THERA BANK - Loan Purchase Modeling

Rajeev

9/11/2020

```
knitr::opts_chunk$set(error = TRUE)
```

#THERA BANK - Loan Purchase Modeling Thera Bank has a growing customer base with majority of their customers being liability customers (depositors). The management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors) and in the process earn more through the interests on the loan. A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. 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.

1. Project Objective

- Build a model from the dataset of previous years campaign to identify the potential customers who have a higher probability of purchasing the loan.
- Increase the success ratio while at the same time reduce the cost of the campaign.
- Build the best model which can classify the right customers who have a higher probability of purchasing the loan.

2.Data Dictionary

Load Packages

```
library(readxl) # read excel file
library(ggplot2) # For graphs and visualisations
library(gridExtra) # To plot multiple ggplot graphs in a grid
library(DataExplorer) # To plot correlation plot between numerical variables
library(caTools) # Split Data into Test and Train Set
library(rpart) # To build CART decision tree
library(rattle) # To visualise decision tree
library(randomForest) # To build a Random Forest
library(ROCR) # To visualise the performance classifiers
library(ineq) # To calculate Gini
library(InformationValue) # For Concordance-Discordance
library(dplyr) # for data manipulation
library(readr)#to read data
library(NbClust) # to find optimal number of clusters
library(xml2) # to work with XML files
library(rvest) #to harvest data
```

```
library(stringr) # #to work with strings
library(dplyr) #for data manipulation
library(psych) #for multivariate analysis
library(factoextra) #to extract & visualize multivariate analysis
library(cluster) #to form clusters for K Means
library(caret) #for multifunction training and plotting
library(rpart) #for splitting data intos ubsets and further split
library(RColorBrewer) # for sensible color schemes
library(rattle) #for grphical user interface
```

3. Import Data

4. Exploratory Data Analysis

Check the dimension of the dataset

```
dim(TB_Data)
```

```
## [1] 5000   14
```

The Dataset has 5000 rows & 14 columns

Sanity Checks

```
# Look at the first and last few rows to ensure that the data is read in properly
head(TB_Data)
```

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

```
tail(TB_Data)
```

```
## # A tibble: 6 x 14
##      ID `Age (in years)` `Experience (in~ `Income (in K/m~ `ZIP Code`
##   <dbl>            <dbl>            <dbl>            <dbl>      <dbl>
## 1  4995               64               40               75      94588
## 2  4996               29                3               40      92697
```

```
## 3  4997               30                4               15      92037
## 4  4998               63               39               24      93023
## 5  4999               65               40               49      90034
## 6  5000               28                4               83      92612
## # ... with 9 more variables: Familymembers <dbl>, CCAvg <dbl>, Education <dbl>,
## #   Mortgage <dbl>, PersonalLoan <dbl>, SecuritiesAccount <dbl>,
## #   CDAccount <dbl>, Online <dbl>, CreditCard <dbl>
```

Values in all fields are consistent in each column.

Check the structure of dataset

```
str(TB_Data)
```

```
## tibble [5,000 x 14] (S3: tbl_df/tbl/data.frame)
##  $ ID                    : num [1:5000] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Age (in years)        : num [1:5000] 25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience (in years): num [1:5000] 1 19 15 9 8 13 27 24 10 9 ...
##  $ Income (in K/month)   : num [1:5000] 49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP Code              : num [1:5000] 91107 90089 94720 94112 91330 ...
##  $ Familymembers         : num [1:5000] 4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg                 : num [1:5000] 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education             : num [1:5000] 1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage              : num [1:5000] 0 0 0 0 0 155 0 0 104 0 ...
##  $ PersonalLoan          : num [1:5000] 0 0 0 0 0 0 0 0 0 1 ...
##  $ SecuritiesAccount     : num [1:5000] 1 1 0 0 0 0 0 0 0 0 ...
##  $ CDAccount             : num [1:5000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Online                : num [1:5000] 0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard            : num [1:5000] 0 0 0 0 1 0 0 1 0 0 ...
```

Observations:

1. ID is an identity variable and not useful for predictive modeling
2. Personal Loan is the response variable and is a numerical variable which may be needed to changed to a factor
3. All the other variable are numerical variables as they should be

Get Summary of the dataset

```
summary(TB_Data)
```

```
##        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     Familymembers       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   :93152   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      PersonalLoan   SecuritiesAccount   CDAccount
##  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
##
```

Obervations: 1. ID is a identity variable can should be dropped 2. Age has a huge range from 23 to 67 3. Corresponding to the age is the huge range in their professional work experience 4. Annual Incomes range from as low as 8000 right up to 224,000. The income distribution is positively skewed. 5. ZIP Code is the Home address code and not a useful variable for predictive modeling for this data. This variable shoud be dropped. 6. Family Members range between 1 & 4. The distribution is negative skewed. There are 18 NA's which mean the data from 18 Age variables are missing. 7. The average spending on the Credit Card per month ranges from 0 to 10,000. There seem to be many outliers as the mean is 1,500 & median is 1,938. the 3rd Quartile is only 2,500. 8. Education Level Range from Under Graduates to Graduates to Professionals. The range seems spread out. 9. The value of house Mortgage is completely right skewed. The median here is 0 while mean is 56,500 stretching right up to 635,000. 10. Customer who accepted Personal Loan offered in the last campaign is 9.6% 11. Customers having security accounts in Thera Bank is around 10.5% 12. Customers having Certificate of Deposit accounts in Thera Bank is around 6% 13. Customers using internet banking facility is close to 60% 14. Customer using There Bank credit card is around 29.5%

Change column names

```
colnames(TB_Data)[colnames(TB_Data)%in% c("Age (in years)",
                                          "Experience (in years)",
                                          "Income (in K/month)",
                                          "ZIP Code",
                                          "Familymembers",
                                          "PersonalLoan")]=
  c("Age", "Experience","Income", "zip","FamilyMember", "PersonalLoan")
```

There seem to be use of symbols like 'brackets' and space used in the column names which could create error

in reading while performing test at various levels. Hence we remove the brackets with the info in them & space between names.

Drop insignificant columns

```
TB_Data=subset(TB_Data, select = -c(ID,zip))
dim(TB_Data)
```

```
## [1] 5000   12
```

The data now has 5000 rows and 12 columns

Missing value treatment

```
sum(is.na(TB_Data))
```

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

```
colSums(is.na(TB_Data))
```

```
##               Age        Experience            Income      FamilyMember
##                 0                 0                 0                18
##             CCAvg         Education          Mortgage      PersonalLoan
##                 0                 0                 0                 0
## SecuritiesAccount         CDAccount            Online        CreditCard
##                 0                 0                 0                 0
```

```
colnames(TB_Data)[apply(TB_Data, 2, anyNA)]
```

```
## [1] "FamilyMember"
```

```
TB_Data$FamilyMember[is.na(TB_Data$FamilyMember)]=median(TB_Data$FamilyMember,na.rm = TRUE)
anyNA(TB_Data)
```

```
## [1] FALSE
```

Observations: 1. The treatment shows 18 values in the “Family members” column missing (NA’s) 2. All NA’s are replaced using the median of the “Family member” column 3. There are no missing values in the Data

Univariate analysis

```
#Distribution of the dependent variable
prop.table(table(TB_Data$PersonalLoan))
```

```
##
##     0     1
## 0.904 0.096
```

1. 9.6% of the total customers accepted the Personal Loan in the last campaign. We need to determine factors that drive customers to accept the Personal Loan so that we suggest measure to increase the percentage of customers accepting the Personal Loan.
2. We will also create models that will help us accurately predict "Potential Customers" for Personal Loans.

```
plot_histogram_n_boxplot = function(variable, variableNameString, binw){
  h = ggplot(data = TB_Data, aes(x= variable))+
    labs(x = variableNameString,y ='count')+
    geom_histogram(fill = 'green',col = 'white',binwidth = binw)+
    geom_vline(aes(xintercept=mean(variable)),
               color="black", linetype="dashed", size=0.5)
  b = ggplot(data = TB_Data, aes('',variable))+
    geom_boxplot(outlier.colour = 'red',col = 'red',outlier.shape = 19)+
    labs(x = '',y = variableNameString)+ coord_flip()
  grid.arrange(h,b,ncol = 2)
}
```

Function to draw histogram and boxplot of numerical variables using ggplot

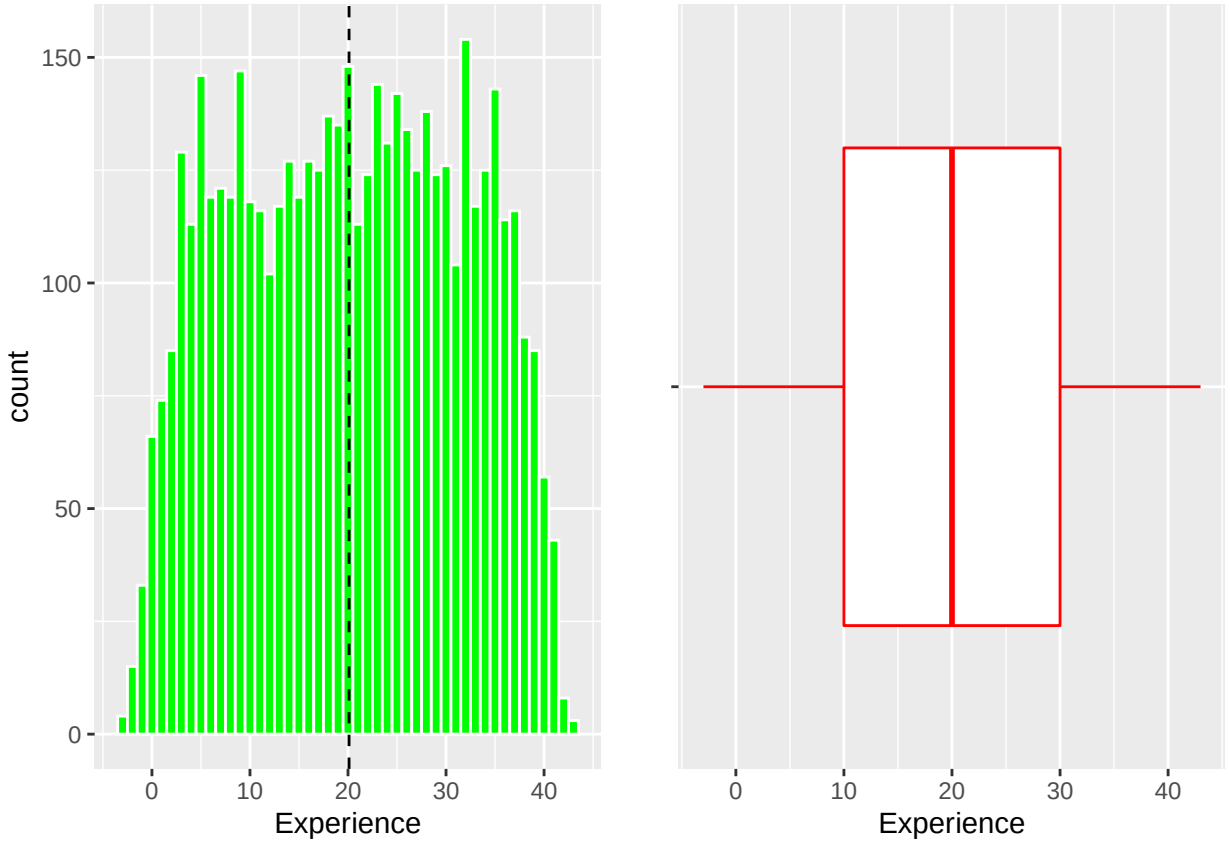
a. Observations on Age

```
plot_histogram_n_boxplot(TB_Data$Age, 'Age', 1)
```

1. As seen in summary the range is huge i.e. between 23 & 67 2. The mean & median are both around 45.

b. Observations on Experience

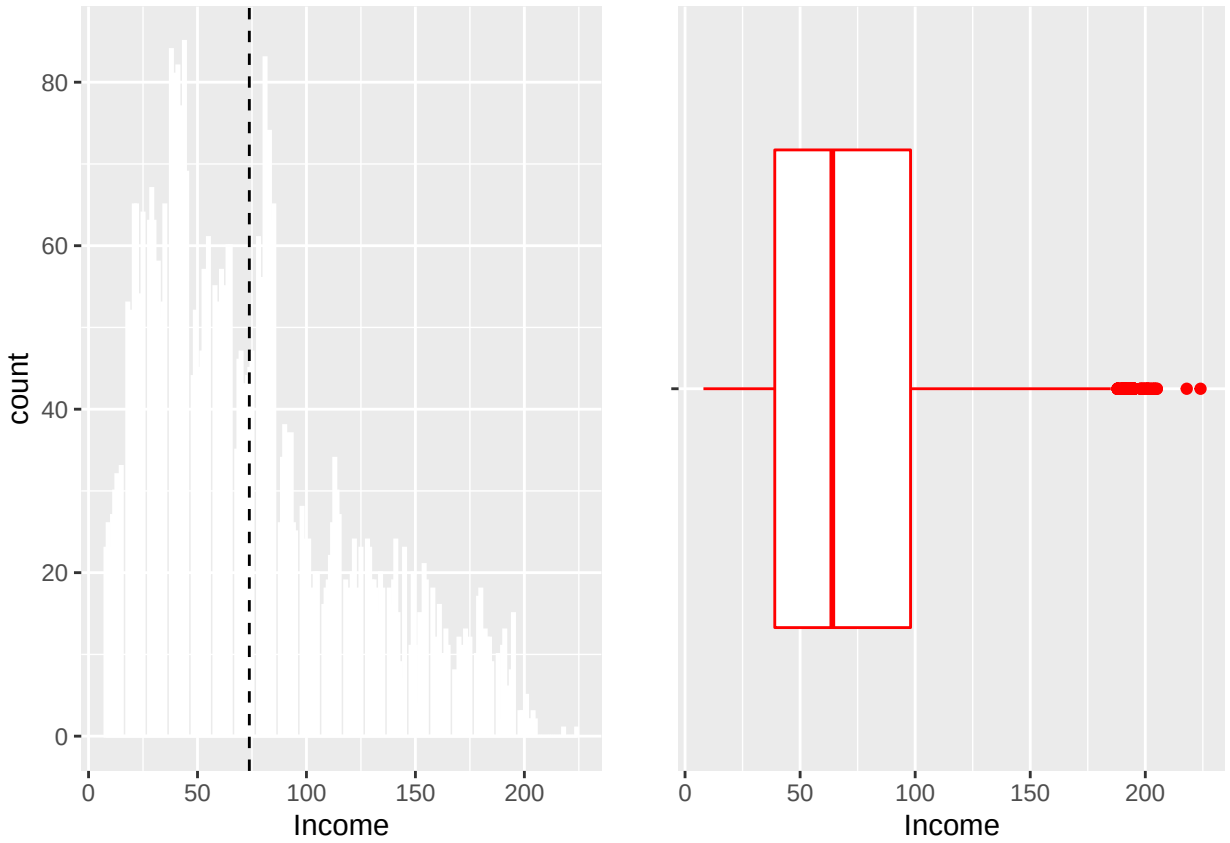
```
plot_histogram_n_boxplot(TB_Data$Experience, 'Experience', 1)
```



1. Proportionate to the Age, the work experiences of the customers also has a wide range. 2. The mean & median experience is at 20. 3. A few are also showing lack of experience.

c. Observations on Income

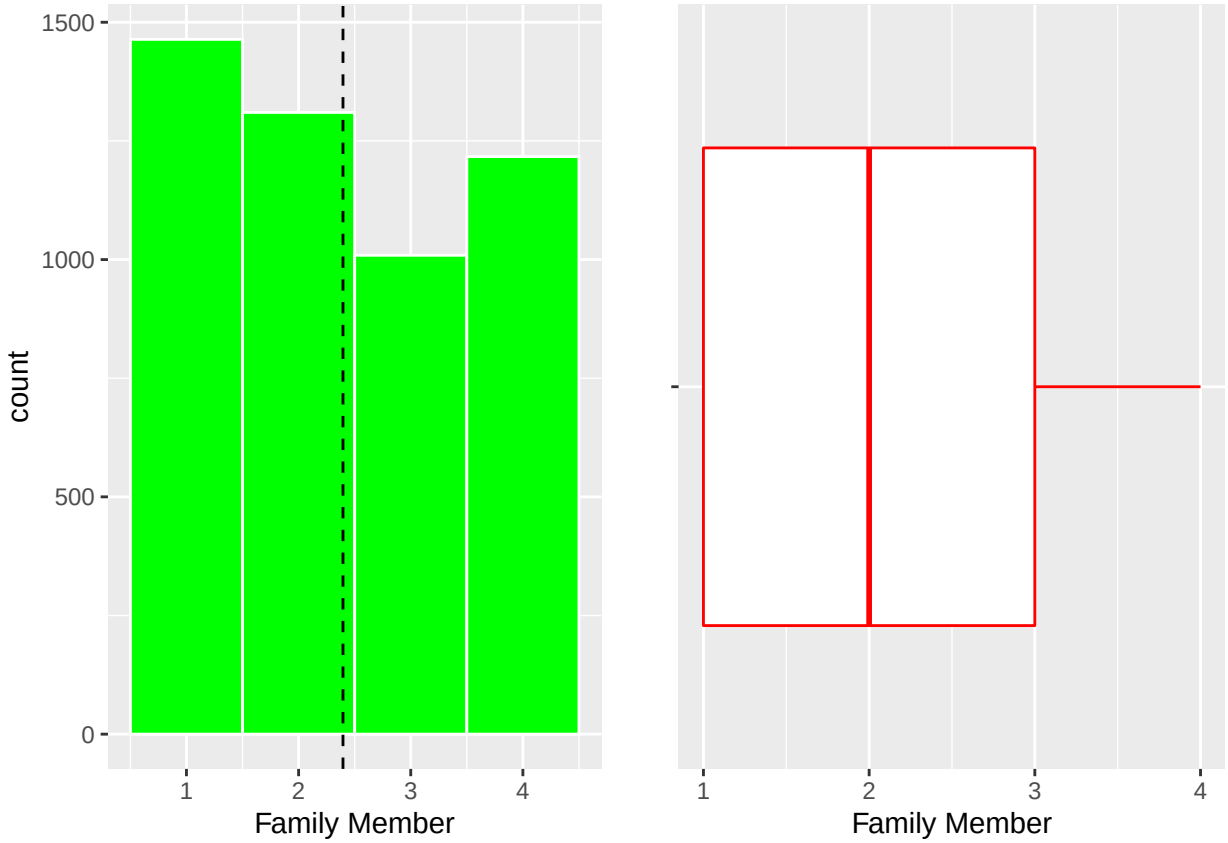
```
plot_histogram_n_boxplot(TB_Data$Income, 'Income', 1)
```



1. Annual Income levels range from as low as 8000 and stretch beyond 200,000 with 2 outliers beyond it. 2. The data is right skewed and has two peaks at around 40,000 and 80,000 range. 3. Can see small concentration at 190,00 & 200,00 ranges.

d. Observations on Family Members

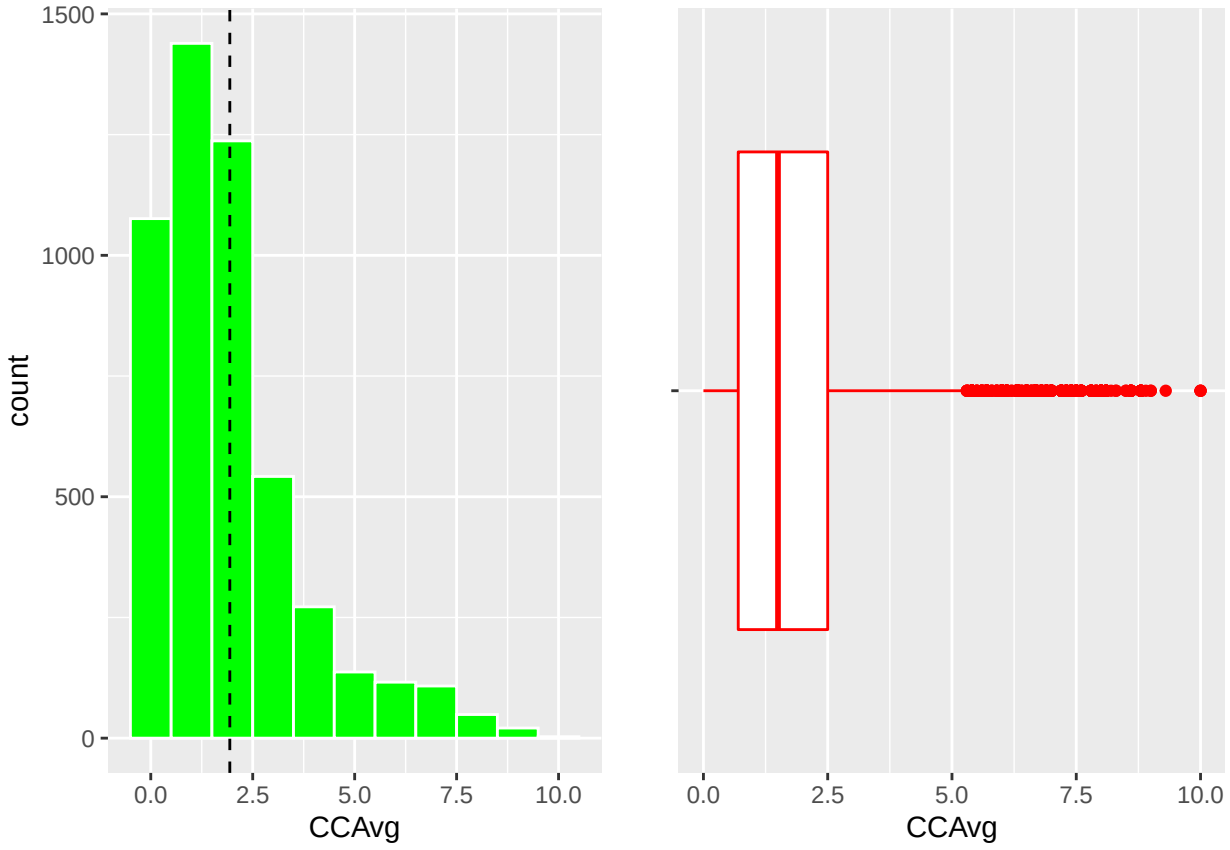
```
plot_histogram_n_boxplot(TB_Data$FamilyMember, 'Family Member', 1)
```



1. Family Members range between 1 to 4, with most with 1 family member, followed by 2 and 4. 2. There are 18 values missing in the data for Age

e. Observations on Average spending on Credit Card per month

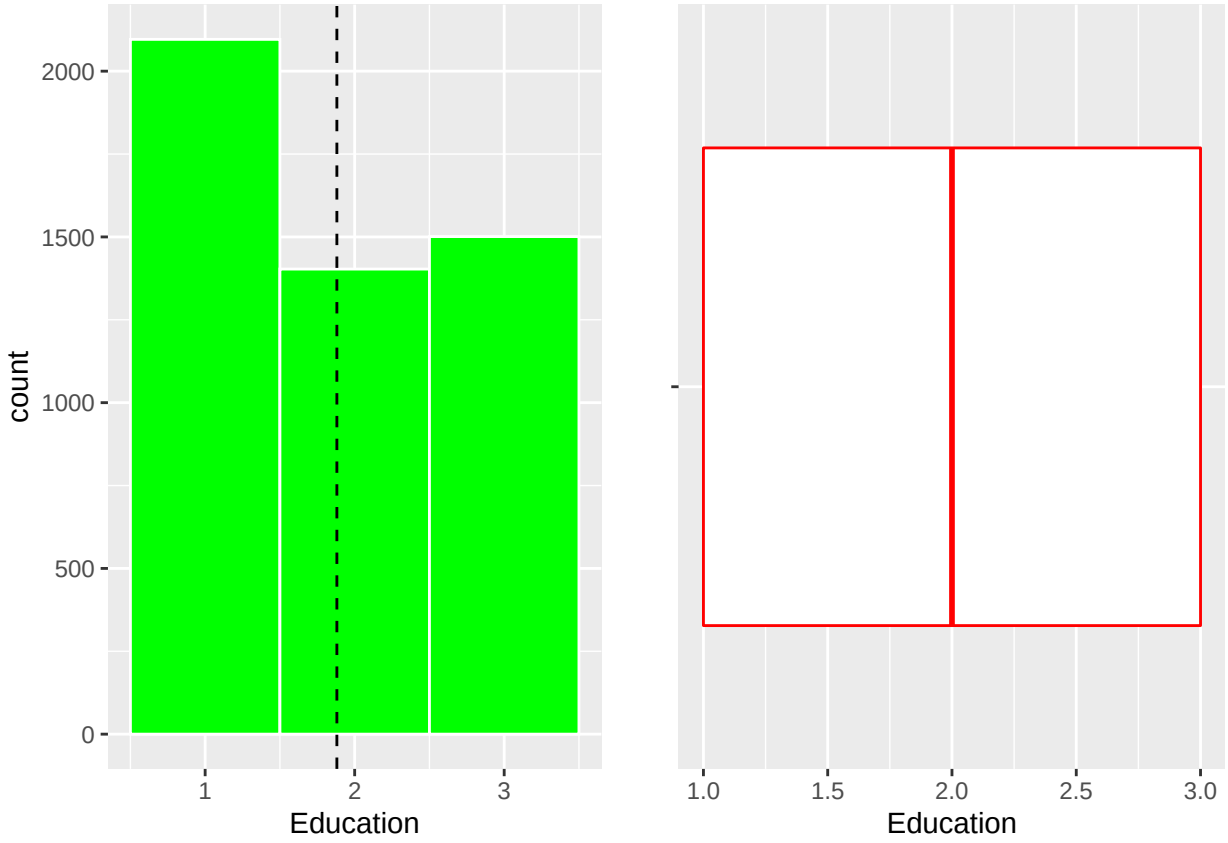
```
plot_histogram_n_boxplot(TB_Data$CCAvg, 'CCAvg', 1)
```



1. Data is right skewed with a peak at around 1000
2. There are many customers who do not use their Credit Card
3. There are many customers who spend a large amount on their credit card monthly compared to others.
4. The people spending higher on their credit card are spending in the range of 5000 to 9000 monthly.

f. Observations on Customer Education Level

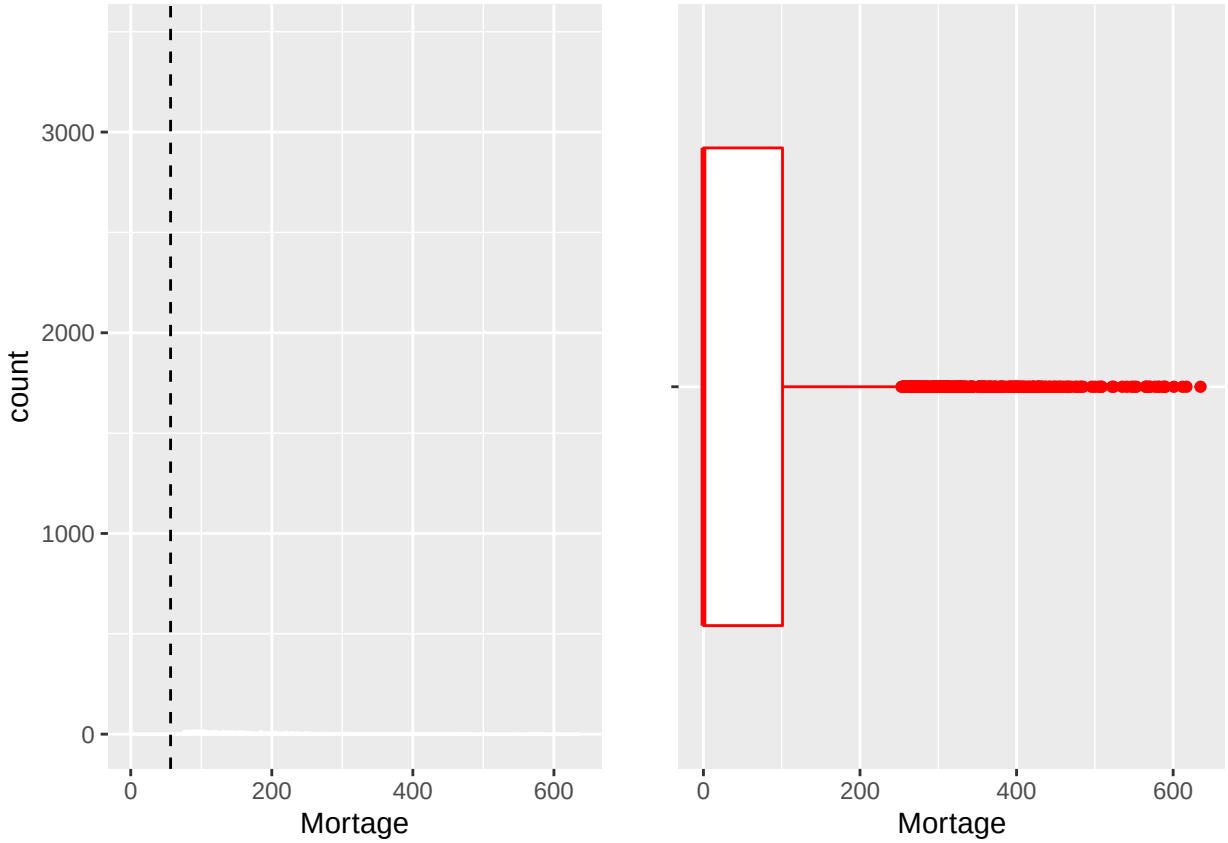
```
plot_histogram_n_boxplot(TB_Data$Education, 'Education', 1)
```



1. There are three levels of Education in the data i.e. 1=Undergraduate, 2=Graduate & 3=Advanced/Professionals.
2. Maximum customers are Undergraduates. There are more Advanced professionals in the data compared to the Graduates.

g. Observations on Customer with Mortgage (and value of Mortage)

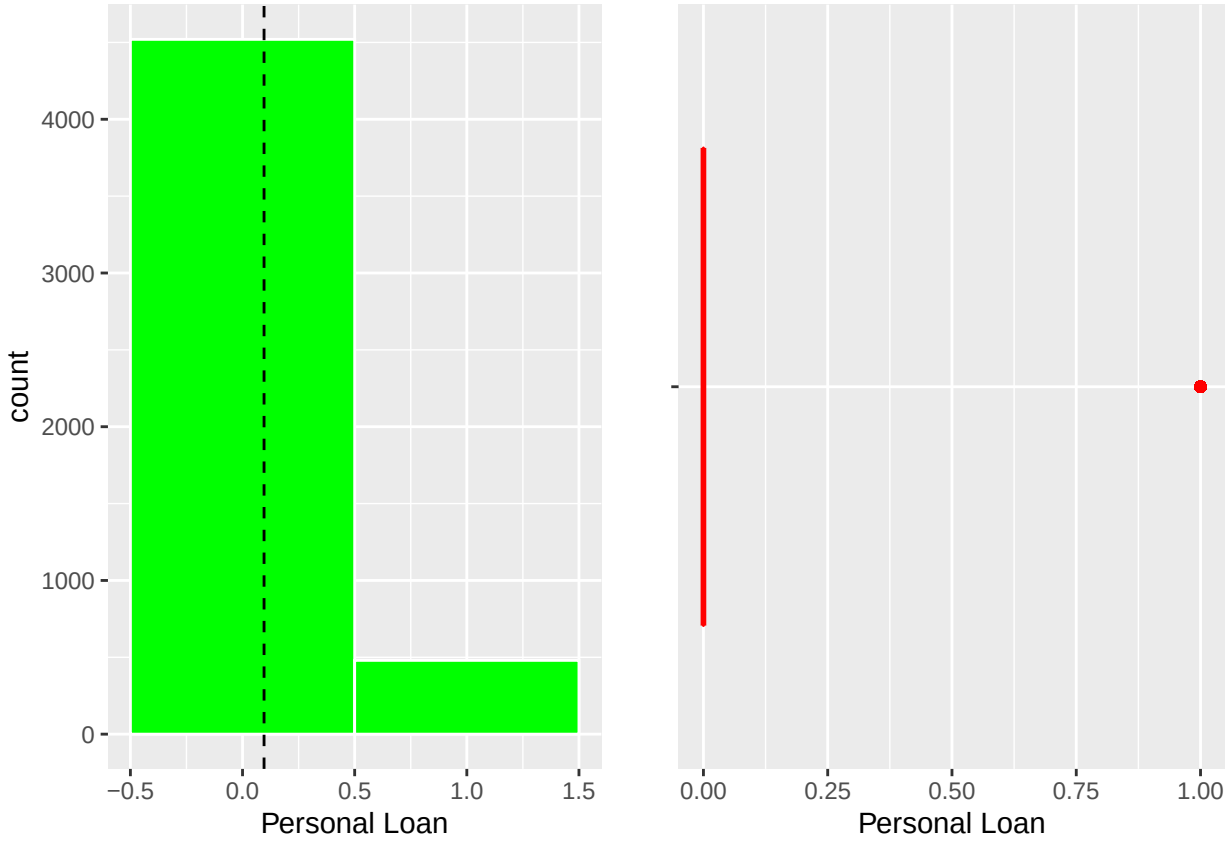
```
plot_histogram_n_boxplot(TB_Data$Mortgage, 'Mortage', 1)
```



1. Most of the customers do not have Mortgage on their house.
2. The maximum Mortgage values are between 250,000 to 500,00

h. Observations on Customer who accepted the Personal Loan during the last campaign

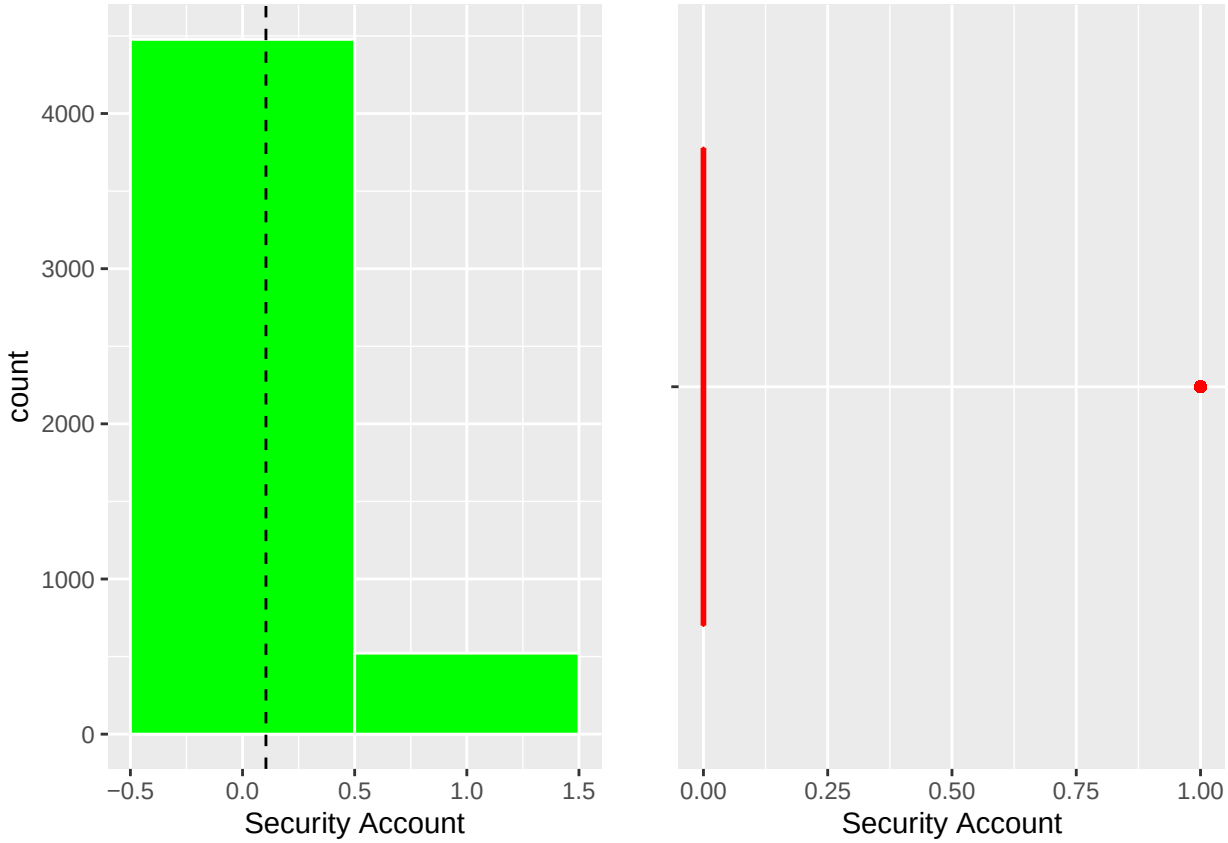
```
plot_histogram_n_boxplot(TB_Data$‘PersonalLoan‘, ’Personal Loan’, 1)
```



1. Only 9.5% of the customer base seem to have accepted the Personal Loan

i. Observations on Customer having Security Account with the bank

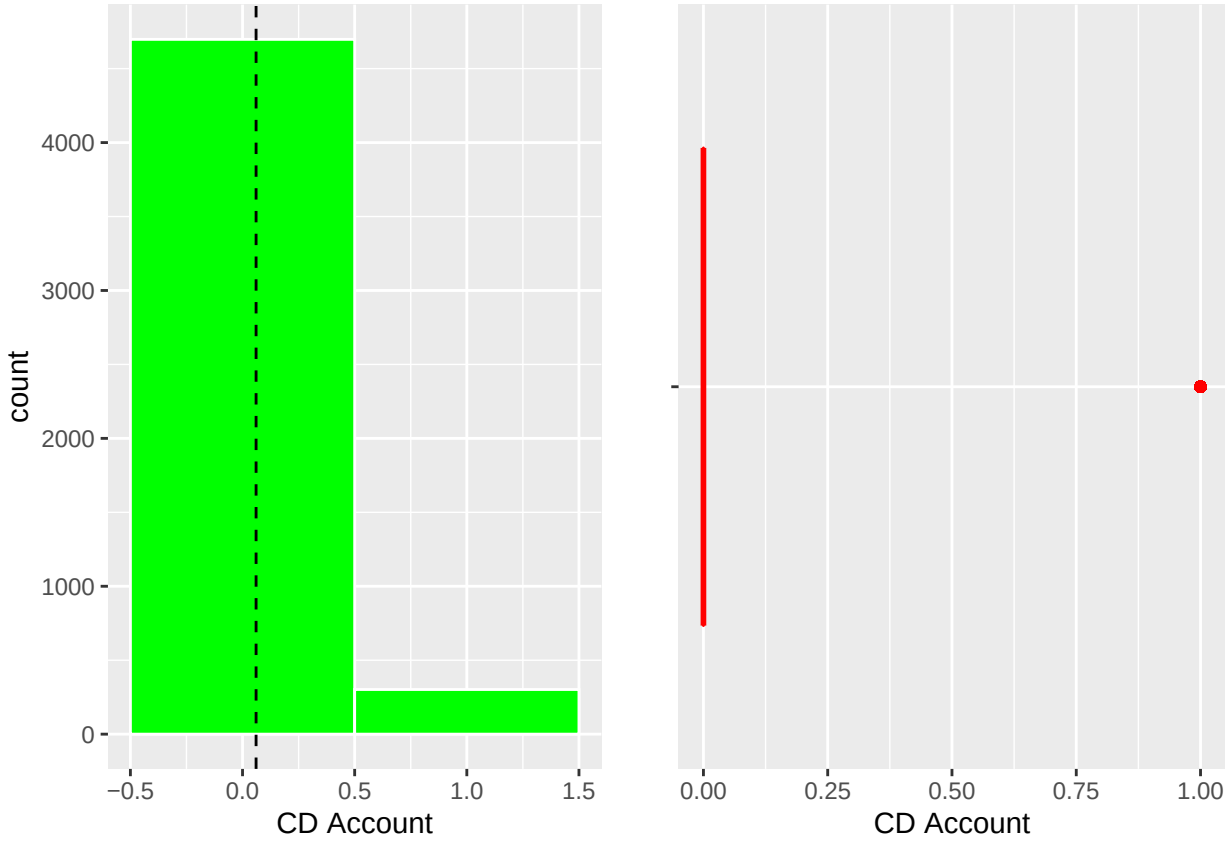
```
plot_histogram_n_boxplot(TB_Data$`SecuritiesAccount`, 'Security Account', 1)
```



1. Around 10.5% of customers have Security Account

j. Observations on Customer having Certificate of Deposit Account with the bank

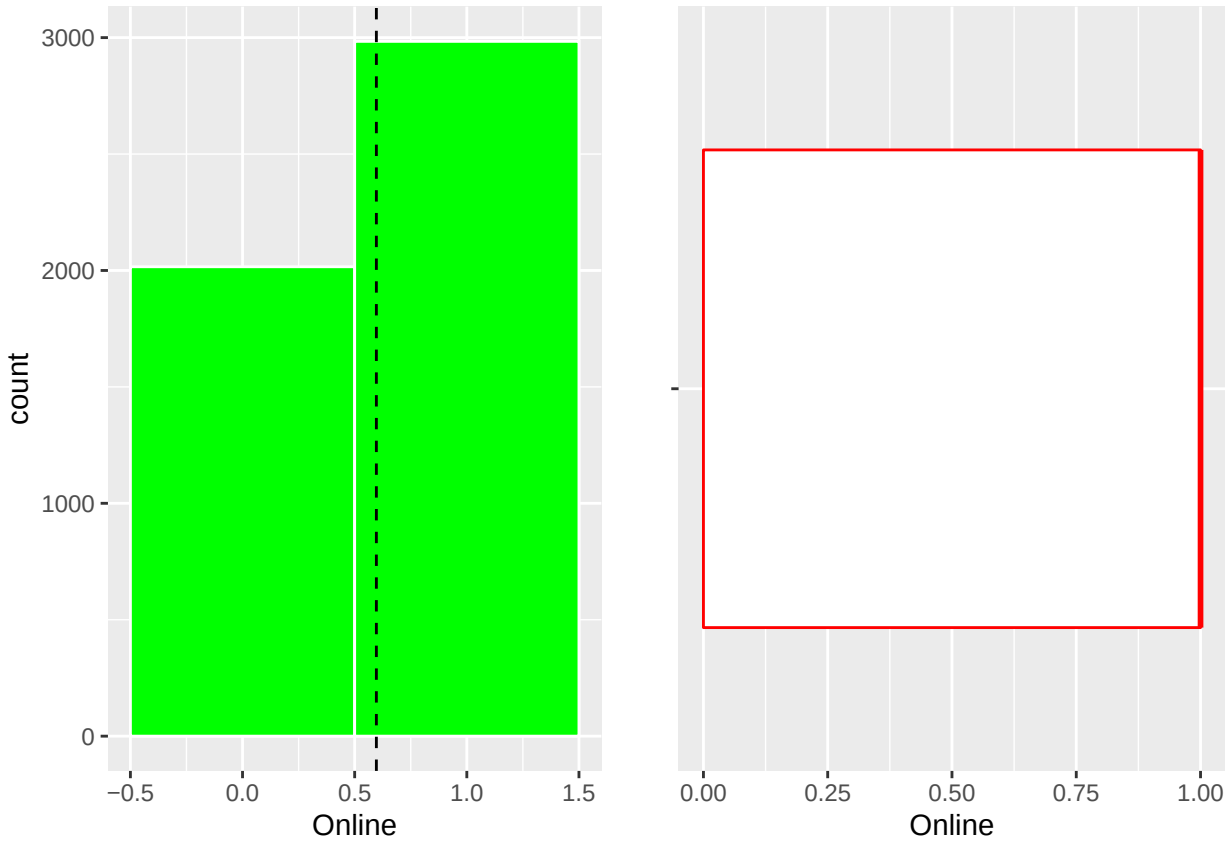
```
plot_histogram_n_boxplot(TB_Data$`CDAccount`, 'CD Account', 1)
```



1. Around 6% of customers have CD Account

k. Observations on Customer using Internet Banking

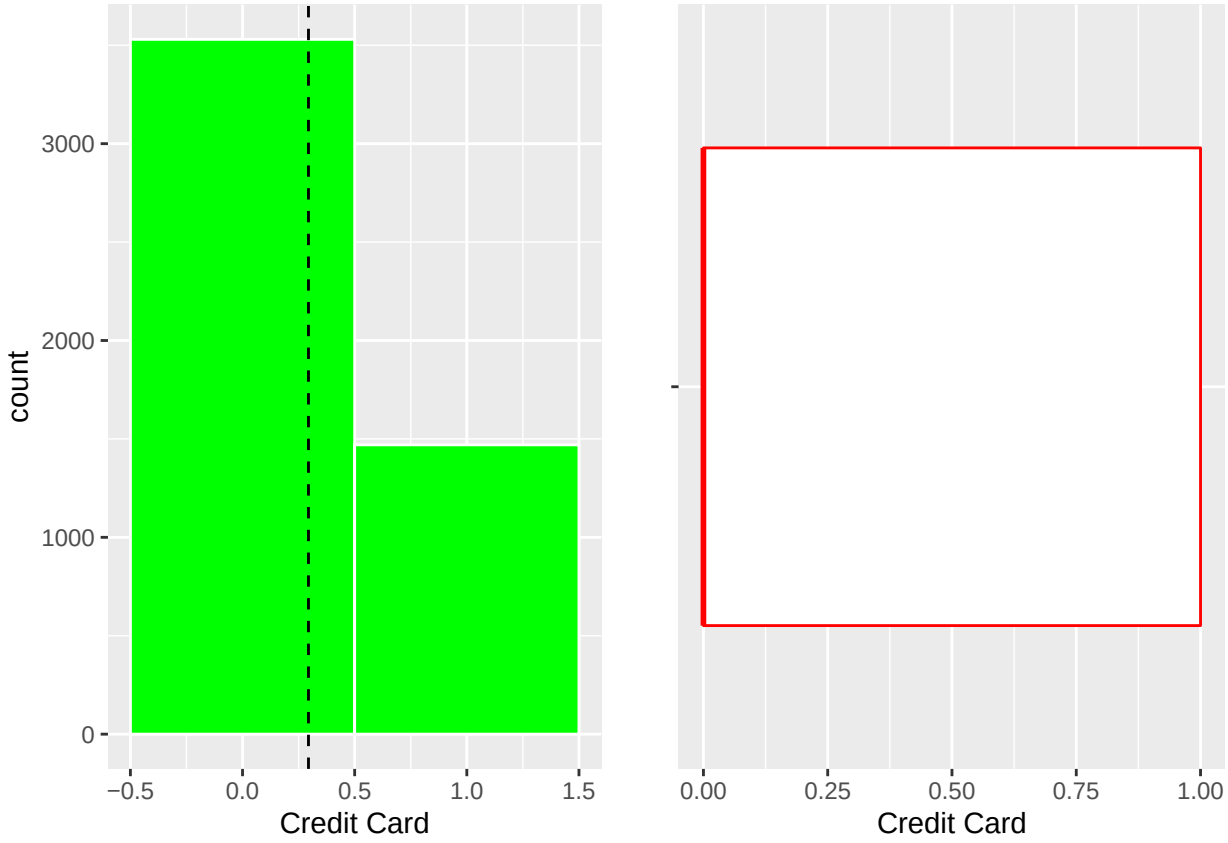
```
plot_histogram_n_boxplot(TB_Data$Online, 'Online', 1)
```



1. Almost 60% of customer use Internet Banking

l. Observations on Customer using Thera Bank Credit Card

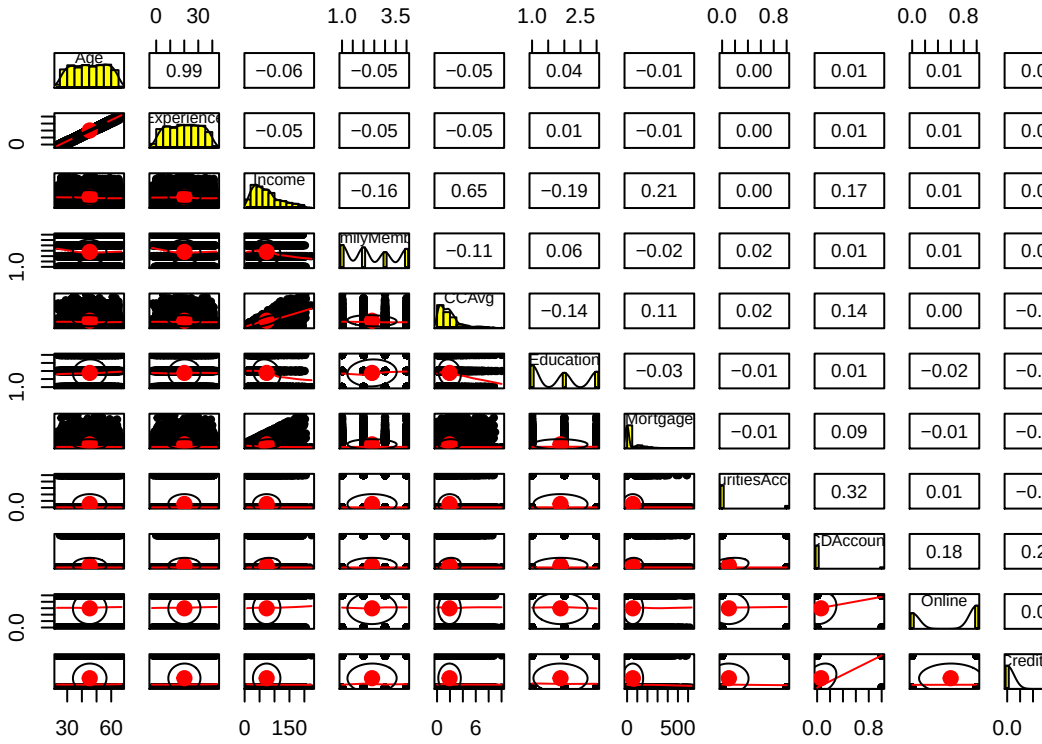
```
plot_histogram_n_boxplot(TB_Data$CreditCard, 'Credit Card', 1)
```



1. Almost 30% customer use bank's credit card

Bivariate Analysis

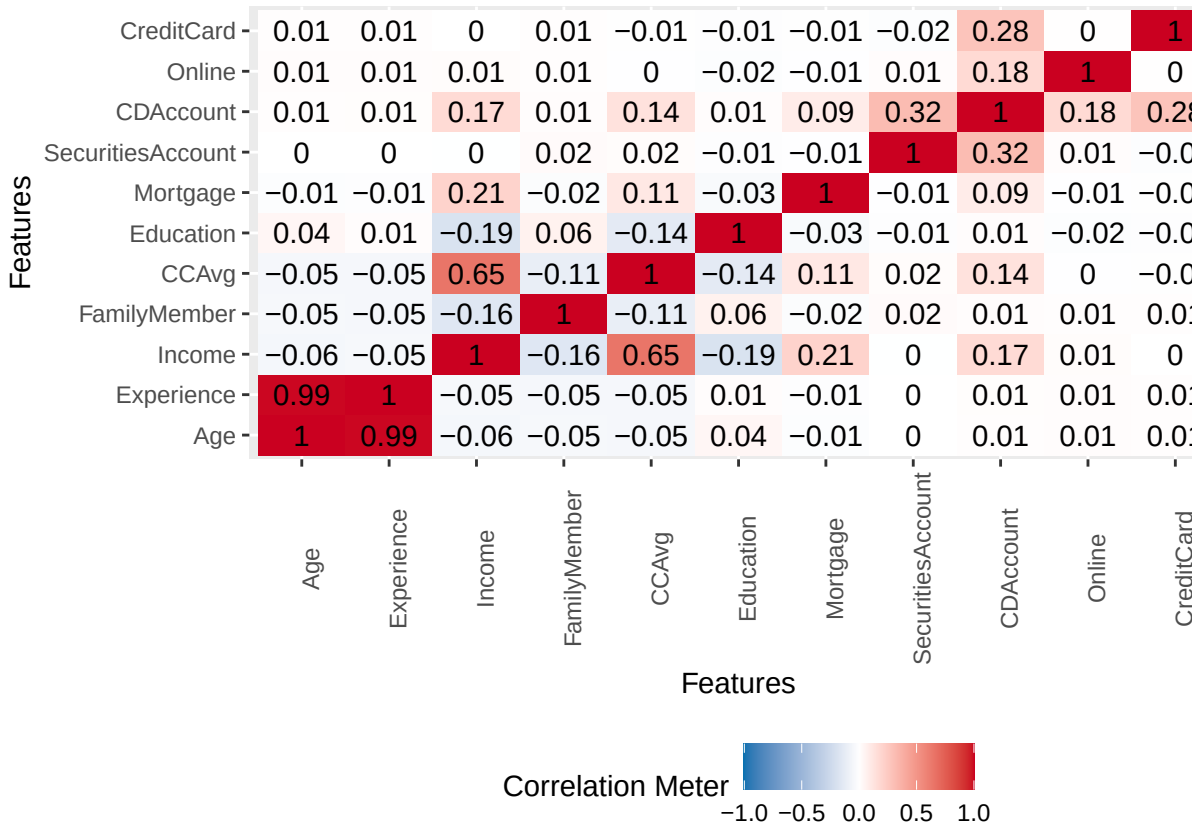
```
pairs.panels(TB_Data[, -8],
             method = "pearson", # correlation method
             hist.col = "yellow",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)
```



Bivariante Scatter Plot:

Observations: 1. Prominent positive correlation between: + Age & Experience + Income & Avg Monthly Spending on Credit card + Income & Mortgage + Mortgage & Avg Monthly Spending on Credit card + Customers with Secutities Account & Customers with CD Account + Avg Monthly Spending on Credit card & Customers with CD Account + Customer using Internet & Customers with CD Account 2. Prominent negative correlation between: + Income & Family Members + Avg Monthly Spending on Credit card & Family Members + Income & Education

```
plot_correlation(TB_Data[, -8])
```



Correlation Plot

1. Average spending on Credit Card per month & Income have high positive correlation

```
cor.test(TB_Data$CCAvg,TB_Data$Income)$p.value
```

Check Correlation between Credit Card spend & Income:

```
## [1] 0
```

1. the CC Avg & Income p.values « 0.001 indicating these correlation is significant

Chi Square Test

```
chisq.test(TB_Data$CCAvg,TB_Data$Income)
```

```
## Warning in chisq.test(TB_Data$CCAvg, TB_Data$Income): Chi-squared approximation
## may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  TB_Data$CCAvg and TB_Data$Income
## X-squared = 40267, df = 17227, p-value < 2.2e-16
```

```r
chisq.test(TB_Data$Mortgage,TB_Data$Income)
```

```
## Warning in chisq.test(TB_Data$Mortgage, TB_Data$Income): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  TB_Data$Mortgage and TB_Data$Income
## X-squared = 81490, df = 55706, p-value < 2.2e-16
```

```r
chisq.test(TB_Data$CDAccount,TB_Data$Income)
```

```
## Warning in chisq.test(TB_Data$CDAccount, TB_Data$Income): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  TB_Data$CDAccount and TB_Data$Income
## X-squared = 407.73, df = 161, p-value < 2.2e-16
```

```r
chisq.test(TB_Data$CDAccount,TB_Data$SecuritiesAccount)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  TB_Data$CDAccount and TB_Data$SecuritiesAccount
## X-squared = 498.21, df = 1, p-value < 2.2e-16
```

```r
chisq.test(TB_Data$CDAccount,TB_Data$CreditCard)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  TB_Data$CDAccount and TB_Data$CreditCard
## X-squared = 385.65, df = 1, p-value < 2.2e-16
```

1. We reject the null hypothesis if the p-value that comes out in the result is less than a predetermined significance level, which is 0.05 usually, then we reject the null hypothesis. H0: The two variables are independent. H1: The two variables relate to each other.
2. The Chi Square test performed between:

- CC Avg & Income
- Mortgage & Income
- CD Account & Income
- CD Account & Securities Account
- CD Account & Credit Card

3. The p-value in all the above cases was less than the predetermined significance level, hence we reject the Null Hypothesis and conclude that the in each test the two variables are in fact dependent and have a positive correlation.

4. Data Modeling - Clustering

Hierarchical Clustering

```
TB_Data.Scaled=scale(TB_Data)
apply(TB_Data.Scaled,2,mean)
```

Scale the Data:

```
##                Age         Experience             Income       FamilyMember
##       6.054575e-18      -1.197687e-16       1.462101e-16       4.797478e-17
##              CCAvg          Education           Mortgage       PersonalLoan
##       4.641524e-17       3.867349e-18      -2.653715e-17       8.276322e-18
## SecuritiesAccount          CDAccount             Online         CreditCard
##      -3.979884e-17      -4.086721e-17       2.209626e-18       5.406587e-17
```

```
apply(TB_Data.Scaled,2,sd)
```

```
##                Age         Experience             Income       FamilyMember
##                  1                  1                  1                  1
##              CCAvg          Education           Mortgage       PersonalLoan
##                  1                  1                  1                  1
## SecuritiesAccount          CDAccount             Online         CreditCard
##                  1                  1                  1                  1
```

- The variables in the data are not in the scale hence the data needs to be scaled to prevent from any variable dominating its effect on the data and data manipulations.

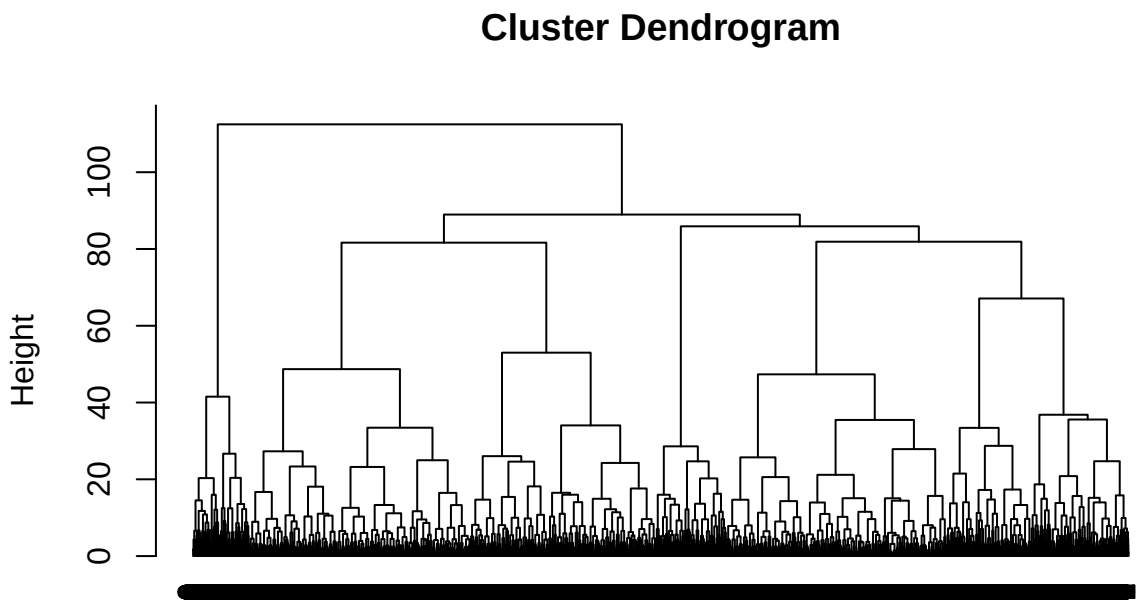
```
eucDistMatrix.scaled <- dist(x=TB_Data.Scaled[, -8], method = "euclidean")
```

Calculate Euclidean Distance between data points:

```
h_cluster <- hclust(eucDistMatrix.scaled, method = "ward.D2")
```

Create dissimilarity matric using hclust() and agglomeration method = Ward's Method

```
plot(h_cluster, labels = as.character(TB_Data$PersonalLoan), hang = -8)
```

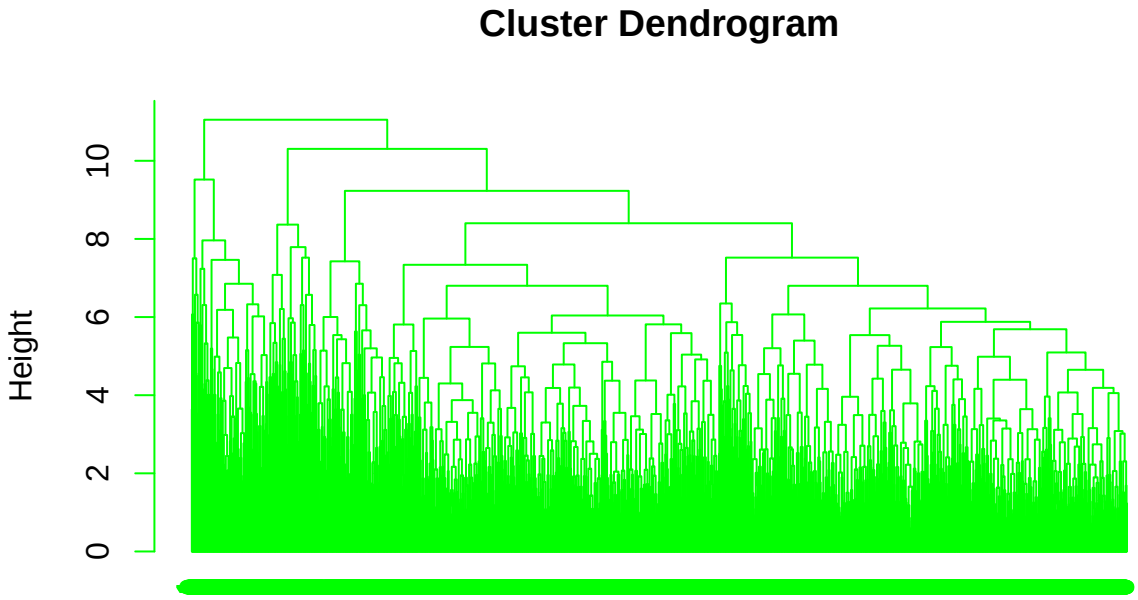


Plot the dendrogram

1. Dendrogram indicates 3/5 clusters of Personal Loan customers in our data

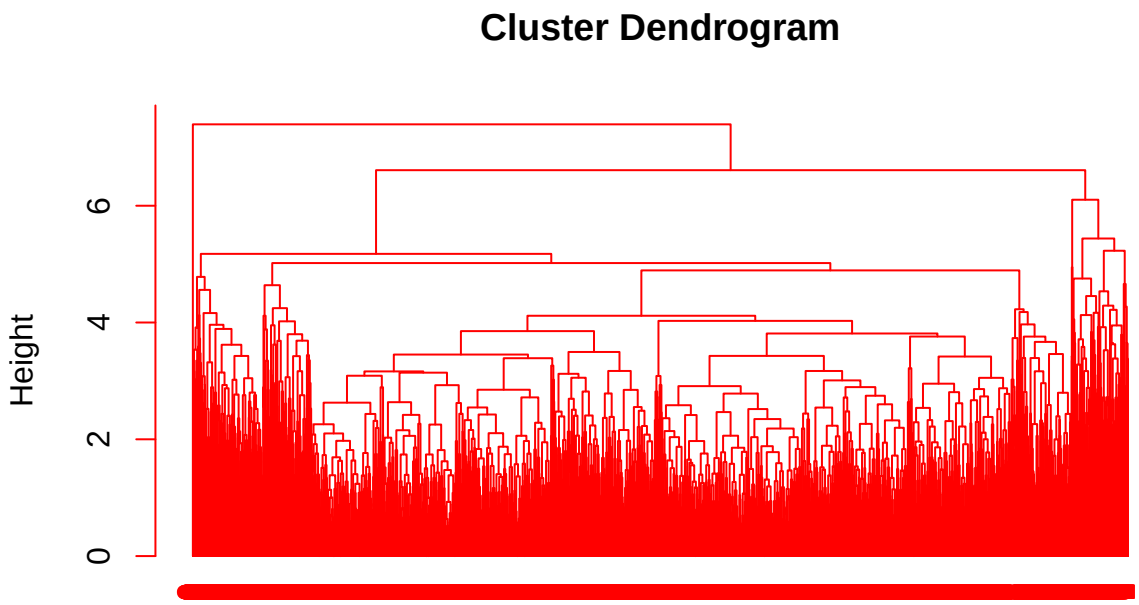
```
h_cluster_euc_comp <- hclust(eucDistMatrix.scaled, method = 'complete')
plot(h_cluster_euc_comp, labels = as.character(TB_Data$PersonalLoan),
     hang = -8, col = 'green')
```

Find optimal number of clusters by creating different dendrograms by varying agglomeration

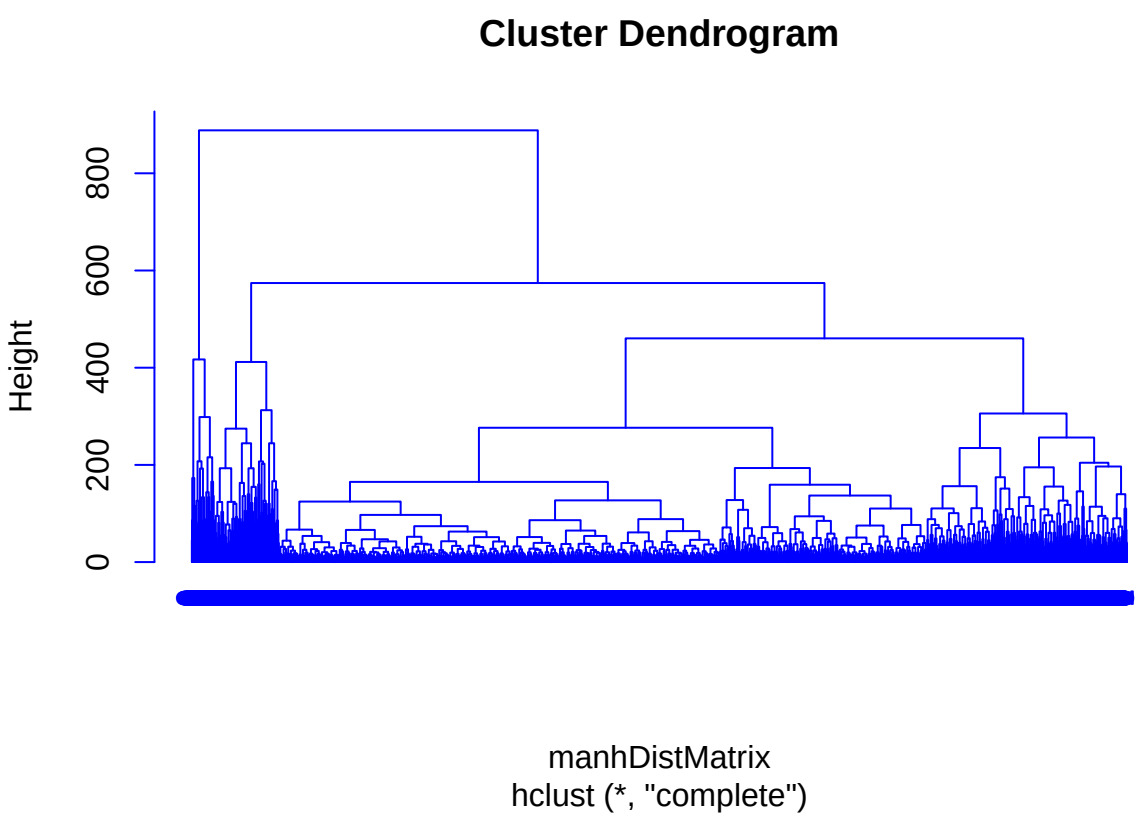


method

```
h_cluster_euc_avg <- hclust(eucDistMatrix.scaled, method = 'average')
plot(h_cluster_euc_avg, labels = as.character(TB_Data$PersonalLoan),
     hang = -8, col = 'red')
```




```
manhDistMatrix <- dist(x=TB_Data[, -8], method = "manhattan")
h_cluster_manh_comp <- hclust(manhDistMatrix, method = 'complete')
plot(h_cluster_manh_comp, labels = as.character(TB_Data$PersonalLoan),
     hang = -8, col = 'blue')
```

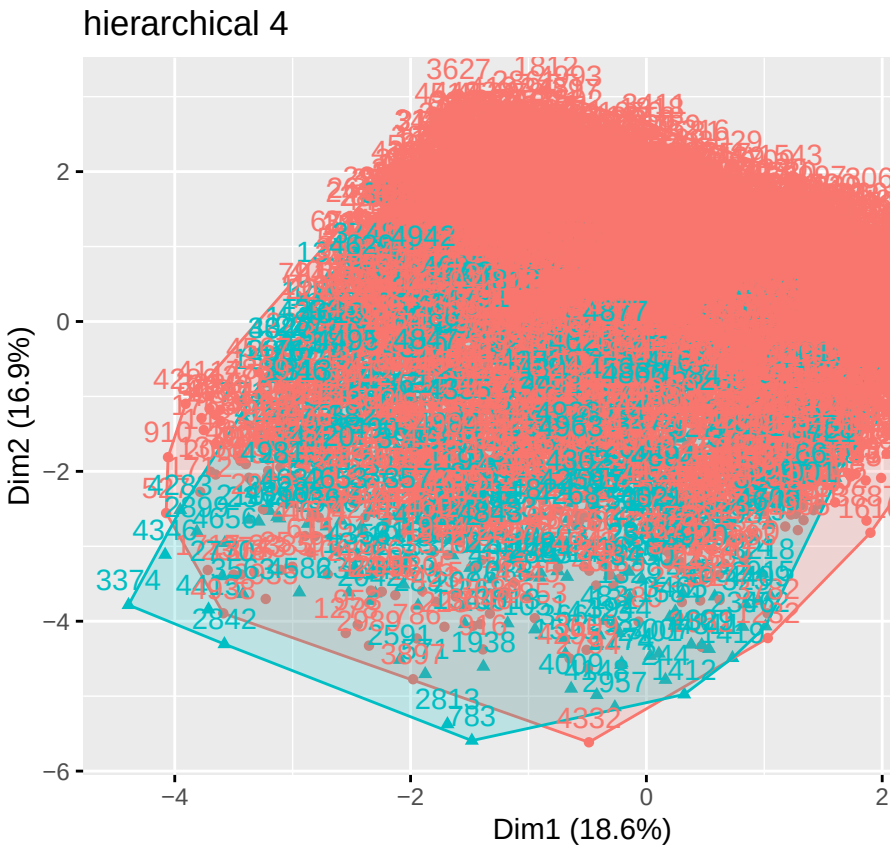


1. All the dendograms indicate a presence of 4 major clusters

```
cluster_name <- cutree(h_cluster, k = 4)
clg_data_hclusters <- cbind(TB_Data,cluster_name)
```

Add cluster membership to original dataset:

```
clg_data_hclusters <- clg_data_hclusters[,colSums(clg_data_hclusters != 0) != 0]
h_clust_viz_4 <- fviz_cluster(list(data = clg_data_hclusters[,-c(8,13)],
                                   cluster = clg_data_hclusters[,8])) + ggtitle("hierarchical 4")
h_clust_viz_4
```



Visualise the clusters in two dimensions
* There is quite an overlap in the 2 Dimensional cluster

```
View(clg_data_hclusters[order(clg_data_hclusters$cluster_name),])
table(clg_data_hclusters$cluster_name)
```

Number of members in each cluster:

```
##
##    1    2    3    4
##  375 2173 2150  302
```

```
aggr_mean <- aggregate(clg_data_hclusters[, -8], list(cluster_name), mean)
```

Observe the differences between identified clusters:

```
hcluster.profile <- data.frame(Cluster = aggr_mean[, 8],
                               PersonalLoan =
```

```
                                  as.vector(table(cluster_name)),
                                aggr_mean[, -8])

View(hcluster.profile)
hcluster.profile
```

Create cluster profiles

```
##    Cluster PersonalLoan Group.1      Age Experience    Income FamilyMember
## 1 46.80533          375       1 45.04000   19.74133  66.93600     2.509333
## 2 30.86516         2173       2 51.13300   25.84860  59.53474     2.387023
## 3 79.06512         2150       3 39.48279   14.29674  85.03023     2.376279
## 4 92.32450          302       4 45.70199   20.57285 104.58940     2.456954
##      CCAvg Education SecuritiesAccount CDAccount    Online CreditCard
## 1 1.838427  1.864000          1.000000         0 0.5040000 0.12000000
## 2 1.351615  1.906121          0.000000         0 0.5904280 0.45697193
## 3 2.415707  1.852093          0.000000         0 0.5716279 0.08930233
## 4 2.878974  1.927152          0.486755         1 0.9370861 0.79470199
##   cluster_name
## 1            1
## 2            2
## 3            3
## 4            4
```

Insights:

CLUSTER 1:

1. Has only 7.5% of customers from the data but 46.80 cluster mean
2. Age mean is 45 almost similar to Cluster 4 and almost similar experience levels
3. Income level is higher than Cluster 2 but much lower than 3 & 4
4. Customers are spending an average of 1838 per month on their credit card
5. Most of the customer having Securities Account belong to this cluster
6. Customers in this cluster have no COD Account with the bank
7. The usage on internet in least in this cluster
8. Only 12% of Customers in this cluster use credit card

CLUSTER 2:

1. Has 43.5% of customers from the data but just 30.86 of cluster mean.
2. Age mean is highest & so is the experience in proportion to age as seen in the data
3. This cluster has least Income level mean
4. Avg spending on Credit Card monthly too is lowest in this cluster
5. Customers in this cluster have no Securities Account with the bank
6. Customers in this cluster have no COD Account with the bank
7. Around 45% of Customers in this cluster use credit card
8. The usage on internet in also low but slightly more than customers of Cluster 3
9. Mean of customer having a Credit Card is 0.45

CLUSTER 3:

1. Has 43% of customers from the data and huge cluster mean of 79.
2. This cluster has the youngest lot of customers
3. This cluster has a substantially high Income level mean compared to Cluster 1 & 2
4. Avg spending on Credit Card monthly is higher in this cluster
5. Lowest Education mean in this cluster
6. Customers in this cluster have no Securities Account with the bank
7. Customers in this cluster have no COD Account with the bank
8. Most customers with credit cards are found in this cluster

CLUSTER 4:

1. Has only 6.5% of customers from the data but highest cluster mean
2. Age mean is 45 almost similar to Cluster 4 and almost similar experience levels
3. Highest income levels among all clusters. Mean is in 3 digit.
4. Highest spend on credit cards. The average monthly spend is 2878
5. Maximum Customer with highest education level found in this cluster
6. Customers have no Securities Account but the mean is half compared to Cluster 1
7. Only cluster where customer hace COD Accounts
8. Maximum number of customers have a credit card which seems proportional to the CC average monthly spend being higher in this cluster of customers.

K -means Clustering

```
set.seed <- 1000
nc=kmeans(TB_Data.Scaled[,8],centers = 2,nstart = 5)
#nc <- NbClust(TB_Data[, -8], min.nc = 2, max.nc = 6, method = "kmeans")
#table(nc$Best.n[1, ])
```

Determine the optimum number of clusters (find optimal k)

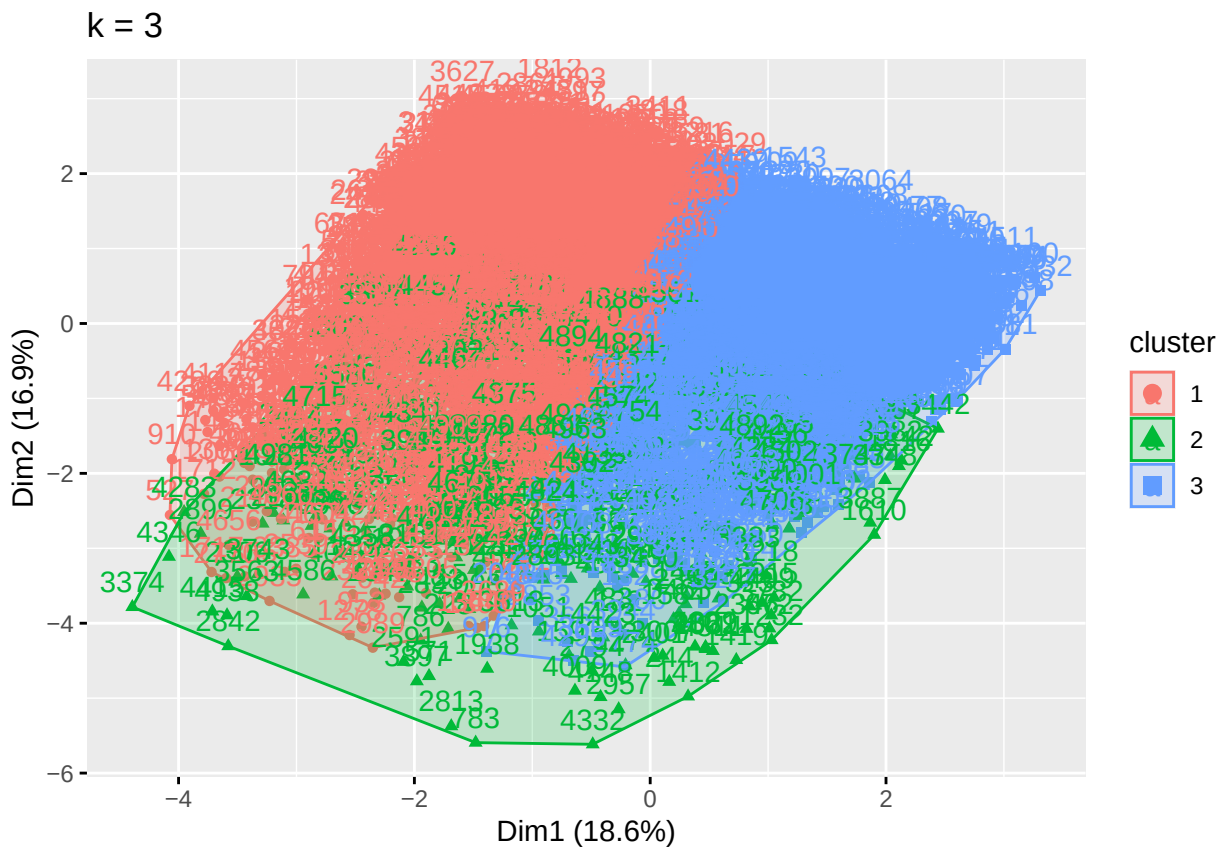
1. Among all the indices, according to the majority rule, the best number of cluster is 4

Create clusters for k=2, k=3 and k=4 for comparative analysis

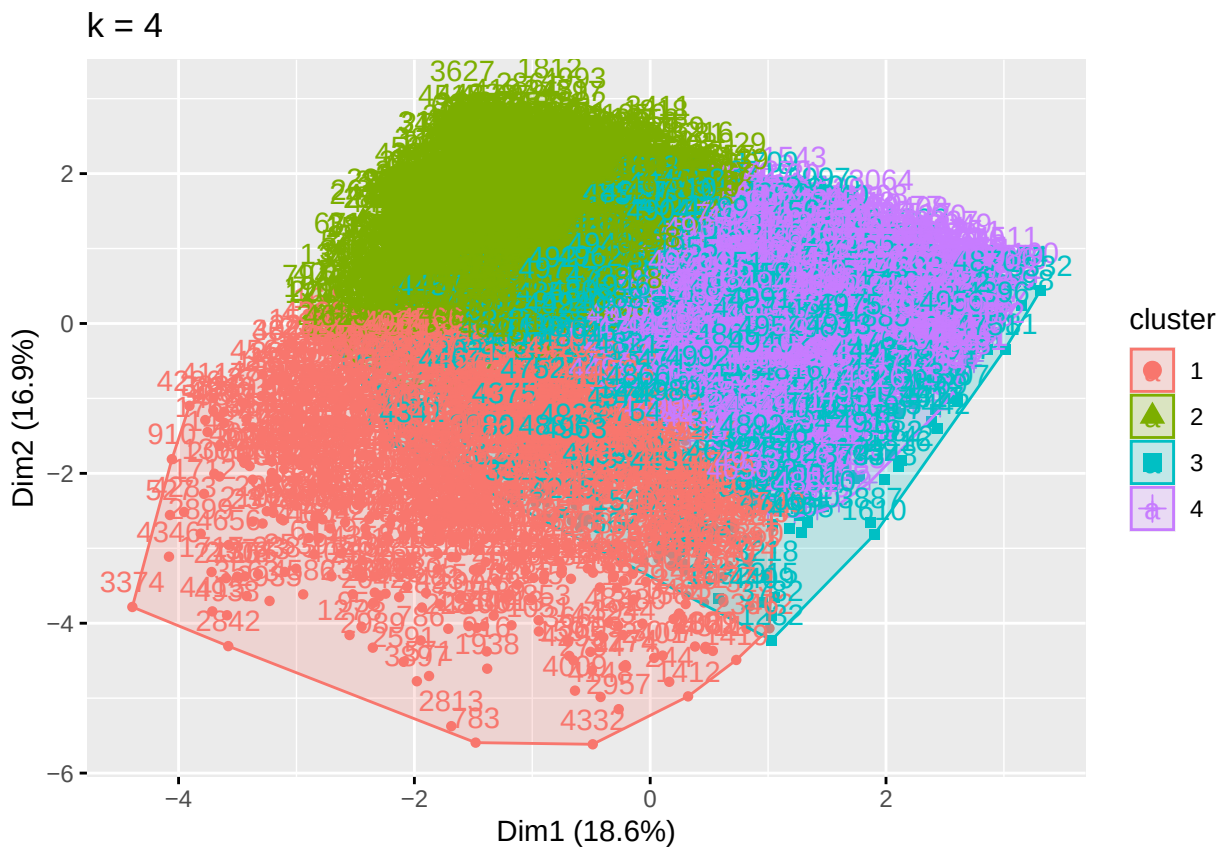
```
kmeans_cluster_3 <- kmeans(x=TB_Data.Scaled[, -8], centers = 3, nstart = 5)
kmeans_cluster_4 <- kmeans(x=TB_Data.Scaled[, -8], centers = 4, nstart = 5)
kmeans_cluster_6 <- kmeans(x=TB_Data.Scaled[, -8], centers = 6, nstart = 5)
```

Visualise clusters in 2 dimensions

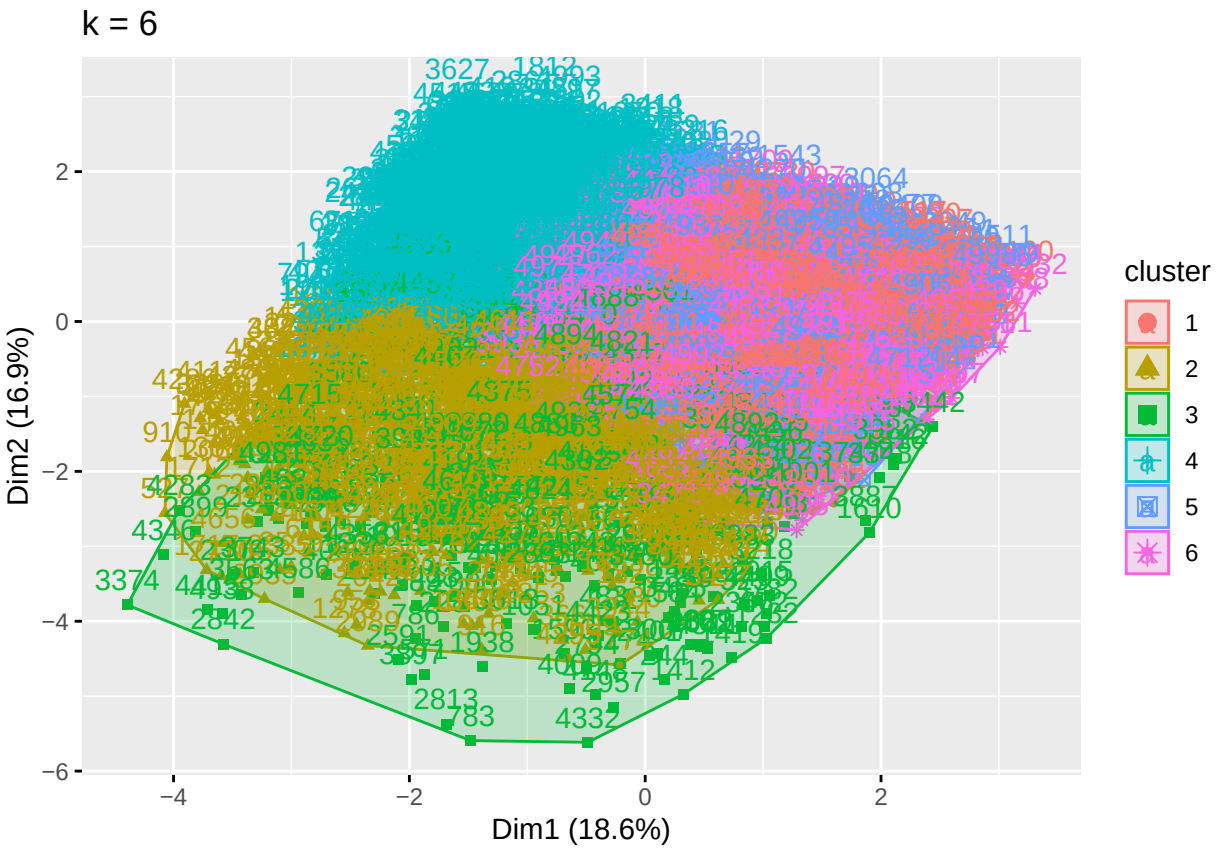
```
k_clust_viz_3 = fviz_cluster(list(data = TB_Data.Scaled[, -8],
                                  cluster = kmeans_cluster_3$cluster)) +
  ggtitle("k = 3")
k_clust_viz_3
```




```
k_clust_viz_4 = fviz_cluster(list(data = TB_Data.Scaled[, -8],
                                  cluster = kmeans_cluster_4$cluster)) +
  ggtitle("k = 4")
k_clust_viz_4
```

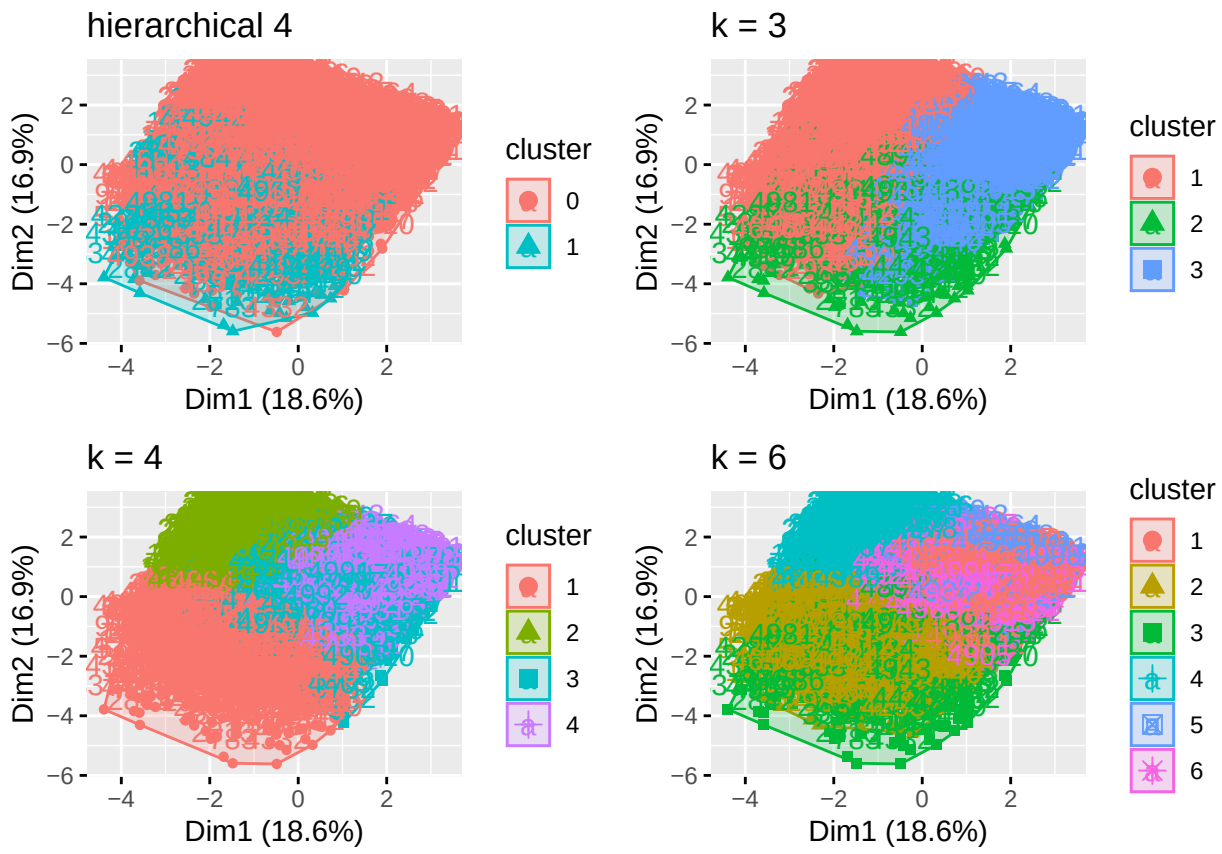



```
k_clust_viz_6 = fviz_cluster(list(data = TB_Data.Scaled[, -8],
                                  cluster = kmeans_cluster_6$cluster)) +
  ggtitle("k = 6")
k_clust_viz_6
```



Visualise all 4 clustering plots together (3 from K-Means and 1 from Hierarchical)

```
grid.arrange(h_clust_viz_4, k_clust_viz_3, k_clust_viz_4, k_clust_viz_6, nrow = 2)
```



Create cluster profiles:

```
aggr_mean_k3 <- aggregate(TB_Data.Scaled[, -8], list(kmeans_cluster_3$cluster), mean)
k3cluster.profile <- data.frame(Cluster = aggr_mean_k3[, 8],
                                PersonalLoan=
                                  as.vector(table(kmeans_cluster_3$cluster)),
                                aggr_mean_k3[, -8])
k3cluster.profile
```

K3

```
##       Cluster PersonalLoan Group.1         Age  Experience     Income
## 1  0.01933594         2350       1 -0.85510670 -0.84920491  0.1086681
## 2  0.35222067          302       2  0.03171783  0.04083097  0.6694049
## 3 -0.06465507         2348       3  0.85175552  0.84467657 -0.1948595
##   FamilyMember      CCAvg   Education SecuritiesAccount  CDAccount       Online
## 1   0.03710501  0.1277102 -0.08732364       -0.06169885 -0.2535149 -0.091492218
## 2   0.05339342  0.5384549  0.05495180        1.25030512  3.9437520  0.693627171
## 3  -0.04400409 -0.1970751  0.08033012       -0.09906296 -0.2535149  0.002355752
##    CreditCard
## 1 -0.09049748
## 2  1.09890428
## 3 -0.05076662
```

1. K3 shows that the Cluster 4 from the Hierarchical cluster as the Group 3 cluster, denoting that this particular group is identified as cluster with the highest mean

```
aggr_mean_k4 <- aggregate(TB_Data.Scaled[, -8], list(kmeans_cluster_4$cluster), mean)
k4cluster.profile <- data.frame(Cluster = aggr_mean_k4[, 8],
                                PersonalLoan =
                                  as.vector(table(kmeans_cluster_4$cluster)),
                                aggr_mean_k4[, -8])
k4cluster.profile
```

K4

```
##      Cluster PersonalLoan Group.1        Age Experience     Income FamilyMember
## 1  0.6207234          767       1 -0.1738523 -0.1531651  1.6188795  -0.41956587
## 2 -0.1008072         1708       2 -0.9824251 -0.9848060 -0.2806634   0.19985958
## 3 -0.1250009          928       3  0.5021674  0.4997933 -0.3102452   0.01570983
## 4 -0.1176677         1597       4  0.8424017  0.8363921 -0.2970569  -0.02137249
##        CCAvg   Education SecuritiesAccount  CDAccount      Online   CreditCard
## 1  1.7165384 -0.42647579        0.10626557  0.5563880  0.06180045  0.001436445
## 2 -0.3011148  0.03642533       -0.03889399 -0.2264833 -0.06484387 -0.232774722
## 3 -0.3139770  0.08266888        0.04269620  0.3887393  0.01794464  1.537652380
## 4 -0.3199188  0.11783078       -0.03424974 -0.2508867  0.02924217 -0.645249803
```

1. K4 shows a new group of 736 customers as Group 1 with the highest cluster mean followed by the previously identified group of 302 customers.
2. Together they constitute highest cluster mean i.e 1038 customers (20%)

```
aggr_mean_k6 <- aggregate(TB_Data.Scaled[, -8], list(kmeans_cluster_6$cluster), mean)
k6cluster.profile <- data.frame(Cluster = aggr_mean_k6[, 8],
                                PersonalLoan =
                                  as.vector(table(kmeans_cluster_6$cluster)),
                                aggr_mean_k6[, -8])
k6cluster.profile
```

K6

```
##       Cluster PersonalLoan Group.1         Age  Experience     Income
## 1 -0.12245889         1092       1  0.71765676  0.71354890 -0.3020246
## 2  0.55881805          603       2 -0.24044933 -0.21735872  1.6384683
## 3  0.35222067          302       3  0.03171783  0.04083097  0.6694049
## 4 -0.09111105         1382       4 -1.13145824 -1.13728846 -0.2779052
## 5 -0.09129622          818       5  0.59094230  0.58901644 -0.2841622
## 6 -0.13576218          803       6  0.53800182  0.53481713 -0.3036563
##   FamilyMember      CCAvg    Education SecuritiesAccount  CDAccount      Online
## 1  -0.01616168 -0.3152927  0.149321197      -0.053915661 -0.2535149  0.82186866
## 2  -0.52077385  1.8040085 -0.592848854      -0.140741727 -0.2535149 -0.09083197
## 3   0.05339342  0.5384549  0.054951800       1.250305123  3.9437520  0.69362717
```

```
## 4   0.25460513 -0.2958683  0.009010087      -0.031424290 -0.2535149  0.02245210
## 5  -0.02749996 -0.3138766  0.109666062      -0.009590968 -0.2535149 -1.21649607
## 6  -0.01720915 -0.2994892  0.094240211      -0.227366381 -0.2535149 -0.10973764
##   CreditCard
## 1 -0.6452498
## 2 -0.2230469
## 3  1.0989043
## 4 -0.1513568
## 5 -0.6452498
## 6  1.5494774
```

1. K6 shows along with previously identified group a new Group 5 with highest cluster mean among all clusters. It has 622 customers.

5.Build CART and Random Forest Model to predict Employee Attrition

Model Building - Approach

```
library(caTools)
set.seed(1000) # To ensure reproducibility

sample= sample.split(TB_Data$PersonalLoan,SplitRatio = 0.7)

train <- subset(TB_Data,sample == TRUE)

test <- subset(TB_Data,sample == FALSE)

nrow(train)
```

Split into train and test

```
## [1] 3500
```

```
nrow(test)
```

```
## [1] 1500
```

1.Train set has 3500 rows 2.Test set has 1500 rows

Check that the distribution of the dependent variable is similar in train and test sets

```
prop.table(table(TB_Data$`PersonalLoan`))
```

```
##
##     0     1
## 0.904 0.096
```

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

```
##
##     0     1
## 0.904 0.096
```

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

```
##
##     0     1
## 0.904 0.096
```

1.The distribution of the dependent variable in the original datat set, train data set & Test data set are similar.

```
r.ctrl = rpart.control(minsplit = 50, minbucket = 10, cp = 0, xval = 10)
cart_model1 <- rpart(formula = `PersonalLoan`~., data = train, method = "class", control = r.ctrl)
cart_model1
```

Build a CART model on the train dataset

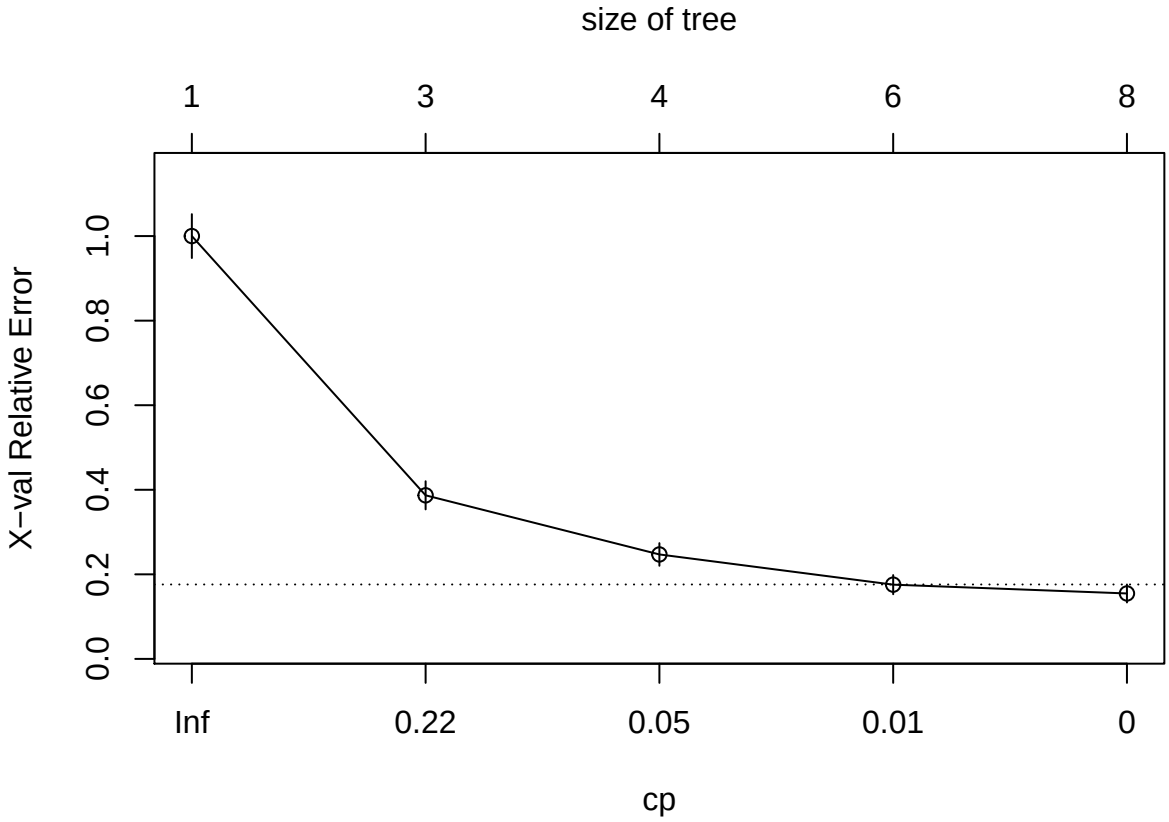
```
## n= 3500
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3500 336 0 (0.904000000 0.096000000)
##    2) Income< 114.5 2827  57 0 (0.979837283 0.020162717)
##      4) CCAvg< 2.95 2613  11 0 (0.995790279 0.004209721) *
##      5) CCAvg>=2.95 214  46 0 (0.785046729 0.214953271)
##       10) CDAccount< 0.5 196  31 0 (0.841836735 0.158163265)
##         20) Income< 98.5 138  14 0 (0.898550725 0.101449275) *
##         21) Income>=98.5 58  17 0 (0.706896552 0.293103448)
##           42) Education< 1.5 36   4 0 (0.888888889 0.111111111) *
##           43) Education>=1.5 22   9 1 (0.409090909 0.590909091) *
##       11) CDAccount>=0.5 18   3 1 (0.166666667 0.833333333) *
##    3) Income>=114.5 673 279 0 (0.585438336 0.414561664)
##      6) Education< 1.5 436  47 0 (0.892201835 0.107798165)
##       12) FamilyMember< 2.5 389   0 0 (1.000000000 0.000000000) *
##       13) FamilyMember>=2.5 47   0 1 (0.000000000 1.000000000) *
##      7) Education>=1.5 237   5 1 (0.021097046 0.978902954) *
```

```
printcp(cart_model1)
```

```
##
## Classification tree:
## rpart(formula = PersonalLoan ~ ., data = train, method = "class",
##     control = r.ctrl)
```

```
##
## Variables actually used in tree construction:
## [1] CCAvg        CDAccount    Education    FamilyMember Income
##
## Root node error: 336/3500 = 0.096
##
## n= 3500
##
##          CP nsplit rel error  xerror     xstd
## 1 0.3377976      0   1.00000 1.00000 0.051870
## 2 0.1398810      2   0.32440 0.38690 0.033298
## 3 0.0178571      3   0.18452 0.24702 0.026791
## 4 0.0059524      5   0.14881 0.17560 0.022667
## 5 0.0000000      7   0.13690 0.15476 0.021302
```

```
plotcp(cart_model1)
```



1. Cart Model 1 shows the use of “CCAvg”, “CDAccount”, “Education” & “Income” in preparing the model. 2. This model shows 7 nsplit

Build Cart Model 2 ### Model Tuning

```
cart_model2 = prune(cart_model1, cp= 0.017 ,"CP")
printcp(cart_model2)
```

```
##
## Classification tree:
## rpart(formula = PersonalLoan ~ ., data = train, method = "class",
##     control = r.ctrl)
##
## Variables actually used in tree construction:
## [1] CCAvg        CDAccount    Education    FamilyMember Income
##
## Root node error: 336/3500 = 0.096
##
## n= 3500
##
##         CP nsplit rel error  xerror     xstd
## 1 0.337798      0   1.00000 1.00000 0.051870
## 2 0.139881      2   0.32440 0.38690 0.033298
## 3 0.017857      3   0.18452 0.24702 0.026791
## 4 0.017000      5   0.14881 0.17560 0.022667
```

```
cart_model2
```

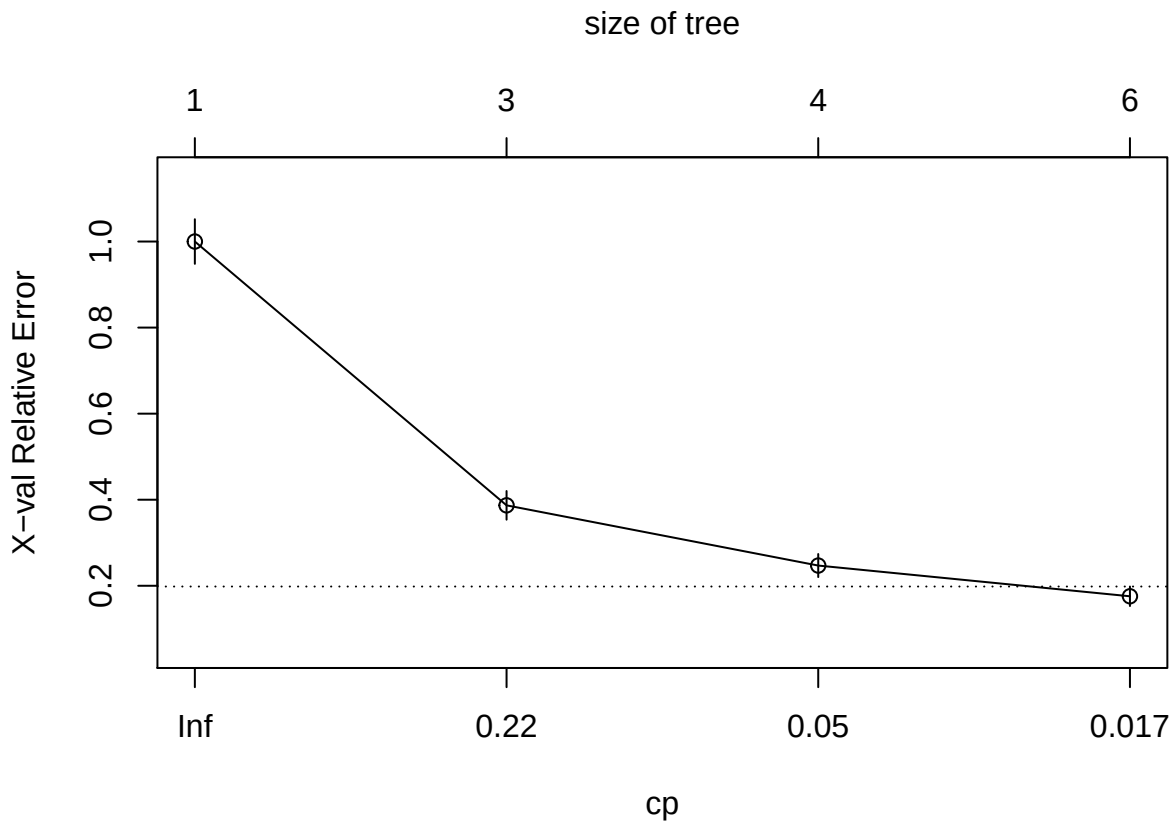
```
## n= 3500
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3500 336 0 (0.904000000 0.096000000)
##    2) Income< 114.5 2827  57 0 (0.979837283 0.020162717)
##      4) CCAvg< 2.95 2613  11 0 (0.995790279 0.004209721) *
##      5) CCAvg>=2.95 214  46 0 (0.785046729 0.214953271)
##       10) CDAccount< 0.5 196  31 0 (0.841836735 0.158163265) *
##       11) CDAccount>=0.5 18   3 1 (0.166666667 0.833333333) *
##    3) Income>=114.5 673 279 0 (0.585438336 0.414561664)
##      6) Education< 1.5 436  47 0 (0.892201835 0.107798165)
##       12) FamilyMember< 2.5 389   0 0 (1.000000000 0.000000000) *
##       13) FamilyMember>=2.5 47   0 1 (0.000000000 1.000000000) *
##      7) Education>=1.5 237   5 1 (0.021097046 0.978902954) *
```

```
printcp(cart_model2)
```

```
##
## Classification tree:
## rpart(formula = PersonalLoan ~ ., data = train, method = "class",
##     control = r.ctrl)
##
## Variables actually used in tree construction:
## [1] CCAvg        CDAccount    Education    FamilyMember Income
##
## Root node error: 336/3500 = 0.096
##
## n= 3500
##
##         CP nsplit rel error  xerror     xstd
## 1 0.337798      0   1.00000 1.00000 0.051870
```

```
## 2 0.139881      2   0.32440 0.38690 0.033298
## 3 0.017857      3   0.18452 0.24702 0.026791
## 4 0.017000      5   0.14881 0.17560 0.022667
```

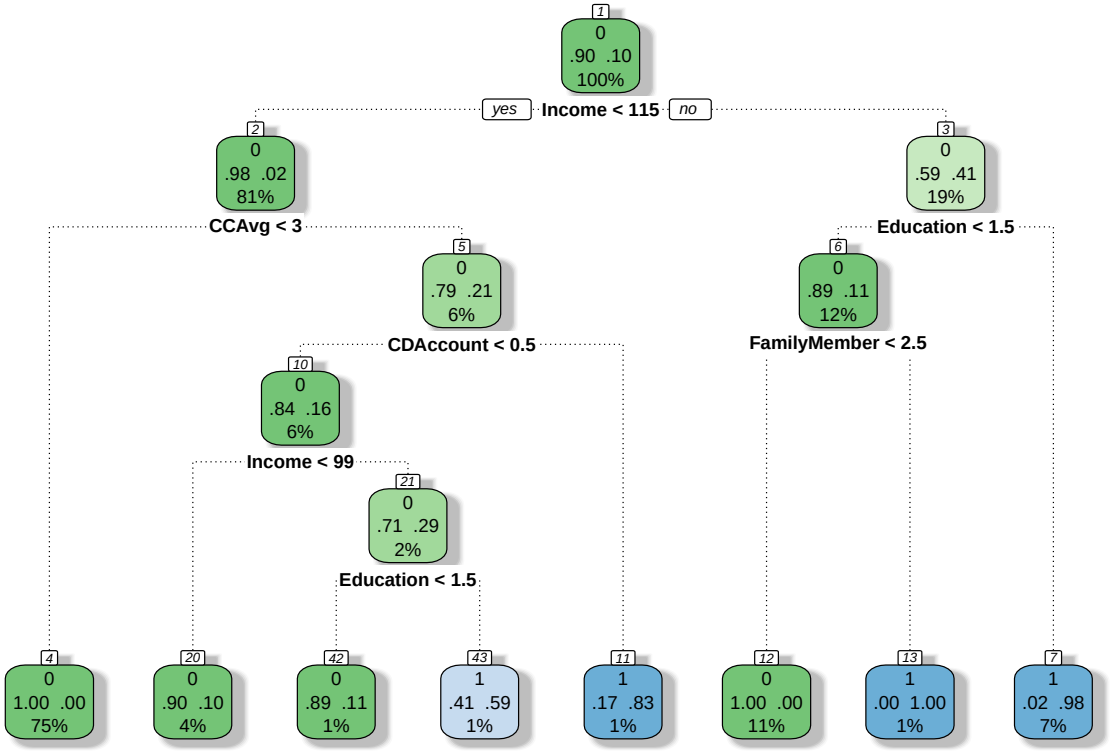
```
plotcp(cart_model2)
```



1. Same variables used as in cart Model 1 to develop Cart Model 2. 2. This model shows 5 nsplits

Visualise the decision tree

```
library(rpart.plot)
library(RColorBrewer)
library(rattle)
fancyRpartPlot(cart_model1)
```



1.The left most node contains 81% of the training data set 2.The right most node contains 19% of the training data set 3.The root node is split on Income $<$ 115000 4. The prominent cluster of customers who could be targeted begins with the high income category who's average spend of Credit Card is high and have CD Accounts with the bank. 5. The group who dont have high income but high level of education with family members more than 2 is another interesting cluster

Check the variable importance

```
cart_model1$variable.importance
```

```
##    Education       Income FamilyMember        CCAvg    CDAccount     Mortgage
##   239.305597   173.255025   148.918117    87.219557    49.442449    21.426347
##   Experience          Age
##     3.550438     1.401711
```

1. Education followed by Income & Family Members are the most important and influential variables in the model.

Model Validation

```
# Predicting on the train dataset
train_predict.class_CART <- predict(cart_model1, train, type="class") # Predicted Classes
```

```
train_predict.score_CART <- predict(cart_model1, train) # Predicted Probabilities

# Create confusion matrix for train data predictions
tab.train_CART = table(train$`PersonalLoan`, train_predict.class_CART)
tab.train_CART
```

```
##    train_predict.class_CART
##        0    1
##   0 3147   17
##   1   29  307
```

```
# Accuracy on train data
accuracy.train_CART = sum(diag(tab.train_CART)) / sum(tab.train_CART)
accuracy.train_CART
```

```
## [1] 0.9868571
```

- CART Model has 98.68% accuracy on training data set shows no improvement on the baseline accuracy.

Model Evaluation

```
# Predicting on the test dataset
test_predict.class_CART <- predict(cart_model1, test, type="class") # Predicted Classes
test_predict.score_CART <- predict(cart_model1, test) # Predicted Probabilities

# Create confusion matrix for test data predictions
tab.test_CART = table(test$`PersonalLoan`, test_predict.class_CART)

# Accuracy on test data
accuracy.test_CART = sum(diag(tab.test_CART)) / sum(tab.test_CART)
accuracy.test_CART
```

```
## [1] 0.9853333
```

- 98.53% accuracy on the test data set is quite close to the training data set accuracy
- Although the CART model does not overfit the data but it is not much of an improvement over the baseline model

Random Forest Model

```
set.seed(1000)
library(randomForest)
rf_model1= randomForest(PersonalLoan~.,data = train,ntree=501,mtry=5,nodesize=10,importance=TRUE)
```

Build the first RF model

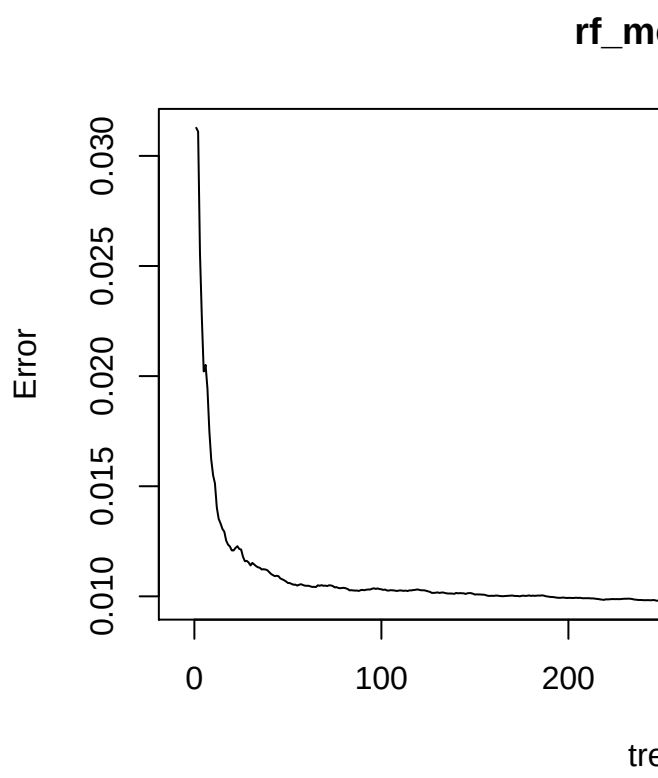
```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
print(rf_model1)
```

```
##
## Call:
##  randomForest(formula = PersonalLoan ~ ., data = train, ntree = 501,      mtry = 5, nodesize = 10, im
##                Type of random forest: regression
##                      Number of trees: 501
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.01002023
##                     % Var explained: 88.45
```

1. The response is showing 5 or fewer unique values.
2. Further regression is not recommended for the Random Forest Model

```
plot(rf_model1)
```



Plot the model to determine the optimum number of trees

```
print(rf_model1$importance)
```

```
##                         %IncMSE IncNodePurity
## Age                4.488531e-03     6.8521508
## Experience         3.662867e-03     5.8783376
## Income             1.578237e-01    94.5700563
## FamilyMember       5.221494e-02    40.7976720
## CCAvg              3.079852e-02    37.0319702
## Education          7.725599e-02    83.3622907
## Mortgage           1.104287e-03     4.5823099
## SecuritiesAccount 5.155035e-05     0.4408576
## CDAccount          4.031029e-03    14.3253399
## Online             3.849061e-04     1.0206409
## CreditCard         6.222775e-04     1.0758297
```

1. The plot reveals that anything more than 50 trees is not of much value
2. The Importance matrix shows high "Mean Decrease Accuracy" for Education,Family Member,Age & CD Account

Tuning the Random Forest Model The response from Random Forest Model 1 shows or fewer unique values, hence no further regression needed.

```
sapply(train, class)
```

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"         "numeric"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

```
sapply(test, class)
```

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"         "numeric"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

```
test$FamilyMember=as.factor(test$FamilyMember)
```

Model Validation

```
# Predicting on the train dataset
train_predict.class_RF <- predict(rf_model1, train, type="class") # Predicted Classes
train_predict.score_RF <- predict(rf_model1, train) # Predicted Probabilities
```

```r
# Create confusion matrix for train data predictions
tab.train_RF = table(train$PersonalLoan, train_predict.class_RF)
```

```r
# Accuracy on train data
accuracy.train_RF = sum(diag(tab.train_RF)) / sum(tab.train_RF)
accuracy.train_RF
```

```
## [1] 0.0002857143
```

- The accuracy model shows this model is not correct and should be discarded

let's see how the model performs on the test data

Model Evaluation

```r
test$FamilyMember= as.numeric(test$FamilyMember)
# Predicting on the test dataset
test_predict.class_RF <- predict(rf_model1, test, type="class") # Predicted Classes
test_predict.score_RF <- predict(rf_model1, test) # Predicted Probabilities
```

```r
# Create confusion matrix for test data predictions
tab.test_RF = table(test$PersonalLoan, test_predict.class_RF)
```

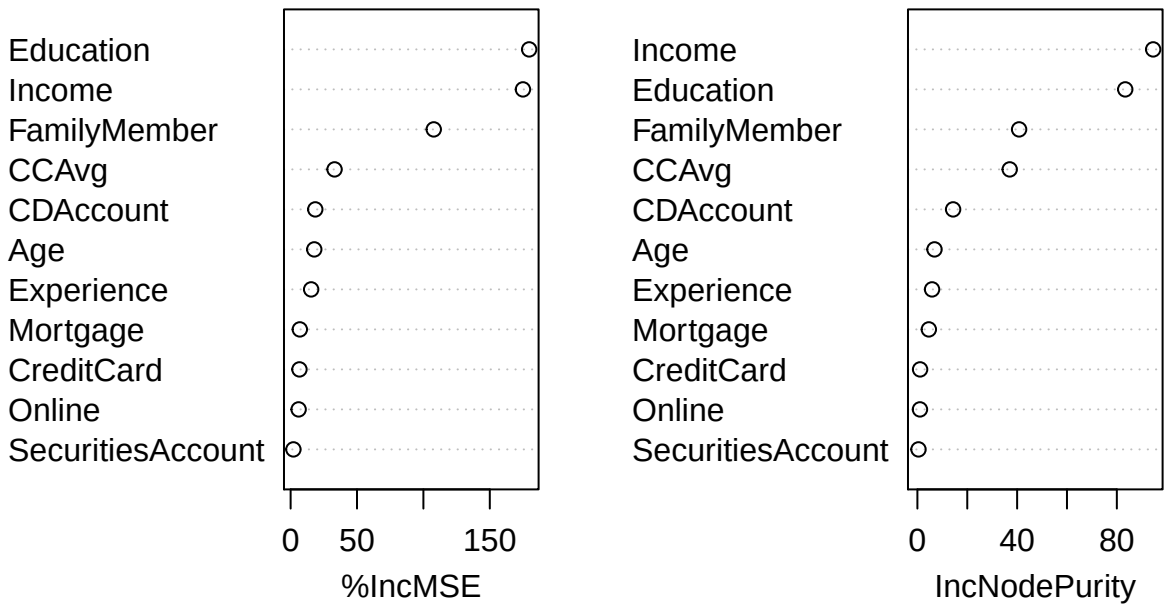
```r
# Accuracy on test data
accuracy.test_RF = sum(diag(tab.test_RF)) / sum(tab.test_RF)
accuracy.test_RF
```

```
## [1] 0.0006666667
```

1. The accuracy of the test data shows this model is not correct and should be discarded

Variable Importance Final Model

```r
varImpPlot(rf_model1, sort = TRUE)
```



1. Its worth noting that most variables indicated by the CART Model and final Random Forest Model are almost same with Income, Education, Family Member & CC Avg being the top influences in both the models. 2. The Random forest generalises over the data in a better way. This randomised feature makes Random Forest more accurate than the decision tree.

Insights:

1. Income, Education, Family & Average monthly spend on Credit Card will be the most important in determing which customers will go for the Loan.
2. The customer with high Income Level fall in the age bracket of 39 to 45.
3. Age is not proportional to Income with all customers as we have many high educated customers in lower income bracket.
4. Also an interesting observation is the customers with higher income possess credit card and spend an average of 2800 per month, and though the number of people with mid level income (comparing in the dataset) do not have credit cards. But the small amount of customers who do, spend an average of 2400 per month. this suggest that the mid-level i.e cluster 3 who form 43% of the base have a propensity to spend more beyond their income and could be the right target customer with high probability to go for a loan.

- The customer on higher income bracker tend to use internet banking more than others.

```
Model_Name = c("Baseline", "CART", "Random Forest")
Train_Accuracy_perc = c(99, accuracy.train_CART*100, accuracy.train_RF*100)
```

```
Test_Accuracy_perc = c(98, accuracy.test_CART*100, accuracy.test_RF*100)
output = data.frame(Model_Name,Train_Accuracy_perc,Test_Accuracy_perc)
output
```

Comparing Models

```
##      Model_Name Train_Accuracy_perc Test_Accuracy_perc
## 1      Baseline         99.00000000        98.00000000
## 2          CART         98.68571429        98.53333333
## 3 Random Forest          0.02857143         0.06666667
```

1. The Random Forest Model is not right for this prediction and will be ignored.
2. The test Accuracy of the CART model shows a slight improvement on the baseline model and will help us predict the right customers who have high probability of purchasing the loan.

Confusion Matrix

We will compare the 2 models that we created earlier - Cart & Random Forest

```
train$PersonalLoan=as.factor(train$PersonalLoan)
levels(test$PersonalLoan)=levels(train$PersonalLoan)
sapply(train, class)
```

Change to factor variable

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"          "factor"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

```
sapply(test, class)
```

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"         "numeric"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

Create Prediction Matrix for both models

```
# Predict on test data using cart_model1
train$PersonalLoan=as.factor(train$PersonalLoan)
levels(test$PersonalLoan)=levels(train$PersonalLoan)
sapply(train, class)
```

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"          "factor"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

```
sapply(test, class)
```

```
##               Age        Experience            Income      FamilyMember
##         "numeric"         "numeric"         "numeric"         "numeric"
##             CCAvg         Education          Mortgage      PersonalLoan
##         "numeric"         "numeric"         "numeric"         "numeric"
## SecuritiesAccount         CDAccount            Online        CreditCard
##         "numeric"         "numeric"         "numeric"         "numeric"
```

```
cart_model1_predict_class = predict(cart_model1, test, type = 'class')
cart_model1_predict_score = predict(cart_model1, test, type = 'prob')

# Predict on test data using rf_model1
levels(train)=levels(test)
str(train)
```

```
## tibble [3,500 x 12] (S3: tbl_df/tbl/data.frame)
##  $ Age               : num [1:3500] 25 39 35 35 37 50 35 65 29 59 ...
##  $ Experience        : num [1:3500] 1 15 9 8 13 24 10 39 5 32 ...
##  $ Income            : num [1:3500] 49 11 100 45 29 22 81 105 45 40 ...
##  $ FamilyMember      : num [1:3500] 4 1 1 4 4 1 3 4 3 4 ...
##  $ CCAvg             : num [1:3500] 1.6 1 2.7 1 0.4 0.3 0.6 2.4 0.1 2.5 ...
##  $ Education         : num [1:3500] 1 1 2 2 2 3 2 3 2 2 ...
##  $ Mortgage          : num [1:3500] 0 0 0 0 155 0 104 0 0 0 ...
##  $ PersonalLoan      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SecuritiesAccount: num [1:3500] 1 0 0 0 0 0 0 0 0 0 ...
##  $ CDAccount         : num [1:3500] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num [1:3500] 0 0 0 0 1 0 1 0 1 1 ...
##  $ CreditCard        : num [1:3500] 0 0 0 1 0 1 0 0 0 0 ...
```

```
str(test)
```

```
## tibble [1,500 x 12] (S3: tbl_df/tbl/data.frame)
##  $ Age               : num [1:1500] 45 53 34 48 67 60 36 40 51 57 ...
##  $ Experience        : num [1:1500] 19 27 9 23 41 30 11 16 25 32 ...
##  $ Income            : num [1:1500] 34 72 180 114 112 22 152 29 71 84 ...
##  $ FamilyMember      : num [1:1500] 3 2 1 2 1 1 2 1 1 3 ...
##  $ CCAvg             : num [1:1500] 1.5 1.5 8.9 3.8 2 1.5 3.9 2 1.4 1.6 ...
##  $ Education         : num [1:1500] 1 2 3 3 1 3 1 2 3 3 ...
##  $ Mortgage          : num [1:1500] 0 0 0 0 0 0 159 0 198 0 ...
##  $ PersonalLoan      : num [1:1500] 0 0 1 0 0 0 0 0 0 0 ...
##   ..- attr(*, "levels")= chr [1:2] "0" "1"
##  $ SecuritiesAccount: num [1:1500] 1 0 0 1 1 0 0 0 0 1 ...
##  $ CDAccount         : num [1:1500] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num [1:1500] 0 1 0 0 0 1 0 1 0 0 ...
##  $ CreditCard        : num [1:1500] 0 0 0 0 0 1 1 0 0 0 ...
```

```
rf_model1_predict_class = predict(rf_model1, test, type = 'class')
rf_model1_predict_score = predict(rf_model1, test,)
```

Create Confusion Matrix for both the models

```
library(caret)
test$PersonalLoan=as.numeric(test$PersonalLoan)
cart_model1_predict_class=as.numeric(cart_model1_predict_class)
rf_model1_predict_class=as.numeric(rf_model1_predict_class)

conf_mat_cart_model1 = table(test$PersonalLoan, cart_model1_predict_class)

conf_mat_rf_model1 = table(test$PersonalLoan, rf_model1_predict_class)


confusionMatrix(test$PersonalLoan, cart_model1_predict_class)
```

```
## Error: `data` and `reference` should be factors with the same levels.
```

```
confusionMatrix(test$PersonalLoan, rf_model1_predict_class)
```

```
## Error: `data` and `reference` should be factors with the same levels.
```

Accuracy of models on test data

```
# Accuracy of models on test data
accuracy_cart_model1 = sum(diag(conf_mat_cart_model1)) / sum(conf_mat_cart_model1)

accuracy_rf_model1 = sum(diag(conf_mat_rf_model1)) / sum(conf_mat_rf_model1)
```

Sensitivity of models on test data

```
# Sensitivity of models on test data
sensitivity_cart_model1 = conf_mat_cart_model1[2,2] / sum(conf_mat_cart_model1[2,])

sensitivity_rf_model1 = conf_mat_rf_model1[2,2] / sum(conf_mat_rf_model1[2,])
```

Specificity of models on test data

```
# Specificity of models on test data
specificity_cart_model1 = conf_mat_cart_model1[1,1] / sum(conf_mat_cart_model1[1,])

specificity_rf_model1 = conf_mat_rf_model1[1,1] / sum(conf_mat_rf_model1[1,])
```

Precision of models on test data

```
# Precision of models on test data
precision_cart_model1 = conf_mat_cart_model1[2,2] / sum(conf_mat_cart_model1[2,])

precision_rf_model1 = conf_mat_rf_model1[2,2] / sum(conf_mat_rf_model1[2,])
```

KS

```
# Using library ROCR functions prediction and performance
library(ROCR)

pred_cart_model1 = prediction(cart_model1_predict_score[,2], test$PersonalLoan)
perf_cart_model1= performance(pred_cart_model1,"tpr","fpr")
ks_cart_model1 = max(attr(perf_cart_model1,'y.values')[[1]] - attr(perf_cart_model1,'x.values')[[1]])


test$PersonalLoan=as.numeric(test$PersonalLoan)
rf_model1_predict_score=as.numeric(rf_model1_predict_score)
pred_rf_model1 = prediction(rf_model1_predict_score, test$PersonalLoan)
perf_rf_model1 = performance(pred_rf_model1,"tpr","fpr")
ks_rf_model1 = max(attr(perf_rf_model1,'y.values')[[1]] - attr(perf_rf_model1,'x.values')[[1]])
```

AUC

```
# Using library ROCR
auc_cart_model1 = performance(pred_cart_model1, measure = "auc")
auc_cart_model1 = auc_cart_model1@y.values[[1]]


auc_rf_model1 = performance(pred_rf_model1, measure = "auc")
auc_rf_model1 = auc_rf_model1@y.values[[1]]
```

Gini

```
# Using library ineq
library(ineq)
gini_cart_model1 = ineq(cart_model1_predict_score[, 2],"gini")

gini_rf_model1 = ineq(rf_model1_predict_score,"gini")
```

Concordance - Discordance

```
# Concordance - Discordance

library(InformationValue)
concordance_cart_model1 = Concordance(actuals = ifelse(test$PersonalLoan == 'Yes', 1,0), predictedScores

concordance_rf_model1 = Concordance(actuals = ifelse(test$PersonalLoan == 'Yes', 1,0), predictedScores =
```

Comparing models

```
cart_model1_metrics = c(accuracy_cart_model1, sensitivity_cart_model1, specificity_cart_model1, precisi


rf_model1_metrics = c(accuracy_rf_model1, sensitivity_rf_model1, specificity_rf_model1, precision_rf_mo


cart_model1_metrics=as.matrix(cart_model1_metrics)
cart_model1_metrics.df=as.data.frame(cart_model1_metrics)
class(cart_model1_metrics.df)
```

```
## [1] "data.frame"
```

```
rf_model1_metrics=as.matrix(rf_model1_metrics)
rf_model1_metrics.df=as.data.frame(rf_model1_metrics)
class(rf_model1_metrics.df)
```

```
## [1] "data.frame"
```

```
comparison_table = data.frame(cart_model1_metrics.df, rf_model1_metrics.df)

rownames(comparison_table) = c("Accuracy", "Sensitivity", "Specificity", "Precision", "KS", "Auc", "Gini
colnames(comparison_table)=c("Cart Model1","RF Model1")
comparison_table
```

```
##             Cart Model1    RF Model1
## Accuracy      0.9853333 0.0006666667
## Sensitivity   0.8888889 0.0000000000
## Specificity   0.9955752 0.0007374631
## Precision     0.8888889 0.0000000000
## KS            0.9242871 0.9757866273
## Auc           0.9842726 0.9982946165
## Gini          0.8748481 0.8957320927
## Concordance         NaN          NaN
```

1.The summary shows the following results: + Accuracy for Cart Model 1 is at 98.5 + Sensitivity of CartModel1 is at 88.88 while RFModel1 is 0 + Specificity of Cart Model1 99 + Precision of Cart Model1 is at 88.88 while RFModel1 is 0 + KS, AUC & Gini are higher for RFModel1

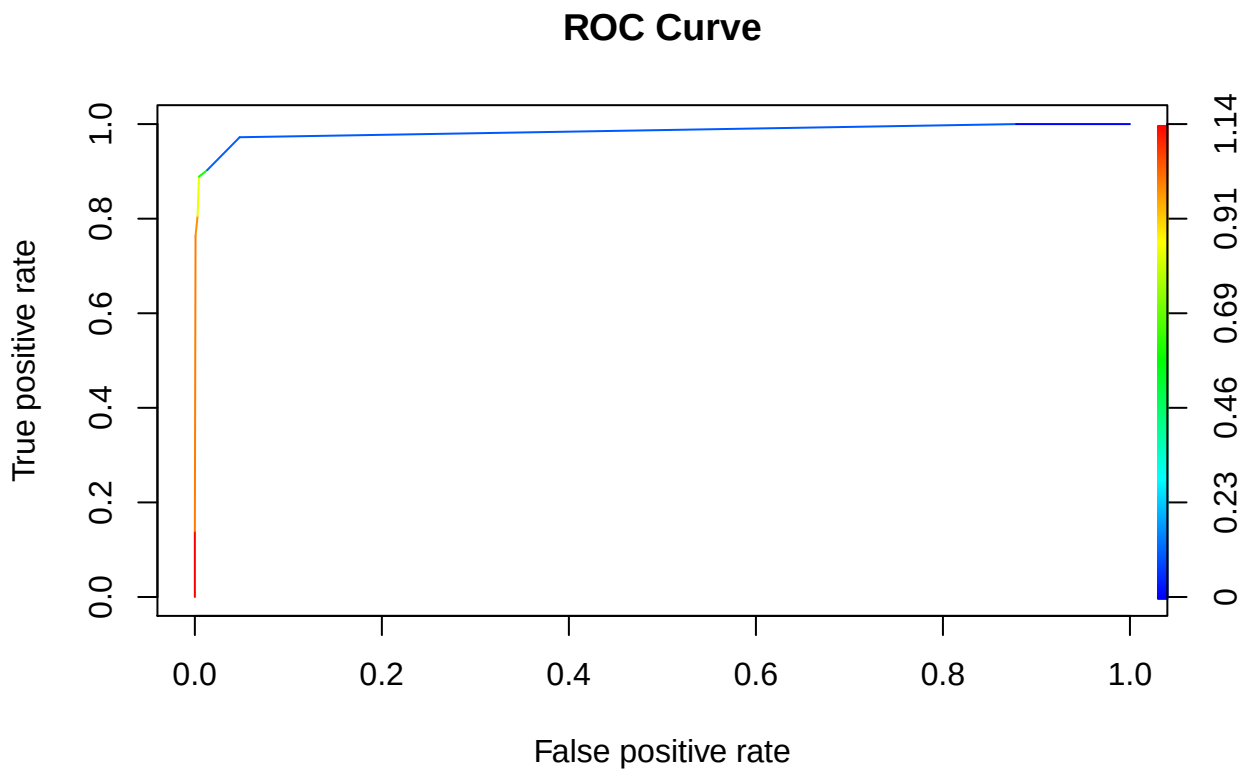
FROM THE ABOVE OBSERVATION WE CAN CONCLUDE THAT CART MODEL 1 IS THE RIGHT MODEL FOR USE

ROC Curves

```
test$PersonalLoan=as.numeric(test$PersonalLoan)
cart_model1_predict_score=as.numeric(cart_model1_predict_score)
pred_cart_model1 = prediction(cart_model1_predict_score, test$PersonalLoan)
```

```
## Error in prediction(cart_model1_predict_score, test$PersonalLoan): Number of predictions in each run
```

```
perf_cart_model1 = performance(pred_cart_model1,'tpr','fpr')
plot(perf_cart_model1, main = "ROC Curve" ,colorize = TRUE)
```




```
str(perf_cart_model1)
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "False positive rate"
##   ..@ y.name      : chr "True positive rate"
##   ..@ alpha.name  : chr "Cutoff"
##   ..@ x.values    :List of 1
##   .. ..$ : num [1:9] 0 0 0.000737 0.00295 0.004425 ...
##   ..@ y.values    :List of 1
##   .. ..$ : num [1:9] 0 0.139 0.764 0.806 0.889 ...
##   ..@ alpha.values:List of 1
##   .. ..$ : num [1:9] Inf 1 0.979 0.833 0.591 ...
```

1. ROC Curve suggests after 0.1 the outcome is linear and stable without any fluctuations.
2. There is further need to measure the model performance (eg. Gain & Lift) as we dont have any modesl to analyse and choose.
3. Also, the Random Model has failed in providing any kind of meaningful Acccuracy, Sensitivity, Specificity or Precision that would be of any consequence to consider it as a model for classifying the right customers for loan.

Conclusion:

1. The Cart Model is the model best model which can classify the right customers who have higher probability of purchasing the loan.
2. The model identifies 2 clusters with high cluster mean and they constitute almost 50% of the total customers approached earlier for loan.
3. Both these cluster have some common features

- Higher Income bracket (85,000 to 104,000)
- Higher Spending on the Credit Card (2400 to 2900)
- They belong to comparatively lower age bracket compared to other cluster (39- to 45)
- Family mean of 2
- One of the cluster has maximum CD Account & almost 50% Securities Account holders
- Experience between 14 to 20 years
- 79% of one cluster (of 302 customers) have credit card, while only 8% of the larger cluster (2150) possess credit card but have a high propensity to spend on those credit card which shows they would be ideal to sell loans to.

4. Compared to the 9% success rate, we can have a much better success in selling loans to this customer base without spending much on marketing. We will be able to identify the right cluster who are likely to buy the loan by filtering and analyzing the data by their similarities, commonalities, segmentation and user behaviors which this dataset shows. The CART model will help us classifying these customers.