Project – Data Mining

EDA of the data available. Showcase the results using appropriate graphs

EDA was performed on the dataset after it being imported in R environment. We discovered the following in the Dataset.

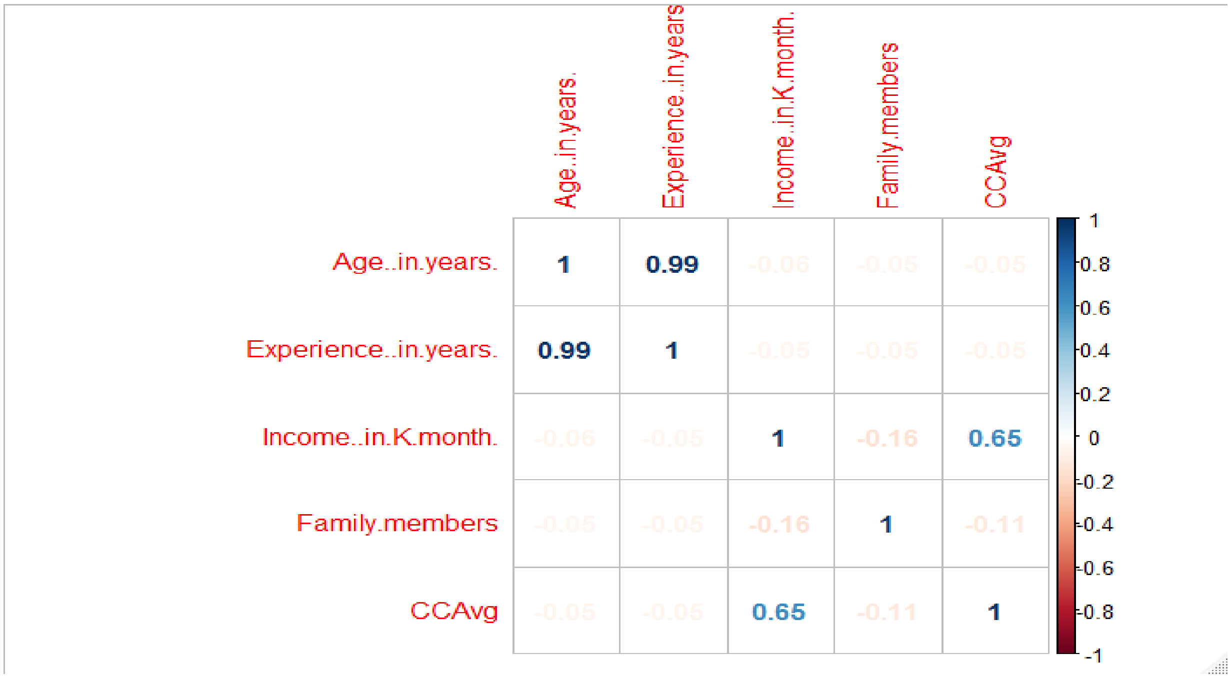
- Family has 18 rows containing NA
- Experience has got negative values like -1, -2, -3
- Around 70% of the rows in mortgage has got value 0
- Experience, Education, Personal Loans, Securities Account, CD Account, Online, Credit Card are as numeric variables in the Data set. We need to convert them into factor variables.

The following remediation were taken due to the above finding:

- The 18 NA rows in Family were omitted.
- We converted the negative values in Experience to 0, and then converted it to a factor variable.
- The rest 30 % of the rows in mortgage having value greater than O were converted to 1, now the entire mortgage column was converted to a factor variable
- Experience, Education, Personal Loans, Securities Account, CD Account, Online, Credit Card are converted to factors.

We have a plot of histograms for the Numeric Variables.

We also found out the correlation between the numeric variables of the dataset.



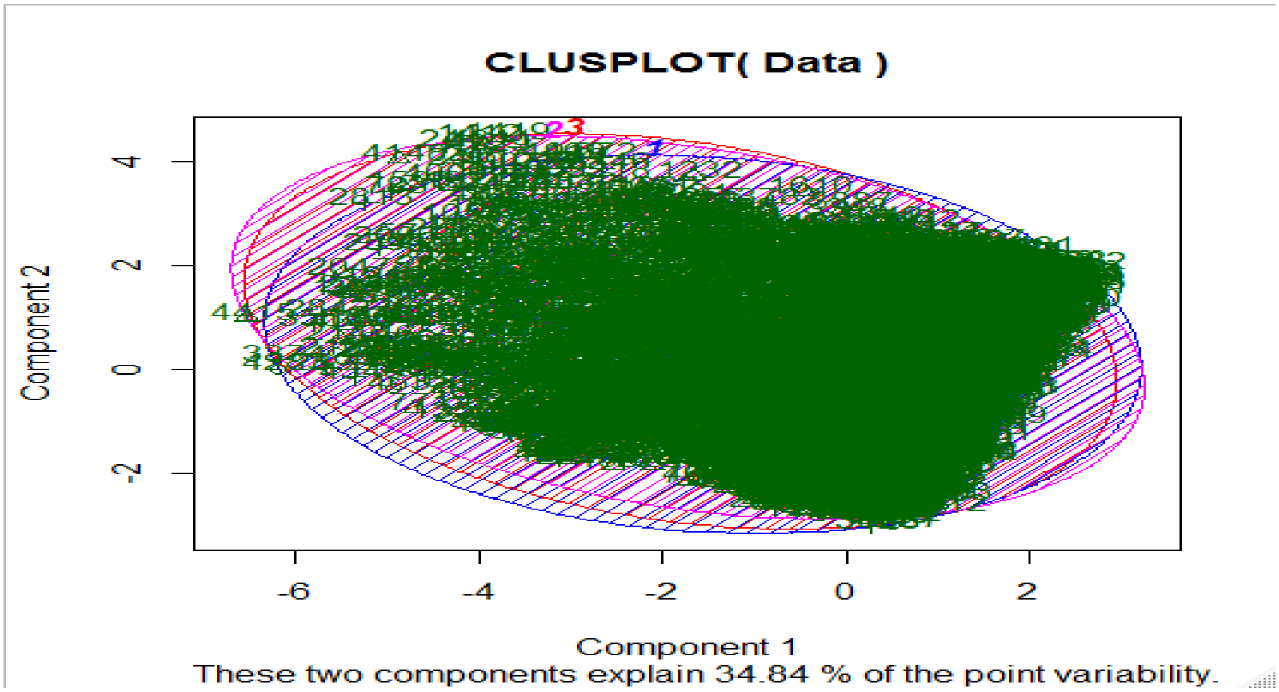
We can see that Age has high correlation with experience, which is obvious.

We also found out that credit card average has got something (high correlation) to do with a person's salary.

APPLY APPROPRIATE CLUSTERING ON THE DATA AND INTERPRET THE OUTPUT

Since our dataset has both Numeric and Factor variables. So, we used Gower method for calculating the distances. Then we used the pam algorithm for clustering. We found that 3 clusters.

We have removed ZIP code during clustering.



After clustering, we find out the customer profile according to each cluster as shown below.

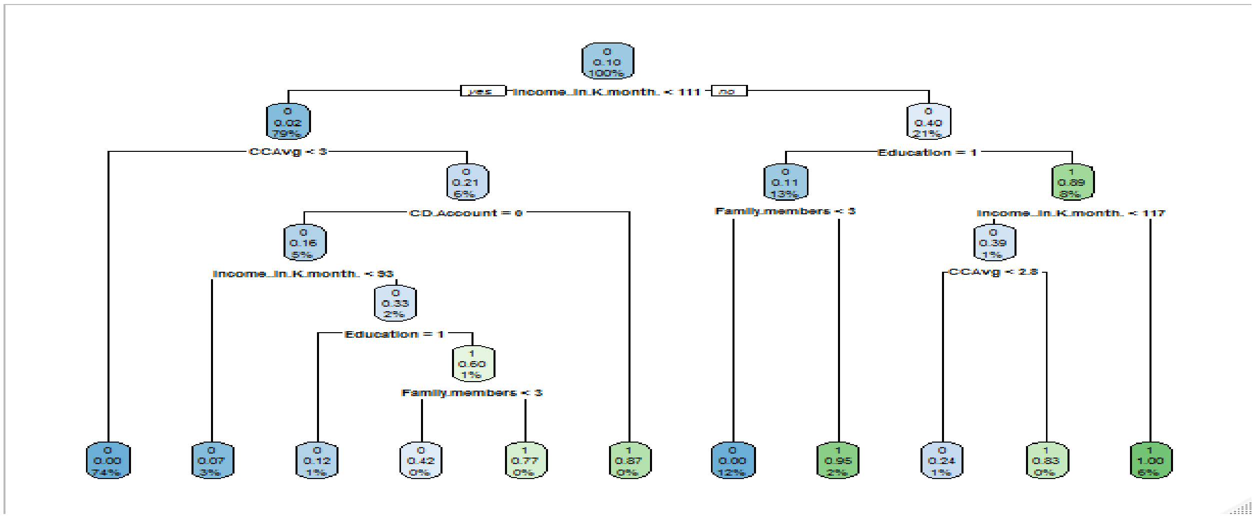
	Group.1	Experience..in.years.	Income..in.K.month.	Family.members	Education	Mortgage	Personal.Loan	Securities.Account	CD.Account	Online	CreditCard	cluster
1	1	20	65	2.9	2.4	30	0.127	0.11	0.024	0.17	0.29	1
2	2	20	79	2.1	1.6	0	0.069	0.10	0.071	0.83	0.30	2
3	3	20	76	2.3	1.7	187	0.103	0.11	0.087	0.73	0.30	3

BUILD APPROPRIATE MODELS ON BOTH THE TEST AND TRAIN DATA (CART & RANDOM FOREST). INTERPRET ALL THE MODEL OUTPUTS AND DO THE NECESSARY MODIFICATIONS WHEREVER ELIGIBLE (SUCH AS PRUNING)

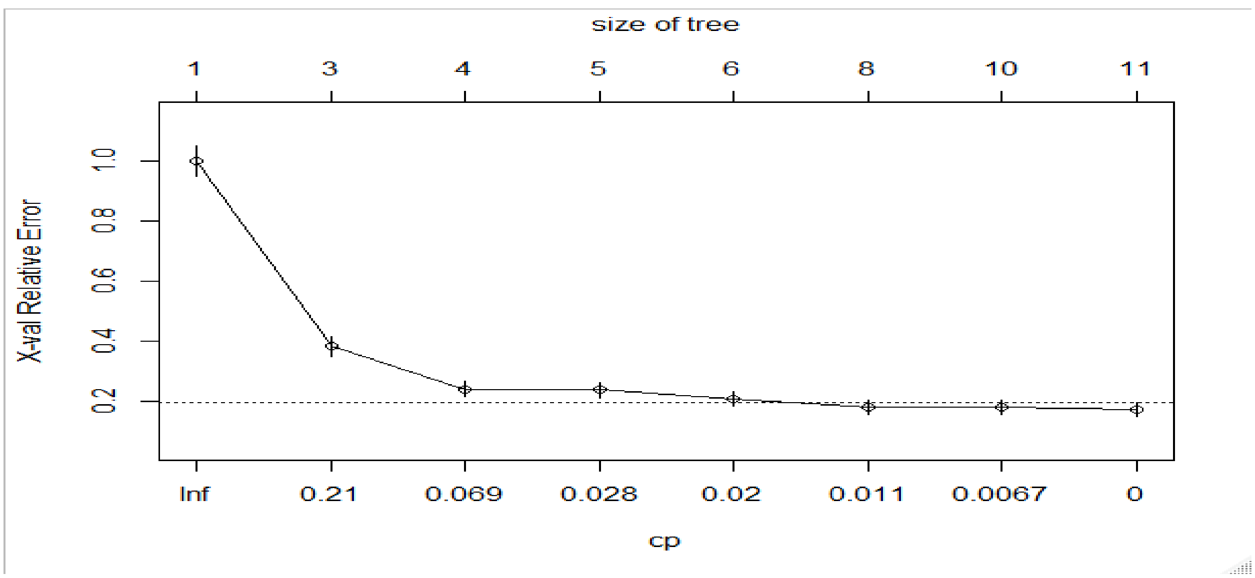
CART

We have split the data set into train & test containing 70 % and 30% of the data respectively.

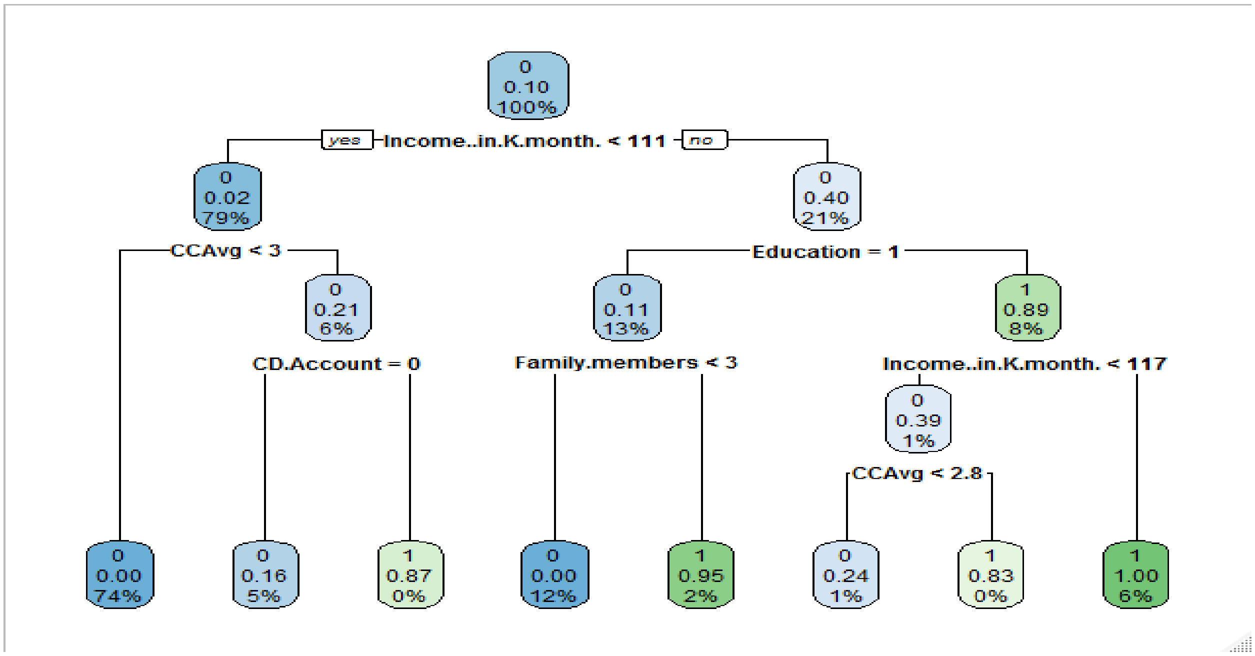
Then we form a tree with cp=0 i.e. with no pruning.



From the below graph, we find cp=0.0085



Now we'll prune the tree with cp=0.0085

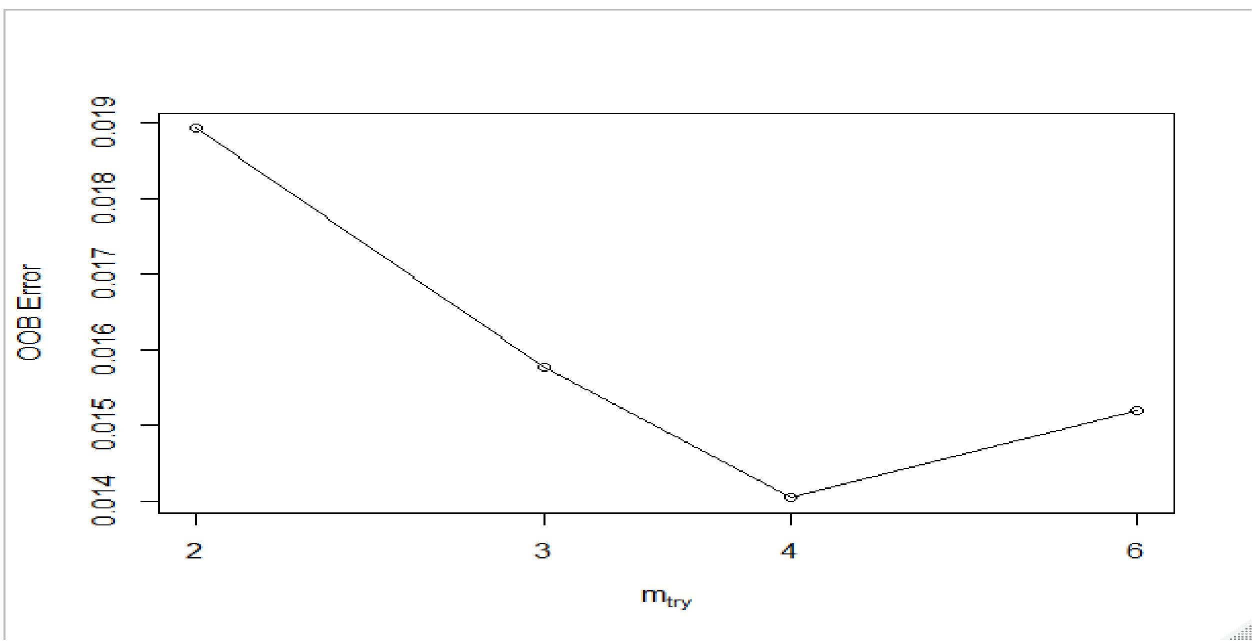


The above is the tree we have got after pruning.

Random Forest:

Here we first find the random forest using a value of m, then we tune this random forest to finf the optimal value of m, where m is the no of variables we use for each tree.

After tuning the random forest, we get m=4 as optimal.



```
> importance(tRndFor,type=1)
                          MeanDecreaseAccuracy
Age..in.years.                           13.64
Experience..in.years.                    13.53
Income..in.K.month.                     140.35
ZIP.Code                                  0.16
Family.members                           81.69
CCAvg                                    34.49
Education                               123.00
Mortgage                                  1.18
Securities.Account                        3.08
CD.Account                               16.58
Online                                    3.84
CreditCard                                7.61
> 
```

The above shows the importance of the various variables in the dataset in the formation of the random forest. Income, education and Family members play the most important role in classifying customers having a higher probability of buying a personal loan.

Check the performance of all the models that you have built (test and train). Use all the model performance measures you have learned so far. Share your remarks on which model performs the best.

CART

Confusion Matrix:

Train Data set:

```
     0     1
0 3146     7
1   40   295
```

Accuracy=0.98

Sensitivity=0.98

Specificity=0.99

Test Data Set:

```
     0     1
0 1345     6
1   23   120
```

Accuracy = 0.98

Sensitivity= 0.95

Specificity= 0.98

Rank Table:

Train Dataset

	deciles	cnt	cnt_tar1	cnt_tar0	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1:	[0.157,1]	517	332	185	64.22	332	185	99	5.9	93
2:	[0.00117,0.157)	2564	3	2561	0.12	335	2746	100	87.1	13
3:	[0,0.00117)	407	0	407	0.00	335	3153	100	100.0	0

KS=0.93

AUC=0.99

Gini = 0.89

Test Dataset

	deciles	cnt	cnt_tar1	cnt_tar0	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1:	[0.157,1]	218	141	77	64.68	141	77	99	5.7	93
2:	[0.00117,0.157)	1084	2	1082	0.18	143	1159	100	85.8	14
3:	[0,0.00117)	192	0	192	0.00	143	1351	100	100.0	0

KS=0.93

AUC=0.99

Gini = 0.89

Random Forest

Train Data Set

	0	1
0	3153	0
1	26	309

Accuracy=0.99

Sensitivity=1

Specificity=0.99

Test Data Set

	0	1
0	1347	4
1	12	131

Accuracy=0.99

Sensitivity=1

Specificity=0.99

Rank Table:

Train Data Set

	deciles	cnt	cnt_tar1	cnt_tar0	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1:	[0.242,1]	349	334	15	95.70	334	15	100	0.48	99
2:	[0.018,0.242)	350	1	349	0.29	335	364	100	11.54	88
3:	[0.004,0.018)	468	0	468	0.00	335	832	100	26.39	74
4:	[0.002,0.004)	338	0	338	0.00	335	1170	100	37.11	63
5:	[0,0.002)	1983	0	1983	0.00	335	3153	100	100.00	0

KS=0.99

AUC=1

Gini=0.89

Test Data Set

	deciles	cnt	cnt_tar1	cnt_tar0	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1:	[0.345,0.998]	150	140	10	93.3	140	10	98	0.74	97
2:	[0.024,0.345)	155	3	152	1.9	143	162	100	11.99	88
3:	[0.008,0.024)	155	0	155	0.0	143	317	100	23.46	77
4:	[0.002,0.008)	312	0	312	0.0	143	629	100	46.56	53
5:	[0,0.002)	722	0	722	0.0	143	1351	100	100.00	0

KS=0.97

AUC=1

Gini=0.88

By the Model performance parameters, we can understand that by using the random forest model, we can better predict the probability of a customer buying a Loan from There Bank since the Sensitvity of the Random Forest model is 100% and accuracy is 99%. AUC is 100% while the first decile das a KS of 99% for Train and 97% for Test. So, we can safely say, Random Forest Model performs better.

APPENDIX

```
setwd("C:\\Users\\Saptarshi Datta\\Desktop\\R PROGRAMMING")
Data=read.csv("Thera Bank.csv", header=TRUE)


Data=Data[,-c(1,5)]

attach(Data)

Data$Mortgage=ifelse(Data$Mortgage>0,1,0)
Data$Experience..in.years.=ifelse(Data$Experience..in.years.<0,0,Data$Experience..in.years.)
Data$Education=as.factor(Data$Education)
Data$Personal.Loan=as.factor(Data$Personal.Loan)
Data$Securities.Account=as.factor(Data$Securities.Account)
Data$CD.Account=as.factor(Data$CD.Account)
Data$Online=as.factor(Data$Online)
Data$CreditCard=as.factor(Data$CreditCard)
##Data$ZIP.Code=as.factor(Data$ZIP.Code)
Data$Mortgage=as.factor(Data$Mortgage)

summary(Data)
str(Data)
is.na(Data)
Data1=na.omit(Data)
Data=Data1


library(funModeling)
library(tidyverse)
library(Hmisc)
```

```
basic_eda <- function(data)
{
  glimpse(data)
  df_status(data)
  freq(data)
  profiling_num(data)
  plot_num(data)
  describe(data)

}
basic_eda(Data)

boxplot(Data)
x=cor(Data[,c(1,2,3,4,5)])
library(corrplot)
corrplot(x,method="number")

library(cluster)
seed=10000
set.seed(seed)
gower_matrix <-daisy(Data,metric="gower")
dist <-gower_matrix
pamxx <-pam(dist,k=3)
sil1 = silhouette(pamxx$clustering,dist)
plot(sil1,col=c("red","blue"))
summary(sil1)
print(pamxx$silinfo$avg.width)
clusplot(Data,pamxx$clustering,color=TRUE,shade=TRUE,label= 2, lines = 1)
Data$cluster=pamxx$clustering
custprofile=aggregate(Data,list(Data$cluster),FUN ="mean")
```

```r
set.seed(seed)
a=rep(0,10)
for(j in 1:100)
{

    pamxx <-pam(dist,j)
    sil = silhouette(pamxx$clustering,dist)
    print(j)
    print(pamxx$silinfo$avg.width)

}
plot(c(1:10),a$avg.width)
```

```r

library(caTools)

split = sample.split(Data$Personal.Loan, SplitRatio = .70)
train  = subset(Data, split == TRUE)
test   = subset(Data, split == FALSE)

table(train$Personal.Loan)

library(rpart)
library(rpart.plot)

tree = rpart(formula = Personal.Loan ~ .,data=train,method="class",cp=0)
rpart.plot(tree)
printcp(tree)
plotcp(tree)

ptree = prune(tree,cp=0.0085,"CP")

##check the updated tree
##plot tree
rpart.plot(ptree)

##print cp value
printcp(ptree)

train$CART.Pred = predict(ptree,data=train,type="class")
train$CART.Score = predict(ptree,data=train,type="prob")[,"1"]
table(train$Personal.Loan,train$CART.Pred)


test$CART.Pred = predict(ptree,test,type="class")
test$CART.Score = predict(ptree,test,type="prob")[,"1"]
table(test$Personal.Loan,test$CART.Pred)

```

```r
#TRAIN data set
probs=seq(0,1,length=11)
qs=quantile(train$CART.Score, probs)
print(qs)
train$deciles=cut(train$CART.Score, unique(qs),include.lowest = TRUE,right=FALSE)
table(train$deciles)

library(data.table)
trainDT=data.table(train)
rankTbl=trainDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(as.numeric(Personal.Loan ==1)),
  cnt_tar0 = sum(as.numeric(Personal.Loan == 0))), by=deciles][order(-deciles)]

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

print(rankTbl)

library(ROCR)
library(ineq)
predobj = prediction(train$CART.Score, train$Personal.Loan)
perf = performance(predobj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
print(KS)
auc = performance(predobj,"auc");
auc = as.numeric(auc@y.values)
print(auc)
gini = ineq(train$CART.Score, type="Gini")
print(gini)

```

```r
#test DATA SET
probs=seq(0,1,length=11)
qs=quantile(test$CART.Score, probs)
print(qs)
test$deciles=cut(test$CART.Score, unique(qs),include.lowest = TRUE,right=FALSE)
table(test$deciles)

library(data.table)
testDT=data.table(test)
rankTbl=testDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(as.numeric(Personal.Loan ==1)),
  cnt_tar0 = sum(as.numeric(Personal.Loan == 0))), by=deciles][order(-deciles)]

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

print(rankTbl)

library(ROCR)
library(ineq)
predobj = prediction(test$CART.Score, test$Personal.Loan)
perf = performance(predobj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
print(KS)
auc = performance(predobj,"auc");
auc = as.numeric(auc@y.values)
print(auc)
gini = ineq(test$CART.Score, type="Gini")
print(gini)
```

```r
library(caTools)
split = sample.split(Data$Personal.Loan, SplitRatio = .70)
train  = subset(Data, split == TRUE)
test   = subset(Data, split == FALSE)


library(randomForest)
seed=1000
set.seed(seed)
rndFor = randomForest(Personal.Loan ~ ., data = train,
                      ntree=501, mtry = 5, nodesize = 10,
                      importance=TRUE)

print(rndFor)
importance(rndFor,type=1)


set.seed(seed)
tRndFor = tuneRF(x = train[,c(-9)],
                 y=train$Personal.Loan,
                 mtryStart = 3,
                 ntreeTry = 501,
                 stepFactor = 1.5,
                 improve = 0.0001,
                 trace=TRUE,
                 plot = TRUE,
                 doBest = TRUE,
                 nodesize = 10,
                 importance=TRUE
)
importance(tRndFor,type=1)

train$predict.class = predict(tRndFor, train, type="class")
train$prob1 = predict(tRndFor, train, type="prob")[,"1"]
tbl=table(train$Personal.Loan, train$predict.class)
print(tbl)
```

```r
test$predict.class = predict(tRndFor, test, type="class")
test$prOb1 = predict(tRndFor, test, type="prob")[,"1"]
tbl=table(test$Personal.Loan, test$predict.class)
print(tbl)

#TRAIN data set
probs=seq(0,1,length=11)
qs=quantile(train$prob1, probs)
print(qs)
train$deciles=cut(train$prob1, unique(qs),include.lowest = TRUE,right=FALSE)
table(train$deciles)

library(data.table)
trainDT=data.table(train)
rankTbl=trainDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(as.numeric(Personal.Loan ==1)),
  cnt_tar0 = sum(as.numeric(Personal.Loan == 0))), by=deciles][order(-deciles)]

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

print(rankTbl)

library(ROCR)
library(ineq)
predObj = prediction(train$prob1, train$Personal.Loan)
perf = performance(predObj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
print(KS)
auc = performance(predObj,"auc");
```

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

#test DATA SET
probs=seq(0,1,length=11)
qs=quantile(test$pr0b1, probs)
print(qs)
test$deciles=cut(test$pr0b1, unique(qs),include.lowest = TRUE,right=FALSE)
table(test$deciles)

library(data.table)
testDT=data.table(test)
rankTbl=testDT[, list(
  cnt = length(Personal.Loan),
  cnt_tar1 = sum(as.numeric(Personal.Loan ==1)),
  cnt_tar0 = sum(as.numeric(Personal.Loan == 0))), by=deciles][order(-deciles)]

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

print(rankTbl)

library(ROCR)
library(ineq)
predObj = prediction(test$pr0b1, test$Personal.Loan)
perf = performance(predObj, "tpr", "fpr")
plot(perf)
KS = max(perf@y.values[[1]]-perf@x.values[[1]])
print(KS)
auc = performance(predObj,"auc");
```

```
auc = as.numeric(auc@y.values)
print(auc)
gini = ineq(test$pr0b1, type="Gini")
print(gini)
```