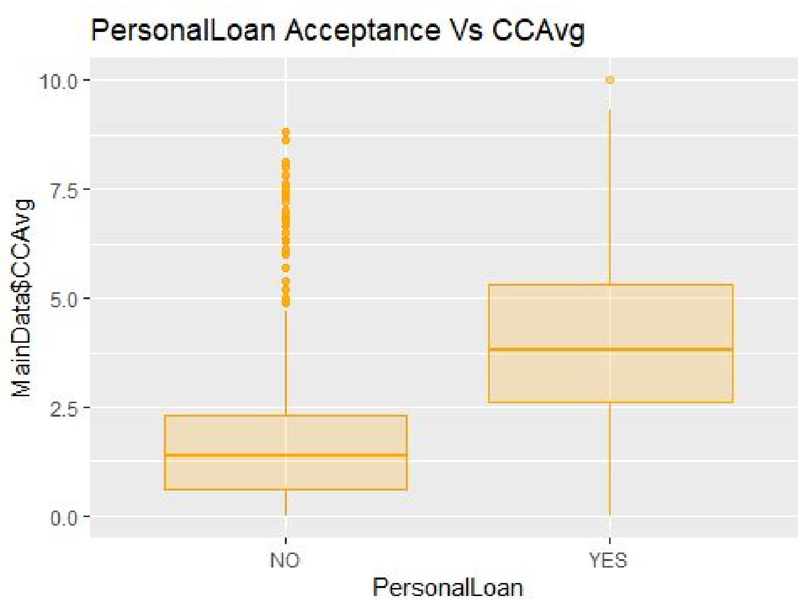


3.5.2 Personal Loan vs Experience

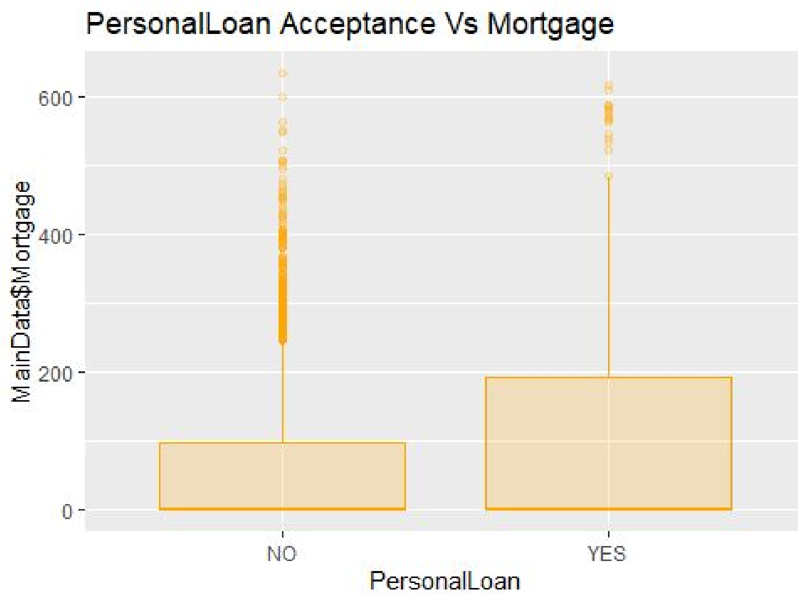
3.5.3 Personal Loan vs Income



3.5.4 Personal Loan vs CCAvg

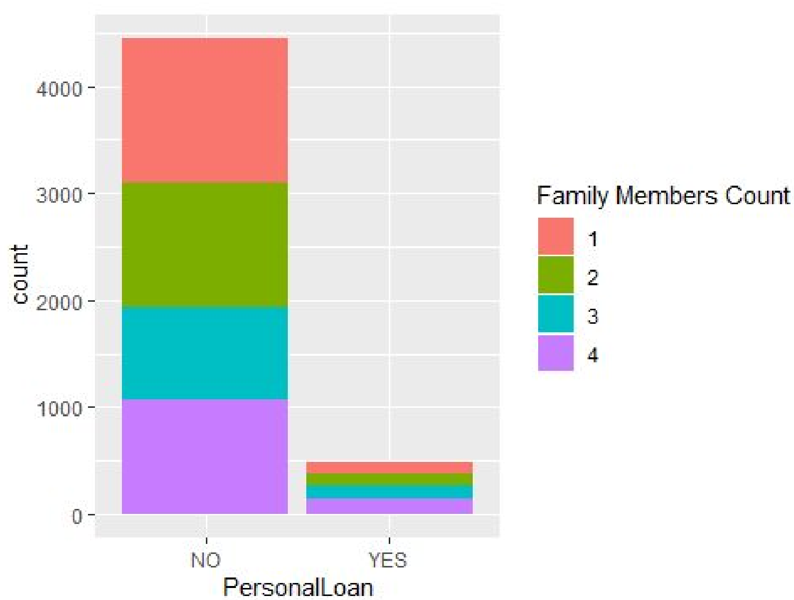


3.5.5 Personal Loan vs Mortgage



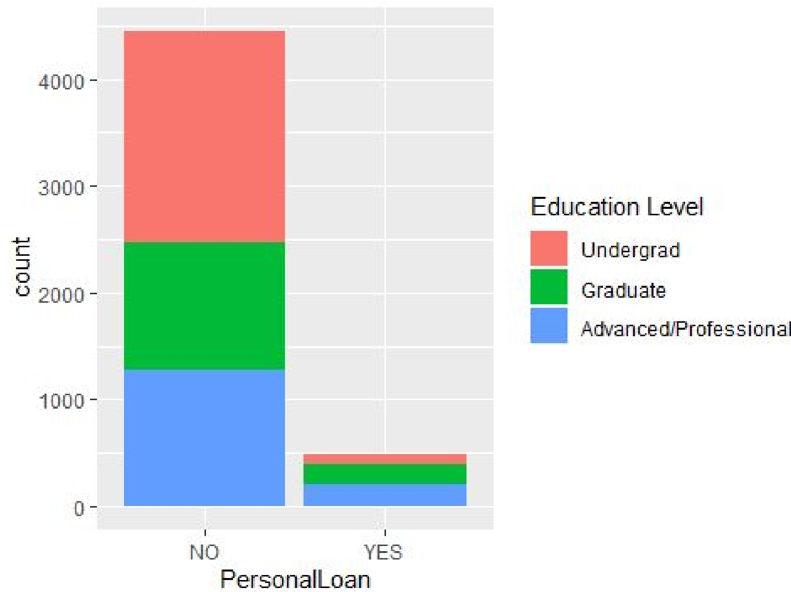
3.5.6 Personal Loan vs Family Count

	(Family Count)			
(Personal Loan)	1	2	3	4
No	1356	1164	867	1065
Yes	106	106	133	133



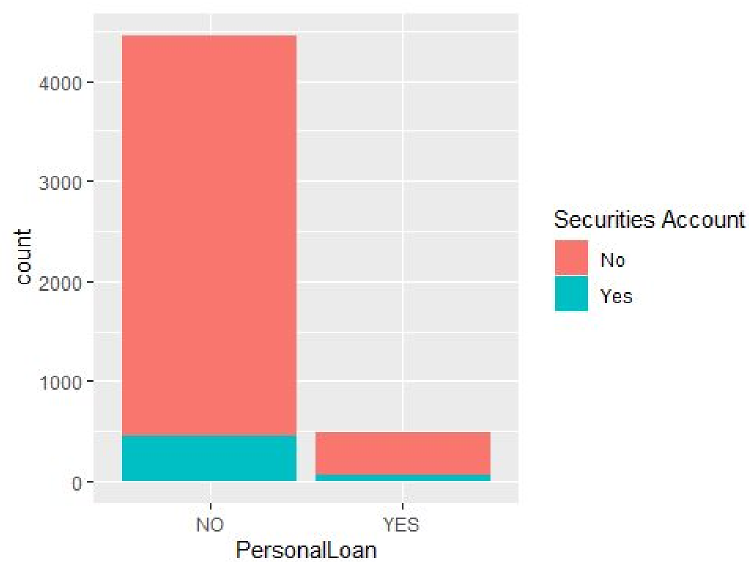
3.5.7 Personal Loan vs Education

	(Education Level)		
(Personal Loan)	UnderGrad	Grad	Professional
No	1979	1202	1271
Yes	93	181	204



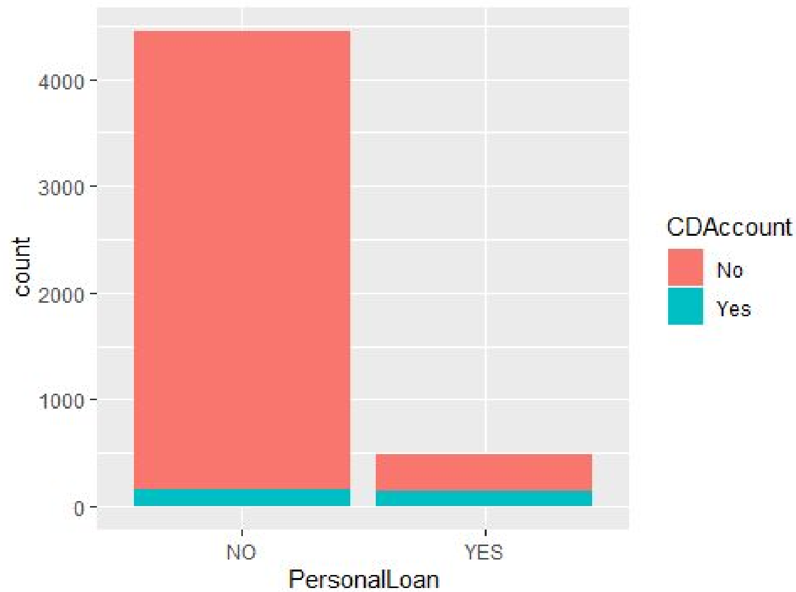
3.5.8 Personal Loan vs Security Account

	(Security Account)	
(Personal Loan)	No	Yes
No	3999	453
Yes	418	60



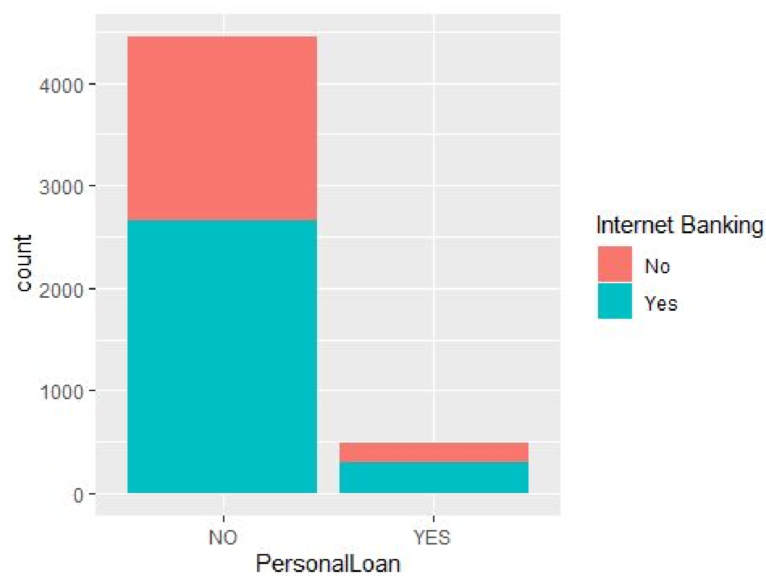
3.5.9 Personal Loan vs CDAccount

	(Security Account)	
(Personal Loan)	No	Yes
No	4291	161
Yes	339	139



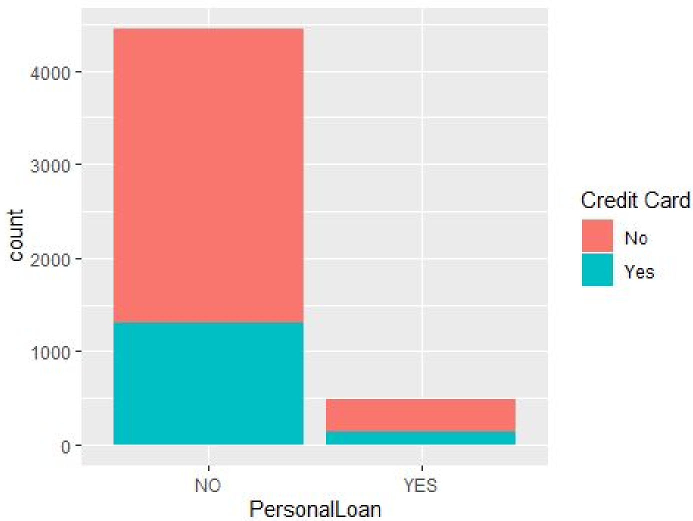
3.5.10 Personal Loan vs Online

	(Internet Banking)	
Personal Loan	No	Yes
No	1802	2650
Yes	189	289

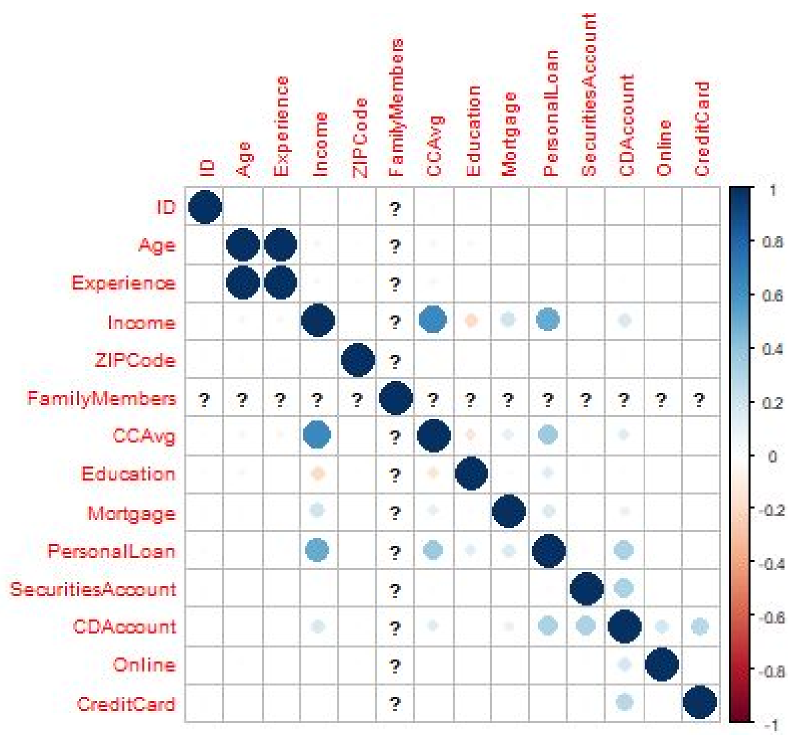


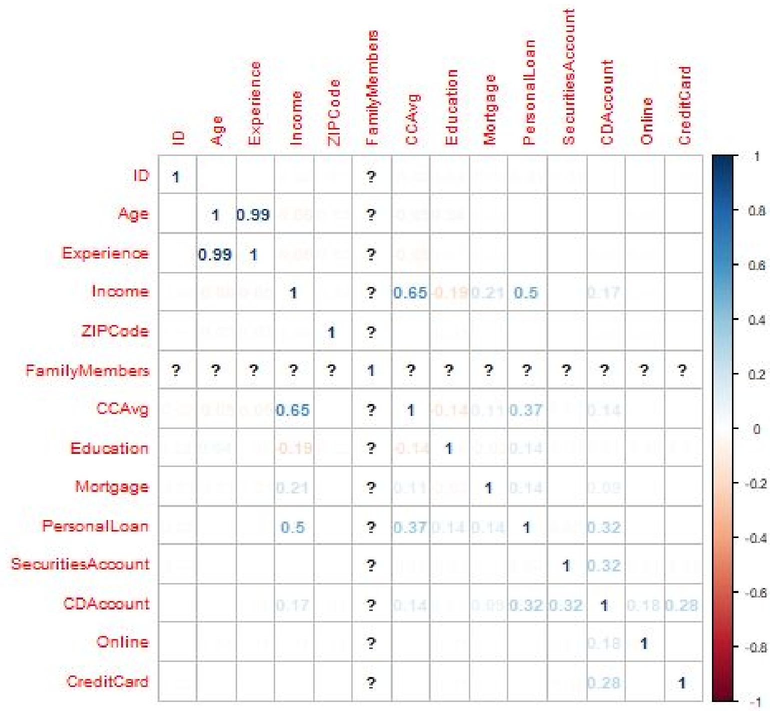
3.5.11 Personal Loan vs CreditCard

	(Credit Card)	
(Personal Loan)	No	Yes
No	3144	1308
Yes	336	142



3.6 Correlation Plot





Age and Experience have high correlation
Income & ccAvg Have high Correlation

Code for Bivariate Analysis

```
> ###PersonalLoan vs Age
> ggplot(MainData, aes(x=PersonalLoan, y=MainData$Age)) +
+   geom_boxplot(color="orange", fill="orange", alpha=0.2) +
+   scale_x_discrete(labels = c('NO','YES'))+
+   labs(title = "PersonalLoan Acceptance Vs Age")
> ###PersonalLoan vs Experience
> ggplot(MainData, aes(x=PersonalLoan, y=MainData$Experience)) +
+   geom_boxplot(color="orange", fill="orange", alpha=0.2) +
+   scale_x_discrete(labels = c('NO','YES'))+
+   labs(title = "PersonalLoan Acceptance Vs Experience")
> ###PersonalLoan vs Income
> ggplot(MainData, aes(x=PersonalLoan, y=MainData$Income)) +
+   geom_boxplot(color="orange", fill="orange", alpha=0.2) +
+   scale_x_discrete(labels = c('NO','YES'))+
+   labs(title = "PersonalLoan Acceptance Vs Income")
> ###PersonalLoan vs CCAvg
> ggplot(MainData, aes(x=PersonalLoan, y=MainData$CCAvg)) +
+   geom_boxplot(color="orange", fill="orange", alpha=0.2) +
+   scale_x_discrete(labels = c('NO','YES'))+
+   labs(title = "PersonalLoan Acceptance Vs CCAvg")
> ###PersonalLoan vs Mortgage
> ggplot(MainData, aes(x=PersonalLoan, y=MainData$Mortgage)) +
+   geom_boxplot(color="orange", fill="orange", alpha=0.2) +
+   scale_x_discrete(labels = c('NO','YES'))+
+   labs(title = "PersonalLoan Acceptance Vs Mortgage")
> ##PersonalLoan vs Family
> table(MainData$PersonalLoan, MainData$FamilyMembers)
```

```
     1    2    3    4
  0 1356 1164  867 1065
  1  106  106  133  133
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = FamilyMembers))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "Family Members Count")
> ##PersonalLoan vs Education
> table(MainData$PersonalLoan, MainData$Education)

     1    2    3
  0 1979 1202 1271
  1   93  181  204
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = Education))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "Education Level", labels = c("Undergrad",
"Graduate","Advanced/Professional"))
> ##PersonalLoan vs Security Account
> table(MainData$PersonalLoan, MainData$SecuritiesAccount)

     0    1
  0 3999  453
  1  418   60
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = SecuritiesAccount))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "Securities Account", labels = c("No", "Yes"))
> ##PersonalLoan vs CDAccount
> table(MainData$PersonalLoan, MainData$CDAccount)

     0    1
  0 4291  161
  1  339  139
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = CDAccount))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "CDAccount", labels = c("No", "Yes"))
> ##PersonalLoan vs Online
> table(MainData$PersonalLoan, MainData$Online)

     0    1
  0 1802 2650
  1  189  289
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = Online))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "Internet Banking", labels = c("No", "Yes"))
> ##PersonalLoan vs CreditCard
> table(MainData$PersonalLoan, MainData$CreditCard)

     0    1
  0 3144 1308
```

```
  1  336  142
> ###BarPlot
> ggplot(MainData, aes(x = PersonalLoan, fill = CreditCard))+
+   geom_bar(position = 'stack')+
+   scale_x_discrete(labels = c('NO','YES'))+
+   scale_fill_discrete(name = "Credit Card", labels = c("No", "Yes"))
> ##Corelation Plot
> Data_cor <- cor(TheraData)
> cex.before <- par("cex")
> par(cex = 0.6 )
> corrplot(Data_cor)
> corrplot(Data_cor, method = "number" , number.digits = 2 )
> par(cex = cex.before)
```

4 CART Model Building

To start with CART model building we need to build two dataset for test and train and id needs to remove.

Following are the two data set with dimension

- **Test**
  - Dimension (3451*13)
  - Distribution of Personal Loan (No - 3116 Yes - 335) i.e (No - 0.902% Yes - 0.097)
- **Train**
  - Dimension (1479*13)
  - Distribution of Personal Loan (No - 1336 Yes - 143) i.e (No - 0.903% Yes - 0.096)

Train Data is ready to build CART Model certain input are required to built CART :

- **minbucket :** minimum records in a terminal node, Generally between 2-3% of Data
  - minbuket = 70
- **minsplit :** 3(minbucket) = 210
- **cp** = 0
- **xval** = 10

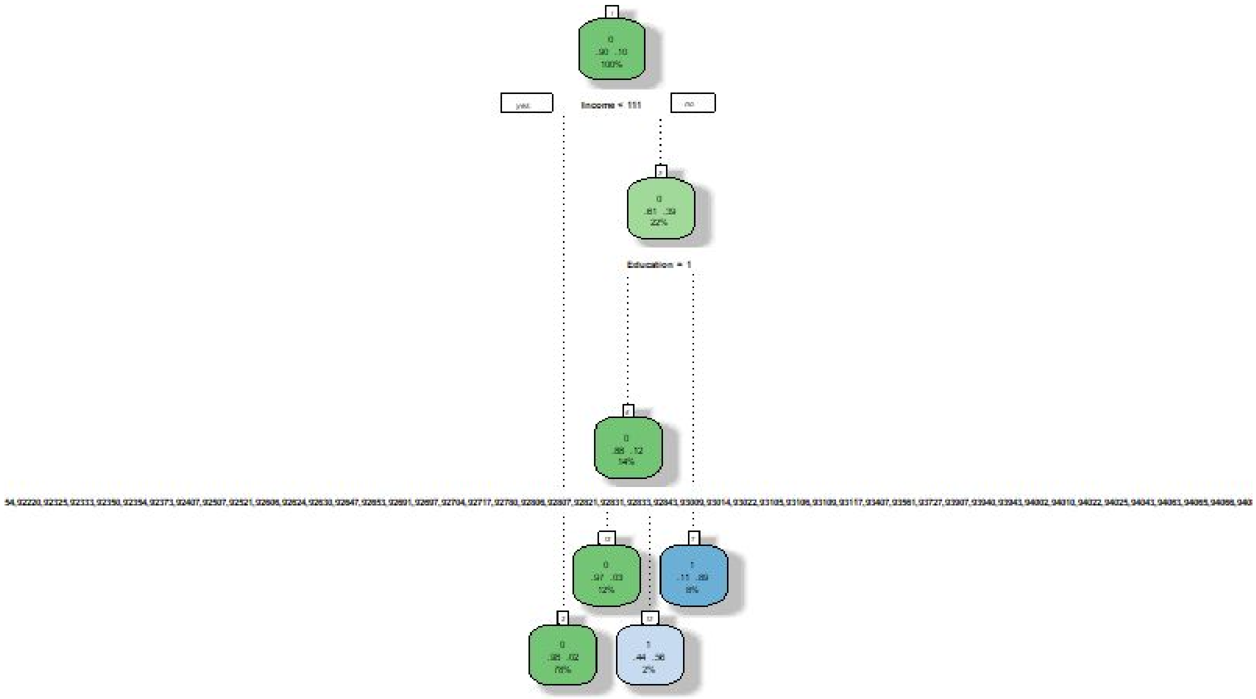
4.1 Initial Model

Initial CART Model

```
> train.t
n= 3451

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

 1) root 3451 335 0 (0.90292669 0.09707331)
   2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
   3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
     6) Education=1 484  57 0 (0.88223140 0.11776860)
      12)
ZIPCode=90005,90007,90009,90024,90025,90027,90028,90029,90034,90035,90036,90045,90064,90065,90066,90071,90086,90095,90212,90230,90232,90245,90250,90254,90266,90274,90277,90401,90405,90502,90503,90630,90638,90720,90740,91024,91030,91040,91107,91125,91203,91302,91304,91311,91320,91326,91335,91342,91360,91367,91380,91423,91604,91605,91710,91711,91763,91765,91768,91770,91801,91902,91911,91942,92007,92009,92028,92037,92056,92096,92104,92115,92120,92121,92123,92126,92152,92154,92220,92325,92333,92350,92354,92373,92407,92507,92521,92606,92624,92630,92647,92653,92691,92697,92704,92717,92780,92806,92807,92821,92831,92833,92843,93009,93014,93022,93105,93106,93109,93117,93407,93561,93727,93907,93940,93943,94002,94010,94022,94025,94043,94063,94065,94066,94080,94085,94102,94104,94105,94110,94111,94112,94117,94123,94131,94132,94234,94301,94303,94305,94402,94501,94507,94521,94539,94542,94545,94546,94550,94551,94566,94571,94575,94577,94583,94591,94596,94606,94607,94608,94609,94611,94701,94706,94709,94720,94801,94920,94949,94960,94998,95003,95014,95023,95035,95037,95039,95051,95053,95054,95064,95112,95120,95133,95136,95138,95193,95307,95348,95351,95449,95521,95616,95621,95630,95747,95762,95814,95819,95827,95833,95842,96001,96091 406  13 0
(0.96798030 0.03201970) *
      13)
ZIPCode=90032,90089,90210,90840,91103,91330,91355,91614,92038,92093,92106,92110,92122,92182,92612,92646,
```

```
92675,92677,93108,93302,93955,94108,94115,94122,94143,94304,94309,94590,94705,94707,94803,94904,95020,95060,95125,95135,95605,95818,95841 78  34 1 (0.43589744 0.56410256) *
     7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
```

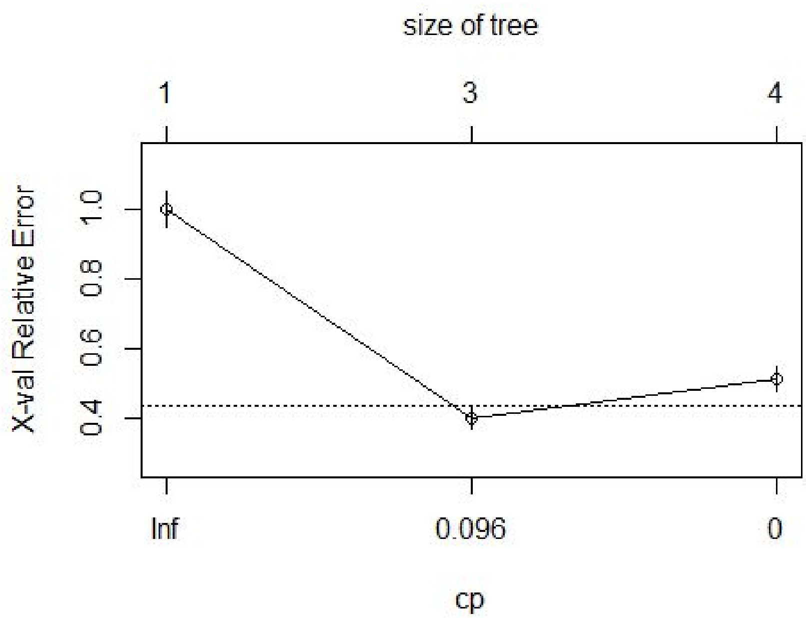
4.1.1 CART Model Plot



Rattle 2019-Nov-24 05:59:26 Digital Pankit

4.1.2 CP Table

```
Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

Variables actually used in tree construction:
[1] Education Income    ZIPCode

Root node error: 335/3451 = 0.097073

n= 3451

        CP nsplit rel error  xerror     xstd
1 0.307463      0   1.00000 1.00000 0.051916
2 0.029851      2   0.38507 0.40000 0.033877
3 0.000000      3   0.35522 0.51045 0.038056
```



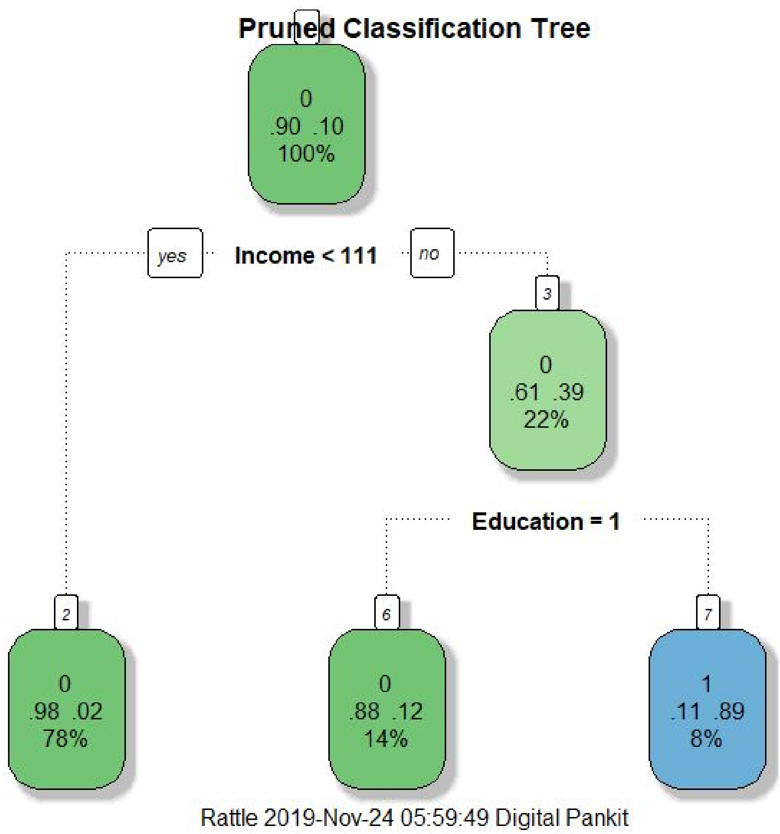
4.2 Pruned CART Tree

xerror of the tree is increasing at cp 0.029851 therefore we can prune the tree at CP 0.030.

4.2.1 Pruned Tree

```
>train.tree
n= 3451

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

1) root 3451 335 0 (0.90292669 0.09707331)
  2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
  3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
    6) Education=1 484  57 0 (0.88223140 0.11776860) *
    7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
```

4.2.2 Pruned Tree Plot



4.3 Model Interpretation

The 89% of Customers of Thera Bank having Income > 111K and Education level is either Graduate or Professional took personal loan from the Bank (Which is 8% Total Population).

4.4 Model Performance Measures

4.4.1 Confusion Matrix

Train DataSet

- Accuracy : 0.9655
- Sensitivity : 0.9823

	Reference	
Prediction	0	1
0	3052	64
1	55	280

```
> confusionMatrix(train$PersonalLoan,train$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3052   64
         1   55  280

               Accuracy : 0.9655
                 95% CI : (0.9589, 0.9714)
    No Information Rate : 0.9003
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8056

 Mcnemar's Test P-Value : 0.4633

            Sensitivity : 0.9823
            Specificity : 0.8140
         Pos Pred Value : 0.9795
         Neg Pred Value : 0.8358
             Prevalence : 0.9003
         Detection Rate : 0.8844
   Detection Prevalence : 0.9029
      Balanced Accuracy : 0.8981

       'Positive' Class : 0
```

Our Model is Performing Good on Train DataSet

Test DataSet

- Accuracy : 0.95
- Sensitivity : 0.9723

	Reference	
Prediction	0	1
0	1299	37
1	37	106

```
> confusionMatrix(test$PersonalLoan,test$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1299   37
         1   37  106

               Accuracy : 0.95
                 95% CI : (0.9376, 0.9605)
    No Information Rate : 0.9033
    P-Value [Acc > NIR] : 2.57e-11

                  Kappa : 0.7136

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9723
            Specificity : 0.7413
         Pos Pred Value : 0.9723
         Neg Pred Value : 0.7413
             Prevalence : 0.9033
         Detection Rate : 0.8783
   Detection Prevalence : 0.9033
      Balanced Accuracy : 0.8568

       'Positive' Class : 0
```

Interpretation

Results on both Train and Test Data are same and Performance is also good which validates our CART model

4.4.2 Other Performance Matrix

Train Data Set

High AUC, KS, Gini Coeff, and Concordance is high presents good performance of Model

KS - 0.8152

AUC - 0.919

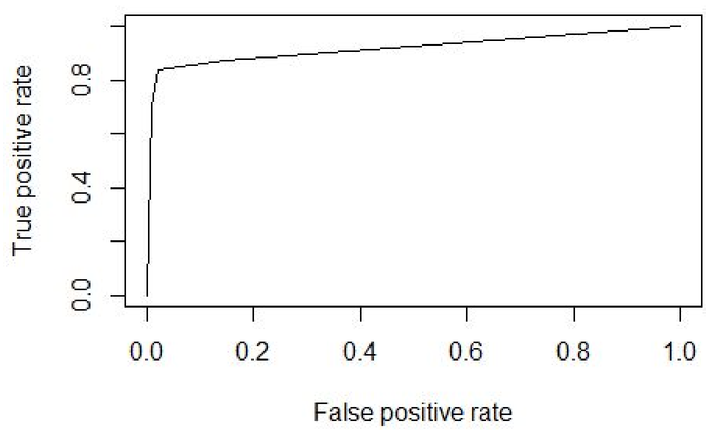
Gini - 0.757

Concordance - 0.8594553

Discordance - 0.1405447

```
> print(rankTbl)
          deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:  [0.032,0.887]  750      293      457 39.07      293          457        87.46            14.67
2: [0.0155,0.032) 2701       42     2659  1.55      335         3116       100.00           100.00
      ks
1: 72.79
2:  0.00
> predObj = prediction(train$prob1, train$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
```

```
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.8152817
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.9195031
> gini = ineq(train$prob1, type="Gini")
> gini
[1] 0.7575611
> ### Concordance and discordcance ratios:
> Concordance(actuals=train$PersonalLoan, predictedScores=train$prob1)
$Concordance
[1] 0.8594553

$Discordance
[1] 0.1405447

$Tied
[1] 2.775558e-17

$Pairs
[1] 1043860
```



Test Data Set

High AUC, KS, Gini Coeff, and Concordance is high presents good performance of Model

KS - 0.71

AUC - 0.88

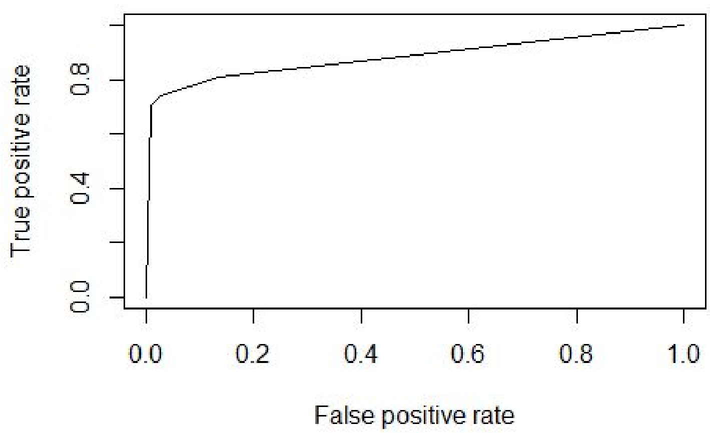
Gini - 0.75

Concordance - 0.79

Discordance - 0.20

```
> print(rankTbl)
            deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp
1:   [0.032,0.887]  296      116      180 39.19      116          180        81.12
2: [0.0155,0.0221) 1183       27     1156  2.28      143         1336       100.00
   cum_rel_non_resp    ks
```

```
1:            13.47 67.65
2:           100.00  0.00
> predObj = prediction(test$prob1, test$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.7135641
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.8836601
> gini = ineq(test$prob1, type="Gini")
> gini
[1] 0.7568962
> ### Concordance and discordcance ratios:
> library("InformationValue")
> Concordance(actuals=test$PersonalLoan, predictedScores=test$prob1)
$Concordance
[1] 0.7949835
$Discordance
[1] 0.2050165
$Tied
[1] 2.775558e-17
$Pairs
[1] 191048
```



Results for KS, Gini Coeff, AUC on both Train and Test Data are similar and indicate good Performance of our CART model

Output for CART

```
> LoanData = MainData[,c(-1)]
> LoanData$ZIPCode = as.factor(LoanData$ZIPCode)
> summary(LoanData)
      Age          Experience        Income          ZIPCode      FamilyMembers     CCAvg
 Min.   :24.00   Min.   : 0.00   Min.   :  8.00   94720  : 163   1:1462         Min.   : 0.000
 1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   94305  : 125   2:1270         1st Qu.: 0.700
 Median :46.00   Median :20.00   Median : 64.00   95616  : 115   3:1000         Median : 1.500
 Mean   :45.55   Mean   :20.32   Mean   : 73.77   90095  :  70   4:1198         Mean   : 1.938
```

```
 3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   93106   :  56                  3rd Qu.: 2.600
 Max.   :67.00   Max.   :43.00   Max.   :224.00   92037   :  54                  Max.   :10.000
                                                  (Other):4347
 Education    Mortgage      PersonalLoan SecuritiesAccount CDAccount Online   CreditCard
 1:2072    Min.   :  0.00   0:4452       0:4417            0:4630    0:1991   0:3480
 2:1383    1st Qu.:  0.00   1: 478       1: 513            1: 300    1:2939   1:1450
 3:1475    Median :  0.00
           Mean   : 56.68
           3rd Qu.:101.00
           Max.   :635.00

> names(LoanData)
 [1] "Age"                "Experience"         "Income"             "ZIPCode"
 [5] "FamilyMembers"      "CCAvg"              "Education"          "Mortgage"
 [9] "PersonalLoan"       "SecuritiesAccount" "CDAccount"          "Online"
[13] "CreditCard"
> str(LoanData)
'data.frame':	4930 obs. of  13 variables:
 $ Age              : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience       : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIPCode          : Factor w/ 467 levels "9307","90005",..: 84 35 368 299 97 161 116 268 35 236 ...
 $ 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        : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : int  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 ...
> ##Creating Test and Train Data
> split <- sample.split(LoanData$PersonalLoan, SplitRatio = 0.7)
> train<- subset(LoanData, split == TRUE)
> test<- subset( LoanData, split == FALSE)
> prop.table(table(train$PersonalLoan))

         0          1 
0.90292669 0.09707331 
> prop.table(table(test$PersonalLoan))

         0          1 
0.90331305 0.09668695 
> table(train$PersonalLoan)

   0    1 
3116  335 
> table(test$PersonalLoan)

   0    1 
1336  143 
> attach(train)
The following objects are masked from train (pos = 4):

    Age, CCAvg, CDAccount, CreditCard, Education, Experience, FamilyMembers, Income,
    Mortgage, Online, PersonalLoan, SecuritiesAccount, ZIPCode

> View(train)
> str(train)
```

```
'data.frame':	3451 obs. of  13 variables:
 $ Age              : int  25 45 35 37 53 50 35 34 65 29 ...
 $ Experience       : int  1 19 9 13 27 24 10 9 39 5 ...
 $ Income           : int  49 34 100 29 72 22 81 180 105 45 ...
 $ ZIPCode          : Factor w/ 467 levels "9307","90005",..: 84 35 299 161 116 268 35 236 367 48 ...
 $ FamilyMembers    : Factor w/ 4 levels "1","2","3","4": 4 3 1 4 2 1 3 1 4 3 ...
 $ CCAvg            : num  1.6 1.5 2.7 0.4 1.5 0.3 0.6 8.9 2.4 0.1 ...
 $ Education        : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 3 2 3 3 2 ...
 $ Mortgage         : int  0 0 0 155 0 0 104 0 0 0 ...
 $ PersonalLoan     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ 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 2 2 1 2 1 1 2 ...
 $ CreditCard       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
> dim(train)
[1] 3451   13
> names(train)
 [1] "Age"               "Experience"        "Income"            "ZIPCode"
 [5] "FamilyMembers"     "CCAvg"             "Education"         "Mortgage"
 [9] "PersonalLoan"      "SecuritiesAccount" "CDAccount"         "Online"
[13] "CreditCard"
> ##Setting the control parameters for rpart
> #minsplit: if the number of records in a given node falls below a threshold, the node will not be
split further.
> #minbucket: minimum records in a terminal node. if the records are less, that bucket will not be
created.
> #Terminal node (minbucket) should not be less than 2-3% of starting population.
> 0.02*3451
[1] 69.02
> 0.03*3451
[1] 103.53
> #minsplit = 3(minbucket)
> #xval divides the entire dataset into mutually exclusive and collectively exhaustive segments.
> #Model is built on xval-1 segments and 1 is used for testing.
> #cp = cost complexity parameter
> r.ctrl = rpart.control(minsplit=210, minbucket = 70, cp = 0, xval = 10)
> train.t <- rpart(formula = PersonalLoan ~ ., data = train[,-c(9)], method = "class", control = r.ctrl)
> train.t
n= 3451

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

 1) root 3451 335 0 (0.90292669 0.09707331)
   2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
   3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
     6) Education=1 484  57 0 (0.88223140 0.11776860)
      12)
ZIPCode=90005,90007,90009,90024,90025,90027,90028,90029,90034,90035,90036,90045,90064,90065,90066,90071,
90086,90095,90212,90230,90232,90245,90250,90254,90266,90274,90277,90401,90405,90502,90503,90630,90638,90
720,90740,91024,91030,91040,91107,91125,91203,91302,91304,91311,91320,91326,91335,91342,91360,91367,9138
0,91423,91604,91605,91710,91711,91763,91765,91768,91770,91801,91902,91911,91942,92007,92009,92028,92037,
92056,92096,92104,92115,92120,92121,92123,92126,92152,92154,92220,92325,92333,92350,92354,92373,92407,92
507,92521,92606,92624,92630,92647,92653,92691,92697,92704,92717,92780,92806,92807,92821,92831,92833,9284
3,93009,93014,93022,93105,93106,93109,93117,93407,93561,93727,93907,93940,93943,94002,94010,94022,94025,
94043,94063,94065,94066,94080,94085,94102,94104,94105,94110,94111,94112,94117,94123,94131,94132,94234,94
301,94303,94305,94402,94501,94507,94521,94539,94542,94545,94546,94550,94551,94566,94571,94575,94577,9458
3,94591,94596,94606,94607,94608,94609,94611,94701,94706,94709,94720,94801,94920,94949,94960,94998,95003,
95014,95023,95035,95037,95039,95051,95053,95054,95064,95112,95120,95133,95136,95138,95193,95307,95348,95
```

```
351,95449,95521,95616,95621,95630,95747,95762,95814,95819,95827,95833,95842,96001,96091 406  13 0
(0.96798030 0.03201970) *
      13)
ZIPCode=90032,90089,90210,90840,91103,91330,91355,91614,92038,92093,92106,92110,92122,92182,92612,92646,
92675,92677,93108,93302,93955,94108,94115,94122,94143,94304,94309,94590,94705,94707,94803,94904,95020,95
060,95125,95135,95605,95818,95841 78  34 1 (0.43589744 0.56410256) *
     7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
> fancyRpartPlot(train.t)
> ##To see how the tree performs
> printcp(train.t)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

Variables actually used in tree construction:
[1] Education Income    ZIPCode

Root node error: 335/3451 = 0.097073

n= 3451

        CP nsplit rel error  xerror     xstd
1 0.307463      0   1.00000 1.00000 0.051916
2 0.029851      2   0.38507 0.40000 0.033877
3 0.000000      3   0.35522 0.51045 0.038056
> plotcp(train.t)
> ##Since Vlaue of x error start increasing we have to prune the tree at cp = 0.030
> train.tree<- prune(train.t, cp=  0.030  ,"CP")
> printcp(train.tree)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

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

Root node error: 335/3451 = 0.097073

n= 3451

       CP nsplit rel error xerror     xstd
1 0.30746      0   1.00000    1.0 0.051916
2 0.03000      2   0.38507    0.4 0.033877
> fancyRpartPlot(train.tree, uniform=TRUE,  main="Pruned Classification Tree")
> ##Scoring
> train$predict.class = predict(train.t, train, type="class")
> train$predict.score = predict(train.t, train)
> library(caret)
> train$predict.class =as.factor(train$predict.class)
> train$PersonalLoan=as.factor(train$PersonalLoan)
> confusionMatrix(train$PersonalLoan,train$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3052   64
         1   55  280
```

```
               Accuracy : 0.9655
                 95% CI : (0.9589, 0.9714)
    No Information Rate : 0.9003
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8056

 Mcnemar's Test P-Value : 0.4633

            Sensitivity : 0.9823
            Specificity : 0.8140
         Pos Pred Value : 0.9795
         Neg Pred Value : 0.8358
             Prevalence : 0.9003
         Detection Rate : 0.8844
   Detection Prevalence : 0.9029
      Balanced Accuracy : 0.8981

       'Positive' Class : 0

> #Scoring the test sample
> test$predict.class <- predict(train.t, test, type="class")
> test$predict.score <- predict(train.t, test)
> test$predict.class <-as.factor(test$predict.class)
> test$PersonalLoan<-as.factor(test$PersonalLoan)
> confusionMatrix(test$PersonalLoan,test$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1299   37
         1   37  106

               Accuracy : 0.95
                 95% CI : (0.9376, 0.9605)
    No Information Rate : 0.9033
    P-Value [Acc > NIR] : 2.57e-11

                  Kappa : 0.7136

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9723
            Specificity : 0.7413
         Pos Pred Value : 0.9723
         Neg Pred Value : 0.7413
             Prevalence : 0.9033
         Detection Rate : 0.8783
   Detection Prevalence : 0.9033
      Balanced Accuracy : 0.8568

       'Positive' Class : 0

> dim(test)
[1] 1479   15
> dim(LoanData)
[1] 4930   13
> train.tree
```

```
n= 3451

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

1) root 3451 335 0 (0.90292669 0.09707331)
  2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
  3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
    6) Education=1 484  57 0 (0.88223140 0.11776860) *
    7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
> plotcp(train.t)
> ##To see how the tree performs
> printcp(train.t)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

Variables actually used in tree construction:
[1] Education Income    ZIPCode

Root node error: 335/3451 = 0.097073

n= 3451

        CP nsplit rel error  xerror     xstd
1 0.307463      0   1.00000 1.00000 0.051916
2 0.029851      2   0.38507 0.40000 0.033877
3 0.000000      3   0.35522 0.51045 0.038056
> plotcp(train.t)
> plotcp(train.t)
> plotcp(train.tree)
> plotcp(train.t)
> fancyRpartPlot(train.tree, uniform=TRUE,  main="Pruned Classification Tree")
> train.tree
n= 3451

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

1) root 3451 335 0 (0.90292669 0.09707331)
  2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
  3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
    6) Education=1 484  57 0 (0.88223140 0.11776860) *
    7) Education=2,3 266  30 1 (0.11278195 0.88721805) *


> ##### Performance Matrix of CART Train
> ### Deciling and Rank Order Table
> str(train)
'data.frame':	3451 obs. of  17 variables:
 $ Age             : int  25 45 35 37 53 50 35 34 65 29 ...
 $ Experience      : int  1 19 9 13 27 24 10 9 39 5 ...
 $ Income          : int  49 34 100 29 72 22 81 180 105 45 ...
 $ ZIPCode         : Factor w/ 467 levels "9307","90005",..: 84 35 299 161 116 268 35 236 367 48 ...
 $ FamilyMembers   : Factor w/ 4 levels "1","2","3","4": 4 3 1 4 2 1 3 1 4 3 ...
 $ CCAvg           : num  1.6 1.5 2.7 0.4 1.5 0.3 0.6 8.9 2.4 0.1 ...
 $ Education       : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 3 2 3 3 2 ...
 $ Mortgage        : int  0 0 0 155 0 0 104 0 0 0 ...
```

```
 $ PersonalLoan     : num  0 0 0 0 0 0 0 1 0 0 ...
 $ 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 2 2 1 2 1 1 2 ...
 $ CreditCard       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
 $ predict.class    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ predict.score    : num [1:3451, 1:2] 0.984 0.984 0.984 0.984 0.984 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr  "1" "2" "4" "6" ...
  .. ..$ : chr  "0" "1"
 $ prob1            : num  0.0155 0.0155 0.0155 0.0155 0.0155 ...
 $ deciles          : Factor w/ 2 levels "[0.0155,0.032)",..: 1 1 1 1 1 1 1 2 1 1 ...
> train$prob1 <-train$predict.score[,2]
> test$prob1 <-test$predict.score[,2]
> probsCart=seq(0,1,length=11)
> probsCart
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qsCart=quantile(train$prob1, probsCart)
> qsCart
       0%       10%       20%       30%       40%       50%       60%       70%       80% 
0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0320197 
      90%      100% 
0.0320197 0.8872180 
> train$deciles=cut(train$prob1, unique(qsCart),include.lowest = TRUE,right=FALSE)
> table(train$deciles)

[0.0155,0.032)  [0.032,0.887] 
          2701            750 
> train$PersonalLoan <-ifelse(train$PersonalLoan == "1", 1,0)
> train$PersonalLoan <-as.numeric(train$PersonalLoan)
> test$PersonalLoan <-ifelse(test$PersonalLoan == "1", 1,0)
> test$PersonalLoan <-as.numeric(test$PersonalLoan)
> trainDT = data.table(train)
> rankTbl = trainDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan == 1),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> print(rankTbl)
          deciles  cnt cnt_tar1 cnt_tar0
1:  [0.032,0.887]  750      293      457
2: [0.0155,0.032) 2701       42     2659
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
> print(rankTbl)
          deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:  [0.032,0.887]  750      293      457 39.07      293          457        87.46            14.67
2: [0.0155,0.032) 2701       42     2659  1.55      335         3116       100.00           100.00
      ks
1: 72.79
2:  0.00
> predObj = prediction(train$prob1, train$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
```

```
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.8152817
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.9195031
> gini = ineq(train$prob1, type="Gini")
> gini
[1] 0.7575611
> ### Concordance and discordcance ratios:
> Concordance(actuals=train$PersonalLoan, predictedScores=train$prob1)
$Concordance
[1] 0.8594553

$Discordance
[1] 0.1405447

$Tied
[1] 2.775558e-17

$Pairs
[1] 1043860

> ##### Performance Matrix of CART Test
> ### Deciling and Rank Order Table
> probs=seq(0,1,length=11)
> probs
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qs=quantile(test$prob1, probs)
> qs
        0%        10%        20%        30%        40%        50%        60%        70%        80%
0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.02213776
       90%       100%
0.03201970 0.88721805
> test$deciles=cut(test$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
> table(test$deciles)

[0.0155,0.0221)  [0.0221,0.032)   [0.032,0.887]
           1183               0             296
> testDT = data.table(test)
> rankTbl = testDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
> print(rankTbl)
           deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp
1:   [0.032,0.887]  296      116      180 39.19      116          180        81.12
2: [0.0155,0.0221) 1183       27     1156  2.28      143         1336       100.00
   cum_rel_non_resp    ks
1:            13.47 67.65
```

```
2:           100.00  0.00
> predObj = prediction(test$prob1, test$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.7135641
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.8836601
> gini = ineq(test$prob1, type="Gini")
> gini
[1] 0.7568962
> ### Concordance and discordcance ratios:
> library("InformationValue")
> Concordance(actuals=test$PersonalLoan, predictedScores=test$prob1)
$Concordance
[1] 0.7949835

$Discordance
[1] 0.2050165

$Tied
[1] 2.775558e-17

$Pairs
[1] 191048
```

5. Random Forest

To start with Random Forest model building we need to build two dataset for test and train and id needs to remove.

Following are the two data set with dimension

- **RFTest**
  - Dimension (3451*13)
  - Distribution of Personal Loan (No - 3116 Yes - 335) i.e (No - 0.902% Yes - 0.097)
- **RFTrain**
  - Dimension (1479*13)
  - Distribution of Personal Loan (No - 1336 Yes - 143) i.e (No - 0.903% Yes - 0.096)

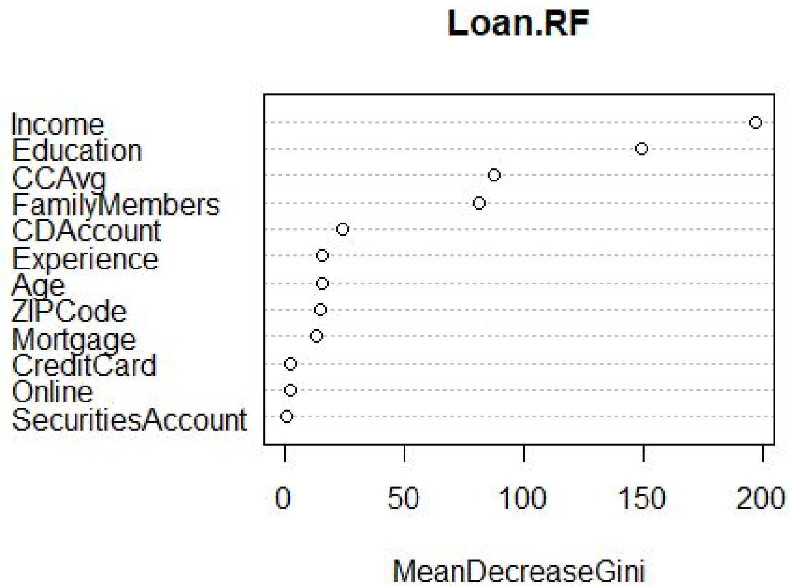
Train Data is ready to build Random Forest Model certain input are required:

- ntree: number of trees to grow = 101
- mtry: number of variables to be considered for split = 4

5.1 Initial Model Random Forest

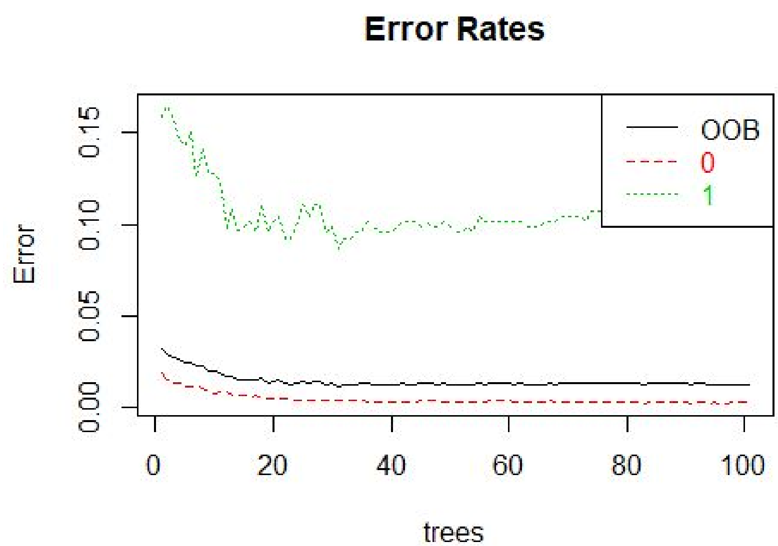
```
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                           ntree=101, mtry=5, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 101,
mtry = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 101
No. of variables tried at each split: 5

        OOB estimate of  error rate: 1.27%
Confusion matrix:
     0   1 class.error
0 3107   9 0.002888318
1   35 300 0.104477612

+ importance(Loan.RF, type=2)
                  MeanDecreaseGini
Age                      16.137480
Experience               16.242118
Income                  196.824539
ZIPCode                  15.182903
FamilyMembers            81.295277
CCAvg                    87.554281
Education               149.540846
Mortgage                 13.910754
SecuritiesAccount         1.034989
CDAccount                24.242179
Online                    2.464974
CreditCard                2.765016
> varImpPlot(Loan.RF, type=2)
```

OOB error Rate is 1.27% which is very less



Income,Education, CCAvg, Family Member are top 4 important factors for the Personal Loan

5.2 Random Forest Plot

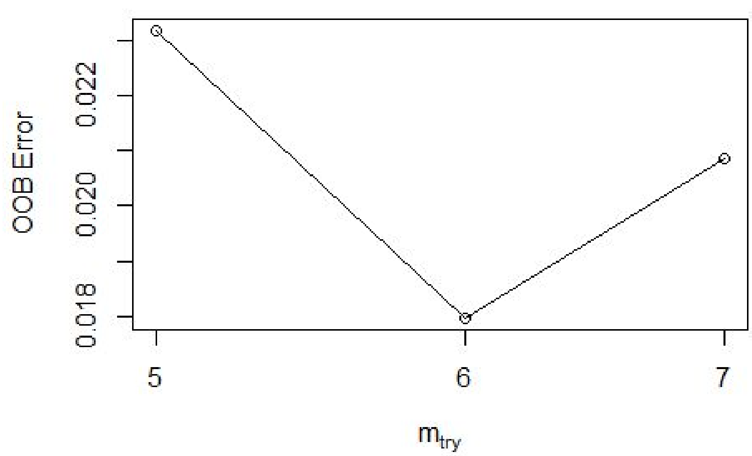


Error Rate Stabilize after 40 trees so we can restrict ntreetry to 30

5.3 Tuned Random forest

Random Forest will be optimal at mtry = 6 and ntreetry = 30

```
> #Tuning Random Forest
> tRF <- tuneRF(x = RFtrain[,-c(9)],
+               y=as.factor(RFtrain$PersonalLoan),
+               ntreeTry=30,
+               mtry=5,
+               stepFactor = 1.2,
+               improve = 0.0001,
+               trace=TRUE,
+               plot = TRUE,
+               doBest = TRUE,
+               nodesize = 100,
+               importance=TRUE
+ )
mtry = 5  OOB error = 2.32%
Searching left ...
Searching right ...
mtry = 6        OOB error = 1.8%
0.2247754 1e-04
mtry = 7        OOB error = 2.09%
-0.1609538 1e-04
```



Optimised Random Forest Tree

```
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                          ntree=30, mtry=6, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 30,
mtry = 6, importance = TRUE)
```

```
               Type of random forest: classification
                     Number of trees: 30
No. of variables tried at each split: 6

        OOB estimate of  error rate: 1.22%
Confusion matrix:
     0   1 class.error
0 3106  10 0.003209243
1   32 303 0.095522388
```

OOB estimate of error rate: 1.22% which is very less so our model is performing good

5.4 Confusion Matrix

Training DataSet

- Accuracy : 1
- Sensitivity : 1

Our Model has an accuracy of 100% which is good theoretically but practically it is capturing noise therefore we have to validate the results with Test Data

	Reference	
Prediction	0	1
0	3116	0
1	0	335

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3116    0
         1    0  335

               Accuracy : 1
                 95% CI : (0.9989, 1)
    No Information Rate : 0.9029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.9029
         Detection Rate : 0.9029
   Detection Prevalence : 0.9029
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

Test DataSet

- Accuracy : 0.98
- Sensitivity : 0.985

Our Model has an accuracy of 100% which is good theoretically but practically it is capturing noise also we will validate the results with Test Data

	Reference	
Prediction	0	1
0	1332	4
1	20	123

```
> confusionMatrix(RFtest$PersonalLoan , RFtest$predict.class )
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1332    4
         1   20  123

               Accuracy : 0.9838
                 95% CI : (0.976, 0.9896)
    No Information Rate : 0.9141
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9022

 Mcnemar's Test P-Value : 0.0022

            Sensitivity : 0.9852
            Specificity : 0.9685
         Pos Pred Value : 0.9970
         Neg Pred Value : 0.8601
             Prevalence : 0.9141
         Detection Rate : 0.9006
   Detection Prevalence : 0.9033
      Balanced Accuracy : 0.9769

       'Positive' Class : 0
```

So, On Test Data our model is performing with 0.98 accuracy with sensitivity of 0.985 similar to th train data performance, it validates our model

5.5 Other Performance Matrix

Test Data

Our Model has AUC, KS, Concordance as 1 which make our model 100% accurate theoretically but practically it is capturing noise Therefore we have to check the performance on test data.

KS - 1

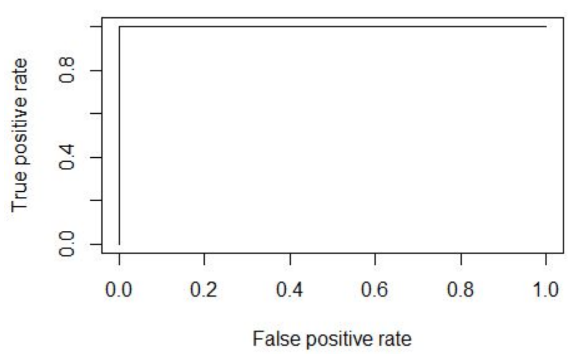
AUC - 1

Gini - 0.90

Concordance - 1

Discordance - 0

```
> print(rankTbl)
     deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1: [0.167,1]  351      335       16 95.44      335           16          100             0.51
2: [0,0.167) 3100        0     3100  0.00      335         3116          100           100.00
      ks
1: 99.49
2:  0.00
> predObj = prediction(RFtrain$prob1, RFtrain$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 1
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 1
> gini = ineq(RFtrain$prob1, type="Gini")
> gini
[1] 0.9006712
> Concordance(actuals=RFtrain$PersonalLoan, predictedScores=RFtrain$prob1)
$Concordance
[1] 1
$Discordance
[1] 0
$Tied
[1] 0
$Pairs
[1] 1043860
```



Test Data

Our Model has high AUC, KS, Concordance, Gini Coeff similar to train data which tell our model is also performing on equally on test data. So, it validate our Model.

KS - 0.94

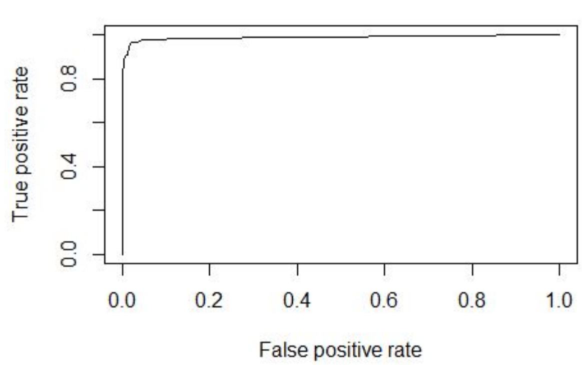
AUC - 0.99

Gini - 0.89

Concordance - 0.98

Discordance - 0.016

```
> print(rankTbl)
        deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:      [0.3,1]  152      134       18 88.16      134           18        93.71             1.35
2: [0.0333,0.3)  161        7      154  4.35      141          172        98.60            12.87
3:   [0,0.0333) 1166        2     1164  0.17      143         1336       100.00           100.00
      ks
1: 92.36
2: 85.73
3:  0.00
> predObj = prediction(RFtest$prob1, RFtest$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.9410829
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.9897015
> gini = ineq(RFtest$prob1, type="Gini")
> gini
[1] 0.8927498
> ### Concordance and discordcance ratios:
> library("InformationValue")
> Concordance(actuals=RFtest$PersonalLoan, predictedScores=RFtest$prob1)
$Concordance
[1] 0.983245
$Discordance
[1] 0.01675495
$Tied
[1] -1.734723e-17
$Pairs
[1] 191048
```



Code Output

```
> LoanData = MainData[,c(-1)]
> LoanData$ZIPCode = as.factor(LoanData$ZIPCode)
> summary(LoanData)
      Age          Experience        Income          ZIPCode      FamilyMembers     CCAvg       
 Min.   :24.00   Min.   : 0.00   Min.   :  8.00   94720  : 163   1:1462        Min.   : 0.000  
 1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   94305  : 125   2:1270        1st Qu.: 0.700  
 Median :46.00   Median :20.00   Median : 64.00   95616  : 115   3:1000        Median : 1.500  
 Mean   :45.55   Mean   :20.32   Mean   : 73.77   90095  :  70   4:1198        Mean   : 1.938  
 3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   93106  :  56                 3rd Qu.: 2.600  
 Max.   :67.00   Max.   :43.00   Max.   :224.00   92037  :  54                 Max.   :10.000  
                                                  (Other):4347                                 
 Education    Mortgage       PersonalLoan SecuritiesAccount CDAccount Online    CreditCard
 1:2072    Min.   :  0.00   0:4452       0:4417            0:4630    0:1991    0:3480    
 2:1383    1st Qu.:  0.00   1: 478       1: 513            1: 300    1:2939    1:1450    
 3:1475    Median :  0.00                                                               
           Mean   : 56.68                                                               
           3rd Qu.:101.00                                                               
           Max.   :635.00                                                               

> names(LoanData)
 [1] "Age"               "Experience"        "Income"            "ZIPCode"          
 [5] "FamilyMembers"     "CCAvg"             "Education"         "Mortgage"         
 [9] "PersonalLoan"      "SecuritiesAccount" "CDAccount"         "Online"           
[13] "CreditCard"       
> str(LoanData)
'data.frame':	4930 obs. of  13 variables:
 $ Age              : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience       : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIPCode          : Factor w/ 467 levels "9307","90005",..: 84 35 368 299 97 161 116 268 35 236 ...
 $ 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        : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : int  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 ...
> ##Creating Test and Train Data
> split <- sample.split(LoanData$PersonalLoan, SplitRatio = 0.7)
> train<- subset(LoanData, split == TRUE)
> test<- subset( LoanData, split == FALSE)
> prop.table(table(train$PersonalLoan))

         0          1 
0.90292669 0.09707331 
> prop.table(table(test$PersonalLoan))

         0          1 
0.90331305 0.09668695 
> table(train$PersonalLoan)

   0    1 
3116  335 
> table(test$PersonalLoan)
```

```
   0    1
1336  143
> attach(train)
The following objects are masked from train (pos = 4):

    Age, CCAvg, CDAccount, CreditCard, Education, Experience, FamilyMembers, Income,
    Mortgage, Online, PersonalLoan, SecuritiesAccount, ZIPCode

> View(train)
> str(train)
'data.frame':	3451 obs. of  13 variables:
 $ Age              : int  25 45 35 37 53 50 35 34 65 29 ...
 $ Experience       : int  1 19 9 13 27 24 10 9 39 5 ...
 $ Income           : int  49 34 100 29 72 22 81 180 105 45 ...
 $ ZIPCode          : Factor w/ 467 levels "9307","90005",..: 84 35 299 161 116 268 35 236 367 48 ...
 $ FamilyMembers    : Factor w/ 4 levels "1","2","3","4": 4 3 1 4 2 1 3 1 4 3 ...
 $ CCAvg            : num  1.6 1.5 2.7 0.4 1.5 0.3 0.6 8.9 2.4 0.1 ...
 $ Education        : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 3 2 3 3 2 ...
 $ Mortgage         : int  0 0 0 155 0 0 104 0 0 0 ...
 $ PersonalLoan     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ 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 2 2 1 2 1 1 2 ...
 $ CreditCard       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
> dim(train)
[1] 3451   13
> names(train)
 [1] "Age"                "Experience"        "Income"            "ZIPCode"
 [5] "FamilyMembers"      "CCAvg"             "Education"         "Mortgage"
 [9] "PersonalLoan"       "SecuritiesAccount" "CDAccount"         "Online"
[13] "CreditCard"
> ##Setting the control parameters for rpart
> #minsplit: if the number of records in a given node falls below a threshold, the node will not be
split further.
> #minbucket: minimum records in a terminal node. if the records are less, that bucket will not be
created.
> #Terminal node (minbucket) should not be less than 2-3% of starting population.
> 0.02*3451
[1] 69.02
> 0.03*3451
[1] 103.53
> #minsplit = 3(minbucket)
> #xval divides the entire dataset into mutually exclusive and collectively exhaustive segments.
> #Model is built on xval-1 segments and 1 is used for testing.
> #cp = cost complexity parameter
> r.ctrl = rpart.control(minsplit=210, minbucket = 70, cp = 0, xval = 10)
> train.t <- rpart(formula = PersonalLoan ~ ., data = train[,-c(9)], method = "class", control = r.ctrl)
> train.t
n= 3451

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

 1) root 3451 335 0 (0.90292669 0.09707331)
   2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
   3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
     6) Education=1 484  57 0 (0.88223140 0.11776860)
      12)
ZIPCode=90005,90007,90009,90024,90025,90027,90028,90029,90034,90035,90036,90045,90064,90065,90066,90071,
```

```
90086,90095,90212,90230,90232,90245,90250,90254,90266,90274,90277,90401,90405,90502,90503,90630,90638,90
720,90740,91024,91030,91040,91107,91125,91203,91302,91304,91311,91320,91326,91335,91342,91360,91367,9138
0,91423,91604,91605,91710,91711,91763,91765,91768,91770,91801,91902,91911,91942,92007,92009,92028,92037,
92056,92096,92104,92115,92120,92121,92123,92126,92152,92154,92220,92325,92333,92350,92354,92373,92407,92
507,92521,92606,92624,92630,92647,92653,92691,92697,92704,92717,92780,92806,92807,92821,92831,92833,9284
3,93009,93014,93022,93105,93106,93109,93117,93407,93561,93727,93907,93940,93943,94002,94010,94022,94025,
94043,94063,94065,94066,94080,94085,94102,94104,94105,94110,94111,94112,94117,94123,94131,94132,94234,94
301,94303,94305,94402,94501,94507,94521,94539,94542,94545,94546,94550,94551,94566,94571,94575,94577,9458
3,94591,94596,94606,94607,94608,94609,94611,94701,94706,94709,94720,94801,94920,94949,94960,94998,95003,
95014,95023,95035,95037,95039,95051,95053,95054,95064,95112,95120,95133,95136,95138,95193,95307,95348,95
351,95449,95521,95616,95621,95630,95747,95762,95814,95819,95827,95833,95842,96001,96091 406  13 0
(0.96798030 0.03201970) *
      13)
ZIPCode=90032,90089,90210,90840,91103,91330,91355,91614,92038,92093,92106,92110,92122,92182,92612,92646,
92675,92677,93108,93302,93955,94108,94115,94122,94143,94304,94309,94590,94705,94707,94803,94904,95020,95
060,95125,95135,95605,95818,95841 78  34 1 (0.43589744 0.56410256) *
     7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
> fancyRpartPlot(train.t)
> ##To see how the tree performs
> printcp(train.t)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

Variables actually used in tree construction:
[1] Education Income    ZIPCode

Root node error: 335/3451 = 0.097073

n= 3451

        CP nsplit rel error  xerror     xstd
1 0.307463      0   1.00000 1.00000 0.051916
2 0.029851      2   0.38507 0.40000 0.033877
3 0.000000      3   0.35522 0.51045 0.038056
> plotcp(train.t)
> ##Since Vlaue of x error start increasing we have to prune the tree at cp = 0.030
> train.tree<- prune(train.t, cp=  0.030  ,"CP")
> printcp(train.tree)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

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

Root node error: 335/3451 = 0.097073

n= 3451

       CP nsplit rel error xerror     xstd
1 0.30746      0   1.00000    1.0 0.051916
2 0.03000      2   0.38507    0.4 0.033877
> fancyRpartPlot(train.tree, uniform=TRUE,  main="Pruned Classification Tree")
> ##Scoring
> train$predict.class = predict(train.t, train, type="class")
> train$predict.score = predict(train.t, train)
```

```
> library(caret)
> train$predict.class =as.factor(train$predict.class)
> train$PersonalLoan=as.factor(train$PersonalLoan)
> confusionMatrix(train$PersonalLoan,train$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3052   64
         1   55  280

               Accuracy : 0.9655
                 95% CI : (0.9589, 0.9714)
    No Information Rate : 0.9003
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8056

 Mcnemar's Test P-Value : 0.4633

            Sensitivity : 0.9823
            Specificity : 0.8140
         Pos Pred Value : 0.9795
         Neg Pred Value : 0.8358
             Prevalence : 0.9003
         Detection Rate : 0.8844
   Detection Prevalence : 0.9029
      Balanced Accuracy : 0.8981

       'Positive' Class : 0

> #Scoring the test sample
> test$predict.class <- predict(train.t, test, type="class")
> test$predict.score <- predict(train.t, test)
> test$predict.class <-as.factor(test$predict.class)
> test$PersonalLoan<-as.factor(test$PersonalLoan)
> confusionMatrix(test$PersonalLoan,test$predict.class)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1299   37
         1   37  106

               Accuracy : 0.95
                 95% CI : (0.9376, 0.9605)
    No Information Rate : 0.9033
    P-Value [Acc > NIR] : 2.57e-11

                  Kappa : 0.7136

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9723
            Specificity : 0.7413
         Pos Pred Value : 0.9723
         Neg Pred Value : 0.7413
             Prevalence : 0.9033
         Detection Rate : 0.8783
```

```
   Detection Prevalence : 0.9033
      Balanced Accuracy : 0.8568

       'Positive' Class : 0

> dim(test)
[1] 1479   15
> dim(LoanData)
[1] 4930   13
> train.tree
n= 3451

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

1) root 3451 335 0 (0.90292669 0.09707331)
  2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
  3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
    6) Education=1 484  57 0 (0.88223140 0.11776860) *
    7) Education=2,3 266  30 1 (0.11278195 0.88721805) *
> plotcp(train.t)
> ##To see how the tree performs
> printcp(train.t)

Classification tree:
rpart(formula = PersonalLoan ~ ., data = train[, -c(9)], method = "class",
    control = r.ctrl)

Variables actually used in tree construction:
[1] Education Income    ZIPCode

Root node error: 335/3451 = 0.097073

n= 3451

        CP nsplit rel error  xerror     xstd
1 0.307463      0   1.00000 1.00000 0.051916
2 0.029851      2   0.38507 0.40000 0.033877
3 0.000000      3   0.35522 0.51045 0.038056
> plotcp(train.t)
> plotcp(train.t)
> plotcp(train.tree)
> plotcp(train.t)
> fancyRpartPlot(train.tree, uniform=TRUE,  main="Pruned Classification Tree")
> train.tree
n= 3451

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

1) root 3451 335 0 (0.90292669 0.09707331)
  2) Income< 110.5 2701  42 0 (0.98445020 0.01554980) *
  3) Income>=110.5 750 293 0 (0.60933333 0.39066667)
    6) Education=1 484  57 0 (0.88223140 0.11776860) *
    7) Education=2,3 266  30 1 (0.11278195 0.88721805) *


> ##### Performance Matrix of CART Train
> ### Deciling and Rank Order Table
```

```
> str(train)
'data.frame':	3451 obs. of  17 variables:
 $ Age              : int  25 45 35 37 53 50 35 34 65 29 ...
 $ Experience       : int  1 19 9 13 27 24 10 9 39 5 ...
 $ Income           : int  49 34 100 29 72 22 81 180 105 45 ...
 $ ZIPCode          : Factor w/ 467 levels "9307","90005",..: 84 35 299 161 116 268 35 236 367 48 ...
 $ FamilyMembers    : Factor w/ 4 levels "1","2","3","4": 4 3 1 4 2 1 3 1 4 3 ...
 $ CCAvg            : num  1.6 1.5 2.7 0.4 1.5 0.3 0.6 8.9 2.4 0.1 ...
 $ Education        : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 3 2 3 3 2 ...
 $ Mortgage         : int  0 0 0 155 0 0 104 0 0 0 ...
 $ PersonalLoan     : num  0 0 0 0 0 0 0 1 0 0 ...
 $ 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 2 2 1 2 1 1 2 ...
 $ CreditCard       : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
 $ predict.class    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
 $ predict.score    : num [1:3451, 1:2] 0.984 0.984 0.984 0.984 0.984 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr  "1" "2" "4" "6" ...
  .. ..$ : chr  "0" "1"
 $ prob1            : num  0.0155 0.0155 0.0155 0.0155 0.0155 ...
 $ deciles          : Factor w/ 2 levels "[0.0155,0.032)",..: 1 1 1 1 1 1 1 2 1 1 ...
> train$prob1 <-train$predict.score[,2]
> test$prob1 <-test$predict.score[,2]
> probsCart=seq(0,1,length=11)
> probsCart
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qsCart=quantile(train$prob1, probsCart)
> qsCart
       0%       10%       20%       30%       40%       50%       60%       70%       80% 
0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0155498 0.0320197 
      90%      100% 
0.0320197 0.8872180 
> train$deciles=cut(train$prob1, unique(qsCart),include.lowest = TRUE,right=FALSE)
> table(train$deciles)

[0.0155,0.032)  [0.032,0.887] 
          2701            750 
> train$PersonalLoan <-ifelse(train$PersonalLoan == "1", 1,0)
> train$PersonalLoan <-as.numeric(train$PersonalLoan)
> test$PersonalLoan <-ifelse(test$PersonalLoan == "1", 1,0)
> test$PersonalLoan <-as.numeric(test$PersonalLoan)
> trainDT = data.table(train)
> rankTbl = trainDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan == 1),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> print(rankTbl)
          deciles  cnt cnt_tar1 cnt_tar0
1:  [0.032,0.887]  750      293      457
2: [0.0155,0.032) 2701       42     2659
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
```

```
> print(rankTbl)
          deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:  [0.032,0.887]  750      293      457 39.07      293          457        87.46            14.67
2: [0.0155,0.032) 2701       42     2659  1.55      335         3116       100.00           100.00
      ks
1: 72.79
2:  0.00
> predObj = prediction(train$prob1, train$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.8152817
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.9195031
> gini = ineq(train$prob1, type="Gini")
> gini
[1] 0.7575611
> ### Concordance and discordcance ratios:
> Concordance(actuals=train$PersonalLoan, predictedScores=train$prob1)
$Concordance
[1] 0.8594553

$Discordance
[1] 0.1405447

$Tied
[1] 2.775558e-17

$Pairs
[1] 1043860

> ##### Performance Matrix of CART Test
> ### Deciling and Rank Order Table
> probs=seq(0,1,length=11)
> probs
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qs=quantile(test$prob1, probs)
> qs
        0%        10%        20%        30%        40%        50%        60%        70%        80%
0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.01554980 0.02213776
       90%       100%
0.03201970 0.88721805
> test$deciles=cut(test$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
> table(test$deciles)

[0.0155,0.0221)  [0.0221,0.032)   [0.032,0.887]
           1183               0             296
> testDT = data.table(test)
> rankTbl = testDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
```

```
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
> print(rankTbl)
           deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp
1:   [0.032,0.887]  296      116      180 39.19      116          180        81.12
2: [0.0155,0.0221) 1183       27     1156  2.28      143         1336       100.00
   cum_rel_non_resp    ks
1:            13.47 67.65
2:           100.00  0.00
> predObj = prediction(test$prob1, test$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 0.7135641
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 0.8836601
> gini = ineq(test$prob1, type="Gini")
> gini
[1] 0.7568962
> ### Concordance and discordcance ratios:
> library("InformationValue")
> Concordance(actuals=test$PersonalLoan, predictedScores=test$prob1)
$Concordance
[1] 0.7949835

$Discordance
[1] 0.2050165

$Tied
[1] 2.775558e-17

$Pairs
[1] 191048


> RFData = MainData[,c(-1)]
> summary(RFData)
      Age          Experience        Income          ZIPCode      FamilyMembers     CCAvg       
 Min.   :24.00   Min.   : 0.00   Min.   :  8.00   Min.   : 9307   1:1462        Min.   : 0.000  
 1st Qu.:36.00   1st Qu.:10.00   1st Qu.: 39.00   1st Qu.:91910   2:1270        1st Qu.: 0.700  
 Median :46.00   Median :20.00   Median : 64.00   Median :93437   3:1000        Median : 1.500  
 Mean   :45.55   Mean   :20.32   Mean   : 73.77   Mean   :93152   4:1198        Mean   : 1.938  
 3rd Qu.:55.00   3rd Qu.:30.00   3rd Qu.: 98.00   3rd Qu.:94608                 3rd Qu.: 2.600  
 Max.   :67.00   Max.   :43.00   Max.   :224.00   Max.   :96651                 Max.   :10.000  
 Education    Mortgage      PersonalLoan SecuritiesAccount CDAccount Online    CreditCard
 1:2072    Min.   :  0.00   0:4452       0:4417            0:4630    0:1991   0:3480    
 2:1383    1st Qu.:  0.00   1: 478       1: 513            1: 300    1:2939   1:1450    
 3:1475    Median :  0.00                                                               
           Mean   : 56.68                                                               
           3rd Qu.:101.00                                                               
           Max.   :635.00                                                               
> names(RFData)
 [1] "Age"               "Experience"        "Income"            "ZIPCode"          
 [5] "FamilyMembers"     "CCAvg"             "Education"         "Mortgage"         
```

```
 [9] "PersonalLoan"      "SecuritiesAccount" "CDAccount"         "Online"
[13] "CreditCard"
> str(RFData)
'data.frame':	4930 obs. of  13 variables:
 $ Age              : int  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience       : int  1 19 15 9 8 13 27 24 10 9 ...
 $ Income           : int  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIPCode          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
 $ 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        : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage         : int  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 ...
> ##Creating Test and Train Data
> RFsplit <- sample.split(RFData$PersonalLoan, SplitRatio = 0.7)
> RFtrain<- subset(RFData, split == TRUE)
> RFtest<- subset( RFData, split == FALSE)
> names(RFtrain)
 [1] "Age"               "Experience"        "Income"            "ZIPCode"
 [5] "FamilyMembers"     "CCAvg"             "Education"         "Mortgage"
 [9] "PersonalLoan"      "SecuritiesAccount" "CDAccount"         "Online"
[13] "CreditCard"
> prop.table(table(RFtrain$PersonalLoan))

         0          1
0.90292669 0.09707331
> prop.table(table(RFtest$PersonalLoan))

         0          1
0.90331305 0.09668695
> table(RFtrain$PersonalLoan)

   0    1
3116  335
> table(RFtest$PersonalLoan)

   0    1
1336  143
> attach(RFtrain)
The following objects are masked from RFtrain (pos = 3):

    Age, CCAvg, CDAccount, CreditCard, Education, Experience, FamilyMembers, Income,
    Mortgage, Online, PersonalLoan, SecuritiesAccount, ZIPCode

The following objects are masked from train (pos = 4):

    Age, CCAvg, CDAccount, CreditCard, Education, Experience, FamilyMembers, Income,
    Mortgage, Online, PersonalLoan, SecuritiesAccount, ZIPCode

The following objects are masked from train (pos = 6):

    Age, CCAvg, CDAccount, CreditCard, Education, Experience, FamilyMembers, Income,
    Mortgage, Online, PersonalLoan, SecuritiesAccount, ZIPCode

> set.seed(123)
```

```
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                         ntree=101, mtry=5, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 101,
mtry = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 101
No. of variables tried at each split: 5

        OOB estimate of  error rate: 1.27%
Confusion matrix:
     0   1 class.error
0 3107   9 0.002888318
1   35 300 0.104477612
> ------------
+
+ ## Importance function
+ ## type is either 1 or 2, specifying the type of importance measure
+ ### (1=mean decrease in accuracy, 2=mean decrease in node impurity).
+
+ importance(Loan.RF, type=2)
                  MeanDecreaseGini
Age                      16.137480
Experience               16.242118
Income                  196.824539
ZIPCode                  15.182903
FamilyMembers            81.295277
CCAvg                    87.554281
Education               149.540846
Mortgage                 13.910754
SecuritiesAccount         1.034989
CDAccount                24.242179
Online                    2.464974
CreditCard                2.765016
> varImpPlot(Loan.RF, type=2)
> #To choose optimum value of ntree
> plot(Loan.RF, main="")
> legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
> title(main="Error Rates")

> #Tuning Random Forest
> tRF <- tuneRF(x = RFtrain[,-c(9)],
+               y=as.factor(RFtrain$PersonalLoan),
+               ntreeTry=30,
+               mtry=5,
+               stepFactor = 1.2,
+               improve = 0.0001,
+               trace=TRUE,
+               plot = TRUE,
+               doBest = TRUE,
+               nodesize = 100,
+               importance=TRUE
+ )
mtry = 5  OOB error = 1.97%
Searching left ...
Searching right ...
mtry = 6        OOB error = 2.09%
```

```
-0.05882353 1e-04
> #Tuning Random Forest
> tRF <- tuneRF(x = RFtrain[,-c(9)],
+               y=as.factor(RFtrain$PersonalLoan),
+               ntreeTry=30,
+               mtry=5,
+               stepFactor = 1.2,
+               improve = 0.0001,
+               trace=TRUE,
+               plot = TRUE,
+               doBest = TRUE,
+               nodesize = 100,
+               importance=TRUE
+ )
mtry = 5  OOB error = 2.32%
Searching left ...
Searching right ...
mtry = 6        OOB error = 1.8%
0.2247754 1e-04
mtry = 7        OOB error = 2.09%
-0.1609538 1e-04
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                          ntree=25, mtry=6, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 25,
mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 25
No. of variables tried at each split: 6

        OOB estimate of  error rate: 1.59%
Confusion matrix:
     0   1 class.error
0 3101  15 0.004813864
1   40 295 0.119402985
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                          ntree=25, mtry=6, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 25,
mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 25
No. of variables tried at each split: 6

        OOB estimate of  error rate: 1.36%
Confusion matrix:
     0   1 class.error
0 3104  12 0.003851091
1   35 300 0.104477612
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                          ntree=25, mtry=6, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 25,
```

```
mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 25
No. of variables tried at each split: 6

        OOB estimate of  error rate: 1.27%
Confusion matrix:
     0   1 class.error
0 3104  12 0.003851091
1   32 303 0.095522388
> Loan.RF <- randomForest(as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,-c(9)],
+                          ntree=30, mtry=6, importance=TRUE)
> print(Loan.RF)

Call:
 randomForest(formula = as.factor(RFtrain$PersonalLoan) ~ ., data = RFtrain[,      -c(9)], ntree = 30,
mtry = 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 30
No. of variables tried at each split: 6

        OOB estimate of  error rate: 1.22%
Confusion matrix:
     0   1 class.error
0 3106  10 0.003209243
1   32 303 0.095522388
> ## Scoring syntax
> RFtrain$predict.class <- predict(Loan.RF, RFtrain, type="class")
> RFtrain$predict.score <- predict(Loan.RF, RFtrain, type="prob")
> ### Checking the model accuracy
> library(caret)
> RFtrain$predict.class <-as.factor( RFtrain$predict.class)
> RFtrain$PersonalLoan <-as.factor( RFtrain$PersonalLoan)
> confusionMatrix(RFtrain$PersonalLoan , RFtrain$predict.class )
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3116    0
         1    0  335

               Accuracy : 1
                 95% CI : (0.9989, 1)
    No Information Rate : 0.9029
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.9029
         Detection Rate : 0.9029
   Detection Prevalence : 0.9029
      Balanced Accuracy : 1.0000
```

```
       'Positive' Class : 0

> #Scoring the holdout sample
> #####
> #####
> RFtest$predict.class <- predict(Loan.RF, RFtest, type="class")
> RFtest$predict.score <- predict(Loan.RF, RFtest, type="prob")
> RFtest$predict.class <-as.factor( RFtest$predict.class)
> RFtest$PersonalLoan <-as.factor( RFtest$PersonalLoan)
> confusionMatrix(RFtest$PersonalLoan , RFtest$predict.class )
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1332    4
         1   20  123

               Accuracy : 0.9838
                 95% CI : (0.976, 0.9896)
    No Information Rate : 0.9141
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9022

 Mcnemar's Test P-Value : 0.0022

            Sensitivity : 0.9852
            Specificity : 0.9685
         Pos Pred Value : 0.9970
         Neg Pred Value : 0.8601
             Prevalence : 0.9141
         Detection Rate : 0.9006
   Detection Prevalence : 0.9033
      Balanced Accuracy : 0.9769

       'Positive' Class : 0

> ##### Performance Matrix
> ### Deciling and Rank Order Table
> RFtrain$prob1 <-RFtrain$predict.score[,2]
> RFtest$prob1 <-RFtest$predict.score[,2]
> head(RFtrain)
  Age Experience Income ZIPCode FamilyMembers CCAvg Education Mortgage PersonalLoan
1  25          1     49   91107             4   1.6         1        0            0
2  45         19     34   90089             3   1.5         1        0            0
4  35          9    100   94112             1   2.7         2        0            0
6  37         13     29   92121             4   0.4         2      155            0
7  53         27     72   91711             2   1.5         2        0            0
8  50         24     22   93943             1   0.3         3        0            0
  SecuritiesAccount CDAccount Online CreditCard predict.class predict.score.0 predict.score.1
1                 1         0      0          0             0               1               0
2                 1         0      0          0             0               1               0
4                 0         0      0          0             0               1               0
6                 0         0      1          0             0               1               0
7                 0         0      1          0             0               1               0
8                 0         0      0          1             0               1               0
  prob1
1     0
2     0
```

```
4     0
6     0
7     0
8     0
> probs=seq(0,1,length=11)
> probs
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qs=quantile(RFtrain$prob1, probs)
> qs
       0%       10%       20%       30%       40%       50%       60%       70%       80% 
0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 
      90%      100% 
0.1666667 1.0000000 
> RFtrain$deciles=cut(RFtrain$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
> table(RFtrain$deciles)

[0,0.167) [0.167,1] 
     3100       351 
> view(RFtrain)
> RFtrain$PersonalLoan <-ifelse(RFtrain$PersonalLoan == "1", 1,0)
> RFtrain$PersonalLoan <-as.numeric(RFtrain$PersonalLoan)
> RFtest$PersonalLoan <-ifelse(RFtest$PersonalLoan == "1", 1,0)
> RFtest$PersonalLoan <-as.numeric(RFtest$PersonalLoan)
> trainDT = data.table(RFtrain)
> rankTbl = trainDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan == 1),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> print(rankTbl)
     deciles  cnt cnt_tar1 cnt_tar0
1: [0.167,1]  351      335       16
2: [0,0.167) 3100        0     3100
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
> print(rankTbl)
     deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1: [0.167,1]  351      335       16 95.44      335           16          100             0.51
2: [0,0.167) 3100        0     3100  0.00      335         3116          100           100.00
      ks
1: 99.49
2:  0.00
> predObj = prediction(RFtrain$prob1, RFtrain$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> KS
[1] 1
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
[1] 1
> gini = ineq(RFtrain$prob1, type="Gini")
> gini
```

```
[1] 0.9006712
> Concordance(actuals=RFtrain$PersonalLoan, predictedScores=RFtrain$prob1)
$Concordance
[1] 1

$Discordance
[1] 0

$Tied
[1] 0

$Pairs
[1] 1043860

> probs=seq(0,1,length=11)
> plot(perf)
> probs=seq(0,1,length=11)
> probs
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> qs=quantile(RFtest$prob1, probs)
> qs
        0%        10%        20%        30%        40%        50%        60%        70%        80%
0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.03333333
       90%       100%
0.30000000 1.00000000
> RFtest$deciles=cut(RFtest$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
> table(RFtest$deciles)

  [0,0.0333) [0.0333,0.3)      [0.3,1]
        1166          161          152
> testDT = data.table(RFtest)
> rankTbl = testDT[, list(
+   cnt = length(PersonalLoan),
+   cnt_tar1 = sum(PersonalLoan),
+   cnt_tar0 = sum(PersonalLoan == 0)
+ ),
+ by=deciles][order(-deciles)]
> rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
> rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
> rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
> rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
> rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
> rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
> print(rankTbl)
        deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:      [0.3,1]  152      134       18 88.16      134           18        93.71             1.35
2: [0.0333,0.3)  161        7      154  4.35      141          172        98.60            12.87
3:   [0,0.0333) 1166        2     1164  0.17      143         1336       100.00           100.00
      ks
1: 92.36
2: 85.73
3:  0.00
> predObj = prediction(RFtest$prob1, RFtest$PersonalLoan)
> perf = performance(predObj, "tpr", "fpr")
> plot(perf)
> KS = max(perf@y.values[[1]]-perf@x.values[[1]])
> auc = performance(predObj,"auc");
> auc = as.numeric(auc@y.values)
> auc
```

```
[1] 0.9897015
> gini = ineq(RFtest$prob1, type="Gini")
> gini
[1] 0.8927498
> ### Concordance and discordcance ratios:
> library("InformationValue")
> Concordance(actuals=RFtest$PersonalLoan, predictedScores=RFtest$prob1)
$Concordance
[1] 0.983245

$Discordance
[1] 0.01675495

$Tied
[1] -1.734723e-17

$Pairs
[1] 191048
```

6. Interpretation of Model

6.1 CART Model

Model Plot Interpretation

The 89% of Customers of Thera Bank having Income > 111K and Education level is either Graduate or Professional took personal loan from the Bank (Which is 8% Total Population).
i.e approx 71%+ of our Personal loan consumers belong to this category only.

Model Prediction Interpretation

If we consider the first decile of our model i.e Range [0.032,0.887] of predicted probability we can get a cumulative response of 81.12% i.e out total customer who take our personal loan 81.12% lie in this decile. By predicting the probability we can narrow down the marketing strategy for customer who can belong to this decile and improve our conversion rate with less marketing budget & Efforts.

```
          deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp
1:   [0.032,0.887]  296      116      180 39.19      116          180        81.12
2: [0.0155,0.0221) 1183       27     1156  2.28      143         1336       100.00
   cum_rel_non_resp    ks
1:            13.47 67.65
2:           100.00  0.00
```

6.2 Random Forest Model

Model Prediction Interpretation

If we consider the first two deciles of our model i.e Range [0.0333,1] of predicted probability we can get a cumulative response of 98.60% i.e out total customer who take our personal loan 98.60% lie in this decile. By predicting the probability we can narrow down the marketing strategy for customer who can belong to this decile and improve our conversion rate with less marketing budget & Efforts.

```
       deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp
1:      [0.3,1]  152      134       18 88.16      134           18        93.71             1.35
2: [0.0333,0.3)  161        7      154  4.35      141          172        98.60            12.87
3:   [0,0.0333) 1166        2     1164  0.17      143         1336       100.00           100.00
      ks
1: 92.36
2: 85.73
3:  0.00
```

7. Conclusion

By Analysing both the Model we can predict that the Random Forest Model is Performing much better because of high KS.
KS = Cumulative Response Rate - Cumulative Non Response Rate
So Higher the KS better the model.

By building the model we are able to uplift the performance of our marketing strategy from 9.6 % to 98.60%.
Using this model we can easily narrow down audience and only target those customers who have a higher probability of taking our Loan. This will reduce our marketing effort also increases our conversion rate. Using this model we can predict the possibility of customer of taking Personal Loan.

8. Suggestion

Here are some suggestions

- Since some of the datapoint were missing so if that can be present that can be great.
- Experience was negative in 52 rows which is not possible so if data can be corrected that would be great

9. Appendix

9.1 Appendix A – Source Code

```
# Load the required libraries

library(tidyverse)
library(dplyr)
library(ggplot2)
library(DataExplorer)
library(rpart)
library(rpart.plot)
library(rattle)
library(RColorBrewer)
library(caTools)
library(caret)
library(randomForest)
library(data.table)
library(ROCR)
library(ineq)
library(corrplot)
library(InformationValue)

#Setting the Working Directory
setwd ("E:/000GL/000 0Projects/004/Project/Final")
getwd()

#Importing the data
TheraData <- read.csv("Thera Bank_Personal_Loan_Modelling-data.csv")

summary(TheraData)

names(TheraData)
names(TheraData) <- c("ID"
                    ,"Age"
                    ,"Experience"
                    ,"Income"
                    ,"ZIPCode"
                    ,"FamilyMembers"
                    ,"CCAvg"
                    ,"Education"
                    ,"Mortgage"
                    ,"PersonalLoan"
                    ,"SecuritiesAccount"
                    ,"CDAccount"
                    ,"Online"
                    ,"CreditCard"
)

summary(TheraData)

#EDA
myData = TheraData
```

```
myData$ID = as.factor(myData$ID)
myData$FamilyMembers = as.factor(myData$FamilyMembers)
myData$Education = as.factor(myData$Education)
myData$PersonalLoan = as.factor(myData$PersonalLoan)
myData$SecuritiesAccount = as.factor(myData$SecuritiesAccount)
myData$CDAccount = as.factor(myData$CDAccount)
myData$Online = as.factor(myData$Online)
myData$CreditCard = as.factor(myData$CreditCard)


summary(myData)

##Check NA
plot_missing(myData)
sum(is.na(myData))

##Also treating the negetive work experience by removing them
myData$Experience[myData$Experience < 0] = NA
sum(is.na(myData))

##70 values are missing which is less than 3% of the data so we can remove the NA Data
MainData = na.omit(myData)

dim(MainData)

summary(MainData)

class(MainData)

str(MainData)
#SUMMARY OF Main Data

names(MainData)
head(MainData)
tail(MainData)


##Univariant Analysis
##Histogram of Continious Variable
plot_histogram(MainData)
names(MainData)

summary(MainData$Age)
sd(MainData$Age)
boxplot(MainData$Age
        ,horizontal = TRUE
        ,las =2
        ,main = "Age"
        ,col = "orange"
        ,border = "brown")

summary(MainData$Experience)
sd(MainData$Experience)
boxplot(MainData$Experience
        ,horizontal = TRUE
        ,las =2
        ,main = "Experience"
        ,col = "orange"
```

```
        ,border = "brown")

summary(MainData$Income)
sd(MainData$Income)
boxplot(MainData$Income
        ,horizontal = TRUE
        ,las =2
        ,main = "Income"
        ,col = "orange"
        ,border = "brown")

summary(MainData$CCAvg)
sd(MainData$CCAvg)
boxplot(MainData$CCAvg
        ,horizontal = TRUE
        ,las =2
        ,main = "CCAvg"
        ,col = "orange"
        ,border = "brown")

summary(MainData$Mortgage)
sd(MainData$Mortgage)
boxplot(MainData$Mortgage
        ,horizontal = TRUE
        ,las =2
        ,main = "Mortgage"
        ,col = "orange"
        ,border = "brown")

summary(MainData$Education)
barplot(table(MainData$Education), main="Education",
        xlab="Education Level",
        names.arg=c("Undergrad","Graduate", "Professional"),
        ylab = "Frequency")

summary(MainData$FamilyMembers)
barplot(table(MainData$FamilyMembers), main="FamilyMembers",
        xlab="FamilyMembers",
        ylab = "Frequency")

summary(MainData$PersonalLoan)
barplot(table(MainData$PersonalLoan), main="Customer accept the personal loan offered in the last
campaign?",
        xlab="Personal Loan",
        names.arg=c("NO","YES"),
        ylab = "Count")

summary(MainData$SecuritiesAccount)
barplot(table(MainData$SecuritiesAccount), main="Customer have a securities account with the bank",
        xlab="Education Level",
        names.arg=c("NO","YES"),
        ylab = "COUNT")

summary(MainData$CDAccount)
barplot(table(MainData$CDAccount), main="Customer have CD account with the bank.",
        xlab="CD account with the bank",
        names.arg=c("NO","YES"),
        ylab = "Frequency")
```

```
summary(MainData$Online)
barplot(table(MainData$Online), main="Online Banking",
        xlab="Education Level",
        names.arg=c("NO","YES"),
        ylab = "Frequency")

summary(MainData$CreditCard)
barplot(table(MainData$CreditCard), main="Uses Bank Credit Card",
        xlab="Education Level",
        names.arg=c("NO","YES"),
        ylab = "Frequency")

ZipTemp = MainData
ZipTemp$ZIPCode = as.factor(ZipTemp$ZIPCode)
summary(ZipTemp)

##Bivarient Analysis

###PersonalLoan vs Age
ggplot(MainData, aes(x=PersonalLoan, y=MainData$Age)) +
  geom_boxplot(color="orange", fill="orange", alpha=0.2) +
  scale_x_discrete(labels = c('NO','YES'))+
  labs(title = "PersonalLoan Acceptance Vs Age")

###PersonalLoan vs Experience
ggplot(MainData, aes(x=PersonalLoan, y=MainData$Experience)) +
  geom_boxplot(color="orange", fill="orange", alpha=0.2) +
  scale_x_discrete(labels = c('NO','YES'))+
  labs(title = "PersonalLoan Acceptance Vs Experience")

###PersonalLoan vs Income
ggplot(MainData, aes(x=PersonalLoan, y=MainData$Income)) +
  geom_boxplot(color="orange", fill="orange", alpha=0.2) +
  scale_x_discrete(labels = c('NO','YES'))+
  labs(title = "PersonalLoan Acceptance Vs Income")

###PersonalLoan vs CCAvg
ggplot(MainData, aes(x=PersonalLoan, y=MainData$CCAvg)) +
  geom_boxplot(color="orange", fill="orange", alpha=0.2) +
  scale_x_discrete(labels = c('NO','YES'))+
  labs(title = "PersonalLoan Acceptance Vs CCAvg")

###PersonalLoan vs Mortgage
ggplot(MainData, aes(x=PersonalLoan, y=MainData$Mortgage)) +
  geom_boxplot(color="orange", fill="orange", alpha=0.2) +
  scale_x_discrete(labels = c('NO','YES'))+
  labs(title = "PersonalLoan Acceptance Vs Mortgage")


##PersonalLoan vs Family
table(MainData$PersonalLoan, MainData$FamilyMembers)
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = FamilyMembers))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "Family Members Count")

##PersonalLoan vs Education
table(MainData$PersonalLoan, MainData$Education)
```

```
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = Education))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "Education Level", labels = c("Undergrad",
"Graduate","Advanced/Professional"))


##PersonalLoan vs Security Account
table(MainData$PersonalLoan, MainData$SecuritiesAccount)
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = SecuritiesAccount))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "Securities Account", labels = c("No", "Yes"))


##PersonalLoan vs CDAccount
table(MainData$PersonalLoan, MainData$CDAccount)
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = CDAccount))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "CDAccount", labels = c("No", "Yes"))

##PersonalLoan vs Online
table(MainData$PersonalLoan, MainData$Online)
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = Online))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "Internet Banking", labels = c("No", "Yes"))



##PersonalLoan vs CreditCard
table(MainData$PersonalLoan, MainData$CreditCard)
###BarPlot
ggplot(MainData, aes(x = PersonalLoan, fill = CreditCard))+
  geom_bar(position = 'stack')+
  scale_x_discrete(labels = c('NO','YES'))+
  scale_fill_discrete(name = "Credit Card", labels = c("No", "Yes"))



##Corelation Plot
Data_cor <- cor(TheraData)
cex.before <- par("cex")
par(cex = 0.6 )
corrplot(Data_cor)

corrplot(Data_cor, method = "number" , number.digits = 2 )
par(cex = cex.before)


####Cart Analaysis

LoanData = MainData[,c(-1)]
LoanData$ZIPCode = as.factor(LoanData$ZIPCode)
```

```r
summary(LoanData)
names(LoanData)
str(LoanData)


##Creating Test and Train Data
split <- sample.split(LoanData$PersonalLoan, SplitRatio = 0.7)
train<- subset(LoanData, split == TRUE)
test<- subset( LoanData, split == FALSE)

prop.table(table(train$PersonalLoan))
prop.table(table(test$PersonalLoan))
table(train$PersonalLoan)
table(test$PersonalLoan)


attach(train)
##Viewing the Development sample

View(train)
str(train)
dim(train)
names(train)

##Setting the control parameters for rpart
#minsplit: if the number of records in a given node falls below a threshold, the node will not be split
further.
#minbucket: minimum records in a terminal node. if the records are less, that bucket will not be
created.
#Terminal node (minbucket) should not be less than 2-3% of starting population.
0.02*3451
0.03*3451

#minsplit = 3(minbucket)
#xval divides the entire dataset into mutually exclusive and collectively exhaustive segments.
#Model is built on xval-1 segments and 1 is used for testing.
#cp = cost complexity parameter
r.ctrl = rpart.control(minsplit=210, minbucket = 70, cp = 0, xval = 10)

#Using rpart to build the tree

train.t <- rpart(formula = PersonalLoan ~ ., data = train[,-c(9)], method = "class", control = r.ctrl)
#train.t <- rpart(formula = PersonalLoan ~ ., data = train[,-9], method = "class")
#LoanData.t <- rpart(formula = PersonalLoan ~ ., data = LoanData.dev[,-1], method = "class")

train.t
fancyRpartPlot(train.t)

##To see how the tree performs
printcp(train.t)
plotcp(train.t)

##Since Vlaue of x error start increasing we have to prune the tree at cp = 0.030
train.tree<- prune(train.t, cp=  0.030  ,"CP")
train.tree
printcp(train.tree)
fancyRpartPlot(train.tree, uniform=TRUE,  main="Pruned Classification Tree")
```

```
##Scoring
train$predict.class = predict(train.t, train, type="class")
train$predict.score = predict(train.t, train)

## We can use the confusionMatrix function of the caret package

library(caret)
train$predict.class =as.factor(train$predict.class)
train$PersonalLoan=as.factor(train$PersonalLoan)
confusionMatrix(train$PersonalLoan,train$predict.class)

#Scoring the test sample
test$predict.class <- predict(train.t, test, type="class")
test$predict.score <- predict(train.t, test)

## Confusion Matrix for Test Data

test$predict.class <-as.factor(test$predict.class)
test$PersonalLoan<-as.factor(test$PersonalLoan)
confusionMatrix(test$PersonalLoan,test$predict.class)


###Perfomance Matrix for Cart

##### Performance Matrix of CART Train
### Deciling and Rank Order Table
str(train)
train$prob1 <-train$predict.score[,2]
test$prob1 <-test$predict.score[,2]


probsCart=seq(0,1,length=11)
probsCart
qsCart=quantile(train$prob1, probsCart)
qsCart
train$deciles=cut(train$prob1, unique(qsCart),include.lowest = TRUE,right=FALSE)
table(train$deciles)

train$PersonalLoan <-ifelse(train$PersonalLoan == "1", 1,0)
train$PersonalLoan <-as.numeric(train$PersonalLoan)
test$PersonalLoan <-ifelse(test$PersonalLoan == "1", 1,0)
test$PersonalLoan <-as.numeric(test$PersonalLoan)

trainDT = data.table(train)

rankTbl = trainDT[, list(
  cnt = length(PersonalLoan),
  cnt_tar1 = sum(PersonalLoan == 1),
  cnt_tar0 = sum(PersonalLoan == 0)
),
by=deciles][order(-deciles)]

print(rankTbl)

rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
```

```
rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);

print(rankTbl)

### ROCR and ineq packages to compute AUC, KS and gini

predObj = prediction(train$prob1, train$PersonalLoan)
perf = performance(predObj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
KS
auc = performance(predObj,"auc");
auc = as.numeric(auc@y.values)
auc
gini = ineq(train$prob1, type="Gini")
gini


### Concordance and discordcance ratios:
Concordance(actuals=train$PersonalLoan, predictedScores=train$prob1)


##### Performance Matrix of CART Test
### Deciling and Rank Order Table
probs=seq(0,1,length=11)
probs
qs=quantile(test$prob1, probs)
qs
test$deciles=cut(test$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
table(test$deciles)


testDT = data.table(test)
rankTbl = testDT[, list(
  cnt = length(PersonalLoan),
  cnt_tar1 = sum(PersonalLoan),
  cnt_tar0 = sum(PersonalLoan == 0)
),
by=deciles][order(-deciles)]

rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
rankTbl$cum_non_resp = cumsum(rankTbl$cnt_tar0)
rankTbl$cum_rel_resp = round(rankTbl$cum_resp / sum(rankTbl$cnt_tar1),4)*100;
rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_tar0),4)*100;
rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);

print(rankTbl)

### ROCR and ineq packages to compute AUC, KS and gini


predObj = prediction(test$prob1, test$PersonalLoan)
perf = performance(predObj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
KS
auc = performance(predObj,"auc");
```

```
auc = as.numeric(auc@y.values)
auc
gini = ineq(test$prob1, type="Gini")
gini



### Concordance and discordcance ratios:
library("InformationValue")
Concordance(actuals=test$PersonalLoan, predictedScores=test$prob1)
```