# Thera Bank - Loan Purchase Modeling

**Muthu Pandian G**

**October 11, 2019**

## Thera Bank - Loan Purchase Modeling

### OBJECTIVE OF THE PROJECT:

This case is about a bank (Thera Bank) which has a growing customer base. Majority of these customers are liability customers (depositors) with varying size of deposits.

The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process, earn more through the interest on loans.

In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors).

A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with a minimal budget.

The department wants to build a model that will help them identify the potential customers who have a higher probability of purchasing the loan. This will increase the success ratio while at the same time reduce the cost of the campaign. The dataset has data on 5000 customers.

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

You are expected to do the following:

- EDA of the data available. Showcase the results using appropriate graphs
- Apply appropriate clustering on the data and interpret the output
- Build appropriate models on both the test and train data (CART & Random Forest). Interpret all the model outputs and do the necessary modifications wherever eligible (such as pruning)
- Check the performance of all the models that you have built (test and train). Use all the model performance measures you have learned so far. Share your remarks on which model performs the best.

# Importing the Dataset

```
setwd("D:/Great Lakes/Projects/DATA MINING")
getwd()
```

```
## [1] "D:/Great Lakes/Projects/DATA MINING"
```

```
library(openxlsx)
thera <- read.xlsx("Thera Bank_Personal_Loan_Modelling-dataset-1.xlsx",sheet = "Bank_Personal_Loan_Modelling",startRow = 1)
```

# Understanding the data

## Structure of Data

```
str(thera)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age.(in.years)       : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience.(in.years): num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income.(in.K/month)  : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : num  91107 90089 94720 94112 91330 ...
##  $ Family.members       : num  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : num  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan        : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities.Account   : num  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ Online             : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard         : num  0 0 0 0 1 0 0 1 0 0 ...
```

- The dataset has data of 5000 customers.
- The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan).
- Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

## Data Description:

| | |
|---|---|
| ID | Customer ID |
| Age | Customer's age in years |
| Experience | Years of professional experience |
| Income | Annual income of the customer ($000) |
| ZIPCode | Home Address ZIP code. |
| Family | Family size of the customer |
| CCAvg | Avg. spending on credit cards per month ($000) |
| Education | Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional |
| Mortgage | Value of house mortgage if any. ($000) |
| Personal Loan | Did this customer accept the personal loan offered in the last campaign? |
| Securities Account | Does the customer have a securities account with the bank? |
| CD Account | Does the customer have a certificate of deposit (CD) account with the bank? |
| Online | Does the customer use internet banking facilities? |
| CreditCard | Does the customer use a credit card issued by the bank? |

## Summary

```
summary(thera)
```

```
##        ID        Age.(in.years)  Experience.(in.years) Income.(in.K/month)
##  Min.   :   1   Min.   :23.00   Min.   :-3.0          Min.   :  8.00
```

```
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0         1st Qu.: 39.00
##  Median :2500   Median :45.00   Median :20.0         Median : 64.00
##  Mean   :2500   Mean   :45.34   Mean   :20.1         Mean   : 73.77
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0         3rd Qu.: 98.00
##  Max.   :5000   Max.   :67.00   Max.   :43.0         Max.   :224.00
##
##     ZIP.Code     Family.members      CCAvg          Education
##  Min.   : 9307   Min.   :1.000   Min.   : 0.000   Min.   :1.000
##  1st Qu.:91911   1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000
##  Median :93437   Median :2.000   Median : 1.500   Median :2.000
##  Mean   :93153   Mean   :2.397   Mean   : 1.938   Mean   :1.881
##  3rd Qu.:94608   3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000
##  Max.   :96651   Max.   :4.000   Max.   :10.000   Max.   :3.000
##                  NA's   :18
##    Mortgage     Personal.Loan   Securities.Account   CD.Account
##  Min.   :  0.0   Min.   :0.000   Min.   :0.0000     Min.   :0.0000
##  1st Qu.:  0.0   1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000
##  Median :  0.0   Median :0.000   Median :0.0000     Median :0.0000
##  Mean   : 56.5   Mean   :0.096   Mean   :0.1044     Mean   :0.0604
##  3rd Qu.:101.0   3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000
##  Max.   :635.0   Max.   :1.000   Max.   :1.0000     Max.   :1.0000
##
##     Online         CreditCard
##  Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.000
##  Median :1.0000   Median :0.000
##  Mean   :0.5968   Mean   :0.294
##  3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.000
##
```

- From the summary, we can see that the age of our customers ranges from 23 years to 67 years.
- It's also seen that the Experience has negative values in it. As it doesn't make any sense, we need to convert them to 0

```
thera$`Experience.(in.years)` <-replace(thera$`Experience.(in.years)`,thera$`Experience.(in.years)`< 0,0)
summary(thera$`Experience.(in.years)`)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   20.00   20.12   30.00   43.00
```

## Checking NA Values/ Missing Values

We can also see that there are 18 NA's Present in the Family Variable.Let's cross check them with is.na command.

```
sum(is.na(thera))
```

```
## [1] 18
```

We can replace the NA's with 0 as the Na's represents only 0.36% of total dataset and it won't affect the data much.

```
thera[is.na(thera)]<- 0
```

## Converting numeric variables into factor Variables

Education,Personal.Loan,Securities.Account,CD.Account,Online, CreditCard can be converted into factor variables.

```
thera$Education <- as.factor(thera$Education)
thera$Family.members <- as.factor(thera$Family.members)
thera$Personal.Loan <- as.factor(thera$Personal.Loan)
thera$Securities.Account <- as.factor(thera$Securities.Account)
thera$CD.Account <- as.factor(thera$CD.Account)
thera$Online <- as.factor(thera$Online)
thera$CreditCard <- as.factor(thera$CreditCard)
str(thera)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age.(in.years)      : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience.(in.years): num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income.(in.K/month)  : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code            : num  91107 90089 94720 94112 91330 ...
##  $ Family.members      : Factor w/ 5 levels "0","1","2","3",..: 5 4 2 2 5 5 3 2 4 2
...
##  $ 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          : num  0 0 0 0 0 155 0 0 104 0 ...
```

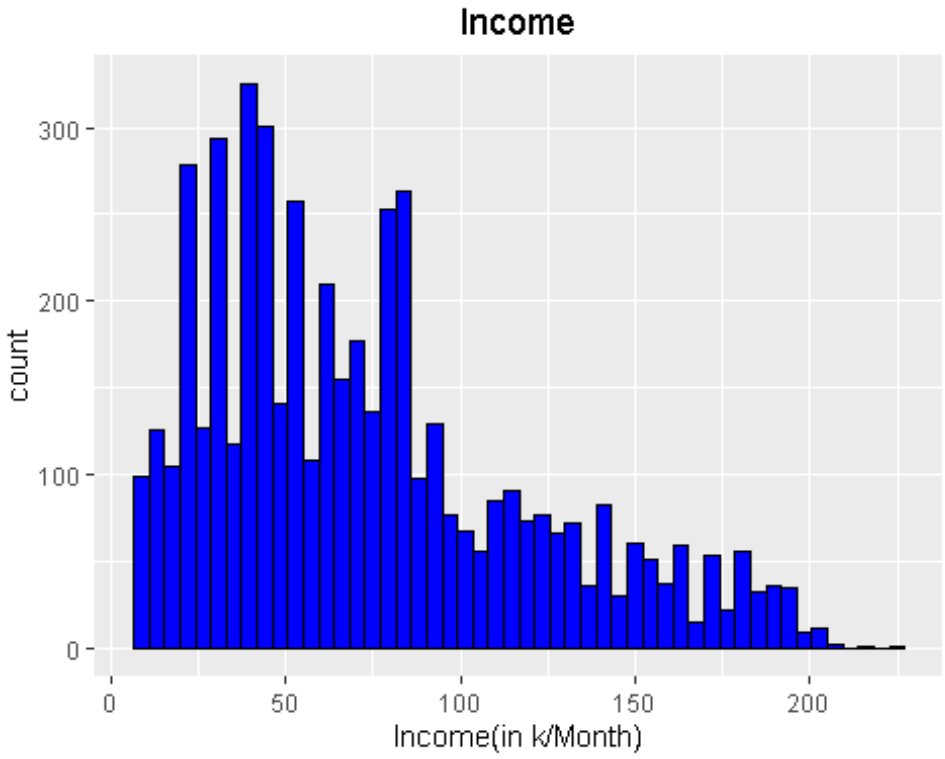
```
##  $ Personal.Loan       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Securities.Account  : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ CD.Account          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Online              : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
##  $ CreditCard          : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
```

# Exploratory Data Analysis

## Univariate Analysis

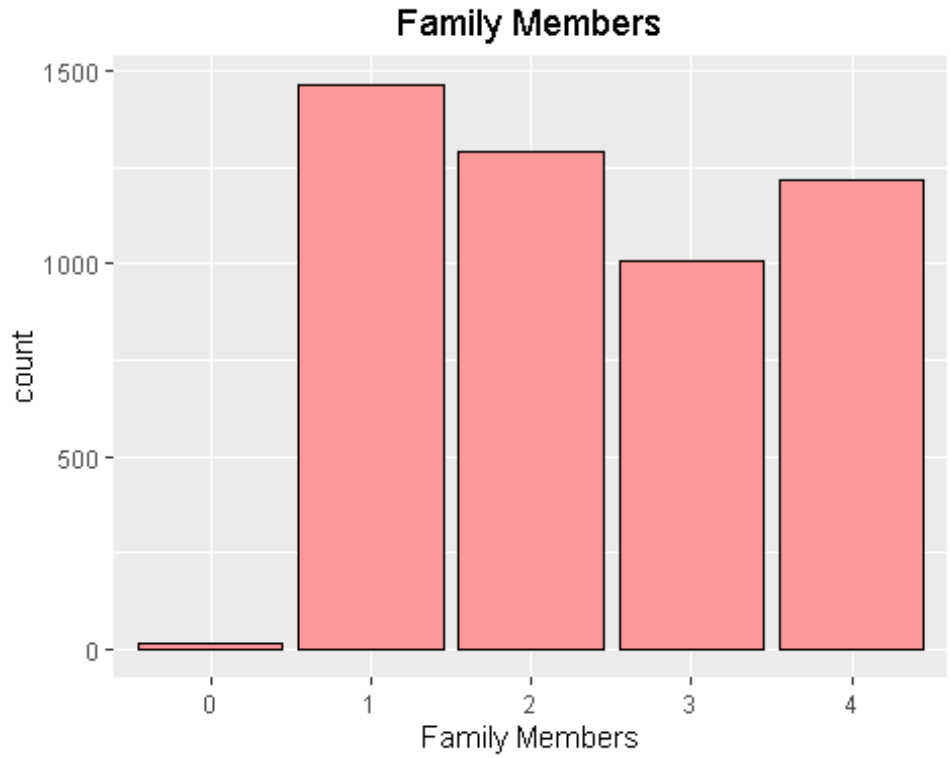
### Frequency Distribution of each Independent numerical Variable

```
library(ggplot2)
ggplot(data=thera,aes(x=`Income.(in.K/month)`))+geom_histogram(bins = 50,fill = "Blue",colour = "Black")+ggtitle("Income")+theme(plot.title = element_text(hjust = 0.5))+xlab("Income(in k/Month)")
```



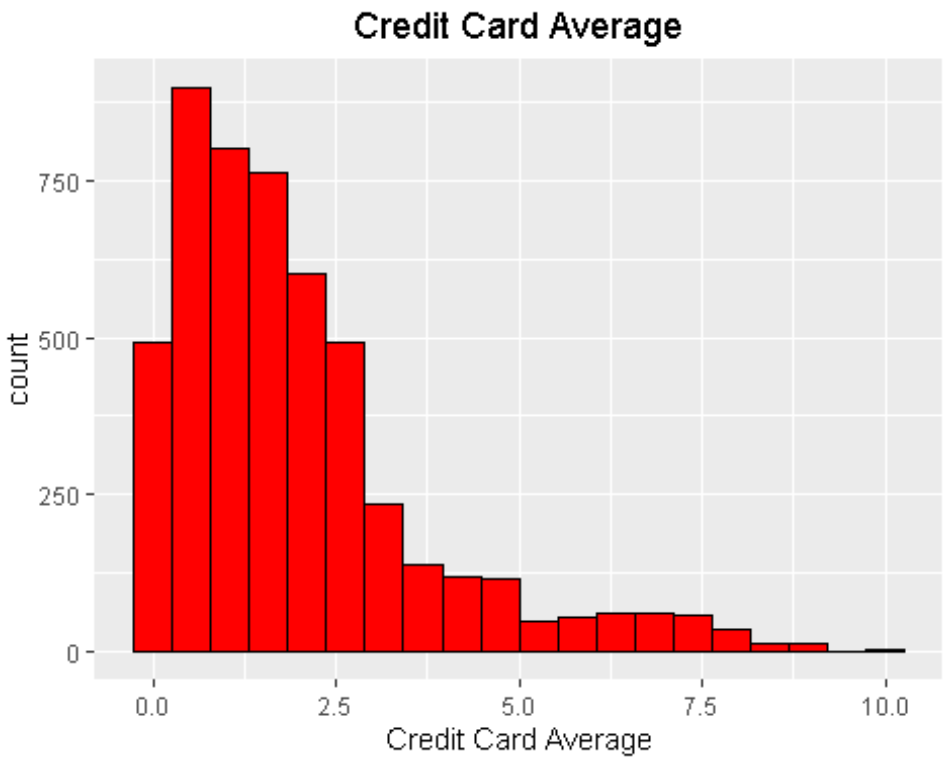

The Income range of our customers varies from lower income people to higher income people with many customers income lies near 45-50 k/month

```
ggplot(data=thera,aes(x=Family.members))+geom_bar(fill = "#FF9999",colour = "Black")+ggtitle("Family Members")+theme(plot.title = element_text(hjust = 0.5))+xlab("Family Members")
```



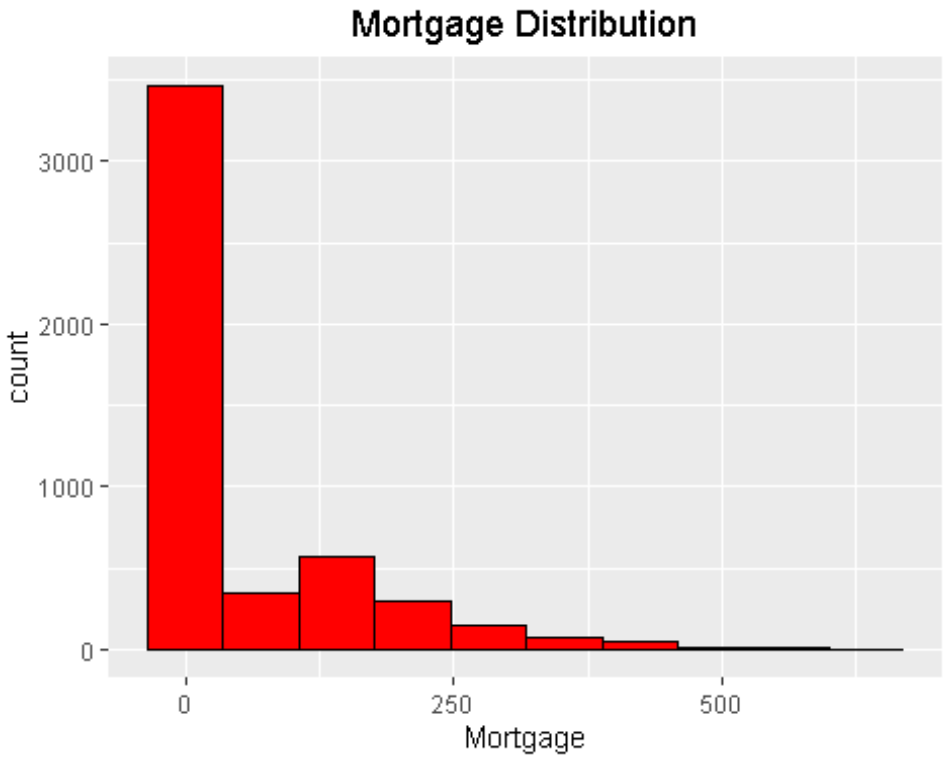

Most Of our customers has a family size of 1 or 2 members but 1 mostly

```
ggplot(data=thera,aes(x=CCAvg))+geom_histogram(bins = 20,fill = "Red",colour = "Black")+ggtitle("Credit Card Average")+theme(plot.title = element_text(hjust = 0.5))+xlab("Credit Card Average")
```



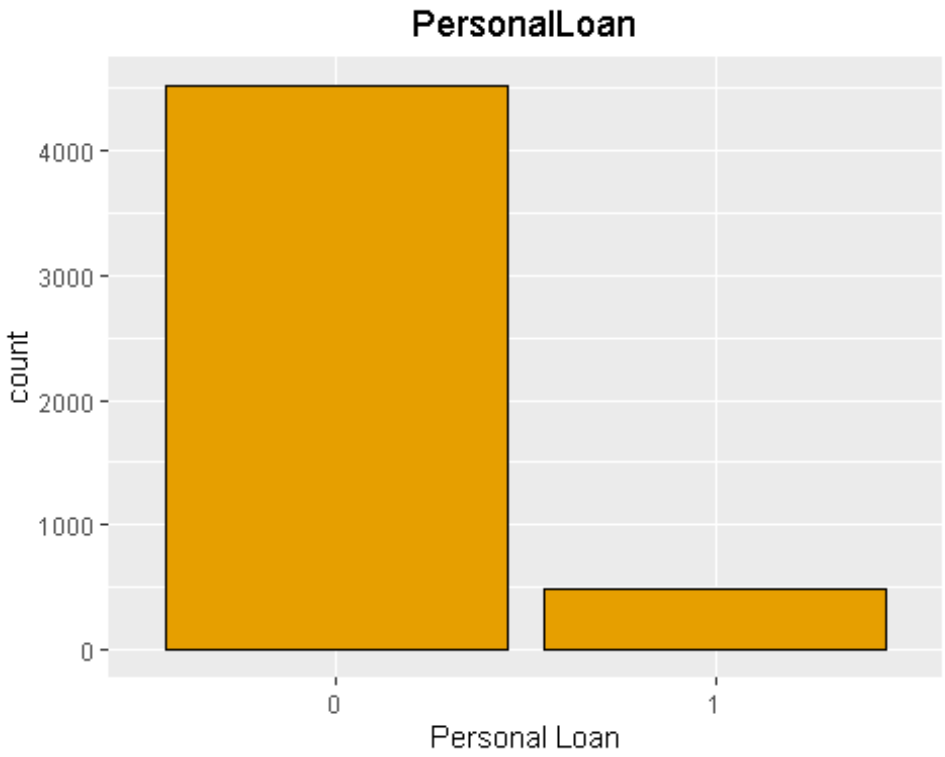

Mostly Our customers Avg. spending on credit cards per month lies around 1 to 2

```
ggplot(data=thera,aes(x=Mortgage))+geom_histogram(bins = 10,fill = "Red",colour = "Black")+ggtitle("Mortgage Distribution")+theme(plot.title = element_text(hjust = 0.5))+xlab("Mortgage")
```



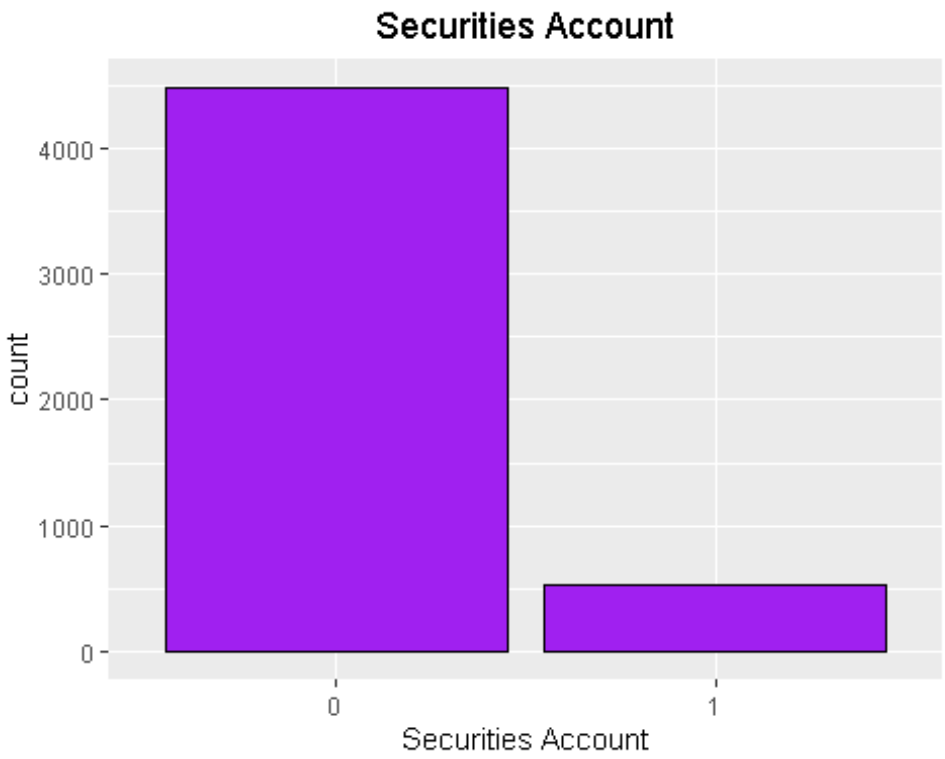

Around 70% of our customers haven’t got mortgage from our bank yet

```
ggplot(data=thera,aes(x=Personal.Loan))+geom_bar(fill = "#E69F00",colour = "
Black")+ggtitle("PersonalLoan")+theme(plot.title = element_text(hjust = 0.5))+xl
ab("Personal Loan")
```



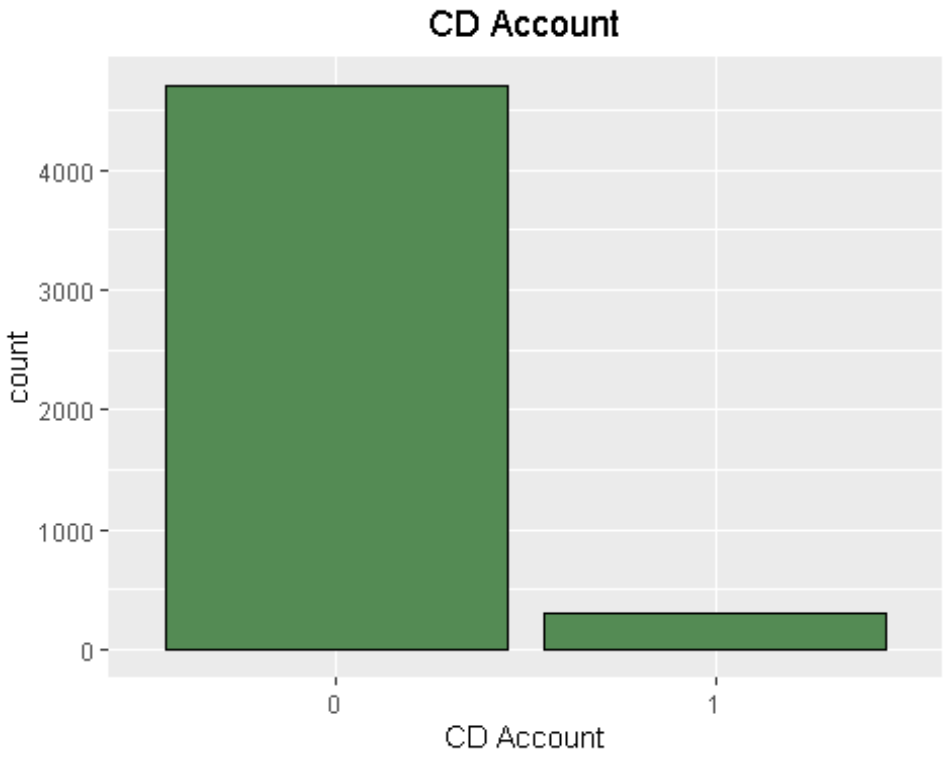

9% (480) of customers only have accepted the personal loan offered in the last campaign

```
ggplot(data=thera,aes(x=Securities.Account))+geom_bar(fill = "purple",colour = "Black")+ggtitle("Securities Account")+theme(plot.title = element_text(hjust = 0.5))+xlab("Securities Account")
```



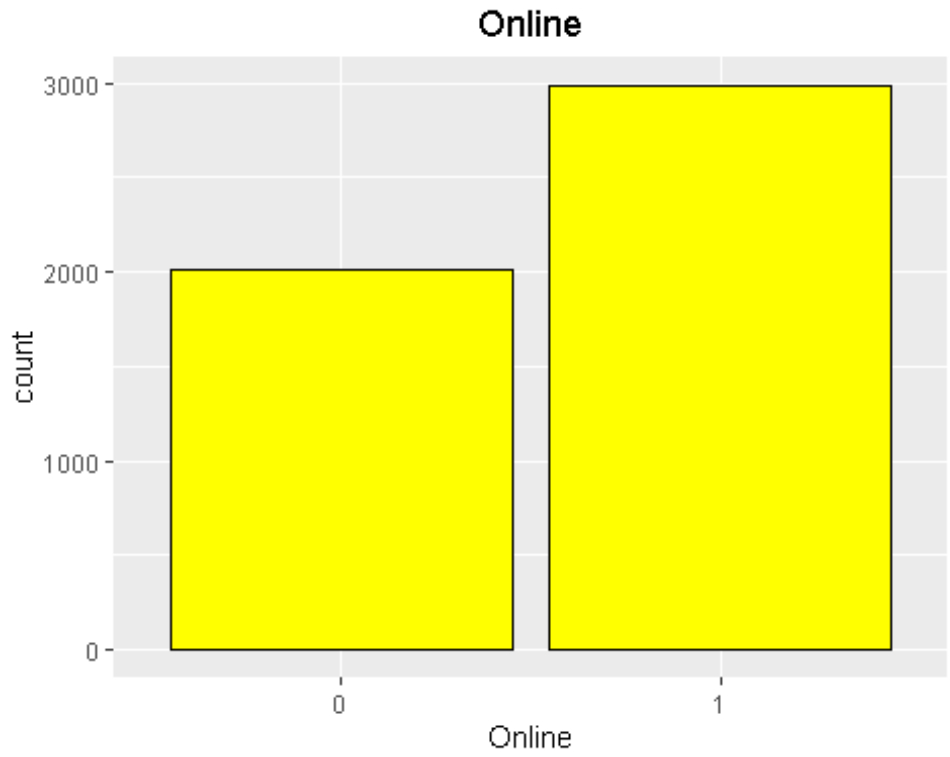

10 % (522) of customers only have securities account with our Bank

```
ggplot(data=thera,aes(x=CD.Account))+geom_bar(fill = "palegreen4",colour = "
Black")+ggtitle("CD Account")+theme(plot.title = element_text(hjust = 0.5))+xla
b("CD Account")
```



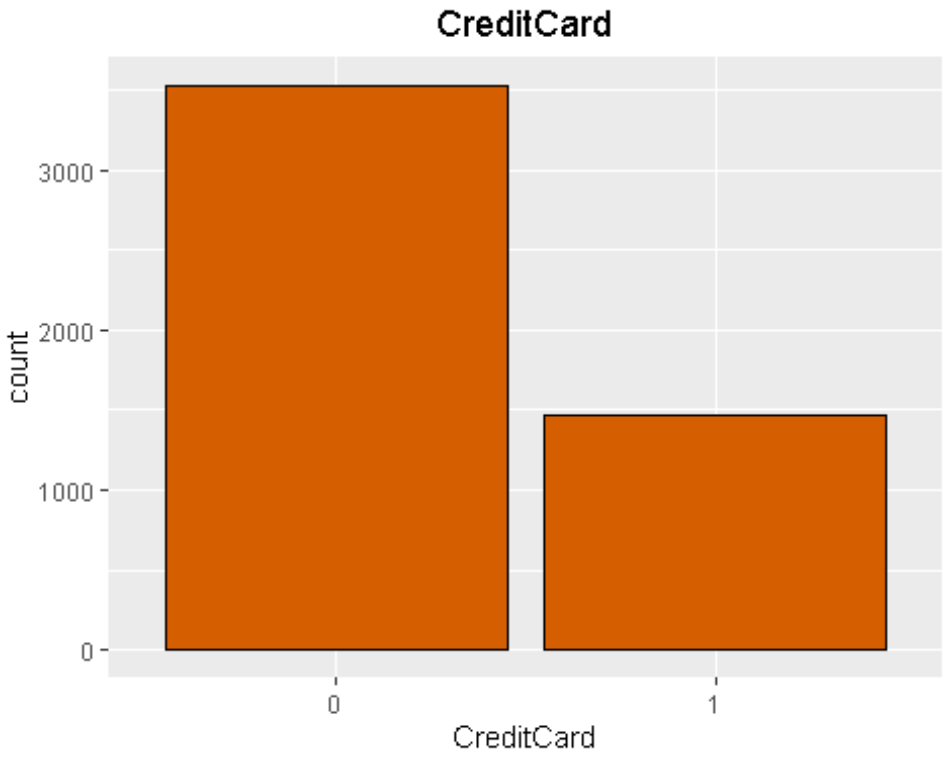

6 % (302) of customers only have Certificate of Deposit (CD)account with thera Bank

```
ggplot(data=thera,aes(x=Online))+geom_bar(fill = "yellow",colour = "Black")+g
gtitle("Online")+theme(plot.title = element_text(hjust = 0.5))+xlab("Online")
```




Nearly 60% (2969) customers use internet banking facilities provided by our thera Bank

```
ggplot(data=thera,aes(x=CreditCard))+geom_bar(fill = "#D55E00",colour = "Black")+ggtitle("CreditCard")+theme(plot.title = element_text(hjust = 0.5))+xlab("CreditCard")
```



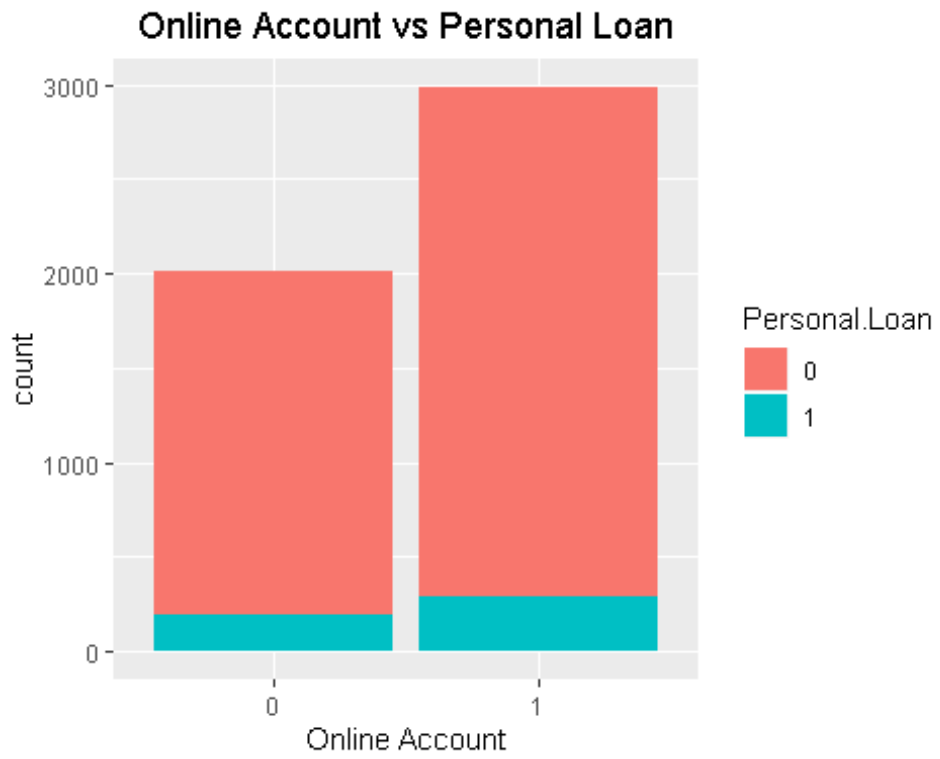

29% of the customer use a credit card issued by thera bank

## Bi Variable and Multi Variable analysis

```
ggplot(thera,aes(CreditCard,fill = Personal.Loan))+geom_bar()+ggtitle("Credit C
ard vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Cred
it Card")
```

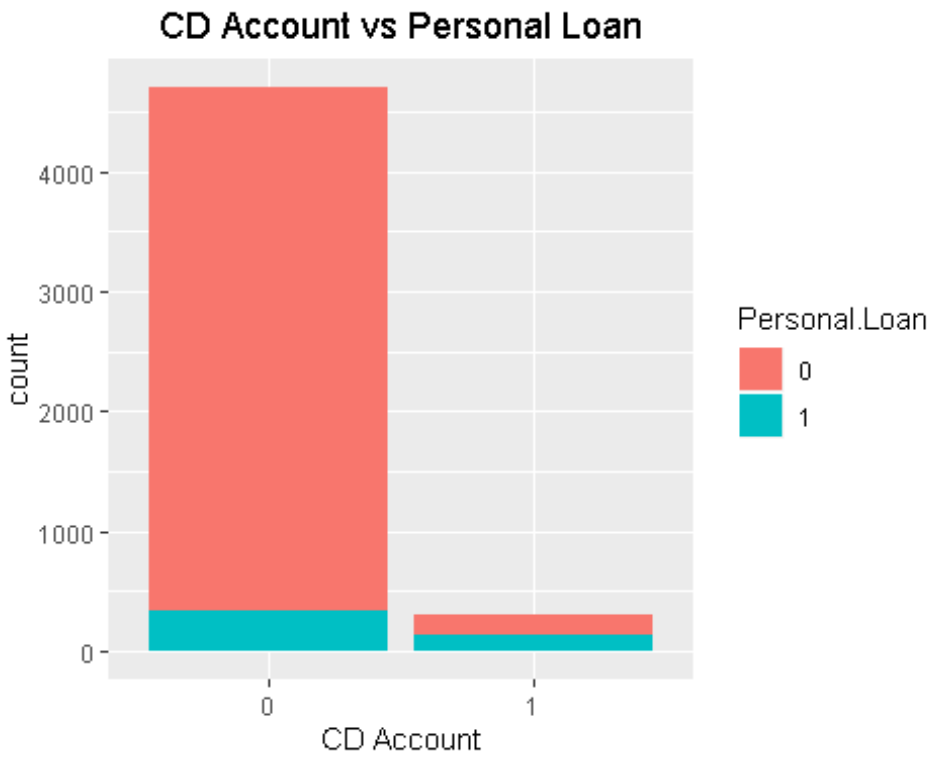

From the plot it's clear that the customers who doesn't have credit card had personal loan when compared to customers who have credit card with the bank.

```
ggplot(thera,aes(Online,fill = Personal.Loan))+geom_bar()+ggtitle("Online Acco
unt vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Onli
ne Account")
```



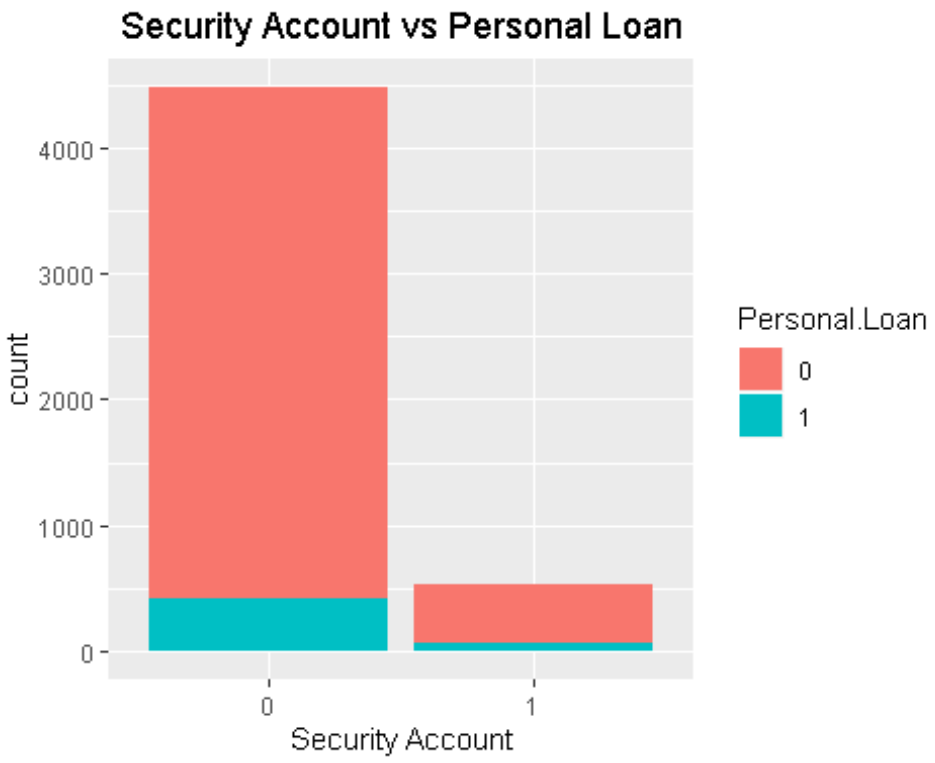

From the above graph we can clearly see that the customers who has Online Account had personal loan when compared to customers who doesn't have Online account.

```
ggplot(thera,aes(CD.Account,fill = Personal.Loan))+geom_bar()+ggtitle("CD Ac
count vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("C
D Account")
```



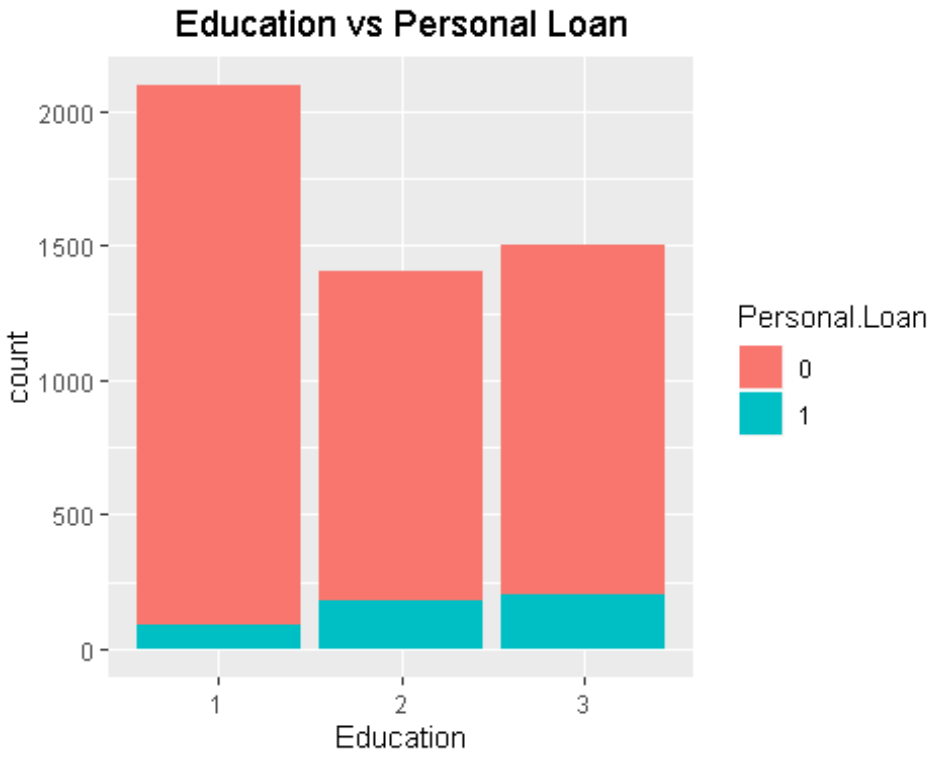

Here we can see that the customers who didn't interested in depositing has acquired personal loan than the customers who have a Certificate of Deposit Account with the Bank.

```
ggplot(thera,aes(Securities.Account,fill = Personal.Loan))+geom_bar()+ggtitle("
Security Account vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5)
)+ xlab("Security Account")
```




Similarly, here the customer who doesn't have Security Account with the bank are more likely to accept the personal Loan than the customers with security account

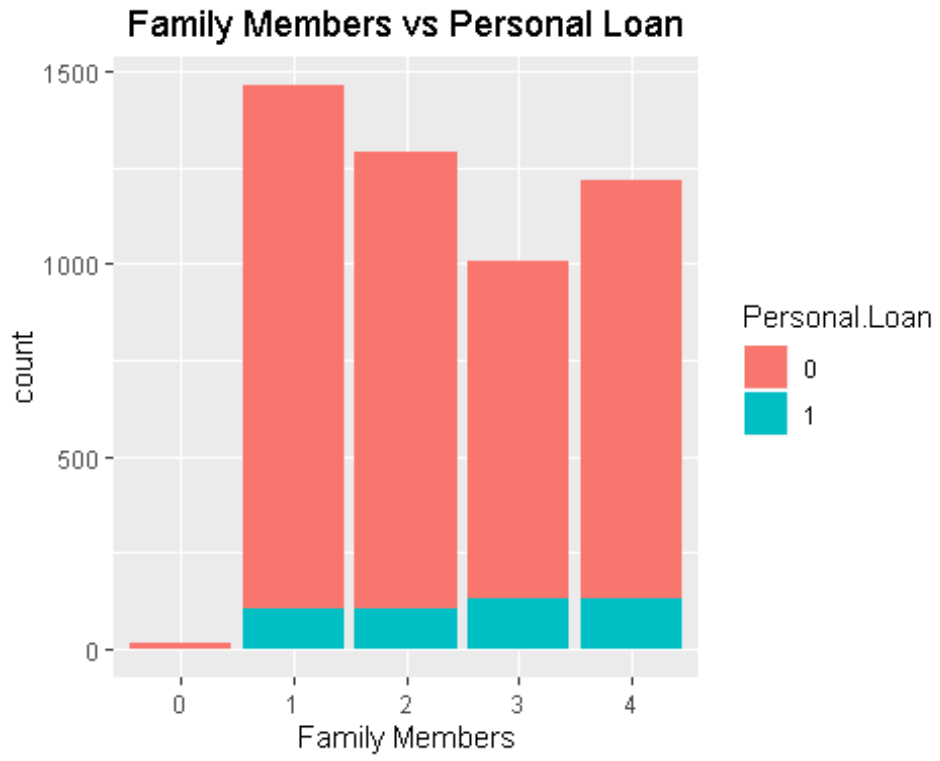
```
ggplot(thera,aes(Education,fill = Personal.Loan))+geom_bar()+ggtitle("Education vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Education")
```




From the above plot we can see that higher the education more the requirement of money.

Hence the graduate and advanced Professionals has acquired Personal Loan than the Undergrad and Graduate

```
ggplot(thera,aes(Family.members,fill = Personal.Loan))+geom_bar()+ggtitle("Family Members vs Personal Loan")+ theme(plot.title = element_text(hjust = 0.5))+ xlab("Family Members")
```




As per our data, we can see that there is no significant relationship between Family Members and Personal Loan.

Since Zip Code and ID doesn't have any impact on Personal.Loan we can remove Zip Code from our dataset

```
cor(as.numeric(thera$Personal.Loan),thera$ZIP.Code)

## [1] 0.0001073764

thera <- thera[,-c(1,5)]
```

# Clustering

## Reason for Choosing K-Means Clustering Over Hierarchical Clustering

- Hierarchical clustering can't handle big data well but K Means clustering can.

- This is because the time complexity of K Means is linear i.e. O(n) while that of hierarchical clustering is quadratic i.e. $O(n^2)$.
- So, we choose K-means Clustering Over Hierarchial CLustering.

## K-Means Clustering

For clustering we need numerical variables, so we remove the categorical variables

```
thera_no_cat <- thera[,-c(4,6,8,9,10,11,12)]
str(thera_no_cat)
```

```
## 'data.frame':    5000 obs. of  5 variables:
##  $ Age.(in.years)       : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience.(in.years): num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income.(in.K/month)  : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ CCAvg                : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Mortgage             : num  0 0 0 0 0 155 0 0 104 0 ...
```

## Step1: we need to give the number of centers (k-value)

Let's figure out the optimum k-value by using elbow method

```
library(factoextra)

library(NbClust)
```

## Scaling the Data

we scale all the columns to mean zero and unit standard deviation, so that we eliminate the dominance of one variable on the other.

```
set.seed(123)
# we use set.seed for the purpose of reproducibility
therascale <- scale(thera_no_cat)
head(therascale)
```

```
##   Age.(in.years) Experience.(in.years) Income.(in.K/month)      CCAvg
## 1    -1.77423939          -1.67122301         -0.5381750 -0.1933661
## 2    -0.02952064          -0.09786299         -0.8640230 -0.2505855
```

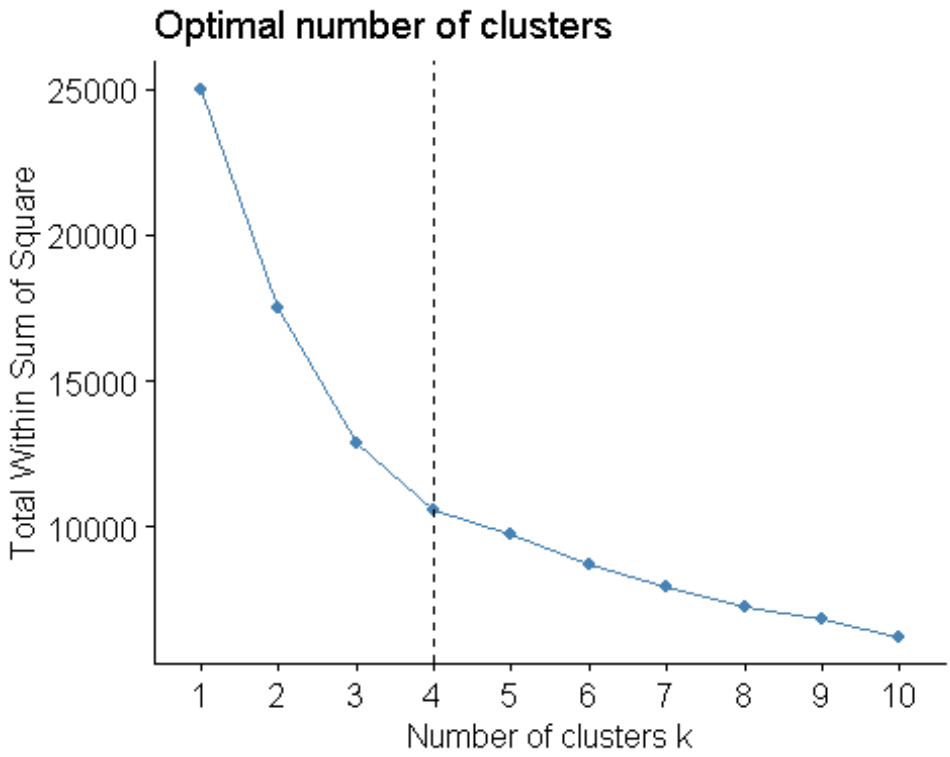
```
## 3    -0.55293627         -0.44749855         -1.3636566 -0.5366825
## 4    -0.90188002         -0.97195189          0.5697084  0.4360473
## 5    -0.90188002         -1.05936078         -0.6250678 -0.5366825
## 6    -0.72740814         -0.62231633         -0.9726390 -0.8799989
##     Mortgage
## 1 -0.5554684
## 2 -0.5554684
## 3 -0.5554684
## 4 -0.5554684
## 5 -0.5554684
## 6  0.9684153
```

Let's check whether we have mean zero and unit SD

```
apply(therascale,2,mean)

##       Age.(in.years) Experience.(in.years)   Income.(in.K/month)
##        6.054575e-18         1.300502e-16         1.462101e-16
##              CCAvg             Mortgage
##        4.641524e-17        -2.653715e-17

apply(therascale,2,sd)

##       Age.(in.years) Experience.(in.years)   Income.(in.K/month)
##                    1                    1                    1
##              CCAvg             Mortgage
##                    1                    1
```

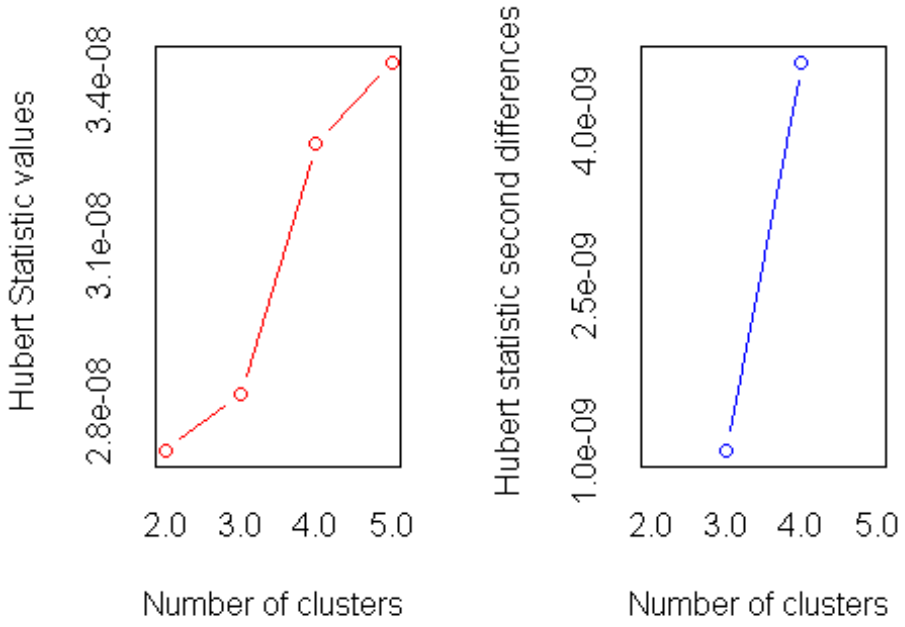
Let's figure out the optimum k-value by using elbow method

```
fviz_nbclust(therascale,kmeans,method = "wss")+geom_vline(xintercept = 4,linetype = 2)
```

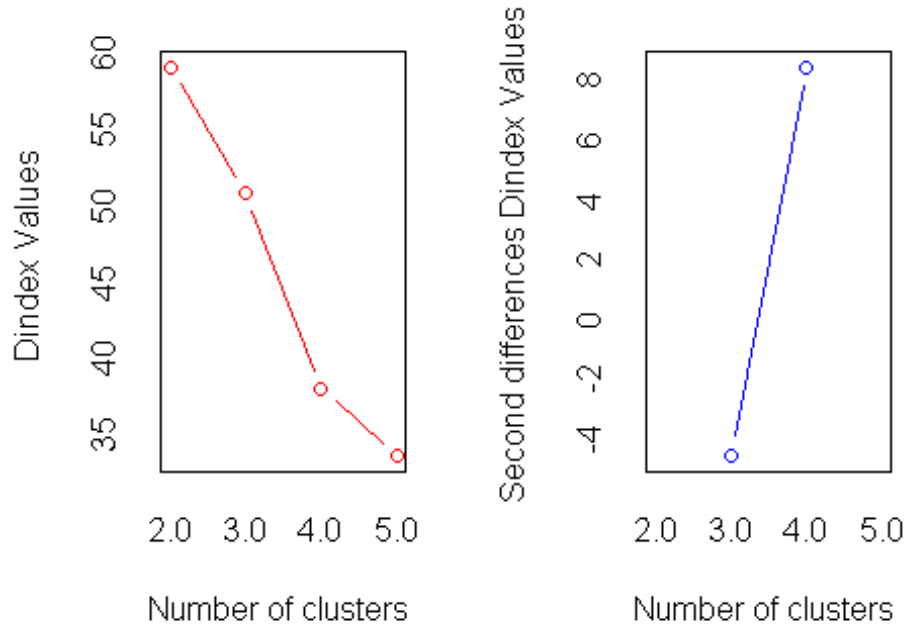



By Using Elbow Method on the above plot, we can say that the optimal number of clusters for our datatset is 4.

```
# Let's cross check it by using NbClust Function
NbClust(thera_no_cat,min.nc = 2,max.nc = 5,method = "kmeans")
```




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




```
## ***: The D index is a graphical method of determining the number of clusters.
##              In the plot of D index, we seek a significant knee (the significant peak
in Dindex
##              second differences plot) that corresponds to a significant increase of t
he value of
##              the measure.
##
## *******************************************************************
***
## * Among all indices:
## * 8 proposed 2 as the best number of clusters
## * 2 proposed 3 as the best number of clusters
## * 12 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
##
##                 ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  4
##
##
```

```
## ************************************************************
***

***: The Hubert index is a graphical method of determining the number of clusters
.
#In the plot of Hubert index, we seek a significant knee that corresponds to a
#significant increase of the value of the measure i.e the significant peak in Hubert
#index second differences plot.
#*** : The D index is a graphical method of determining the number of clusters.
#In the plot of D index, we seek a significant knee (the significant peak in Dindex
#                                    second differences plot) that corresponds to a sig
nificant increase of the value of
#the measure.
#* Among all indices:
#  * 8 proposed 2 as the best number of clusters
#* 2 proposed 3 as the best number of clusters
#* 12 proposed 4 as the best number of clusters
#* 2 proposed 5 as the best number of clusters
#* According to the majority rule, the best number of clusters is  4
#****** Conclusion ******
```

Let's do K-means clustering with k-value as 4

```
set.seed(123)
kmeans.cluster=kmeans(therascale,4,nstart = 10)
print(kmeans.cluster)
```

```
## K-means clustering with 4 clusters of sizes 1904, 2033, 636, 427
##
## Cluster means:
##   Age.(in.years) Experience.(in.years) Income.(in.K/month)      CCAvg
## 1     -0.8975732          -0.90001401         -0.3433621 -0.3405017
## 2      0.8950601           0.89097963         -0.3822979 -0.3525390
## 3     -0.1384284          -0.12095213          1.5605050  1.7863982
## 4     -0.0530151          -0.04873387          1.0269131  0.5360138
##     Mortgage
## 1 -0.2109990
## 2 -0.2057075
## 3 -0.3935731
## 4  2.5064591
```

```
## Within cluster sum of squares by cluster:
## [1] 3086.469 3371.240 2073.525 2022.995
##  (between_SS / total_SS =  57.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

- As shown above, K-means has formed 4 clusters of sizes 900,327,2581,1192.
- K-means cluster also gives Within cluster sum of squares by cluster (Variance) i.e., how similar are the members within the same group.
- In this case we have 57.8% similarity within the groups. let's add the cluster column in our original dataset

```
thera_no_cat$cluster <- kmeans.cluster$cluster
head(thera_no_cat,10)
```

```
##    Age.(in.years) Experience.(in.years) Income.(in.K/month) CCAvg Mortgage
## 1              25                    1                  49   1.6        0
## 2              45                   19                  34   1.5        0
## 3              39                   15                  11   1.0        0
## 4              35                    9                 100   2.7        0
## 5              35                    8                  45   1.0        0
## 6              37                   13                  29   0.4      155
## 7              53                   27                  72   1.5        0
## 8              50                   24                  22   0.3        0
## 9              35                   10                  81   0.6      104
## 10             34                    9                 180   8.9        0
##    cluster
## 1        1
## 2        1
## 3        1
## 4        1
## 5        1
## 6        1
## 7        2
```

```
## 8       2
## 9       1
## 10      3
```

```
clusterProfile <- aggregate(thera_no_cat[,-6],list(thera_no_cat$cluster),FUN = "mean")
clusterProfile
```

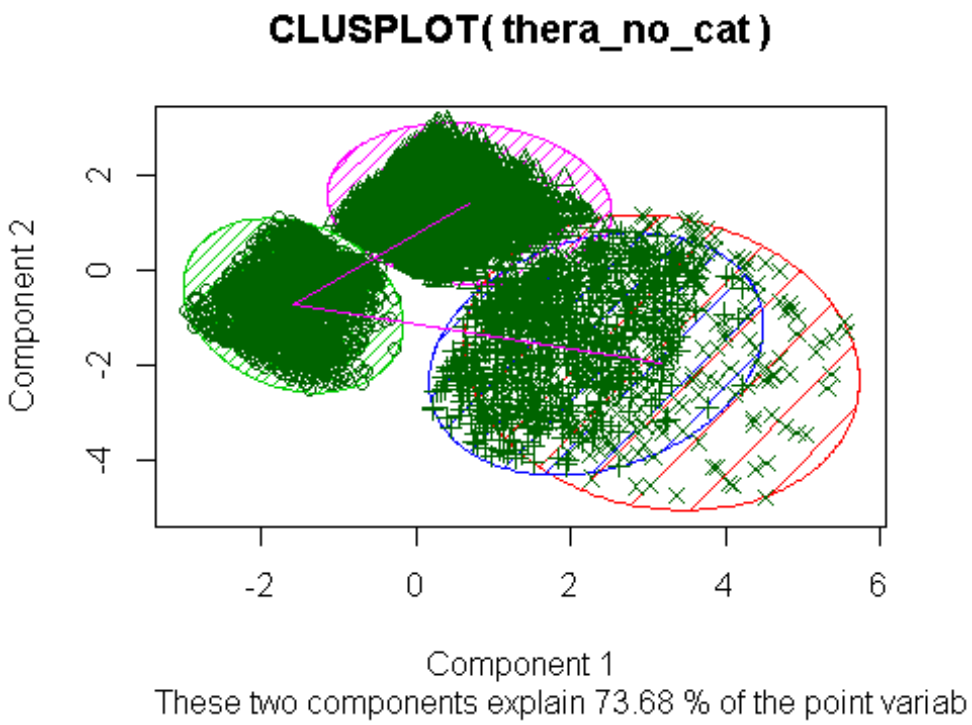
```
##   Group.1 Age.(in.years) Experience.(in.years) Income.(in.K/month)
## 1       1       35.04937             9.823004            57.96796
## 2       2       55.59862            30.312838             56.17560
## 3       3       43.75157            18.735849            145.61006
## 4       4       44.73068            19.562061            121.04684
##      CCAvg  Mortgage
## 1 1.342857  35.03729
## 2 1.321820  35.57550
## 3 5.059953  16.46698
## 4 2.874707 311.44028
```

Here we have 4 groups,

- The 1st segment of customers are **" comparably younger customers"** of all the groups as their average age group is 35 years.
- The 2nd segment of customers are **"older customers"** as their average age group is 55 years.
- The 3rd and 4th segment of customers are **"Middle Aged customers"** as their average age group is 43 & 44years.
- Here we have 57.8% similarity within the groups(as we have scaled the data).

Let's plot and see the 4 clusters

```
library(cluster)
clusplot(thera_no_cat,kmeans.cluster$cluster,color = TRUE,shade = TRUE,lines = 1)
```




# CART MODEL

Let's build a Model Since Our Dependent Variable is categorical (0,1) we use Cart model

```
library(rpart)# TO Build CART Decision Trees
library(rpart.plot)# To Visualise the Decision Trees
library(caTools) # To use sample.split function
```

We need to split the data into train and test data so we can test our model performances later on.

```
split <- sample.split(thera$Personal.Loan, SplitRatio = 0.7)
train <- subset(thera,split== TRUE)
test <- subset(thera,split == FALSE)
```

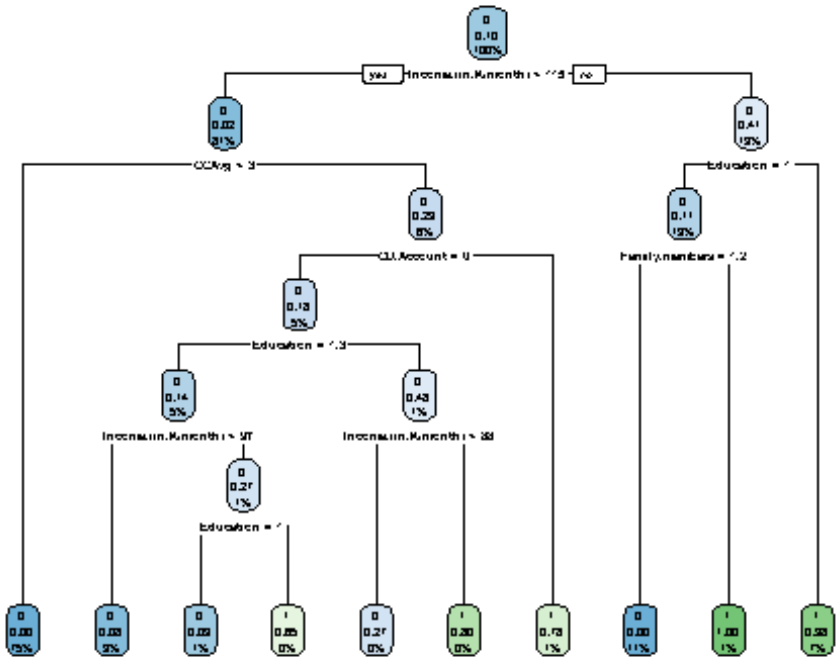
Now, Let's Build a Complex tree model with no cp value and see how the model looks like.

```
cart_tree <- rpart(formula = train$Personal.Loan~.,data = train,method = "class",cp=0)
print(cart_tree)
```

```
## n= 3500
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3500 336 0 (0.904000000 0.096000000)
##    2) Income.(in.K/month)< 114.5 2829  61 0 (0.978437610 0.021562390)
##      4) CCAvg< 2.95 2619  12 0 (0.995418099 0.004581901) *
##      5) CCAvg>=2.95 210  49 0 (0.766666667 0.233333333)
##       10) CD.Account=0 192  35 0 (0.817708333 0.182291667)
##         20) Education=1,3 167  23 0 (0.862275449 0.137724551)
##           40) Income.(in.K/month)< 96.5 115   9 0 (0.921739130 0.078260870) *
##           41) Income.(in.K/month)>=96.5 52  14 0 (0.730769231 0.269230769)
##             82) Education=1 35   3 0 (0.914285714 0.085714286) *
##             83) Education=3 17   6 1 (0.352941176 0.647058824) *
##         21) Education=2 25  12 0 (0.520000000 0.480000000)
##           42) Income.(in.K/month)< 87.5 15   4 0 (0.733333333 0.266666667) *
##           43) Income.(in.K/month)>=87.5 10   2 1 (0.200000000 0.800000000) *
##       11) CD.Account=1 18   4 1 (0.222222222 0.777777778) *
##    3) Income.(in.K/month)>=114.5 671 275 0 (0.590163934 0.409836066)
##      6) Education=1 438  46 0 (0.894977169 0.105022831)
##       12) Family.members=1,2 392   0 0 (1.000000000 0.000000000) *
##       13) Family.members=3,4 46   0 1 (0.000000000 1.000000000) *
##      7) Education=2,3 233   4 1 (0.017167382 0.982832618) *
```

Let’s Visualize the tree so that we can have clear interpretation

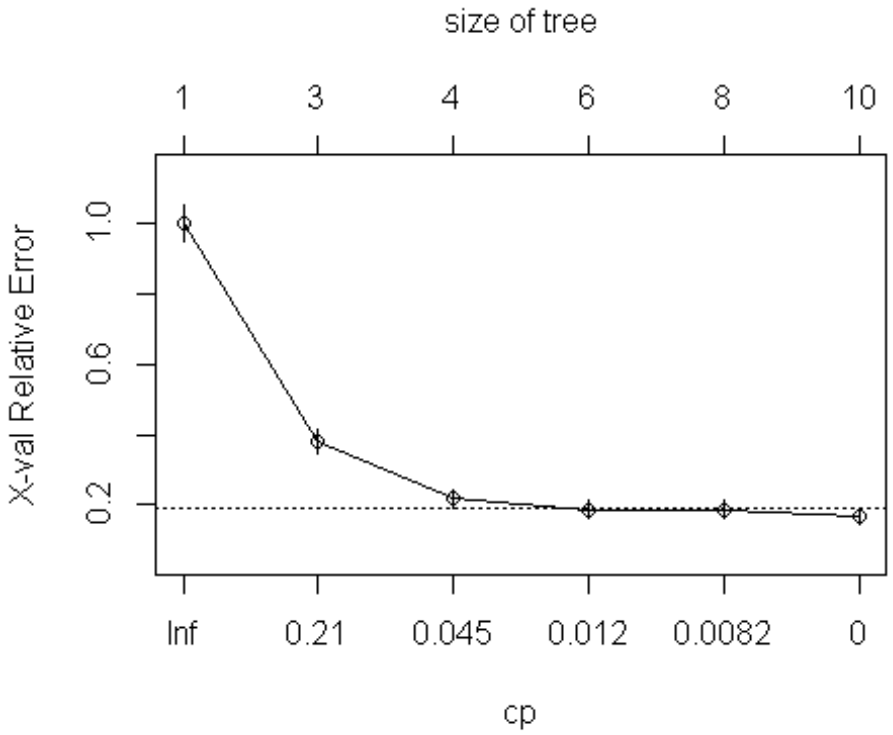
```
rpart.plot(cart_tree)
```



- ✓ Our Tree has 1root node,16 branches(decision nodes) and 17 leaf nodes.
- ✓ Here the tree splits the root node based on the income variable as it has the least gini impurity at income level of 115 k/month.
- ✓ Likewise, it has splitted further 13 branches. The root node has predicted 0 (Customer didn't accept the personal loan offered by the bank) with a probability of 90 %
- ✓ As per Our Model when the income is greater than 115 k/month and the customer has education level of either graduate or Advanced/Professionals then they are more likely to accept our loan offer.
- ✓ Likewise our model has predicted 1 on 7 various subsets (Combinations) from the root node. But mostly our leaf nodes has little observations, this shows how deep (OverFitting) our model has classified the dataset.

- ✓ So now we need to Prune our model with cp value (threshold Value) Now, let's see the complexity chart to set the threshold Value (cp)

```
printcp(cart_tree)

##
## Classification tree:
## rpart(formula = train$Personal.Loan ~ ., data = train, method = "class",
##     cp = 0)
##
## Variables actually used in tree construction:
## [1] CCAvg              CD.Account         Education
## [4] Family.members     Income.(in.K/month)
##
## Root node error: 336/3500 = 0.096
##
## n= 3500
##
##         CP nsplit rel error  xerror     xstd
## 1 0.3348214      0   1.00000 1.00000 0.051870
## 2 0.1369048      2   0.33036 0.38095 0.033050
## 3 0.0148810      3   0.19345 0.21726 0.025162
## 4 0.0089286      5   0.16369 0.18750 0.023409
## 5 0.0074405      7   0.14583 0.18750 0.023409
## 6 0.0000000      9   0.13095 0.16964 0.022286

plotcp(cart_tree)
```




From the above plot it's clear that the X-Val Relative Error remains almost the same after the cp value of 0.010.

So, we use this cp value (0.010) as a threshold to prune the tree.

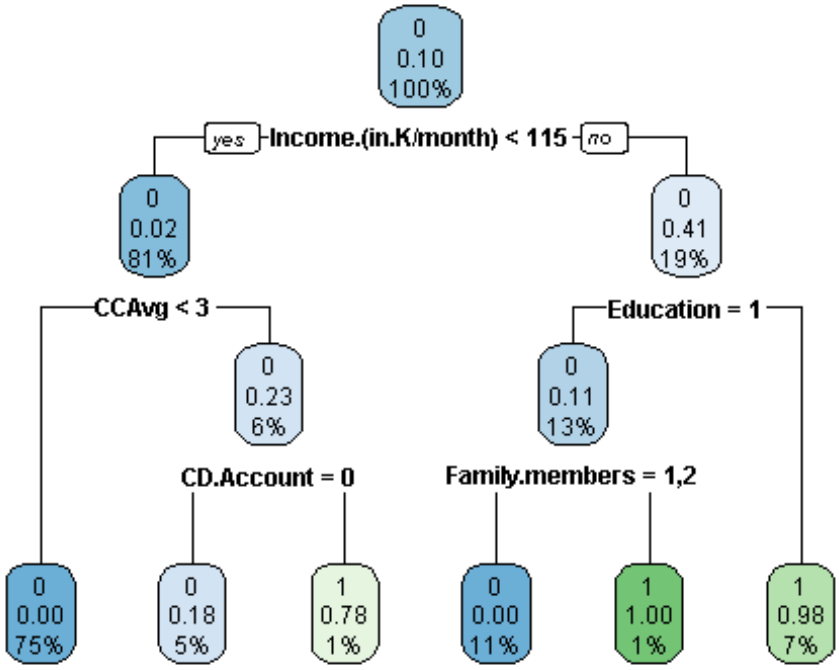
## Pruning

```
pcart_tree <- prune(cart_tree,cp=0.010,"CP")
print(pcart_tree)
```

```
## n= 3500
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3500 336 0 (0.904000000 0.096000000)
##    2) Income.(in.K/month)< 114.5 2829  61 0 (0.978437610 0.021562390)
##      4) CCAvg< 2.95 2619  12 0 (0.995418099 0.004581901) *
##      5) CCAvg>=2.95 210  49 0 (0.766666667 0.233333333)
##       10) CD.Account=0 192  35 0 (0.817708333 0.182291667) *
##       11) CD.Account=1 18   4 1 (0.222222222 0.777777778) *
##    3) Income.(in.K/month)>=114.5 671 275 0 (0.590163934 0.409836066)
```

```
##      6) Education=1 438  46 0 (0.894977169 0.105022831)
##       12) Family.members=1,2 392   0 0 (1.000000000 0.000000000) *
##       13) Family.members=3,4 46   0 1 (0.000000000 1.000000000) *
##      7) Education=2,3 233   4 1 (0.017167382 0.982832618) *
```

```
rpart.plot(pcart_tree)
```




1. Now Our Pruned Tree has 1root node,5 branches (decision nodes) and 6 leaf nodes but still explains the dataset.
2. However, our root node remains the same (That our tree has predicted 0).
3. Out of 6 leaf's, here Our Pruned tree predicted that at three leafs our customer will accept the personal loan offered by thera bank.
4. Let's see the rules at each decision nodes to get the customers who can accept our personal Loan.

```
path.rpart(pcart_tree,c(4,5,6,7))
```

```
##
##  node number: 4
```

```
##    root
##    Income.(in.K/month)< 114.5
##    CCAvg< 2.95
##
##  node number: 5
##    root
##    Income.(in.K/month)< 114.5
##    CCAvg>=2.95
##
##  node number: 6
##    root
##    Income.(in.K/month)>=114.5
##    Education=1
##
##  node number: 7
##    root
##    Income.(in.K/month)>=114.5
##    Education=2,3
```

If the Customer Falls either in the below mentioned three conditions, then there is a huge possibility of accepting the personal loan offered by our bank.

1. Income. (in. K/month)>=114.5 Education=2,3
2. Income. (in. K/month)>=114.5 Education=1 Family members = 3,4
3. Income. (in.K/month)< 114.5 CCAvg>=2.95 C.D. Account = 1

Finally, let's use the decision tree to predict the class for each row and probability score on both train and test dataset.

Let's see how well the model perform in test and train dataset.

```
train$Prediction <- predict(pcart_tree,data = train,type = "class")
train$Prob1 <- predict(pcart_tree,data = train,type = "prob")[,"1"]
head(train)
```

```
##    Age.(in.years) Experience.(in.years) Income.(in.K/month) Family.members
## 1              25                    1                 49              4
```

```
## 2            45                19               34           3
## 3            39                15               11           1
## 5            35                 8               45           4
## 9            35                10               81           3
## 10           34                 9              180           1
##    CCAvg Education Mortgage Personal.Loan Securities.Account CD.Account
## 1    1.6         1        0            0                  1          0
## 2    1.5         1        0            0                  1          0
## 3    1.0         1        0            0                  0          0
## 5    1.0         2        0            0                  0          0
## 9    0.6         2      104            0                  0          0
## 10   8.9         3        0            1                  0          0
##    Online CreditCard Prediction       Prob1
## 1       0          0          0 0.004581901
## 2       0          0          0 0.004581901
## 3       0          0          0 0.004581901
## 5       0          1          0 0.004581901
## 9       1          0          0 0.004581901
## 10      0          0          1 0.982832618
```

Similarly let's do this for test dataset

```
test$Prediction <- predict(pcart_tree,test,type = "class")
test$Prob1 <- predict(pcart_tree,test,type = "prob")["1"]
head(test)
```

```
##    Age.(in.years) Experience.(in.years) Income.(in.K/month) Family.members
## 4              35                    9                 100              1
## 6              37                   13                  29              4
## 7              53                   27                  72              2
## 8              50                   24                  22              1
## 11             65                   39                 105              4
## 13             48                   23                 114              2
##    CCAvg Education Mortgage Personal.Loan Securities.Account CD.Account
## 4    2.7         2        0            0                  0          0
## 6    0.4         2      155            0                  0          0
## 7    1.5         2        0            0                  0          0
## 8    0.3         3        0            0                  0          0
## 11   2.4         3        0            0                  0          0
## 13   3.8         3        0            0                  1          0
##    Online CreditCard Prediction       Prob1
```

```
## 4       0         0         0 0.004581901
## 6       1         0         0 0.004581901
## 7       1         0         0 0.004581901
## 8       0         1         0 0.004581901
## 11      0         0         0 0.004581901
## 13      0         0         0 0.182291667
```

# Model Evaluation on both Test and Train dataset

## Confusion Matrix

```
library(caret)

## Loading required package: lattice

set.seed(1234)
caret::confusionMatrix(train$Prediction,train$Personal.Loan,positive ="1")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 3156   47
##          1    8  289
##
##                Accuracy : 0.9843
##                  95% CI : (0.9796, 0.9881)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9045
##
##  Mcnemar's Test P-Value : 2.992e-07
##
##             Sensitivity : 0.86012
##             Specificity : 0.99747
##          Pos Pred Value : 0.97306
##          Neg Pred Value : 0.98533
##              Prevalence : 0.09600
##          Detection Rate : 0.08257
##    Detection Prevalence : 0.08486
##       Balanced Accuracy : 0.92880
```

```
##
##       'Positive' Class : 1
##
```

- ✓ Our cart Model performed well in train dataset with a accuracy of 98.5%
- ✓ Our model has a sensitivity of 86.01% (True Positive Rate), i.e., how well our model predicted 1(customers who will accept the loan) in training dataset.
- ✓ Our model has a sensitivity of 99.7% (True Negative Rate), i.e., how well our model predicted 0(customers who will not accept the loan) in training dataset.
- ✓ Our Model has done well in predicting the customers who will not accept loan more accurately(0) loan more accurately than customers who will accept the loan(1).

## Building Rank Order Table for K-S,Gain,and Lift Chart on Training Dataset

Next let's calculate the decile thresholds and use those thresholds to compute various columns in a rank order table

```
probs=seq(0,1,length=11)
qs=quantile(train$Prob1, probs)
train$deciles=cut(train$Prob1, unique(qs),include.lowest = TRUE,right=FALSE)
table(train$deciles,train$Personal.Loan)
```

```
##
##                   0    1
##   [0,0.00458)     392    0
##   [0.00458,0.182) 2607   12
##   [0.182,1]       165  324
```

Now Let's construct a table with number of 1's and 0's and its probability deciles on Training Dataset.

```
library(data.table)
trainDT = data.table(train)
```

```
rankTbl = trainDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(Personal.Loan == 1),
  cnt_tar0 = sum(Personal.Loan == 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
## 1:       [0.182,1]  489      324      165 66.26      324          165
## 2: [0.00458,0.182) 2619       12     2607  0.46      336         2772
## 3:     [0,0.00458)  392        0      392  0.00      336         3164
##    cum_rel_resp cum_rel_non_resp    ks
## 1:        96.43            5.21 91.22
## 2:       100.00           87.61 12.39
## 3:       100.00          100.00  0.00
```

- According to KS - 91.22 if we target the top 10 decile group (0.182,1) there is high chance that the customer will respond to our loan offer.

We will next use the ROCR and ineq packages to compute AUC, KS and gini

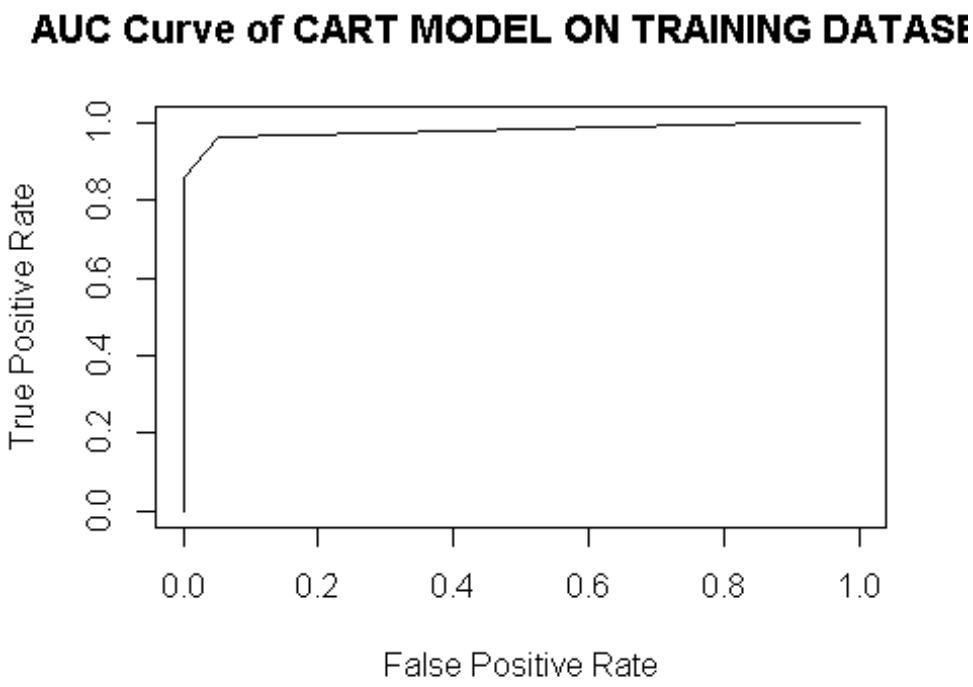
```
library(ROCR) # To Compute AUC
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

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

```
library(ineq)# To Compute the ks and gini scores
predObj = prediction(train$Prob1, train$Personal.Loan)
perf = performance(predObj, "tpr", "fpr")
plot(perf,main="AUC Curve of CART MODEL ON TRAINING DATASET",,xla
b="False Positive Rate",ylab="True Positive Rate")
```




```
KS = max(perf@y.values[[1]]-perf@x.values[[1]]) # The Maximum the Better
print(KS)
```

```
## [1] 0.9121365
```

```
auc = performance(predObj,"auc");
auc = as.numeric(auc@y.values)
print(auc)
```

```
## [1] 0.9800664
```

```
gini = ineq(train$Prob1, type="Gini")
print(gini)
```

```
## [1] 0.86796
```

## Thumb Rule - Larger the auc and gini coefficient better the model

- We have a auc of 98% and gini coefficient of 87% which conveys the message that our model is a good model in training datset.

Finally, we use the Concordance function in the InformationValue package to find the concordance and discordcance ratios:

```
library(InformationValue)

##
## Attaching package: 'InformationValue'

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

print(Concordance(actuals=train$Personal.Loan, predictedScores=train$Prob1))# Concordance(actual,predicted score)

## $Concordance
## [1] 0.9623113
##
## $Discordance
## [1] 0.03768869
##
## $Tied
## [1] -6.938894e-18
##
## $Pairs
## [1] 1063104
```

## Thumb Rule - Larger the Concordance Ratio better the model

- Here, Concordance Ratio of 97% says that our model is a good model on training dataset.

Now Let's Evaluate the model performance measure on testing dataset as well

```
caret::confusionMatrix(test$Prediction,test$Personal.Loan,positive ="1")

## Confusion Matrix and Statistics
##
```

```
##          Reference
## Prediction    0    1
##         0 1352   23
##         1    4  121
##
##               Accuracy : 0.982
##                 95% CI : (0.9739, 0.9881)
##    No Information Rate : 0.904
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.8898
##
##  Mcnemar's Test P-Value : 0.000532
##
##             Sensitivity : 0.84028
##             Specificity : 0.99705
##       'Positive' Class : 1
##
```

- ✓ Our cart Model also performed well in test dataset with a accuracy of 98.2%
- ✓ Our model has a sensitivity of 84% (True Positive Rate),i.e., how well our model predicted 1(customers who will accept the loan) in training dataset.
- ✓ Our model has a specificity of 99.7% (True Negative Rate),i.e., how well our model predicted 0(customers who will not accept the loan) in training dataset.
- ✓ Our Model has done well in predicting the customers who will not accept (0) loan more accurately than customers who will accept the loan(1).

As per Confusion Matrix results our model has done a good job in both training and test dataset.

## Building Rank Order Table for K-S,Gain,and Lift Chart for Test Dataset

Now it's time to calculate the decile thresholds and use those thresholds to compute various columns in a rank order table

```
probs=seq(0,1,length=11)
qs1=quantile(test$Prob1, probs)
test$deciles=cut(test$Prob1, unique(qs1),include.lowest = TRUE,right=FALSE)
table(test$deciles,test$Personal.Loan)
```

```
##
##                   0    1
##   [0,0.00458)     162    0
##   [0.00458,0.182) 1121    3
##   [0.182,1]         73  141
```

We need to construct a table with number of 1's and 0's and its prob deciles in test dataset.

```
testDT = data.table(test)
rankTbl = testDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(Personal.Loan == 1),
  cnt_tar0 = sum(Personal.Loan == 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)*10
0;
rankTbl$cum_rel_non_resp = round(rankTbl$cum_non_resp / sum(rankTbl$cnt_t
ar0),4)*100;
rankTbl$ks = abs(rankTbl$cum_rel_resp - rankTbl$cum_rel_non_resp);
print(rankTbl)
```

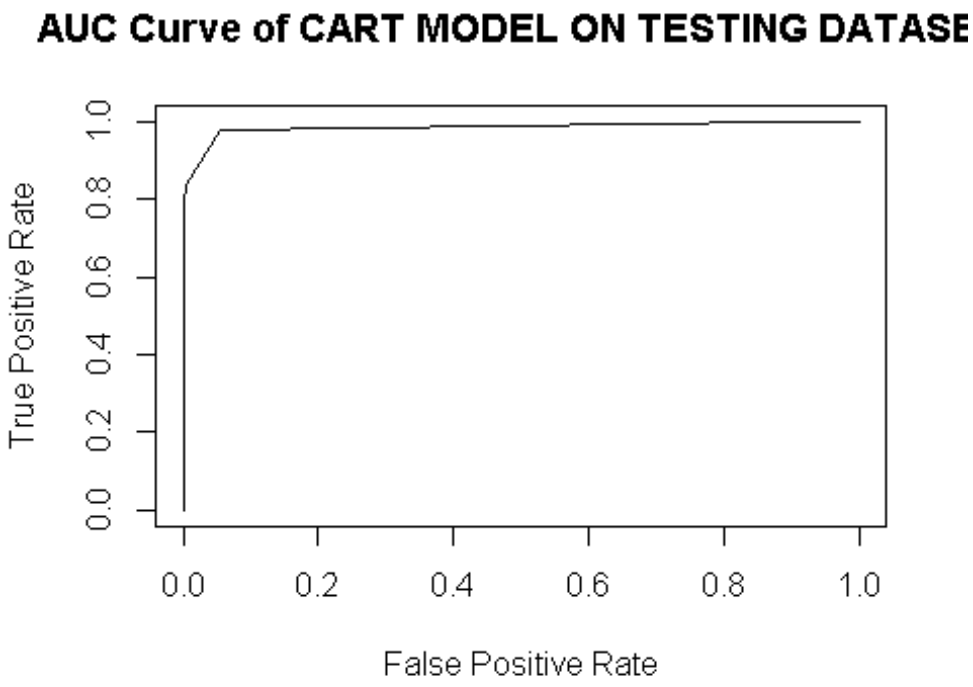
```
##           deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp
## 1:       [0.182,1]  214      141       73 65.89      141          73
## 2: [0.00458,0.182) 1124        3     1121  0.27      144        1194
## 3:     [0,0.00458)  162        0      162  0.00      144        1356
##    cum_rel_resp cum_rel_non_resp    ks
```

```
## 1:       97.92          5.38 92.54
## 2:      100.00         88.05 11.95
## 3:      100.00        100.00  0.00
```

- According to KS - 92.54 if we target the top 10 decile group (0.103,1) there is high chance that the customer will respond to our loan offer

We will next use the ROCR and ineq packages to compute AUC, KS and gini

```
predObj1 = prediction(test$Prob1, test$Personal.Loan)
perf1 = performance(predObj1, "tpr", "fpr")
plot(perf1,main="AUC Curve of CART MODEL ON TESTING DATASET",xlab
="False Positive Rate",ylab="True Positive Rate")
```




```
KS1 = max(perf1@y.values[[1]]-perf1@x.values[[1]]) # The Maximum the Better
print(KS1)
```

```
## [1] 0.9253319
```

```
auc1 = performance(predObj1,"auc");
auc1 = as.numeric(auc1@y.values)
print(auc1)
```

```
## [1] 0.9857398
```

```
gini1 = ineq(test$Prob1, type="Gini")
print(gini1)
```

```
## [1] 0.867303
```

### Thumb Rule - Larger the auc and gini coefficient better the model

- We have a auc of 98% and gini coefficient of 86% which conveys the message that our model is a good model in testing datset.

Finally, we use the Concordance function in the InformationValue package to find the concordance and discordcance ratios

```
print(Concordance(actuals=test$Personal.Loan, predictedScores=test$Prob1))# C
oncordance(actual,predicted score)
```

```
## $Concordance
## [1] 0.9730826
##
## $Discordance
## [1] 0.0269174
##
## $Tied
## [1] -2.428613e-17
##
## $Pairs
## [1] 195264
```

### Thumb Rule - Larger the Concordance Ratio better the model

- Here Concordance Ratio of 97% says that our model is a good model on testing dataset.

# RANDOM FOREST MODEL

Let’s build Random Forest model and check its performance

```
library(randomForest)

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed(1235)
library(caTools)
splitrf <- sample.split(thera$Personal.Loan, SplitRatio = 0.7)
trainrf <- subset(thera,split== TRUE)
testrf <- subset(thera,split == FALSE)
```

Since, Random Forest Function doesnot accept the column names with "()".

So, we need to replace the column names so that we can build the model.

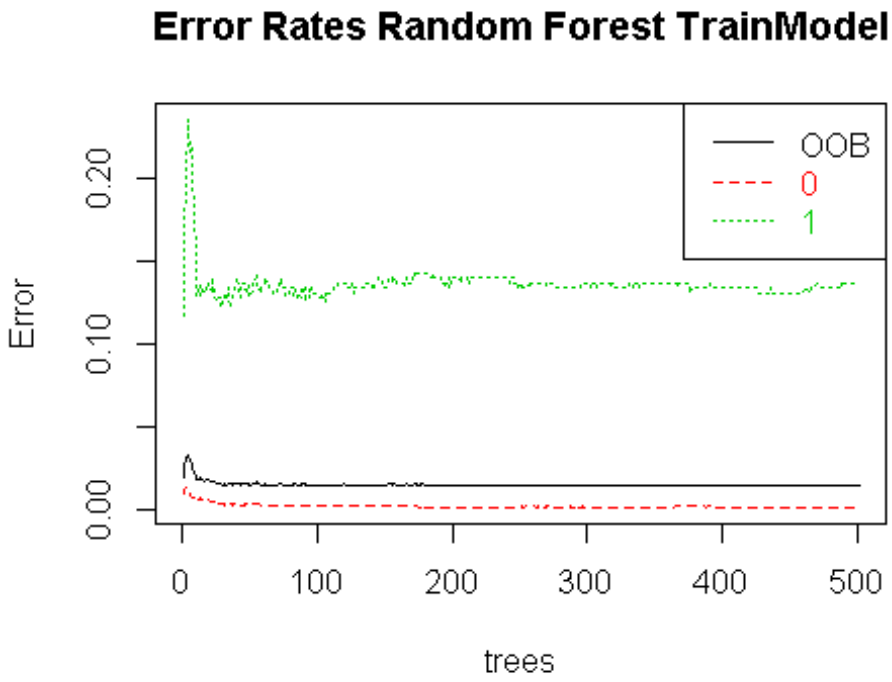
```
trainrf$Ageinyears <- trainrf$`Age.(in.years)`
trainrf$Experienceinyears <- trainrf$`Experience.(in.years)`
trainrf$IncomeinKpermonth <- trainrf$`Income.(in.K/month)`
trainrf<- trainrf[,-c(1,2,3)]
# Similarily for test dataset
testrf$Ageinyears <- testrf$`Age.(in.years)`
testrf$Experienceinyears <- testrf$`Experience.(in.years)`
testrf$IncomeinKpermonth <- testrf$`Income.(in.K/month)`
testrf<- testrf[,-c(1,2,3)]
set.seed(1235)
RFModel <- randomForest(trainrf$Personal.Loan~.,data = trainrf,ntree = 501,mtry = 3,nodesize = 10,importance = TRUE)
print(RFModel)

##
## Call:
##  randomForest(formula = trainrf$Personal.Loan ~ ., data = trainrf,      ntree = 501, mtry = 3, nodesize = 10, importance = TRUE)
```

```
##               Type of random forest: classification
##                     Number of trees: 501
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.51%
## Confusion matrix:
##      0   1 class.error
## 0 3157   7 0.002212389
## 1   46 290 0.136904762
```

## RF Model Error Rate

```
print(RFModel$err.rate,10)
```

```
##              OOB          0         1
##   [1,] 0.02020202 0.010282776 0.1166667
##   [2,] 0.02664129 0.013068479 0.1640212
##   [3,] 0.03115385 0.013977128 0.2008368
##   [4,] 0.03366202 0.012762763 0.2346570
##   [5,] 0.03319502 0.011992945 0.2348993
##   [6,] 0.02934882 0.009797297 0.2154341
##   [7,] 0.02806808 0.007926024 0.2180685
##   [8,] 0.02643948 0.007792208 0.2037037
##   [9,] 0.02356021 0.007397877 0.1762918
##  [10,] 0.02137493 0.007028754 0.1566265
plot(RFModel, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:5, lty=1:3, col=1:3)
title(main="Error Rates Random Forest TrainModel")
```




- From the RFModel error rate plot with respect to number of trees reveals that anything more than 251 trees, it doesn't add any value to the model instead it overfits the model.

Let's check the Important Variables

```
print(RFModel$importance)
```

```
##                              0           1 MeanDecreaseAccuracy
## Family.members     4.200458e-02  0.052631567        0.0430315065
## CCAvg              3.263125e-02  0.067789787        0.0359773813
## Education          6.420644e-02  0.128493012        0.0703895473
## Mortgage           2.114535e-03 -0.002466827        0.0016822491
## Securities.Account 3.824029e-05  0.002396575        0.0002657684
## CD.Account         4.630393e-03  0.020603948        0.0061527947
## Online             7.086747e-04  0.002350748        0.0008691745
## CreditCard         1.223000e-03  0.003232546        0.0014080335
## Ageinyears         5.575233e-03  0.003396351        0.0053551675
## Experienceinyears  5.463750e-03  0.003618297        0.0052753619
## IncomeinKpermonth  1.127523e-01  0.408679041        0.1411083765
##                    MeanDecreaseGini
## Family.members           61.233813
```

```
## CCAvg                86.693128
## Education           121.157647
## Mortgage             15.729333
## Securities.Account    1.523079
## CD.Account           28.198836
## Online                2.151581
## CreditCard            3.425648
## Ageinyears           12.905639
## Experienceinyears    12.704821
## IncomeinKpermonth   193.972604
```

**Thumb Rule : Larger the Mean Decrease values, the more important the corresponding variable.**

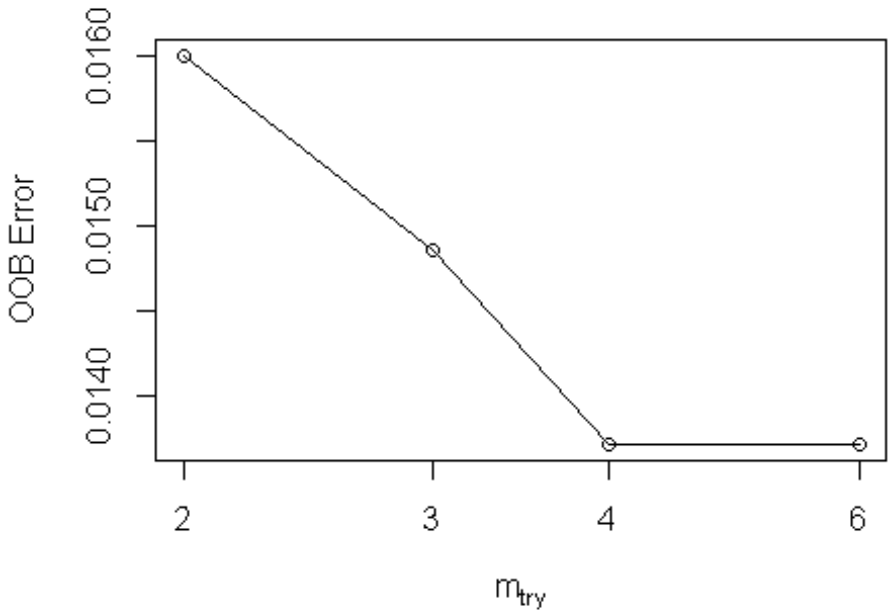
Here as per Mean Decrease (both accuracy and Gini) Income in k/Month is more important Variable.

Now it's time to choose the right m-value to avoid overfitting for that we are going to tune our Random Forest Model.

```
set.seed(1235)
TRFModel = tuneRF(x = trainrf[,-5],
                  y=trainrf$Personal.Loan,
                  mtryStart = 3,
                  ntreeTry = 251,
                  stepFactor = 1.5,
                  improve = 0.0001,
                  trace=TRUE,
                  plot = TRUE,
                  doBest = TRUE,
                  nodesize = 10,
                  importance=TRUE
)
```

```
## mtry = 3  OOB error = 1.49%
## Searching left ...
## mtry = 2     OOB error = 1.6%
## -0.07692308 1e-04
## Searching right ...
## mtry = 4     OOB error = 1.37%
```

```
## 0.07692308 1e-04
## mtry = 6     OOB error = 1.37%
## 0 1e-04
```




- From the mtry vs OOBError plot we can see that at m-4 error is minimum so we can fix m value as 4.

```
importance(TRFModel)
```

```
##                          0        1 MeanDecreaseAccuracy
## Family.members      88.5543183 47.328538          89.805888
## CCAvg               30.7338333 30.517009          33.758996
## Education          154.5750159 72.050732         154.108889
## Mortgage             8.6091560 -3.822972           7.181428
## Securities.Account   0.2572641  2.094952           1.927636
## CD.Account          11.7654940 11.100867          15.072716
## Online               3.4552515  2.767955           4.646940
## CreditCard           6.2867786  3.512775           7.387449
## Ageinyears          14.4595950  2.104480          14.977732
## Experienceinyears   13.5314559  1.425199          13.608598
## IncomeinKpermonth  131.6799779 87.837611         135.408131
##                  MeanDecreaseGini
```

```
## Family.members          69.237979
## CCAvg                   75.691980
## Education              152.549615
## Mortgage                11.327818
## Securities.Account       1.034840
## CD.Account              26.234253
## Online                  1.570634
## CreditCard               2.321123
## Ageinyears              11.415172
## Experienceinyears       10.493404
## IncomeinKpermonth      191.479570
```

- After tuning still the important variable didn't change, i.e.,Income in k/Month is the important variable

Now let's see how well our tuned random forest model predicts(perform) in our training and test dataset.

```
trainrf$Prediction<- predict(TRFModel,data = trainrf,type = "class")
trainrf$Prob1<- predict(TRFModel,data = trainrf,type = "prob")["1"]
head(trainrf)
```

```
##    Family.members CCAvg Education Mortgage Personal.Loan
## 1               4   1.6         1        0             0
## 2               3   1.5         1        0             0
## 3               1   1.0         1        0             0
## 5               4   1.0         2        0             0
## 9               3   0.6         2      104             0
## 10              1   8.9         3        0             1
##    Securities.Account CD.Account Online CreditCard Ageinyears
## 1                   1          0      0          0         25
## 2                   1          0      0          0         45
## 3                   0          0      0          0         39
## 5                   0          0      0          1         35
## 9                   0          0      1          0         35
## 10                  0          0      0          0         34
##    Experienceinyears IncomeinKpermonth Prediction     Prob1
## 1                  1                49          0 0.0000000
## 2                 19                34          0 0.0000000
## 3                 15                11          0 0.0000000
## 5                  8                45          0 0.0000000
```

```
## 9              10             81         0 0.0000000
## 10              9            180         1 0.9836066
```

Similarily let's do this for test dataset

```
testrf$Prediction <- predict(TRFModel,testrf,type = "class")
testrf$Prob1 <- predict(TRFModel,testrf,type = "prob")["1"]
head(testrf)
```

```
##    Family.members CCAvg Education Mortgage Personal.Loan
## 4               1   2.7         2        0             0
## 6               4   0.4         2      155             0
## 7               2   1.5         2        0             0
## 8               1   0.3         3        0             0
## 11              4   2.4         3        0             0
## 13              2   3.8         3        0             0
##    Securities.Account CD.Account Online CreditCard Ageinyears
## 4                   0          0      0          0         35
## 6                   0          0      1          0         37
## 7                   0          0      1          0         53
## 8                   0          0      0          1         50
## 11                  0          0      0          0         65
## 13                  1          0      0          0         48
##    Experienceinyears IncomeinKpermonth Prediction Prob1
## 4                  9               100          0 0.018
## 6                 13                29          0 0.000
## 7                 27                72          0 0.000
## 8                 24                22          0 0.000
## 11                39               105          0 0.028
## 13                23               114          1 0.696
```

# Model Evaluation

## Confusion Matrix

```
caret::confusionMatrix(trainrf$Prediction,trainrf$Personal.Loan,positive ="1")
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 3156   40
```

```
##         1    8  296
##
##               Accuracy : 0.9863
##                 95% CI : (0.9819, 0.9899)
##    No Information Rate : 0.904
##    P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9175
##
##  Mcnemar's Test P-Value : 7.66e-06
##
##             Sensitivity : 0.900495
##             Specificity : 0.99747
##          'Positive' Class : 1
```

- ✓ Our RFModel performed well in train dataset with a accuracy of 98.6%
- ✓ Our model has a sensitivity of 90.4% (True Positive Rate),i.e., how well our model predicted 1(customers who will accept the loan) in training dataset.
- ✓ Our model has a sensitivity of 99.7% (True Negative Rate),i.e., how well our model predicted 0(customers who will not accept the loan) in training dataset
- ✓ Our Model has done well in predicting the customers who will not accept (0) loan more accurately than customers who will accept the loan (1).

As per Confusion Matrix results our model has done a good job in both training and test dataset.

## Building Rank Order Table for K-S,Gain,and Lift Chart for RF Model on Training Dataset

Next let's calculate the decile thresholds and use those thresholds to compute various columns in a rank order table

```
probs=seq(0,1,length=11)
qs2=quantile(trainrf$Prob1, probs)
trainrf$deciles=cut(trainrf$Prob1, unique(qs2),include.lowest = TRUE,right=FALSE)
table(trainrf$deciles,trainrf$Personal.Loan)

##
##                   0    1
##   [0,0.00556)     2443    0
##   [0.00556,0.0253)  356    1
##   [0.0253,0.286)    329   21
##   [0.286,1]         36  314
```

Now Let's construct a table with number of 1's and 0's and its prob deciles for random forest training dataset.

```
trainDTRF = data.table(trainrf)
rankTbl = trainDTRF[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(Personal.Loan == 1),
  cnt_tar0 = sum(Personal.Loan == 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
## 1:        [0.286,1]  350      314       36 89.71      314          36
## 2:   [0.0253,0.286)  350       21      329  6.00      335         365
## 3: [0.00556,0.0253)  357        1      356  0.28      336         721
## 4:      [0,0.00556) 2443        0     2443  0.00      336        3164
##    cum_rel_resp cum_rel_non_resp    ks
## 1:        93.45            1.14 92.31
## 2:        99.70           11.54 88.16
```

```
## 3:      100.00          22.79 77.21
## 4:      100.00         100.00  0.00
```

- According to KS - 92.31 if we target the top 10 decile group (0.286,1) there is high chance that the customer will respond to our loan offer.

Let's Compute AUC, KS and gini

```
predObj2 = prediction(trainrf$Prob1, trainrf$Personal.Loan)
perf2 = performance(predObj2, "tpr", "fpr")
plot(perf2,main="AUC Curve of RF MODEL ON TRAINING DATASET",xlab="False Positive Rate",ylab="True Positive Rate")
```


```
KS2 = max(perf2@y.values[[1]]-perf2@x.values[[1]]) # The Maximum the Better
print(KS2)
```

```
## [1] 0.9497735
```

```
auc2 = performance(predObj2,"auc");
auc2 = as.numeric(auc2@y.values)
print(auc2)
```

```
## [1] 0.9972463
```

```
gini2 = ineq(trainrf$Prob1, type="Gini")
print(gini2)
```

```
## [1] 0.8896336
```

## Thumb Rule - Larger the auc and gini coefficient better the model

- We have a auc of 99% and gini coefficient of 89% which conveys the message that our model is a good model in training datset.

Finally, we use the Concordance function in the InformationValue package to find the concordance and discordcance ratios:

```
print(Concordance(actuals=trainrf$Personal.Loan, predictedScores=trainrf$Prob1
))# Concordance(actual,predicted score)
```

```
## $Concordance
## [1] 0.997243
##
## $Discordance
## [1] 0.002757021
##
## $Tied
## [1] -3.252607e-17
##
## $Pairs
## [1] 1063104
```

## Thumb Rule - Larger the Concordance Ratio better the model

- Here Concordance Ratio of 99.7% says that our RFmodel is a good model on training dataset.

Now Let's Evaluate the model performance measure on testing dataset as well

```
caret::confusionMatrix(testrf$Prediction,testrf$Personal.Loan,positive ="1")
```

```
## Confusion Matrix and Statistics
##
```

```
##          Reference
## Prediction    0    1
##         0 1353   18
##         1    3  126
##
##               Accuracy : 0.986
##                 95% CI : (0.9787, 0.9913)
##    No Information Rate : 0.904
##    P-Value [Acc > NIR] : < 2e-16
##
##                  Kappa : 0.9154
##
##  Mcnemar's Test P-Value : 0.00225
##
##             Sensitivity : 0.8750
##             Specificity : 0.9978
##       'Positive' Class : 1
##
```

- Our RFModel also performed well in test dataset with a accuracy of 98.6%
- Our model has a same sensitivity of 87.5% (True Positive Rate), i.e., how well our model predicted 1(customers who will accept the loan) in training dataset.
- Our model has a sensitivity of 99.78% (True Negative Rate), i.e., how well our model predicted 0(customers who will not accept the loan) in training dataset.

As per Confusion Matrix results our model has done a good job in both training dataset than test dataset.

## Building Rank Order Table for K-S,Gain,and Lift Chart for RF Test Dataset:

Next let's calculate the decile thresholds and use those thresholds to compute various columns in a rank order table

```
probs=seq(0,1,length=11)
qs3=quantile(testrf$Prob1, probs)
testrf$deciles=cut(testrf$Prob1, unique(qs3),include.lowest = TRUE,right=FALSE)
table(testrf$deciles,testrf$Personal.Loan)

##
##                 0   1
##   [0,0.002)     866   0
##   [0.002,0.006) 177   0
##   [0.006,0.026) 156   0
##   [0.026,0.299) 143   8
##   [0.299,0.998]  14 136
```

Now Let's construct a table with number of 1's and 0's and its prob deciles in test dataset.

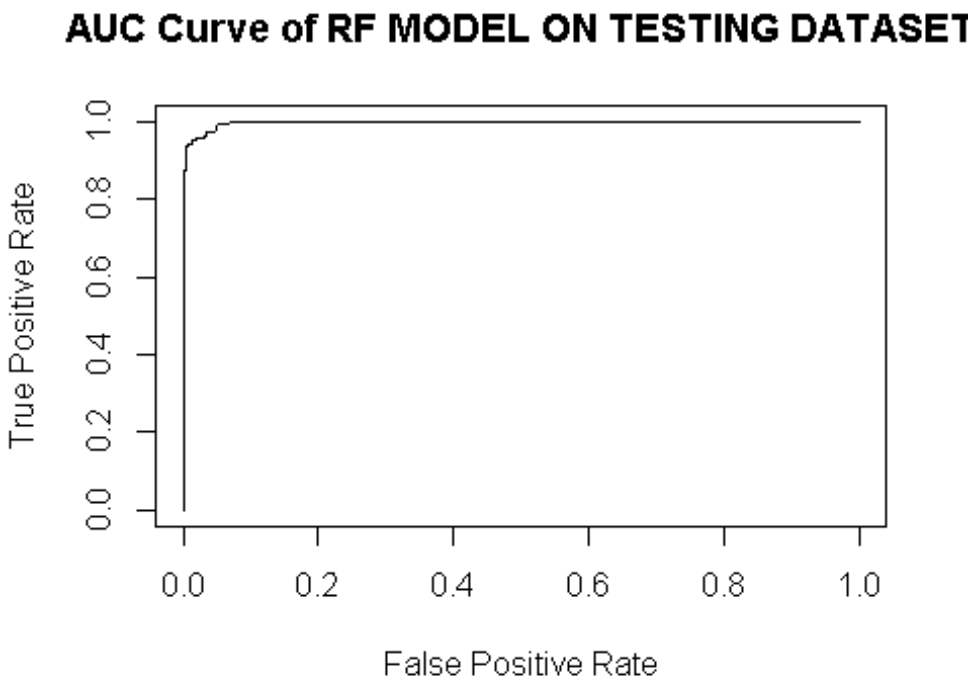
```
testDTRF = data.table(testrf)
rankTbl = testDTRF[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(Personal.Loan == 1),
  cnt_tar0 = sum(Personal.Loan == 0)
),by=deciles][order(-deciles)]
rankTbl$rrate = round(rankTbl$cnt_tar1 / rankTbl$cnt,4)*100;
rankTbl$cum_resp = cumsum(rankTbl$cnt_tar1)
rankTbl$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
## 1: [0.299,0.998] 150      136       14 90.67      136          14
## 2: [0.026,0.299) 151        8      143  5.30      144         157
## 3: [0.006,0.026) 156        0      156  0.00      144         313
## 4: [0.002,0.006) 177        0      177  0.00      144         490
## 5:     [0,0.002) 866        0      866  0.00      144        1356
##    cum_rel_resp cum_rel_non_resp    ks
## 1:        94.44            1.03 93.41
```

```
## 2:      100.00          11.58 88.42
## 3:      100.00          23.08 76.92
## 4:      100.00          36.14 63.86
## 5:      100.00         100.00  0.00
```

- According to KS - 93.41 if we target the top 10 decile group(0.299,1) there is high chance that the customer will respond to our loan offer.

Computing AUC, KS and gini:

```
predObj3 = prediction(testrf$Prob1, testrf$Personal.Loan)
perf3 = performance(predObj3, "tpr", "fpr")
plot(perf3,main="AUC Curve of RF MODEL ON TESTING DATASET",xlab="False Positive Rate",ylab="True Positive Rate")
```




```
KS3 = max(perf3@y.values[[1]]-perf3@x.values[[1]]) # The Maximum the Better
print(KS3)
```

```
## [1] 0.9421706
```

```
auc3 = performance(predObj3,"auc");
auc3 = as.numeric(auc3@y.values)
print(auc3)
```

```
## [1] 0.9974701
```

```
gini3 = ineq(testrf$Prob1, type="Gini")
print(gini3)
```

```
## [1] 0.8884141
```

## Thumb Rule - Larger the auc and gini coefficient better the model

- We have a auc of 99.8% and gini coefficient of 88.9% which conveys the message that our model is a good model in training datset.

Finally, we use the Concordance function in the InformationValue package to find the concordance and discordcance ratios

```
print(Concordance(actuals=testrf$Personal.Loan, predictedScores=testrf$Prob1))
# Concordance(actual,predicted score)
```

```
## $Concordance
## [1] 0.9974445
##
## $Discordance
## [1] 0.002555515
##
## $Tied
## [1] -6.938894e-18
##
## $Pairs
## [1] 195264
```

## Thumb Rule - Larger the Concordance Ratio better the model

- Here Concordance Ratio of 99.7% says that our model is a good model on testing dataset.

# CART MODEL vs RANDOM FOREST MODEL

| PERFORMANCE MEASURES | | Model Evaluation | | | |
|---|---|---|---|---|---|
| | | CART | | RANDOM FOREST | |
| | | TRAIN | TEST | TRAIN | TEST |
| CONFUSION MATRIX | Accuracy | 98.5 | 98.2 | 98.6 | **98.6** |
| | Sensitivity (1) | 86.01 | 84 | 90.4 | **87.5** |
| | Specificity (0) | 99.7 | 99.7 | 99.7 | **99.7** |
| AUC | | 98 | 98 | 99 | **99.8** |
| KS | | 91.22 | 92.54 | 92.31 | **93.4** |
| GINI | | 87 | 87 | 89 | **89** |
| CONCORDANCE RATIO | | 97 | 97 | 99.7 | **99.7** |

Though both models did their best, the above table shows that **Random Forest Model** edged **better** than **CART Model** in predicting the customer who will accept the loan offer.