Thera Bank-Loan Purchase modeling

***Table of contents***

1. Project Objective
2. Exploratory data analysis
   1. Environment Set up and Data Import
   2. Variable Identification
   3. Uni-Variate Analysis
   4. Bi-Variate Analysis
3. Unsupervised Learning Technique
   1. Hierarchical Clustering
   2. K-means Clustering
4. Supervised Learning Technique
   1. CART Model with model performance
   2. Random Forest with model performance
5. Conclusions
6. Appendix A-Source code

**1 Project Objective**

The primary objective of this report is to help  the Thera bank, the management wants to explore ways of converting its liability customers to personal loan customers .For the better understanding of dataset,imported advanced graphical libraries and plotted some important attributes of loan prediction data.  The dataset has data on 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).

**2 Exploratory data analysis**

Generally, Exploratory data analysis are carried out to discover the patterns in the given dataset,missing values ,central tendency of the given dataset.

**2.1 Environment set up and data Import**

**The working directory should be one , where the code and dataset are placed.**

The following R packages used for the analysis of loan prediction dataset

* Ggplot2 package-To infer about the attributes of the loan prediction data using graphical plots
* Cluster-To group the similar data with respect to attributes
* NbClust-To get optimum number of clusters
* caTools-To split the given dataset into training and testing
* rpart-Building the classification and Regression model
* rpart.plot-The package to plot the CART model
* Data.table-Its an improved version of data frame
* ROCR- To compute model performance measures such as KS,Area under curve
* Ineq-To compute the gini index of data
* InformationValue-Package to calculate the concordance ratio
* randomForest-To build the random forest

**2.2 Variable Identification:**

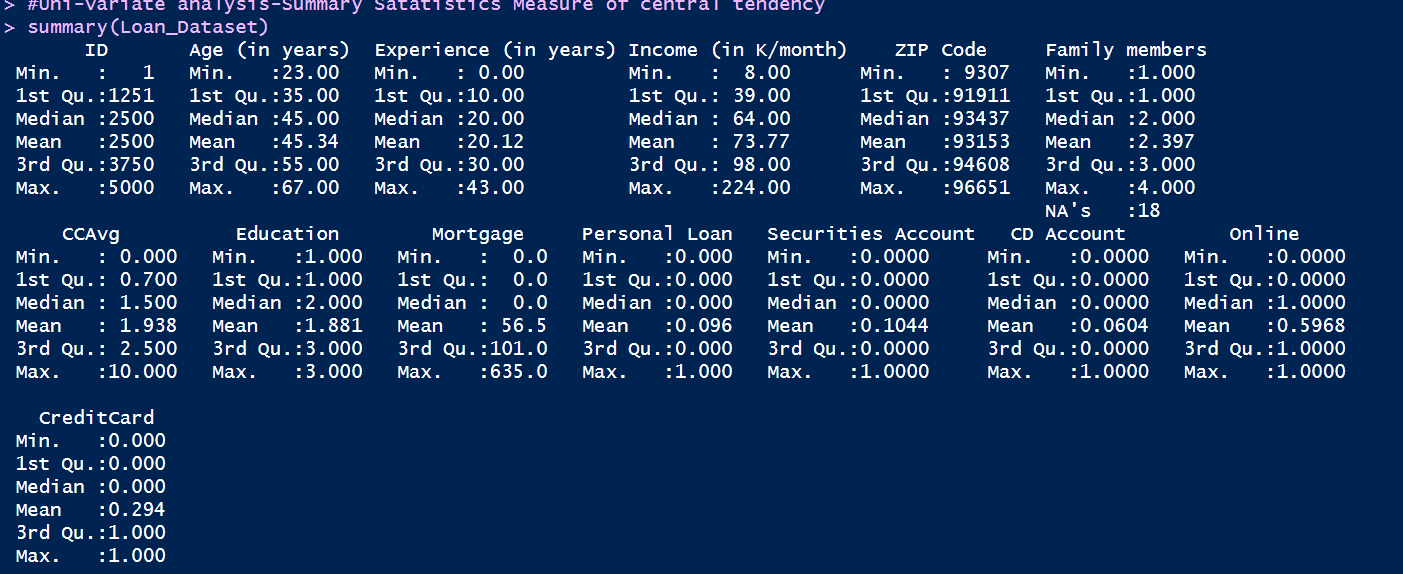
Some basics R functions such as mean , sd ,round are used for the statistical calculation. Here are the following functions are getting used for the better understanding of data and to make further decision **.**

**Structure(Str)-** To get the structure of the personal loan prediction such as class category ,name and the count of the fields

**Summary** –Generally , summary function will give the classification of attributes. Here, the personal loan prediction data having fourteen columns/ fields such as age, working experience, income, personal loan, credit card ,online banking, etc .If the field is categorical data, summary function give the count of each sub-category If the field is integer, then it will return the five stats notably minimum,25th percentile , median or 50th percentile , 75th percentile and maximum.

**2.3 Uni-Variate Analysis:**

**Summary statistics**

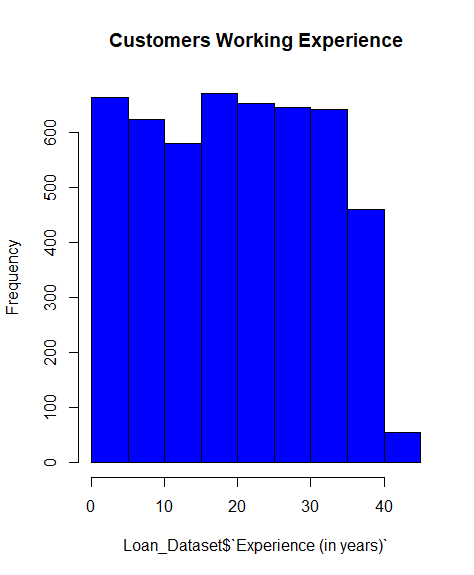
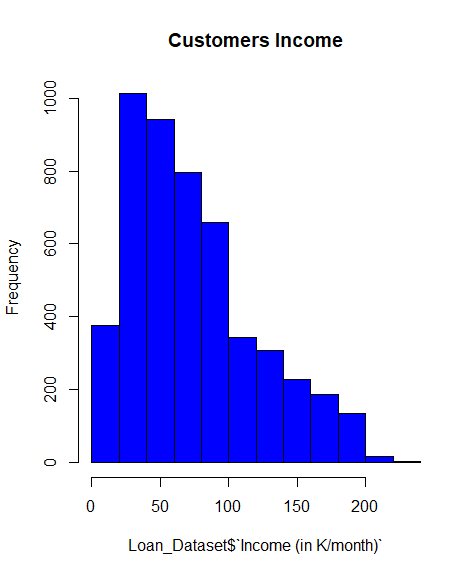
****

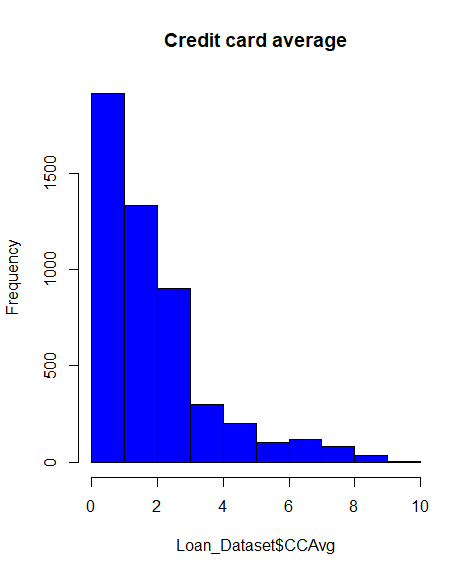
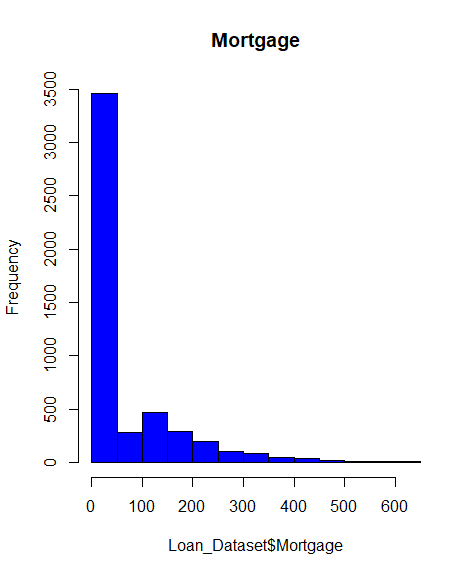
**Frequency distribution of dependent variable**

|  |  |
| --- | --- |
| Category | Count |
| Customers with Personal loan(1) | **480** |
| Customers without Personal loan(0) | **4520** |
| Total | **5000** |

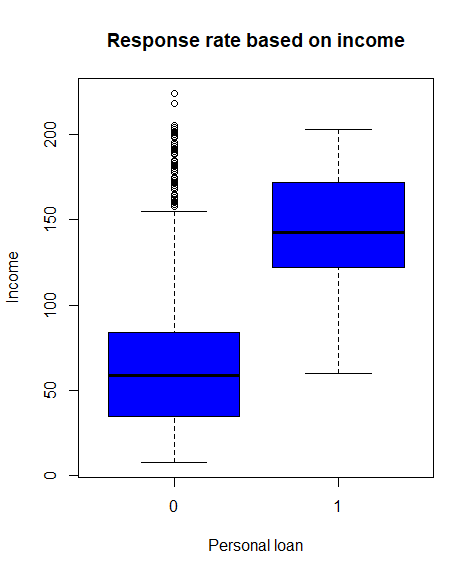
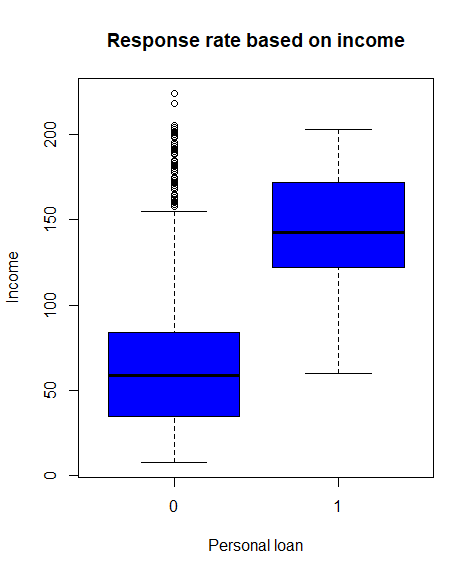
Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

**Frequency distribution of independent variables**

** **

** **

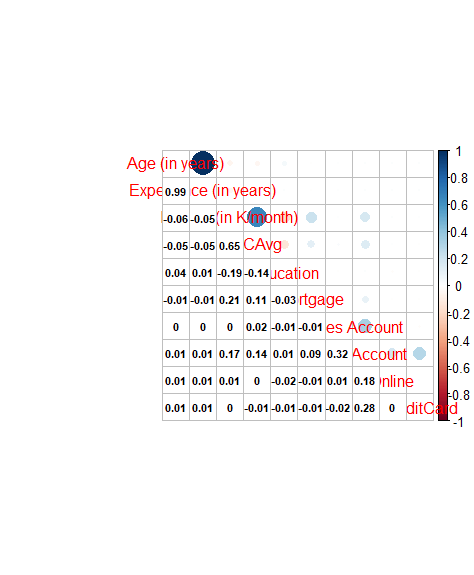
**2.4 Bi-Variate Analysis:**

****

**Correlation:**

1)In the given loan prediction data, customer age and working experience are highly correlated.

2)Customer Income and credit card average are get correlated.



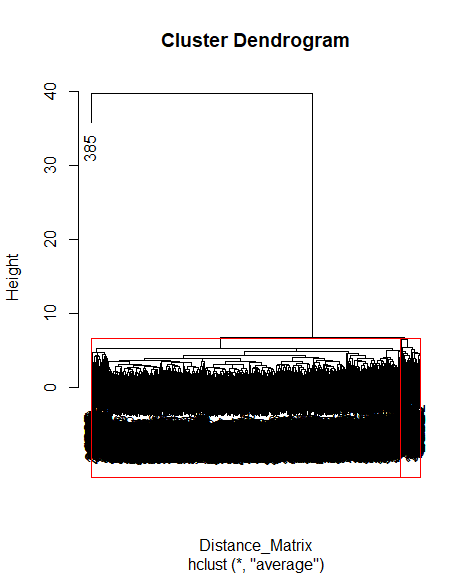
**3.Unsupervised Learning Technique:**

Unsupervised learning is one of the important machine learning technique ,it is neither classified nor labeled,So **splitting the dataset into training and testing doesn’t give much difference in the result**. It is divided into two types

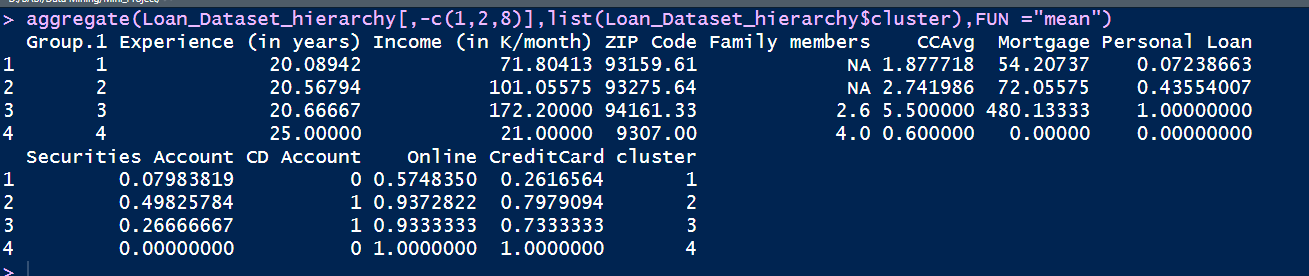
**3.1 Hierachial Clustering:**

1. Scale the given customer details,because each and every column vary with units.
2. Calculate the distance of scaled matrix,cluster them with the hclust() function.
3. Aggregate the result based on the newly created cluster.

Dendrogram



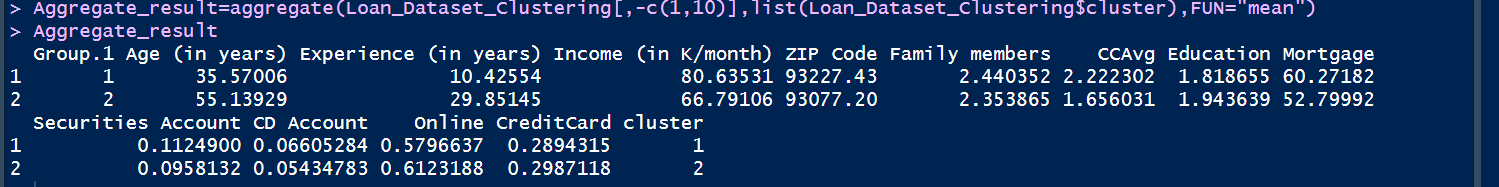
Aggregated Result:



**Disadvantage:** It is computationally cost effective.

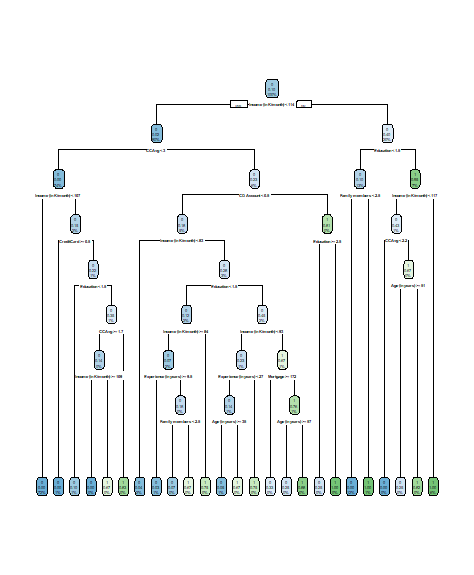
**3.2 K-Means Clustering:**

1. Omit the missing values to have a valid dataset.
2. Scale the given customer details,because each and every column vary with units.
3. Once the scaling is done,cluster them with the Kmeans() function.
4. Find the optimal cluster(K value) using the Nbclust function or the traditional elbow method.

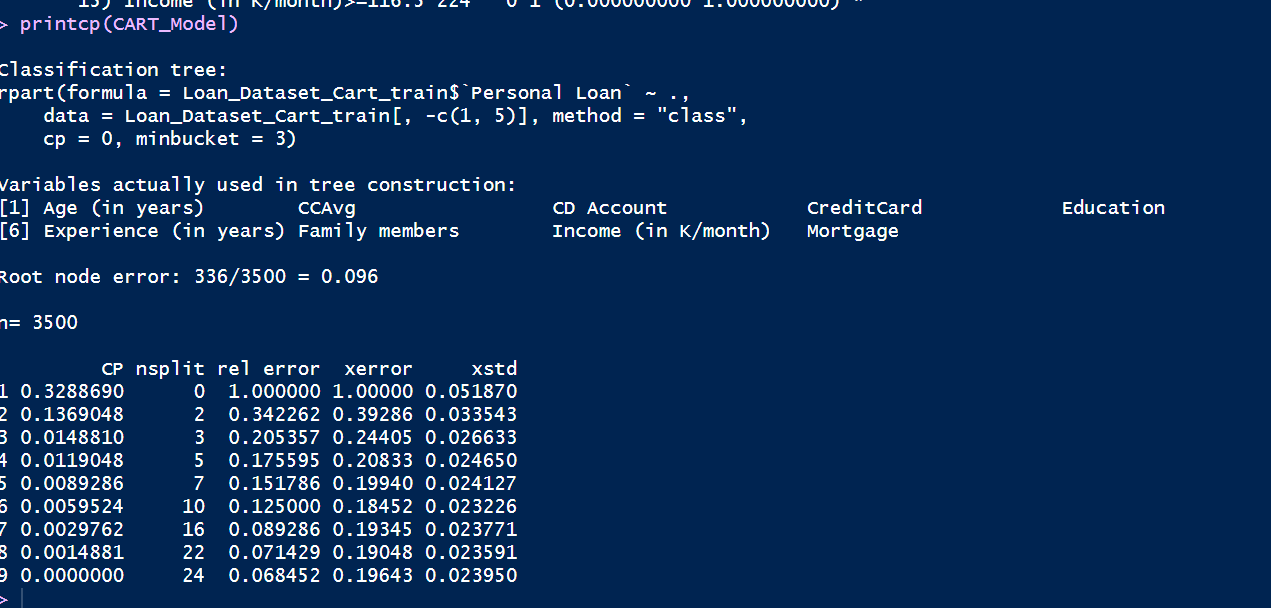


**4.Supervised Learning Technique:**

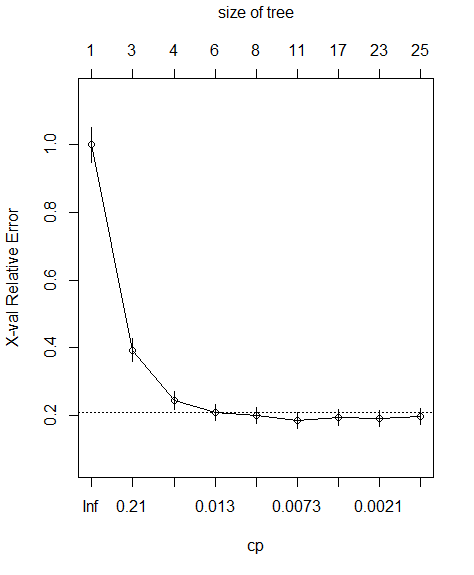
**4.1 CART Model**

****

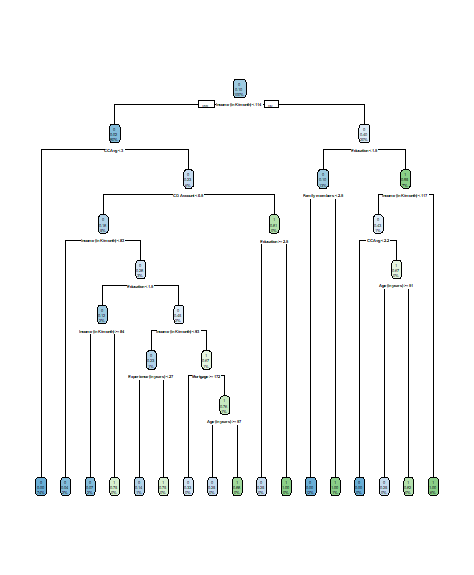
**Complex parameter:**

****

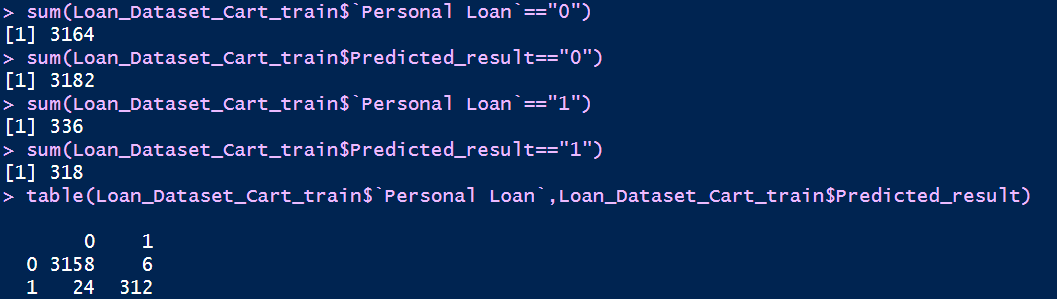
**CP Plot:**

****

Pruned Model:

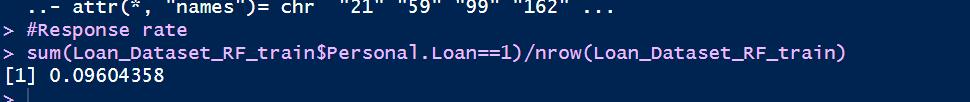


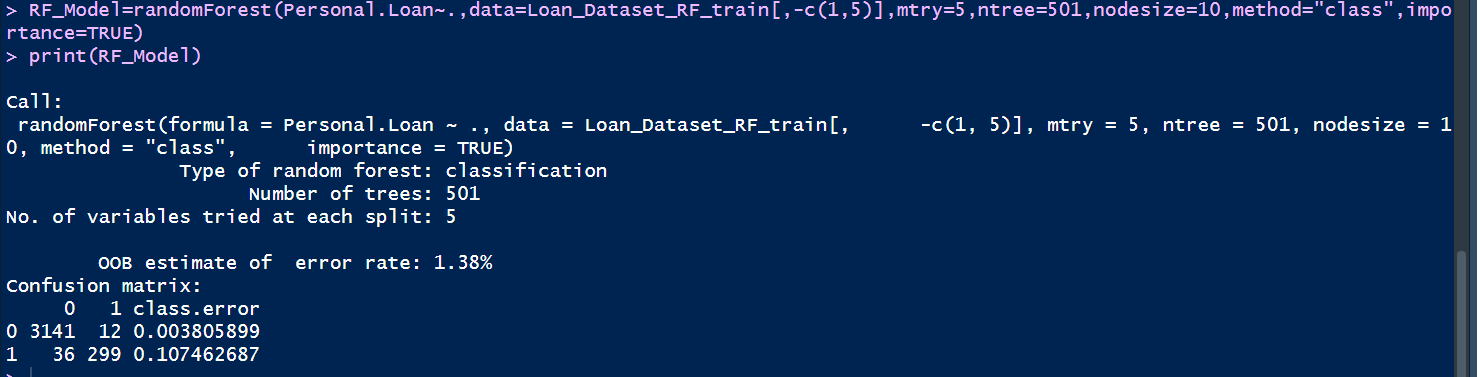
**Model Performance on CART training dataset:**

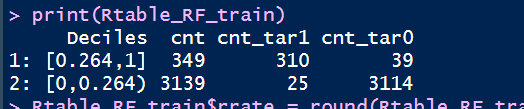
****

**Random Forest:**

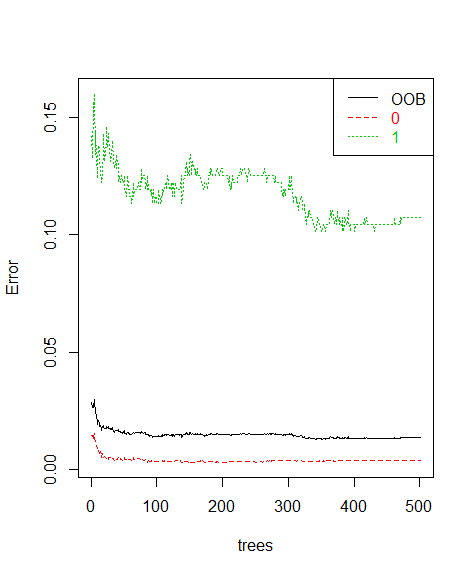
Response rate:







Out of bag error:



It is observed that as the number of tress increases, the OOB error rate starts decreasing till it reaches around 23rd tree with OOB = 0.014 (the minimum value).

After this, the OOB doesn’t decrease further and remains largely steady. Hence, the optimal number of trees would be 23.

Conclusions:

When compared to CART model , random forest is more efficient and it is the best model.

**Appendix A-Source code**

#Bank personal loan modeling

rm(list=ls())

setwd("D:/BABI/Data Mining/Mini\_Project")

library(readxl)

Loan\_Dataset=read\_excel("Thera Bank\_Personal\_Loan.xlsx",2)

#colnames(Loan\_Dataset\_Clustering)=make.names(colnames(Loan\_Dataset\_Clustering))

#Converting Negative working experience into zero,to make it meaningful

for ( i in 1 : nrow(Loan\_Dataset))

{

if (Loan\_Dataset$`Experience (in years)`[i]<0)

{

Loan\_Dataset$`Experience (in years)`[i] =0

}

}

#####################Exploratory data analysis###################

#Uni-variate analysis-Summary Satatistics Measure of central tendency

summary(Loan\_Dataset)

##Frequency distribution for categorical variable (Univariate Analysis)

table(Loan\_Dataset[,c(10)])

library(ggplot2)

hist(Loan\_Dataset$`Age (in years)`,col="BLUE")

qplot(`Age (in years)`, data = Loan\_Dataset, geom = "histogram",fill = `Personal Loan`,ylab = "Frequency")

boxplot(Loan\_Dataset$`Age (in years)`,col="BLUE")#no outliers

hist(Loan\_Dataset$`Experience (in years)`,col="BLUE",main="Customers Working Experience")

boxplot(Loan\_Dataset$`Experience (in years)`,col="BLUE",main="Customers Working Experience")#no outliers

hist(Loan\_Dataset$`Income (in K/month)`,col="BLUE",main="Customers Income")

boxplot(Loan\_Dataset$`Income (in K/month)`,col="BLUE",ylab="Income",main="Customers Income")

hist(Loan\_Dataset$CCAvg,col="BLUE",main = "Credit card average")

boxplot(Loan\_Dataset$CCAvg,col="BLUE",ylab="Credit card average")

hist(Loan\_Dataset$Mortgage,col="BLUE",main="Mortgage")

boxplot(Loan\_Dataset$Mortgage,col="BLUE",ylab="Mortgage")

#Bi-variate analysis

boxplot(Loan\_Dataset$`Income (in K/month)`~ Loan\_Dataset$`Personal Loan`,xlab = "Personal loan", ylab = "Income",col="blue",main="Response rate based on income")

boxplot(Loan\_Dataset$`Experience (in years)`~ Loan\_Dataset$`Personal Loan`,xlab = "Personal loan", ylab = "Working Experience",col="blue",main="Response rate based on experience")

##correlation between continous variables (Bivariate Analysis)

Cor\_Matrix=round(cor(Loan\_Dataset[,c(2,3,4,7,8,9,11,12,13,14)]),2)

library(corrplot)

corrplot.mixed(Cor\_Matrix, lower.col ="black", number.cex = .6)

##################Hierarcial clustering#################

Loan\_Dataset\_hierarchy=Loan\_Dataset

str(Loan\_Dataset\_hierarchy)

View(Loan\_Dataset\_hierarchy)

#Every columns are in different units ,so we need to scale linearly all the columns

Scaled\_Matrix=scale(Loan\_Dataset\_hierarchy[,-c(1,10)])

print(Scaled\_Matrix,digits = 2)

#To check whether the data are linearly scaled or not

apply(Scaled\_Matrix,2,mean)

apply(Scaled\_Matrix,2,sd)

Distance\_Matrix=dist(Scaled\_Matrix,method="minkowski",p=2)

Distance\_Matrix

cluster=hclust(Distance\_Matrix,method="average")

cluster

plot(cluster,labels=as.character("0","1"))

plot(cluster)

plot(cluster,labels=as.character(Loan\_Dataset\_hierarchy[,2]))

rect.hclust(cluster,k=3,border = "Red")

Loan\_Dataset\_hierarchy$cluster=cutree(cluster,k=4)

Loan\_Dataset\_hierarchy

aggregate(Loan\_Dataset\_hierarchy[,-c(1,2,8)],list(Loan\_Dataset\_hierarchy$cluster),FUN ="mean")

##################K-means clustering##########################

Loan\_Dataset\_Clustering=Loan\_Dataset

str(Loan\_Dataset\_Clustering)

scaled\_Matrix=scale(Loan\_Dataset\_Clustering[,-c(1,10)])

head(scaled\_Matrix)

cluster\_data=kmeans(na.omit(scaled\_Matrix),centers = 2,nstart = 5)

cluster\_data$cluster

cluster\_data$totss

cluster\_data$size

cluster\_data$centers

cluster\_data$withinss

cluster\_data$tot.withinss

cluster\_data$betweenss

tot=rep(1,5)

for (i in 1:5)

{

set.seed(100)

cluster\_data\_new=kmeans(na.omit(scaled\_Matrix),centers = i,nstart = 5)

tot[i]=cluster\_data\_new$tot.withinss

print(cluster\_data\_new$tot.withinss)

}

library(cluster)

plot(tot,type='b')

#by the tradition method(Elbow),k value is 3

clusplot(scaled\_Matrix,cluster\_data$cluster,color =TRUE,shade =TRUE)

library(NbClust)

Nb\_cluster=NbClust(na.omit(scaled\_Matrix),min.nc = 2,max.nc = 5,method = "kmeans")

cluster\_data=kmeans(na.omit(scaled\_Matrix),centers = 3,nstart = 5)

Loan\_Dataset\_Clustering=na.omit(Loan\_Dataset\_Clustering)

nrow(Loan\_Dataset\_Clustering)

Loan\_Dataset\_Clustering$cluster=cluster\_data$cluster

Aggregate\_result=aggregate(Loan\_Dataset\_Clustering[,-c(1,10)],list(Loan\_Dataset\_Clustering$cluster),FUN="mean")

Aggregate\_result

#CART model

Loan\_Dataset\_Cart=Loan\_Dataset

library(caTools)

sample=sample.split(Loan\_Dataset\_Cart$`Personal Loan`,SplitRatio = 0.7)

Loan\_Dataset\_Cart\_train=subset(Loan\_Dataset\_Cart,sample==TRUE)

nrow(Loan\_Dataset\_Cart\_train)

Loan\_Dataset\_Cart\_test=subset(Loan\_Dataset\_Cart,sample==FALSE)

Loan\_Dataset\_Cart\_train$`Personal Loan`=as.factor(Loan\_Dataset\_Cart\_train$`Personal Loan`)

dim(Loan\_Dataset\_Cart\_train)

dim(Loan\_Dataset\_Cart\_test)

library("rpart")

library("rpart.plot")

set.seed(1000)

CART\_Model=rpart(formula=Loan\_Dataset\_Cart\_train$`Personal Loan`~.,data=Loan\_Dataset\_Cart\_train[,-c(1,5)],method = "class",cp=0,minbucket=3)

rpart.plot(CART\_Model)

CART\_Model

printcp(CART\_Model)

plotcp(CART\_Model)

Pruned\_Model=prune(CART\_Model,cp=0.004,"CP")

printcp(Pruned\_Model)

plotcp(Pruned\_Model)

rpart.plot(Pruned\_Model)

path.rpart(Pruned\_Model,c(5))#To get the path of tree w.r.t node number

#################Model performance on CART training dataset################

#1)scoring

Loan\_Dataset\_Cart\_train$Predicted\_result=predict(Pruned\_Model,data=Loan\_Dataset\_Cart\_train[,-c(1,5)],type="class")

Loan\_Dataset\_Cart\_train$Probabilty=predict(Pruned\_Model,data=Loan\_Dataset\_Cart\_train[,-c(1,5)],type ="prob")

head(Loan\_Dataset\_Cart\_train)

sum(Loan\_Dataset\_Cart\_train$`Personal Loan`=="0")

sum(Loan\_Dataset\_Cart\_train$Predicted\_result=="0")

sum(Loan\_Dataset\_Cart\_train$`Personal Loan`=="1")

sum(Loan\_Dataset\_Cart\_train$Predicted\_result=="1")

table(Loan\_Dataset\_Cart\_train$`Personal Loan`,Loan\_Dataset\_Cart\_train$Predicted\_result)

#2)Confusion matrix

Table\_cart\_train=table(Loan\_Dataset\_Cart\_train$Personal.Loan,Loan\_Dataset\_Cart\_train$Prediction)

Accuracy\_cart\_train=(Table\_cart\_train[1,1]+Table\_cart\_train[2,2])/nrow(Loan\_Dataset\_Cart\_train)

Error\_cart\_train=round((1-Accuracy\_cart\_train),4)

#Sensitivity/Recall/True positive rate

Sensitivity\_cart\_train=(Table\_cart\_train[2,2])/(Table\_cart\_train[2,1]+Table\_cart\_train[2,2])

#Specificity /True Negative Rate

Specificity\_cart\_train=(Table\_cart\_train[1,1])/(Table\_cart\_train[1,2]+Table\_cart\_train[1,1])

#3)Deciling code-Rank ordering

qs\_cart\_train=quantile(Loan\_Dataset\_Cart\_train$Probablity,prob = seq(0,1,length=11))

print(qs\_cart\_train)

print(qs\_cart\_train[10])

threshold=qs\_cart\_train[10]

mean((Loan\_Dataset\_Cart\_train$Personal.Loan[Loan\_Dataset\_Cart\_train$Probablity>threshold])=="1")

Loan\_Dataset\_Cart\_train$Deciles=cut(Loan\_Dataset\_Cart\_train$Probablity,unique(qs\_cart\_train),include.lowest = TRUE,right = FALSE)

head(Loan\_Dataset\_Cart\_train)

print(Loan\_Dataset\_Cart\_train$Deciles)

#Rank ordering

library(data.table)

#Loan\_Dataset\_Cart\_train$Personal.Loan=as.numeric(Loan\_Dataset\_Cart\_train$Personal.Loan)

DT\_cart\_train=data.table(Loan\_Dataset\_Cart\_train)

#Aggregate columns

Rtable\_cart\_train=DT\_cart\_train[,list(cnt=length(Personal.Loan),

cnt\_tar1 = sum(Personal.Loan==1),

cnt\_tar0 = sum(Personal.Loan==0)),by=Deciles][order(-Deciles)]

print(Rtable\_cart\_train)

Rtable\_cart\_train$rrate = round(Rtable\_cart\_train$cnt\_tar1 / Rtable\_cart\_train$cnt,4)\*100;

Rtable\_cart\_train$cum\_resp = cumsum(Rtable\_cart\_train$cnt\_tar1)

Rtable\_cart\_train$cum\_non\_resp = cumsum(Rtable\_cart\_train$cnt\_tar0)

Rtable\_cart\_train$cum\_rel\_resp = round(Rtable\_cart\_train$cum\_resp / sum(Rtable\_cart\_train$cnt\_tar1),4)\*100;

Rtable\_cart\_train$cum\_rel\_non\_resp = round(Rtable\_cart\_train$cum\_non\_resp / sum(Rtable\_cart\_train$cnt\_tar0),4)\*100;

Rtable\_cart\_train$ks = abs(Rtable\_cart\_train$cum\_rel\_resp - Rtable\_cart\_train$cum\_rel\_non\_resp);

print(Rtable\_cart\_train)

#4)ROC curve(FPR Vs TPR)

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

library(ROCR)

library(ineq)

ROC\_cart\_train=prediction(Loan\_Dataset\_Cart\_train$Probablity,Loan\_Dataset\_Cart\_train$Prediction)

perf\_cart\_train=performance(ROC\_cart\_train,"tpr","fpr")

plot(perf\_cart\_train,main="TPR Vs FPR")

KS\_cart\_train = max(perf\_cart\_train@y.values[[1]]-perf\_cart\_train@x.values[[1]])

auc\_cart\_train = performance(ROC\_cart\_train,"auc");

auc\_cart\_train = as.numeric(auc\_cart\_train@y.values)

gini\_cart\_train = ineq(Loan\_Dataset\_Cart\_train$Probablity, type="Gini")

# 5)Concordance Function

library(InformationValue)

Concordance(actuals=Loan\_Dataset\_Cart\_train$Personal.Loan, predictedScores=Loan\_Dataset\_Cart\_train$Probablity)

#################Model performance on CART Testing dataset################

# 1)Scoring

Loan\_Dataset\_Cart\_test$Predicted\_result=predict(Pruned\_Model,newdata=Loan\_Dataset\_Cart\_test[,-c(1,5)],type="class")

Loan\_Dataset\_Cart\_test$Probabilty=predict(Pruned\_Model,newdata=Loan\_Dataset\_Cart\_test[,-c(1,5)],type ="prob")

Accuracy\_CART\_train=round((3157+310)/nrow(Loan\_Dataset\_Cart\_train),4)

Accuracy\_CART\_test=round((1352+125)/nrow(Loan\_Dataset\_Cart\_test),4)

#2)Confusion matrix

Table\_cart\_test=table(Loan\_Dataset\_Cart\_test$Personal.Loan,Loan\_Dataset\_Cart\_test$Prediction)

Accuracy\_cart\_test=(Table\_cart\_test[1,1]+Table\_cart\_test[2,2])/nrow(Loan\_Dataset\_Cart\_test)

Error\_cart\_train=round((1-Accuracy\_cart\_test),4)

#Sensitivity/Recall/True positive rate

Sensitivity\_cart\_train=(Table\_cart\_test[2,2])/(Table\_cart\_test[2,1]+Table\_cart\_test[2,2])

#Specificity /True Negative Rate

Specificity\_cart\_train=(Table\_cart\_test[1,1])/(Table\_cart\_test[1,2]+Table\_cart\_test[1,1])

#3)Deciling code-Rank ordering

qs\_cart\_test=quantile(Loan\_Dataset\_Cart\_test$Probablity,prob = seq(0,1,length=11))

print(qs\_cart\_test)

print(qs\_cart\_test[10])

threshold=qs\_cart\_test[10]

mean((Loan\_Dataset\_Cart\_test$Personal.Loan[Loan\_Dataset\_Cart\_test$Probablity>threshold])=="1")

Loan\_Dataset\_Cart\_test$Deciles=cut(Loan\_Dataset\_Cart\_test$Probablity,unique(qs\_cart\_test),include.lowest = TRUE,right = FALSE)

head(Loan\_Dataset\_Cart\_test)

print(Loan\_Dataset\_Cart\_test$Deciles)

#Rank ordering

library(data.table)

#Loan\_Dataset\_Cart\_test$Personal.Loan=as.numeric(Loan\_Dataset\_Cart\_test$Personal.Loan)

DT\_cart\_test=data.table(Loan\_Dataset\_Cart\_test)

#Aggregate columns

Rtable\_cart\_test=DT\_cart\_test[,list(cnt=length(Personal.Loan),

cnt\_tar1 = sum(Personal.Loan==1),

cnt\_tar0 = sum(Personal.Loan==0)),by=Deciles][order(-Deciles)]

print(Rtable\_cart\_test)

Rtable\_cart\_test$rrate = round(Rtable\_cart\_test$cnt\_tar1 / Rtable\_cart\_test$cnt,4)\*100;

Rtable\_cart\_test$cum\_resp = cumsum(Rtable\_cart\_test$cnt\_tar1)

Rtable\_cart\_test$cum\_non\_resp = cumsum(Rtable\_cart\_test$cnt\_tar0)

Rtable\_cart\_test$cum\_rel\_resp = round(Rtable\_cart\_test$cum\_resp / sum(Rtable\_cart\_test$cnt\_tar1),4)\*100;

Rtable\_cart\_test$cum\_rel\_non\_resp = round(Rtable\_cart\_test$cum\_non\_resp / sum(Rtable\_cart\_test$cnt\_tar0),4)\*100;

Rtable\_cart\_test$ks = abs(Rtable\_cart\_test$cum\_rel\_resp - Rtable\_cart\_test$cum\_rel\_non\_resp);

print(Rtable\_cart\_test)

#4)ROC curve(FPR Vs TPR)

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

library(ROCR)

library(ineq)

ROC\_cart\_test=prediction(Loan\_Dataset\_Cart\_test$Probablity,Loan\_Dataset\_Cart\_test$Prediction)

perf\_cart\_test=performance(ROC\_cart\_test,"tpr","fpr")

plot(perf\_cart\_test,main="TPR Vs FPR")

KS\_cart\_test = max(perf\_cart\_test@y.values[[1]]-perf\_cart\_test@x.values[[1]])

auc\_cart\_test = performance(ROC\_cart\_test,"auc");

auc\_cart\_test = as.numeric(auc\_cart\_test@y.values)

gini\_cart\_test = ineq(Loan\_Dataset\_Cart\_test$Probablity, type="Gini")

# 5)Concordance Function

library(InformationValue)

Concordance(actuals=Loan\_Dataset\_Cart\_test$Personal.Loan, predictedScores=Loan\_Dataset\_Cart\_test$Probablity)

#######################Random Forest#####################

library(randomForest)

Loan\_Dataset\_RF=na.omit(Loan\_Dataset)

samples=sample.split(Loan\_Dataset\_RF$`Personal Loan`,SplitRatio = 0.7)

Loan\_Dataset\_RF\_train=subset(Loan\_Dataset\_RF,samples=="TRUE")

Loan\_Dataset\_RF\_test=subset(Loan\_Dataset\_RF,samples=="FALSE")

set.seed(1234)

Loan\_Dataset\_RF\_train$`Personal Loan`=as.factor(Loan\_Dataset\_RF\_train$`Personal Loan`)

colnames(Loan\_Dataset\_RF\_train)=make.names(colnames(Loan\_Dataset\_RF\_train))

str(Loan\_Dataset\_RF\_train)

#Response rate

sum(Loan\_Dataset\_RF\_train$Personal.Loan==1)/nrow(Loan\_Dataset\_RF\_train)

##Build the first RF model

RF\_Model=randomForest(Personal.Loan~.,data=Loan\_Dataset\_RF\_train[,-c(1,5)],mtry=5,ntree=501,nodesize=10,method="class",importance=TRUE)

print(RF\_Model)

##Plot the RF to know the optimum number of trees

#Out of Bag estimate errror

plot(RF\_Model,main="")

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

title(main="Error Rates Random Forest - Training data")

RF\_Model$err.rate

#It is observed that as the number of tress increases, the OOB error rate starts decreasing

#till it reaches around 23rd tree with OOB = 0.014 (the minimum value).

#After this, the OOB doesn’t decrease further and remains largely steady. Hence, the optimal number of trees would be 23.

##Identify the importance of the variables

importance(RF\_Model)

##Tune up the RF model to find out the best mtry

RF\_Tuned=tuneRF(Loan\_Dataset\_RF\_train[,-c(1,10)],Loan\_Dataset\_RF\_train$Personal.Loan,stepFactor = 2,improve = 0.0001,ntreeTry = 23)

##Build the refined RF model

Tuned\_RF\_Model=randomForest(Personal.Loan~.,data=Loan\_Dataset\_RF\_train[,-c(1,5)],mtry=6,ntree=51,nodesize=10,method="class",importance=TRUE)

######################RF Model performance-Training dataset#####################

#1)Scoring

Loan\_Dataset\_RF\_train$Prediction=predict(Tuned\_RF\_Model,data=Loan\_Dataset\_RF\_train[,-c(1,5)],type = "class")

Loan\_Dataset\_RF\_train$Probablity=predict(Tuned\_RF\_Model,data=Loan\_Dataset\_RF\_train[,-c(1,5)],type = "prob")[,"1"]

#2)Confusion matrix

Table\_RF\_train=table(Loan\_Dataset\_RF\_train$Personal.Loan,Loan\_Dataset\_RF\_train$Prediction)

Accuracy\_RF\_train=(Table\_RF\_train[1,1]+Table\_RF\_train[2,2])/nrow(Loan\_Dataset\_RF\_train)

Error\_RF\_train=round((1-Accuracy\_RF\_train),4)

#Sensitivity/Recall/True positive rate

Sensitivity\_RF\_train=(Table\_RF\_train[2,2])/(Table\_RF\_train[2,1]+Table\_RF\_train[2,2])

#Specificity /True Negative Rate

Specificity\_RF\_train=(Table\_RF\_train[1,1])/(Table\_RF\_train[1,2]+Table\_RF\_train[1,1])

#3)Deciling code-Rank ordering

qs=quantile(Loan\_Dataset\_RF\_train$Probablity,prob = seq(0,1,length=11))

print(qs)

print(qs[10])

threshold=qs[10]

mean((Loan\_Dataset\_RF\_train$Personal.Loan[Loan\_Dataset\_RF\_train$Probablity>threshold])=="1")

Loan\_Dataset\_RF\_train$Deciles=cut(Loan\_Dataset\_RF\_train$Probablity,unique(qs),include.lowest = TRUE,right = FALSE)

head(Loan\_Dataset\_RF\_train)

print(Loan\_Dataset\_RF\_train$Deciles)

#Rank ordering

library(data.table)

#Loan\_Dataset\_RF\_train$Personal.Loan=as.numeric(Loan\_Dataset\_RF\_train$Personal.Loan)

DT\_RF\_train=data.table(Loan\_Dataset\_RF\_train)

#Aggregate columns

Rtable\_RF\_train=DT\_RF\_train[,list(cnt=length(Personal.Loan),

cnt\_tar1 = sum(Personal.Loan==1),

cnt\_tar0 = sum(Personal.Loan==0)),by=Deciles][order(-Deciles)]

print(Rtable\_RF\_train)

Rtable\_RF\_train$rrate = round(Rtable\_RF\_train$cnt\_tar1 / Rtable\_RF\_train$cnt,4)\*100;

Rtable\_RF\_train$cum\_resp = cumsum(Rtable\_RF\_train$cnt\_tar1)

Rtable\_RF\_train$cum\_non\_resp = cumsum(Rtable\_RF\_train$cnt\_tar0)

Rtable\_RF\_train$cum\_rel\_resp = round(Rtable\_RF\_train$cum\_resp / sum(Rtable\_RF\_train$cnt\_tar1),4)\*100;

Rtable\_RF\_train$cum\_rel\_non\_resp = round(Rtable\_RF\_train$cum\_non\_resp / sum(Rtable\_RF\_train$cnt\_tar0),4)\*100;

Rtable\_RF\_train$ks = abs(Rtable\_RF\_train$cum\_rel\_resp - Rtable\_RF\_train$cum\_rel\_non\_resp);

print(Rtable\_RF\_train)

#4)ROC curve(FPR Vs TPR)

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

library(ROCR)

library(ineq)

ROC\_RF\_train=prediction(Loan\_Dataset\_RF\_train$Probablity,Loan\_Dataset\_RF\_train$Prediction)

perf\_RF\_train=performance(ROC\_RF\_train,"tpr","fpr")

plot(perf\_RF\_train,main="TPR Vs FPR")

KS\_RF\_train = max(perf\_RF\_train@y.values[[1]]-perf\_RF\_train@x.values[[1]])

auc\_RF\_train = performance(ROC\_RF\_train,"auc");

auc\_RF\_train = as.numeric(auc\_RF\_train@y.values)

gini = ineq(Loan\_Dataset\_RF\_train$Probablity, type="Gini")

# 5)Concordance Function

library(InformationValue)

Concordance(actuals=Loan\_Dataset\_RF\_train$Personal.Loan, predictedScores=Loan\_Dataset\_RF\_train$Probablity)

######################RF Model performance-Testing dataset#####################

#1)Scoring

colnames(Loan\_Dataset\_RF\_test)=make.names(colnames(Loan\_Dataset\_RF\_test))

Loan\_Dataset\_RF\_test$Prediction=predict(Tuned\_RF\_Model,newdata=Loan\_Dataset\_RF\_test[,-c(1,5)],type = "class")

Loan\_Dataset\_RF\_test$Probablity=predict(Tuned\_RF\_Model,newdata=Loan\_Dataset\_RF\_test[,-c(1,5)],type = "prob")[,"1"]

#2)Confusion matrix

Table\_RF\_test=table(Loan\_Dataset\_RF\_test$Personal.Loan,Loan\_Dataset\_RF\_test$Prediction)

Accuracy\_RF\_test=(Table\_RF\_test[1,1]+Table\_RF\_test[2,2])/nrow(Loan\_Dataset\_RF\_test)

Error\_RF\_test=round((1-Accuracy\_RF\_test),4)

#Sensitivity/Recall/True positive rate

Sensitivity\_RF\_test=(Table\_RF\_test[2,2])/(Table\_RF\_test[2,1]+Table\_RF\_test[2,2])

#Specificity /True Negative Rate

Specificity\_RF\_test=(Table\_RF\_test[1,1])/(Table\_RF\_test[1,2]+Table\_RF\_test[1,1])

#3)Deciling code-Rank ordering

qs\_RF\_test=quantile(Loan\_Dataset\_RF\_train$Probablity,prob = seq(0,1,length=11))

print(qs\_RF\_test)

print(qs\_RF\_test[10])

threshold=qs\_RF\_test[10]

mean((Loan\_Dataset\_RF\_test$Personal.Loan[Loan\_Dataset\_RF\_test$Probablity>threshold])=="1")

Loan\_Dataset\_RF\_test$Deciles=cut(Loan\_Dataset\_RF\_test$Probablity,unique(qs),include.lowest = TRUE,right = FALSE)

head(Loan\_Dataset\_RF\_test)

print(Loan\_Dataset\_RF\_test$Deciles)

#Rank ordering

library(data.table)

#Loan\_Dataset\_RF\_test$Personal.Loan=as.numeric(Loan\_Dataset\_RF\_test$Personal.Loan)

DT\_RF\_train=data.table(Loan\_Dataset\_RF\_test)

#Aggregate columns

Rtable\_RF\_train=DT\_RF\_train[,list(cnt=length(Personal.Loan),

cnt\_tar1 = sum(Personal.Loan==1),

cnt\_tar0 = sum(Personal.Loan==0)),by=Deciles][order(-Deciles)]

print(Rtable\_RF\_train)

Rtable\_RF\_train$rrate = round(Rtable\_RF\_train$cnt\_tar1 / Rtable\_RF\_train$cnt,4)\*100;

Rtable\_RF\_train$cum\_resp = cumsum(Rtable\_RF\_train$cnt\_tar1)

Rtable\_RF\_train$cum\_non\_resp = cumsum(Rtable\_RF\_train$cnt\_tar0)

Rtable\_RF\_train$cum\_rel\_resp = round(Rtable\_RF\_train$cum\_resp / sum(Rtable\_RF\_train$cnt\_tar1),4)\*100;

Rtable\_RF\_train$cum\_rel\_non\_resp = round(Rtable\_RF\_train$cum\_non\_resp / sum(Rtable\_RF\_train$cnt\_tar0),4)\*100;

Rtable\_RF\_train$ks = abs(Rtable\_RF\_train$cum\_rel\_resp - Rtable\_RF\_train$cum\_rel\_non\_resp);

print(Rtable\_RF\_train)

#4)ROC curve(FPR Vs TPR)

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

library(ROCR)

library(ineq)

ROC\_RF\_test=prediction(Loan\_Dataset\_RF\_test$Probablity,Loan\_Dataset\_RF\_test$Prediction)

perf\_RF\_test=performance(ROC\_RF\_test,"tpr","fpr")

plot(perf\_RF\_test,main="TPR Vs FPR- RF Testing dataset")

KS\_RF\_test = max(perf\_RF\_test@y.values[[1]]-perf\_RF\_test@x.values[[1]])

auc\_RF\_test = performance(ROC\_RF\_test,"auc");

auc\_RF\_test = as.numeric(auc\_RF\_test@y.values)

gini\_RF\_test = ineq(Loan\_Dataset\_RF\_test$Probablity, type="Gini")

# 5)Concordance Function

library(InformationValue)

Concordance(actuals=Loan\_Dataset\_RF\_test$Personal.Loan, predictedScores=Loan\_Dataset\_RF\_test$Probablity)