<u>Project: Thera Bank Case Study</u>

Machine Learning

By: Khaled Majzoub

Table of Contents

Case Study:

Thera Bank - Loan Purchase Modeling

This case is about a bank (Thera Bank) which has a growing customer base. Majority of these customers are liability customers (depositors) with varying size of deposits. The number of customers who are also borrowers (asset customers) is quite small, and the bank is interested in expanding this base rapidly to bring in more loan business and in the process, earn more through the interest on loans. In particular, the management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors). A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with a minimal budget. The department wants to build a model that will help them identify the potential customers who have a higher probability of purchasing the loan. This will increase the success ratio while at the same time reduce the cost of the campaign. The dataset has data on 5000 customers. The data include customer demographic information (age, income, etc.), the customer's relationship with the bank (mortgage, securities account, etc.), and the customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

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

- EDA of the data available. Showcase the results using appropriate graphs - **(10 Marks)**
- Apply appropriate clustering on the data and interpret the output(Thera Bank wants to understand what kind of customers exist in their database and hence we need to do customer segmentation) - **(10 Marks)**
- 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) - **(20 Marks)**
- 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. - **(20 Marks)**

Hint : split <- sample.split(Thera_Bank$Personal Loan, SplitRatio = 0.7)
#we are splitting the data such that we have 70% of the data is Train Data and 30% of the data is my Test Data

1- EDA - Showcase the Results:

```
> setwd("C:/Users/khaled Majzoub/Desktop/R/Machine Learning/Project - Thera B
ank/Final")
> Thera_Bank_Personal_Loan_Modelling_dataset_1 <- read_excel("Thera Bank_Pers
onal_Loan_Modelling-dataset-1.xlsx",
+     sheet = "Bank_Personal_Loan_Modelling")
> View(Thera_Bank_Personal_Loan_Modelling_dataset_1)
bpl = Thera_Bank_Personal_Loan_Modelling_dataset_1
> anyNA(bpl)
[1] TRUE
```

I have to get rid of the NAs

```
> anyNA(bpl_w_na)
[1] FALSE
```

As shown I only lost 18 obs out of 5000. I ll accept the loss.

```
> str(bpl_w_na)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':     4982 obs. of  14 variables:
 $ ID                       : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Age (in years)           : num  25 45 39 35 35 37 53 50 35 34 ...
 $ Experience (in years): num  1 19 15 9 8 13 27 24 10 9 ...
 $ Income (in K/month)      : num  49 34 11 100 45 29 72 22 81 180 ...
 $ ZIP Code                 : num  91107 90089 94720 94112 91330 ...
 $ Family members           : num  4 3 1 1 4 4 2 1 3 1 ...
 $ CCAvg                    : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
 $ Education                : num  1 1 1 2 2 2 2 3 2 3 ...
 $ Mortgage                 : num  0 0 0 0 0 155 0 0 104 0 ...
 $ Personal Loan            : num  0 0 0 0 0 0 0 0 0 1 ...
 $ Securities Account       : num  1 1 0 0 0 0 0 0 0 0 ...
 $ CD Account               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Online                   : num  0 0 0 0 0 1 1 0 1 0 ...
 $ CreditCard               : num  0 0 0 0 1 0 0 1 0 0 ...
 - attr(*, "na.action")= 'omit' Named int  21 59 99 162 236 290 488 722 1461
1462 ...
  ..- attr(*, "names")= chr  "21" "59" "99" "162" ...
```

I have to change variables (Edu,personal loan, Securities loan, CD account , online, Creditcard) to factors

```
> bpl_w_na$Education = as.factor(bpl_w_na$Education)
> bpl_w_na$`Securities Account` = as.factor(bpl_w_na$`Securities Account`)
> bpl_w_na$`CD Account` = as.factor(bpl_w_na$`CD Account`)
> bpl_w_na$Online = as.factor(bpl_w_na$Online)
> bpl_w_na$CreditCard = as.factor(bpl_w_na$CreditCard)
> bpl_w_na$`Personal Loan` = as.factor(bpl_w_na$`Personal Loan`)
> summary(bpl_w_na)
       ID         Age (in years)  Experience (in years) Income (in K/month)
 Min.   :   1    Min.   :23.00   Min.   :-3.0          Min.   :  8.00
 1st Qu.:1254    1st Qu.:35.00   1st Qu.:10.0          1st Qu.: 39.00
 Median :2502    Median :45.00   Median :20.0          Median : 64.00
 Mean   :2502    Mean   :45.33   Mean   :20.1          Mean   : 73.73
 3rd Qu.:3750    3rd Qu.:55.00   3rd Qu.:30.0          3rd Qu.: 98.00
 Max.   :5000    Max.   :67.00   Max.   :43.0          Max.   :224.00
    ZIP Code      Family members       CCAvg          Education    Mortgage
 Min.   : 9307    Min.   :1.000    Min.   : 1.00     1:2088     Min.   :  0.00
```

```
 1st Qu.:91911   1st Qu.:1.000   1st Qu.:  9.00   2:1399    1st Qu.:  0.00
 Median :93437   Median :2.000   Median : 19.00   3:1495    Median :  0.00
 Mean   :93153   Mean   :2.397   Mean   : 25.05             Mean   : 56.55
 3rd Qu.:94608   3rd Qu.:3.000   3rd Qu.: 32.00             3rd Qu.:101.00
 Max.   :96651   Max.   :4.000   Max.   :108.00             Max.   :635.00
 Personal Loan Securities Account CD Account Online   CreditCard
 0:4504        0:4463             0:4682     0:2013   0:3517

 1: 478        1: 519             1: 300     1:2969   1:1465
```

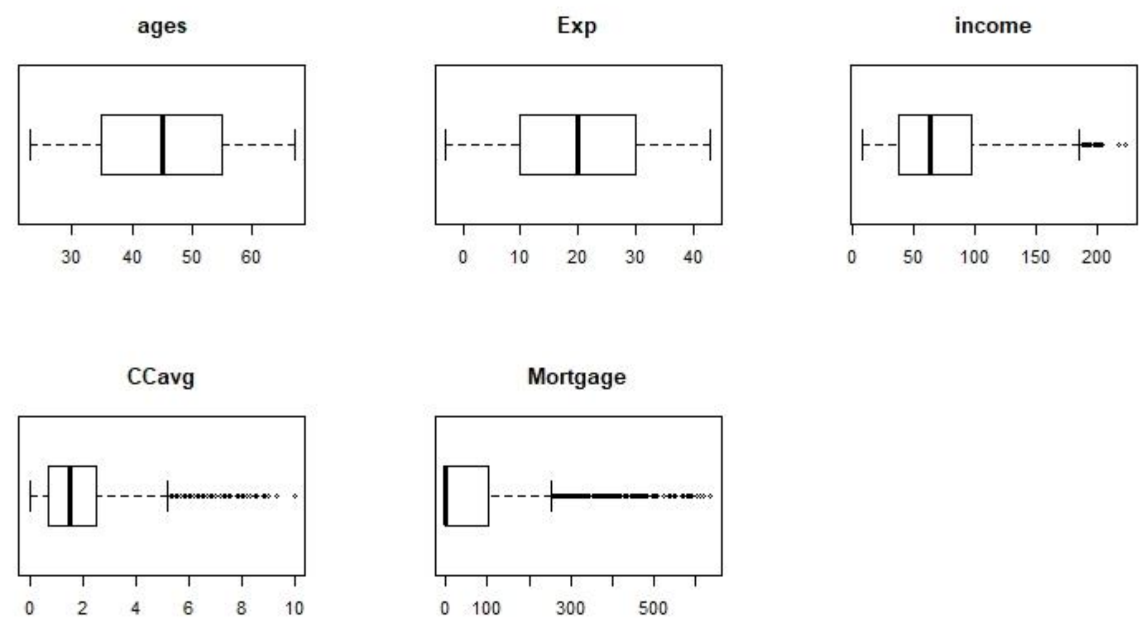
As mentioned in the task there are 478(9.5%) whom they responded in positive on taking the loan from the campaign.

I will remove the ID + ZIP Code + Family Members , because I don't think they will affect the outcomes.

```
> bpl_w_na = bpl_w_na[,-c(1,5,6)]
```

Checking outliers and its effect.

```
library(ggplot2)
> par(mfrow=c(2,3))
> boxplot(bpl_w_na$`Age (in years)`,horizontal = TRUE, main = "ages")
> boxplot(bpl_w_na$`Experience (in years)`,horizontal = TRUE, main = "Exp")
> boxplot(bpl_w_na$`Income (in K/month)`,horizontal = TRUE, main = "income")
> boxplot(bpl_w_na$CCAvg,horizontal = TRUE, main = "CCavg")
> boxplot(bpl_w_na$Mortgage,horizontal = TRUE, main = "Mortgage")
```



Ages looks normal

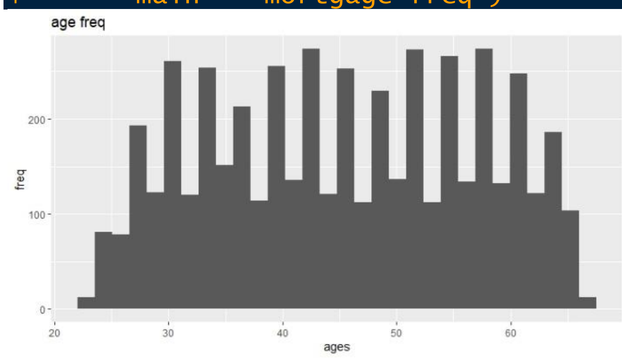
Experience with negative values (I will assume it is a typo and change the minus to +)

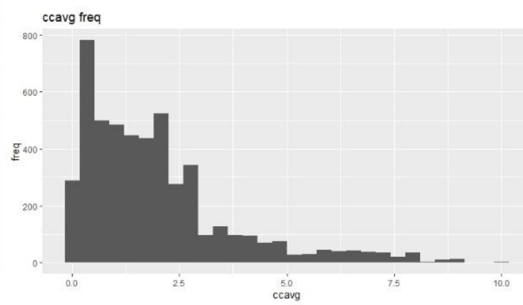
```
> bpl_w_na$`Experience (in years)` = abs(bpl_w_na$`Experience (in years)`)
```

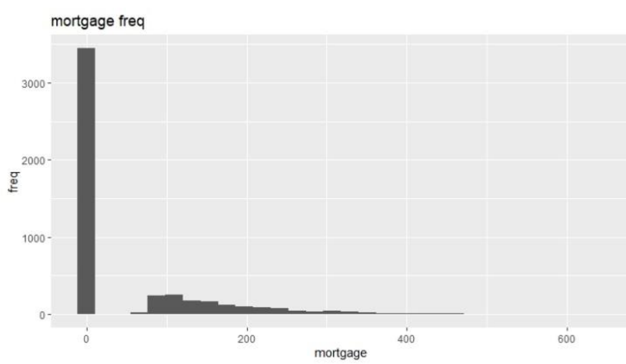
I will keep all the data as is even though there are outliers, but I think excluding them will affect the results , in clustering , CART and the performance.

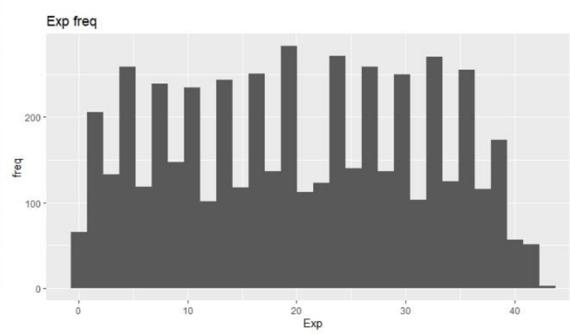
Some Histograms to understand the distribution

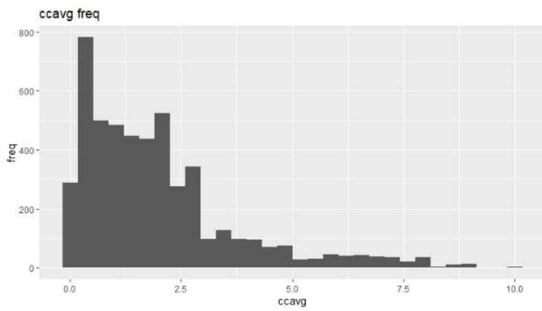
```
> qplot(`Age (in years)`, data = bpl_w_na , geom = "histogram",xlab = "ages"
,ylab = "freq",
+       main = "age freq")
> qplot(`Experience (in years)`, data = bpl_w_na , geom = "histogram",xlab =
"Exp" ,ylab = "freq",
+       main = "Exp freq")
> qplot(`Income (in K/month)`, data = bpl_w_na , geom = "histogram",xlab = "i
ncome" ,ylab = "freq",
+       main = "income freq")
> qplot(CCAvg, data = bpl_w_na , geom = "histogram",xlab = "ccavg" ,ylab = "f
req",
+       main = "ccavg freq")
> qplot(Mortgage, data = bpl_w_na , geom = "histogram",xlab = "mortgage" ,yla
b = "freq",
+       main = "mortgage freq")
```

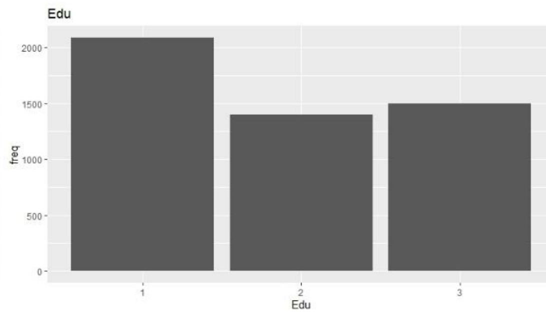


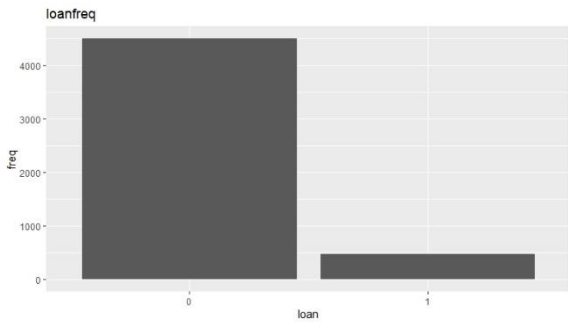












non of the variables are normal distribution

```
> bpl_w_na %>% group_by(`Personal Loan`,Education) %>% summarise (counts = n(
))
# A tibble: 6 x 3
# Groups:   Personal Loan [2]
  `Personal Loan` Education counts
  <fct>           <fct>      <int>
1 0               1           1995
2 0               2           1218
3 0               3           1291
4 1               1             93
5 1               2            181
6 1               3            204
```

You see, level 3 education has the highest number in taking the loan = 204 below them level 2 education and level 1. **This indicates that Education level is related to taking the loans**.

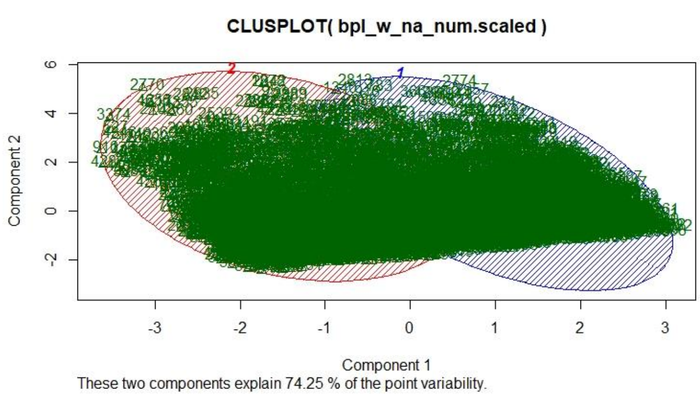
```
> bpl_w_na %>% group_by(`Personal Loan`,Online) %>% summarise (counts = n())
# A tibble: 4 x 3
# Groups:   Personal Loan [2]
  `Personal Loan` Online counts
  <fct>           <fct>   <int>
1 0               0        1824
2 0               1        2680
3 1               0         189
4 1               1         289
```

There is not pattern between loan and online

2- Apply appropriate clustering on the data and interpret the output (Thera Bank wants to understand what kind of customers exist in their database and hence we need to do customer segmentation)

I will use Kmeans Clustering , so I will create a new dataset with numbers only

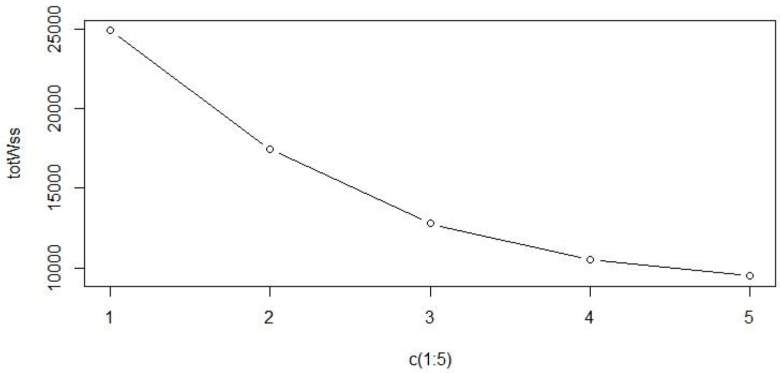
```
> bpl_w_na_num = bpl_w_na[,-c(5,7,8,9,10,11)]
> View(bpl_w_na_num)
> bpl_w_na_num.scaled = scale(bpl_w_na_num)
> bpl_w_na_num.scaled

> seed = 10
> set.seed(seed)
> clust2= kmeans(x=bpl_w_na_num.scaled, centers = 2, nstart = 5)
> clust2
Cluster means:
  Age (in years) Experience (in years) Income (in K/month)      CCAvg    Mort
gage
1      0.8692624             0.8638134          -0.1333010 -0.1449705 -0.0260
7209
2     -0.8458525            -0.8405503           0.1297111  0.1410664  0.0253
6995
Within cluster sum of squares by cluster:
[1] 7519.642 9913.609
 (between_SS / total_SS =  30.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss
"
[6] "betweenss"    "size"         "iter"         "ifault"
> clusplot(bpl_w_na_num.scaled,clust2$cluster,color = TRUE,shade = TRUE,label
s = 2,lines = 1)
```



Obvious 2 clusters needs more , now with choosing the right number of clusters

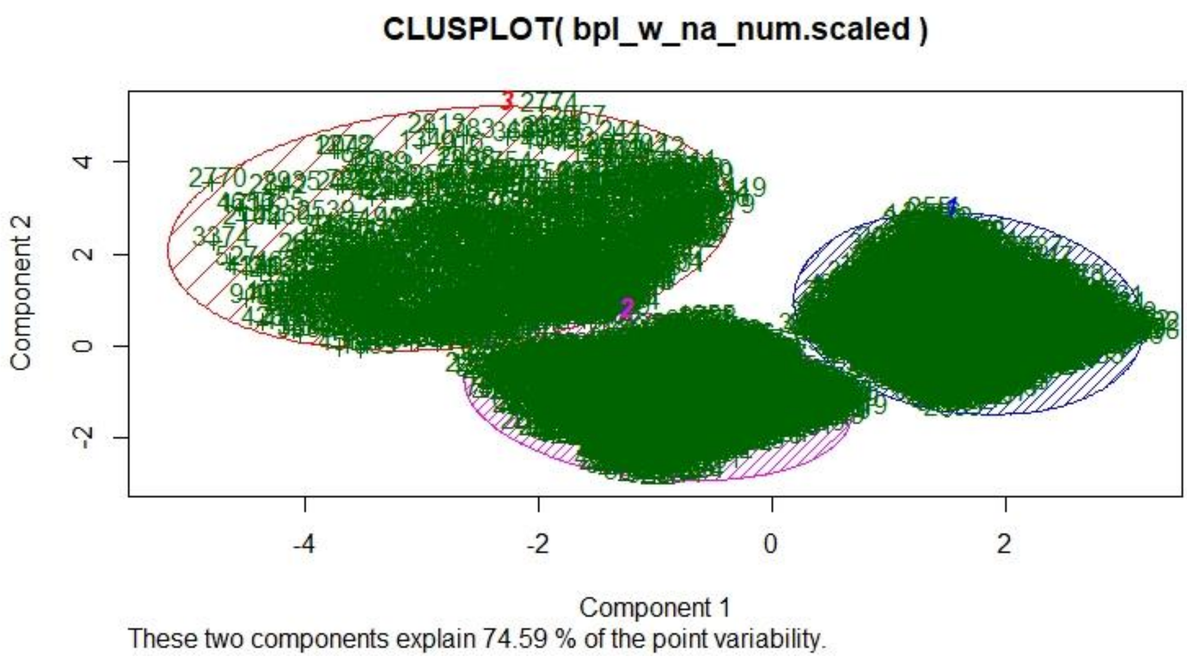
```
> totWss= rep(0,5)
> for(k in 1:5){set.seed(seed)
```

```
+    clust= kmeans(x=bpl_w_na_num.scaled, centers = k, nstart = 5)
+    totWss[k]= clust$tot.withinss}
> plot(c(1:5),totWss,type = "b")
```



###I will pick 3 clusters now from the elbow method

```
> seed = 10
> set.seed(seed)
> clust_final= kmeans(x=bpl_w_na_num.scaled, centers = 3, nstart = 5)
> clusplot(bpl_w_na_num.scaled,clust_final$cluster,color = TRUE,shade = TRUE,
labels = 2,lines = 3)
```



adding the clusters on the data set & profile

```
> bpl_w_na$cluster = clust_final$cluster
> cust_profile = aggregate(bpl_w_na ,list(bpl_w_na$cluster), FUN = "mean")
> cust_profile
```

```
  Group.1 Age (in years) Experience (in years) Income (in K/month)    CCAvg
1       1       43.66543             18.635688           147.64312 4.857906
2       2       35.11078              9.871414            60.09446 1.383264
3       3       55.54482             30.246168            58.83744 1.369066
  Education  Mortgage Personal Loan Securities Account CD Account Online Cred
itCard
1        NA 116.18587            NA                 NA         NA      NA
NA
2        NA  44.88229            NA                 NA         NA      NA
NA
3        NA  45.14631            NA                 NA         NA      NA
NA
  cluster
1       1
2       2
3       3
```

groups of the clusters

```
> head(bpl_w_na)
  Age (in years) Experience (in years) Income (in K/month) CCAvg Education Mo
rtgage
1             25                     1                  49   1.6         1
0
2             45                    19                  34   1.5         1
0
3             39                    15                  11   1.0         1
0
4             35                     9                 100   2.7         2
0
5             35                     8                  45   1.0         2
0
6             37                    13                  29   0.4         2
155
  Personal Loan Securities Account CD Account Online CreditCard cluster
1             0                  1          0      0          0       2
2             0                  1          0      0          0       2
3             0                  0          0      0          0       2
4             0                  0          0      0          0       2
5             0                  0          0      0          1       2
6             0                  0          0      1          0       2
```

I ll make groups now for each Group

```
> group1 = subset(bpl_w_na, cluster == 1)
> View(group1)
> group2 = subset(bpl_w_na, cluster == 2)
> group3 = subset(bpl_w_na, cluster == 3)
```

now the bank have groups for the customers

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)

I will have to make 70% train data , 30% test data

```
> train = subset(bpl_w_na,split==TRUE)
> test = subset(bpl_w_na, split==FALSE)
> seed = 10
> set.seed(seed)
> split = sample.split(bpl_w_na$`Personal Loan`,SplitRatio = 0.7)
> train = subset(bpl_w_na,split==TRUE)
> test = subset(bpl_w_na, split==FALSE)
```

making sure the split is correct , test = 1494 obs , train = 3488 obs

```
> table(train$`Personal Loan`)

   0    1
3153  335
> table(test$`Personal Loan`)

   0    1
1351  143
  > library(rpart, lib.loc = "C:/Program Files/R/R-3.6.1/library")
> library(rpart.plot)
```

CART :we will start with a very complex trees by choosing CP = 0 then we will prune

```
> tree_train_CART = rpart(formula = `Personal Loan` ~.,data = train , method
= "class", minbucket = 3 , cp = 0 )
> tree_test_CART = rpart(formula = `Personal Loan` ~.,data = test , method =
"class", minbucket = 3 , cp = 0 )
> tree_test_CART
> tree_train_CART
```

visualizing the tree

```
> rpart.plot(tree_test_CART)
> rpart.plot(tree_train_CART)
```

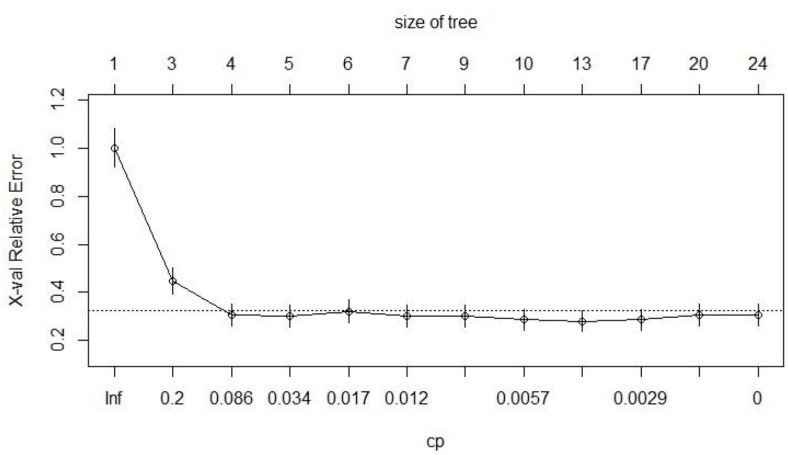
A very complex trees

```
> printcp(tree_test_CART)

Classification tree:
rpart(formula = `Personal Loan` ~ ., data = test, method = "class",
    minbucket = 3, cp = 0)

Variables actually used in tree construction:
[1] Age (in years)      CCAvg               CD Account          Education
[5] Income (in K/month) Mortgage            Online

Root node error: 143/1494 = 0.095716

n= 1494
```

```
           CP nsplit rel error  xerror     xstd
1  0.2902098      0   1.00000 1.00000 0.079521
2  0.1328671      2   0.41958 0.44755 0.054733
3  0.0559441      3   0.28671 0.30769 0.045698
4  0.0209790      4   0.23077 0.30070 0.045191
5  0.0139860      5   0.20979 0.32168 0.046693
6  0.0104895      6   0.19580 0.30070 0.045191
7  0.0069930      8   0.17483 0.30070 0.045191
8  0.0046620      9   0.16783 0.28671 0.044158
9  0.0034965     12   0.15385 0.27972 0.043632
10 0.0023310     16   0.13986 0.28671 0.044158
11 0.0017483     19   0.13287 0.30769 0.045698
12 0.0000000     23   0.12587 0.30769 0.045698
```

here we can see that at nsplit# 12 xerror is at minimum = 0.27972 then it starts to increase

```
> plotcp(tree_test_CART)
```



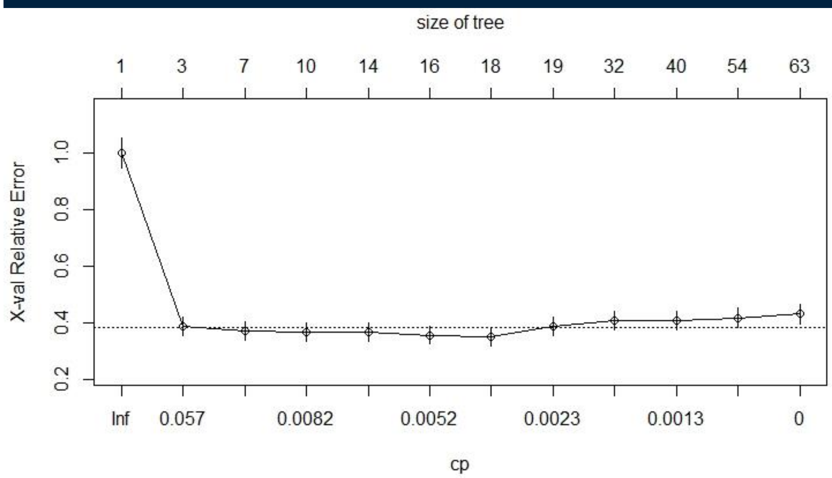
Same for the train data

```
> printcp(tree_train_CART)

Classification tree:
rpart(formula = `Personal Loan` ~ ., data = train, method = "class",
    minbucket = 3, cp = 0)

Variables actually used in tree construction:
 [1] Age (in years)        CCAvg                 CD Account
 [4] CreditCard            Education             Experience (in years)
 [7] Income (in K/month)   Mortgage              Online
[10] Securities Account

Root node error: 335/3488 = 0.096044

n= 3488
```

```
           CP nsplit rel error  xerror     xstd
1  0.30895522      0   1.00000 1.00000 0.051946
2  0.01044776      2   0.38209 0.38806 0.033395
3  0.00895522      6   0.32239 0.37313 0.032771
4  0.00746269      9   0.29552 0.37015 0.032644
5  0.00597015     13   0.26567 0.37015 0.032644
6  0.00447761     15   0.25373 0.35821 0.032132
7  0.00298507     17   0.24478 0.35224 0.031873
8  0.00179104     18   0.24179 0.38806 0.033395
9  0.00149254     31   0.21194 0.40896 0.034246
10 0.00119403     39   0.20000 0.40896 0.034246
11 0.00099502     53   0.17910 0.41791 0.034604
12 0.00000000     62   0.17015 0.43284 0.035190
```

Here we can see that at nsplit# 17 xerror is at minimum = 0.31873 then it starts to increase

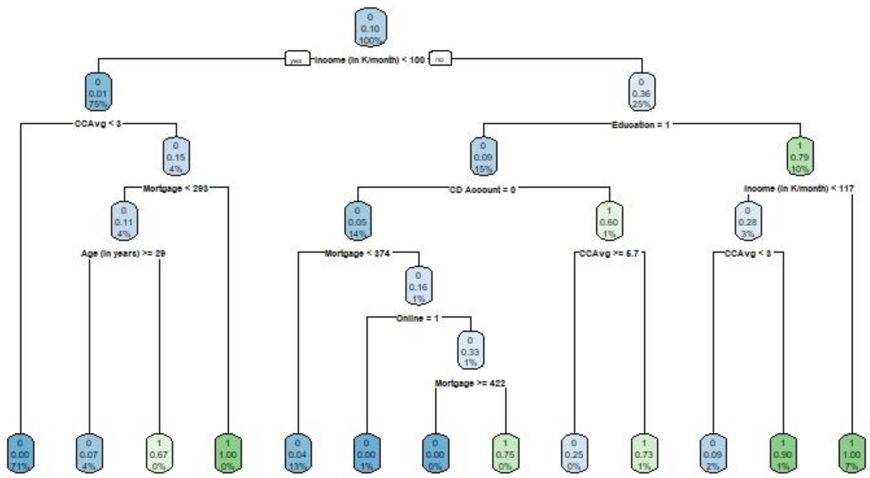
```
> plotcp(tree_train_CART)
```



pruning from the results at cp = 0.004 for test --- cp = 0.003 for train

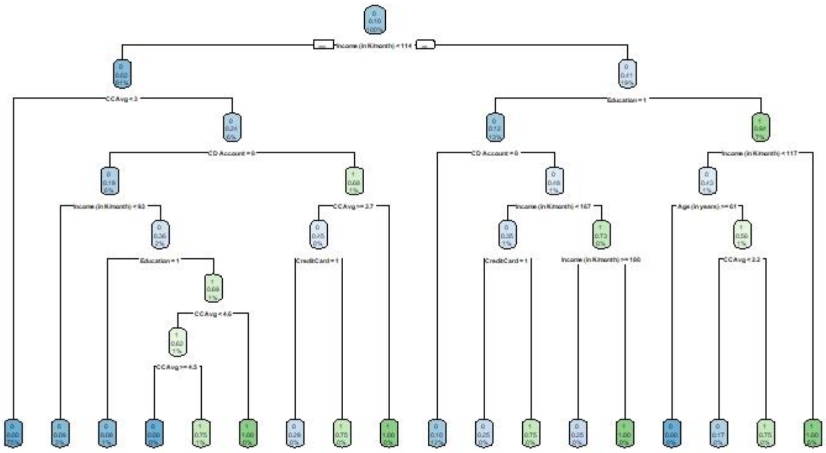
```
> ptree_test = prune(tree_test_CART,cp = 0.004, "CP")
printcp(ptree_test)

Classification tree:
rpart(formula = `Personal Loan` ~ ., data = test, method = "class",
    minbucket = 3, cp = 0)

Variables actually used in tree construction:
[1] Age (in years)      CCAvg               CD Account          Education
[5] Income (in K/month) Mortgage            Online

Root node error: 143/1494 = 0.095716

n= 1494

        CP nsplit rel error  xerror     xstd
1 0.290210      0   1.00000 1.00000 0.079521
2 0.132867      2   0.41958 0.44755 0.054733
```

```
3 0.055944      3   0.28671 0.30769 0.045698
4 0.020979      4   0.23077 0.30070 0.045191
5 0.013986      5   0.20979 0.32168 0.046693
6 0.010490      6   0.19580 0.30070 0.045191
7 0.006993      8   0.17483 0.30070 0.045191
8 0.004662      9   0.16783 0.28671 0.044158
9 0.004000     12   0.15385 0.27972 0.043632
```



This is the prune tree test

```
> ptree_train = prune(tree_train_CART,cp = 0.003, "CP")
> printcp(ptree_train)

Classification tree:
rpart(formula = `Personal Loan` ~ ., data = train, method = "class",
    minbucket = 3, cp = 0)

Variables actually used in tree construction:
[1] Age (in years)      CCAvg               CD Account          CreditCard
[5] Education           Income (in K/month)

Root node error: 335/3488 = 0.096044

n= 3488

         CP nsplit rel error  xerror     xstd
1 0.3089552      0   1.00000 1.00000 0.051946
2 0.0104478      2   0.38209 0.38806 0.033395
3 0.0089552      6   0.32239 0.37313 0.032771
4 0.0074627      9   0.29552 0.37015 0.032644
5 0.0059701     13   0.26567 0.37015 0.032644
6 0.0044776     15   0.25373 0.35821 0.032132
7 0.0030000     17   0.24478 0.35224 0.031873
> rpart.plot(ptree_train)
```



Final train tree

```
> path.rpart(ptree_test, c(4:7))

 node number: 4 
   root
   Income (in K/month)< 99.5
   CCAvg< 2.95

 node number: 5 
   root
   Income (in K/month)< 99.5
   CCAvg>=2.95

 node number: 6 
   root
   Income (in K/month)>=99.5
   Education=1

 node number: 7 
   root
   Income (in K/month)>=99.5
   Education=2,3
path.rpart(ptree_train, c(4:7))

 node number: 4 
   root
   Income (in K/month)< 113.5
   CCAvg< 2.95

 node number: 5 
   root
   Income (in K/month)< 113.5
   CCAvg>=2.95

 node number: 6 
   root
   Income (in K/month)>=113.5
   Education=1
```

```
 node number: 7
   root
   Income (in K/month)>=113.5
   Education=2,3
```

adding the predictions and probabilities

```
> test$prediction = predict(ptree_test, data = test, type = "class")
> test$score = predict(ptree_test, data = test, type = "prob")
> train$prediction = predict(ptree_train, data = test, type = "class")
> train$score = predict(ptree_train, data = test, type = "prob")
> head(test)
   Age (in years) Experience (in years) Income (in K/month) CCAvg Education M
ortgage
19             46                    21                 193   8.1         3
0
23             44                    18                  43   0.7         1
163
24             36                    11                 152   3.9         1
159
26             40                    16                  83   0.2         3
0
31             40                    16                  29   2.0         2
0
33             30                     6                  18   0.9         3
0
   Personal Loan Securities Account CD Account Online CreditCard prediction
19             1                  0          0      0          0          1
23             0                  1          0      0          0          0
24             0                  0          0      0          1          0
26             0                  0          0      0          0          0
31             0                  0          0      1          0          0
33             0                  0          0      0          0          0
      score.0    score.1
19 0.00000000 1.00000000
23 1.00000000 0.00000000
24 0.95767196 0.04232804
26 1.00000000 0.00000000
31 1.00000000 0.00000000
33 1.00000000 0.00000000
```

```
> head(train)
  Age (in years) Experience (in years) Income (in K/month) CCAvg Education Mo
rtgage
1             25                     1                  49   1.6         1
0
2             45                    19                  34   1.5         1
0
3             39                    15                  11   1.0         1
0
4             35                     9                 100   2.7         2
0
5             35                     8                  45   1.0         2
0
6             37                    13                  29   0.4         2
155
  Personal Loan Securities Account CD Account Online CreditCard prediction
1             0                  1          0      0          0          0
2             0                  1          0      0          0          0
3             0                  0          0      0          0          0
4             0                  0          0      0          0          0
```

```
5              0                  0          0      0          1          0
6              0                  0          0      1          0          0
      score.0      score.1
1 0.996155325 0.003844675
2 0.996155325 0.003844675
3 0.996155325 0.003844675
4 0.996155325 0.003844675
5 0.996155325 0.003844675
6 0.996155325 0.003844675
```

Random Forest

an error occurred when I tried to randomforest the test – tried a lot to solve it , asked the academic team , without solutions.. I tried to cancel colums (prob, clust , score) and create a new test_wo_p_c_s and it worked !

The error:

```
> train_rndforest = randomForest( `Personal Loan` ~ ., data = train , ntree=
251 , mtry = 3 , nodesize = 5, importance = TRUE)
Error in eval(predvars, data, env) : object 'Age (in years)' not found
```

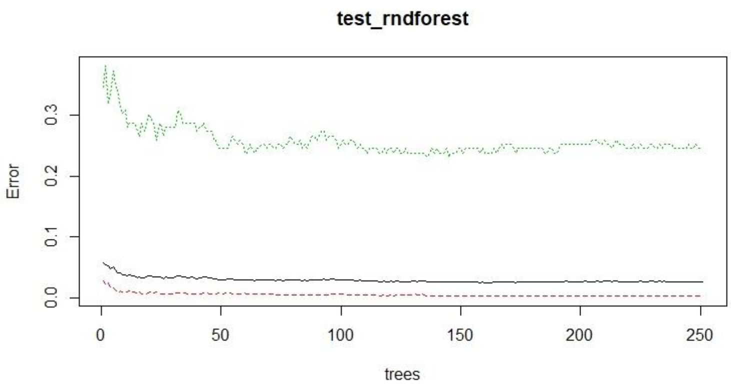
the solution

```
test_w_c_p_s = test[,-c(12,13,14)]
> test_rndforest = randomForest(`Personal Loan`~.,data = test_w_c_p_s , ntree
=251, mtry = 3 , nodesize = 5 , importance = TRUE)
> train_rndforest = randomForest( `Personal Loan` ~ ., data = train_w_c_p_s ,
ntree= 251 , mtry = 3 , nodesize = 5, importance = TRUE)
> print(test_rndforest)

Call:
 randomForest(formula = `Personal Loan` ~ ., data = test_w_c_p_s,      ntree
= 251, mtry = 3, nodesize = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 251
No. of variables tried at each split: 3

        OOB estimate of  error rate: 2.68%
Confusion matrix:
     0   1 class.error
0 1346   5 0.003700962
1   35 108 0.244755245
```

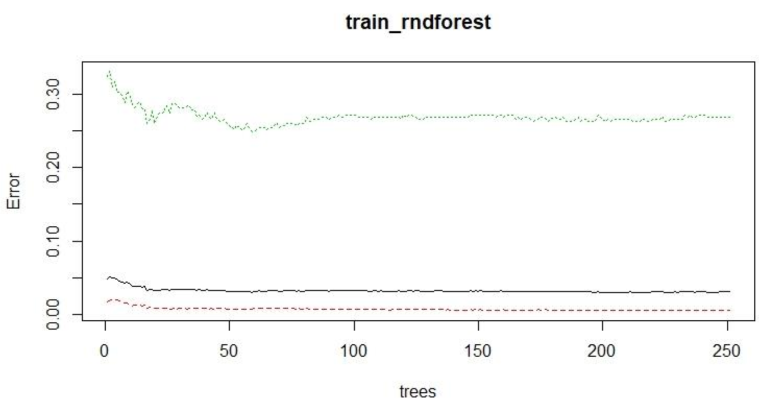
OOB = 2.68% which is low and good ---- class error is also low

```
> print(train_rndforest)

Call:
 randomForest(formula = `Personal Loan` ~ ., data = train_w_c_p_s,      ntree
= 251, mtry = 3, nodesize = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 251
No. of variables tried at each split: 3

        OOB estimate of  error rate: 3.07%
Confusion matrix:
```

```
     0   1 class.error
0 3136  17  0.00539169
1   90 245  0.26865672
```
OOB = 3.07% which is low and good ---- class error is also low

```
plot(test_rndforest)
```


after 55 the error stable

```
plot(train_rndforest)
```


after 40 the error stable

Importance

```
importance(test_rndforest)
                               0          1 MeanDecreaseAccuracy MeanDecrease
Gini
Age (in years)         8.9018411 -1.2507005             8.766160        10.33
6987
Experience (in years)  9.5498776 -1.5933302             8.811758        10.72
4543
Income (in K/month)   44.5349977 41.0203270            51.853742        75.75
0340
```

```
CCAvg                  19.5787057 19.3205019          24.412841        46.63
0213
Education              54.7289933 30.8806237          57.363887        59.58
8181
Mortgage                4.8720419  0.6463121           4.841762        10.15
4636
Securities Account     -0.3335703  2.7163729           1.400055         1.31
4805
CD Account              9.8833828 10.6293316          13.471092        15.52
9474
Online                  0.2866372  2.0518395           1.240431         1.63
2318
CreditCard              5.8833973  0.4173481           5.554369         2.70
6921
```

the 2 most important variables are (education & income) in the test

```
> importance(train_rndforest)
                               0          1 MeanDecreaseAccuracy MeanDecrease
Gini
Age (in years)         13.5319811  0.4504116           13.7175945        24.29
8068
Experience (in years) 13.2280704 -1.3047865           12.8773249        22.24
9928
Income (in K/month)    81.2271683 61.5941416           89.5997478       203.98
2730
CCAvg                  20.7783743 16.8262643           23.2094004        88.80
9618
Education              86.4779604 43.6209767           87.4801560       134.29
1289
Mortgage                1.9382919 -0.0886352            1.9433802        21.96
5813
Securities Account      1.9179783  0.3233846            1.6424755         2.39
0792
CD Account             10.5360977 15.1640029           16.3652688        36.31
3038
Online                  0.7781329 -0.3720187            0.5785617         3.98
7552
CreditCard              4.3799018  4.6772369            6.5304882         4.80
1762
```

the 2 most important variables are (education & income) in the train as well as the test

this means both samples matches

Tuning it keeps give me errors , I couldn't solve

```
 > tuned_test_rndforest = tuneRF(x = test_w_c_p_s, y = test_w_c_p_s$`Personal
Loan`, mtryStart = 3, stepFactor = 1.5, ntreeTry = 55
+                                ,improve = 0.0001 , nodesize=5 , trace = TRUE
, plot = TRUE, doBest = TRUE, importance= TRUE )
mtry = 3  OOB error = 0%
Searching left ...
mtry = 2       OOB error = 0%
NaN 1e-04
Error in if (Improve > improve) { : missing value where TRUE/FALSE needed
> tuned_train_rndforest = tuneRF(x = train_w_c_p_s, y = train_w_c_p_s$`Person
al Loan`, mtryStart = 3, stepFactor = 1.5, ntreeTry = 40
+                                ,improve = 0.0001 , nodesize=5 , trace = TRUE
, plot = TRUE, doBest = TRUE, importance= TRUE )
mtry = 3  OOB error = 0%
Searching left ...
```

```
mtry = 2        OOB error = 0%
NaN 1e-04
Error in if (Improve > improve) { : missing value where TRUE/FALSE needed
```

3 mtry will be assumed as a good number

Prediction

```
> test_w_c_p_s$predict.class = predict(test_rndforest,test_w_c_p_s, type = "c
lass")
> test_w_c_p_s$prob1 = predict(test_rndforest,test_w_c_p_s, type = "prob")[,"
1"]
> head(test_w_c_p_s)
   Age (in years) Experience (in years) Income (in K/month) CCAvg Education M
ortgage
7              53                    27                  72   1.5         2
0
12             29                     5                  45   0.1         2
0
15             67                    41                 112   2.0         1
0
26             43                    19                  29   0.5         1
97
27             40                    16                  83   0.2         3
0
28             46                    20                 158   2.4         1
0
   Personal Loan Securities Account CD Account Online CreditCard predict.clas
s
7              0                  0          0      1          0
0
12             0                  0          0      1          0
0
15             0                  1          0      0          0
0
26             0                  0          0      1          0
0
27             0                  0          0      0          0
0
28             0                  0          0      1          1
0
         prob1
7  0.000000000
12 0.000000000
15 0.023904382
26 0.000000000
27 0.000000000
28 0.007968127
```

```
> train_w_c_p_s$predict.class = predict(train_rndforest,train_w_c_p_s, type =
"class")
> train_w_c_p_s$prob1 = predict(train_rndforest,train_w_c_p_s, type = "prob")
[,"1"]
> head(train_w_c_p_s)
  Age (in years) Experience (in years) Income (in K/month) CCAvg Education Mo
rtgage
1             25                     1                  49   1.6         1
0
2             45                    19                  34   1.5         1
0
```

```
3            39                    15                  11   1.0         1
0
4            35                     9                 100   2.7         2
0
5            35                     8                  45   1.0         2
0
6            37                    13                  29   0.4         2
155
  Personal Loan Securities Account CD Account Online CreditCard predict.class
1             0                  1          0      0          0             0
2             0                  1          0      0          0             0
3             0                  0          0      0          0             0
4             0                  0          0      0          0             0
5             0                  0          0      0          1             0
6             0                  0          0      1          0             0
       prob1
1 0.00000000
2 0.00000000
3 0.00000000
4 0.01593625
5 0.00000000
6 0.00000000
```

Comparison prediction vs loans

```
table(test_w_c_p_s$`Personal Loan`, test_w_c_p_s$predict.class)

       0    1
  0 1351    0
  1   13  130
```

```
> table(train_w_c_p_s$`Personal Loan`, train_w_c_p_s$predict.class)

       0    1
  0 3153    0
  1   40  295
```

a very low rate on both samples

quantiles

```
> test_qs = quantile(test_w_c_p_s$prob1, prob= seq(0,1,length=11))
> test_qs
         0%         10%         20%         30%         40%         50%
60%
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.003
984064
        70%         80%         90%        100%
0.007968127 0.039840637 0.255378486 1.000000000
```

```
train_qs = quantile(train_w_c_p_s$prob1, prob= seq(0,1,length=11))
> train_qs
         0%         10%         20%         30%         40%         50%
60%
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000
000000
        70%         80%         90%        100%
0.007968127 0.035856574 0.302788845 1.000000000
```

error rate

```
> (tbl_test[1,2]+tbl_test[2,1])/nrow(test_w_c_p_s)
[1] 0.008701473
```

```
> (tbl_train[1,2]+tbl_train[2,1])/nrow(train_w_c_p_s)
[1] 0.01146789
```

Both train and test data are close so we don't have an over fitting and the sample is good

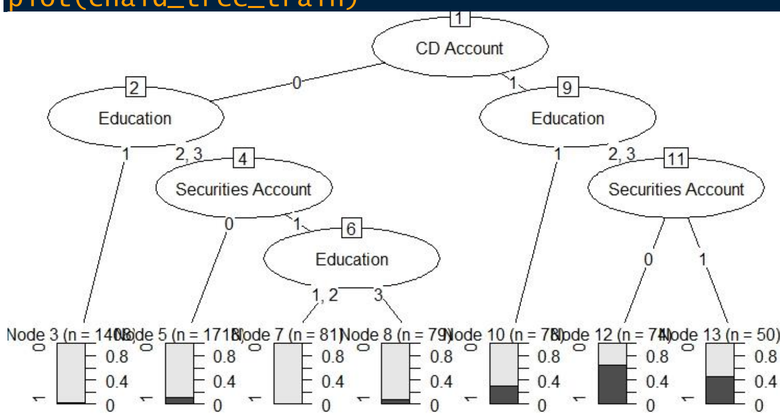
CHAID model

```
> bpl_chaid = bpl_w_na[,-c(1,2,3,4,6,12)]
> str(bpl_chaid)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':     4982 obs. of  6 variables:
 $ Education         : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ..
.
 $ Personal Loan     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
 $ Securities Account: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
 $ CD Account        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Online            : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 1 2 1 ...
 $ CreditCard        : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
> tedu= table(bpl_chaid$`Personal Loan`, bpl_chaid$Education)
> tsl=table(bpl_chaid$`Personal Loan`, bpl_chaid$`Securities Account`)
> tcd=table(bpl_chaid$`Personal Loan`, bpl_chaid$`CD Account`)
> tonline=table(bpl_chaid$`Personal Loan`, bpl_chaid$Online)
> tcc=table(bpl_chaid$`Personal Loan`, bpl_chaid$CreditCard)
> chisq.test(tedu)

        Pearson's Chi-squared test

data:  tedu
X-squared = 109.92, df = 2, p-value < 2.2e-16

> chisq.test(tsl)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tsl
X-squared = 2.3352, df = 1, p-value = 0.1265

> chisq.test(tcd)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tcd
X-squared = 492.24, df = 1, p-value < 2.2e-16

> chisq.test(tonline)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tonline
X-squared = 0.1272, df = 1, p-value = 0.7214
```

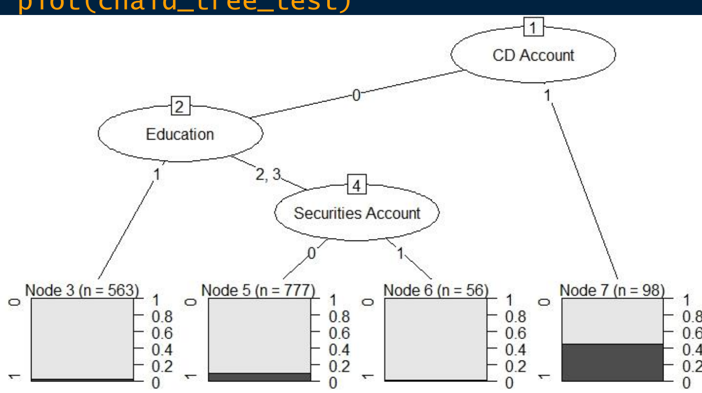
```
> chisq.test(tcc)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tcc
X-squared = 0.0098495, df = 1, p-value = 0.9209
```

####Lowest values are: education – cd account- securities account) will concentrate on

```
> View(bpl_chaid)
> split_chaid = sample.split(bpl_chaid$`Personal Loan`,SplitRatio = 0.7)
> chaid_train_bpl = subset(bpl_chaid,split==TRUE)
> chaid_test_bpl = subset(bpl_chaid,split==FALSE)
> library(CHAID)
> chaid.cntrl = chaid_control(minbucket = 30, minsplit = 100, alpha2 = .05,al
pha4 = 0.05)
> chaid_tree_train = chaid(`Personal Loan` ~Education + `CD Account`+ `Securi
ties Account`,
+                     data = chaid_train_bpl , control = chaid.cntrl)
plot(chaid_tree_train)
```




```
> chaid_tree_test = chaid(`Personal Loan` ~Education + `CD Account`+ `Securit
ies Account`,
+                             data = chaid_test_bpl , control = chaid.cntrl)
> plot(chaid_tree_test)
```



there is some differences , will know which one is better in performance

4- 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

we will use the following data to measure the performance (test - train – test_w_c_p_s – train_w_c_p_s – chaid_test – chaid_train)

Error rate

```
> (forest_tbl_test[1,2]+forest_tbl_test[2,1])/nrow(test_w_c_p_s)
[1] 0.008701473
> (forest_tbl_train[1,2]+forest_tbl_train[2,1])/nrow(train_w_c_p_s)
[1] 0.01146789
```

```
> (cart_tbl_test[1,2]+cart_tbl_test[2,1])/nrow(test)
[1] 0.01807229
> (cart_tbl_train[1,2]+cart_tbl_train[2,1])/nrow(train)
[1] 0.01978211
```

both are close accuracy rate = 1- the error

Rank ordering table

```
> cart_tbl_test = table(test$`Personal Loan`, test$prediction)
> cart_tbl_train = table(train$`Personal Loan`, train$prediction)
> (cart_tbl_test[1,2]+cart_tbl_test[2,1])/nrow(test)
[1] 0.01807229
> (cart_tbl_train[1,2]+cart_tbl_train[2,1])/nrow(train)
[1] 0.01978211
> probs=seq(0,1,length=11)
> forest_test_qs = quantile(test$score, probs)
> forest_train_qs = quantile(train$score, probs)
> forest_test_qs
        0%        10%        20%        30%        40%        50%        60% 
0.00000000 0.00000000 0.00000000 0.00000000 0.06161137 0.50000000 0.93838863 
       70%        80%        90%       100% 
1.00000000 1.00000000 1.00000000 1.00000000 
> forest_train_qs
        0%        10%        20%        30%        40%        50%        60% 
0.00000000 0.00000000 0.00000000 0.00000000 0.01754386 0.50000000 0.98245614 
       70%        80%        90%       100% 
1.00000000 1.00000000 1.00000000 1.00000000 
```

```
> cart_test_qs = quantile(test_w_c_p_s$prob1, probs)
> cart_train_qs = quantile(train_w_c_p_s$prob1, probs)
> cart_test_qs
         0%         10%         20%         30%         40%         50%
60%
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.003
984064
        70%         80%         90%        100%
0.007968127 0.039840637 0.255378486 1.000000000
> cart_train_qs
```

```
            0%           10%           20%           30%           40%           50%
60%
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000
000000
          70%           80%           90%          100%
0.007968127 0.035856574 0.302788845 1.000000000
```
both numbers are close

deciles

```
> test$deciles = cut(test$score, unique(forest_test_qs), include.lowest = TRU
E)
Error in `$<-.data.frame`(`*tmp*`, deciles, value = c(4L, 4L, 3L, 4L,  :
  replacement has 2988 rows, data has 1494
> train$deciles = cut(train$score, unique(forest_train_qs), include.lowest =
TRUE)
Error in `$<-.data.frame`(`*tmp*`, deciles, value = c(4L, 4L, 4L, 4L,  :
  replacement has 6976 rows, data has 3488
```
it didn't work on the forest – I don't know why , I ll try different approach

```
> test_w_c_p_s$deciles = cut(test_w_c_p_s$prob1, unique(cart_test_qs), includ
e.lowest = TRUE)
> train_w_c_p_s$deciles = cut(train_w_c_p_s$prob1, unique(cart_train_qs), inc
lude.lowest = TRUE)
```
it worked on the **CART**. Below all CART performance

```
> library(data.table)
> cart_test_dt=data.table(test_w_c_p_s)
> cart_train_dt=data.table(train_w_c_p_s)
> test_rank_table = cart_test_dt[,list(cnt=length(`Personal Loan`)), by=decil
es][order(-deciles)]
> train_rank_table = cart_train_dt[,list(cnt=length(`Personal Loan`)), by=dec
iles][order(-deciles)]
> test_rank_table = cart_test_dt[,list(cnt=length(`Personal Loan`),
+                                     cnt_tar1=sum(`Personal Loan`==1),
+                                     cnt_tar0=sum(`Personal Loan`==0)), by=
deciles][order(-deciles)]
> train_rank_table = cart_train_dt[,list(cnt=length(`Personal Loan`),
+                                     cnt_tar1=sum(`Personal Loan`==1),
+                                     cnt_tar0=sum(`Personal Loan`==0)), by=
deciles][order(-deciles)]
> test_rank_table
               deciles  cnt cnt_tar1 cnt_tar0
1:           (0.255,1]  150      143        7
2:      (0.0398,0.255]  140        0      140
3:    (0.00797,0.0398]  148        0      148
4:   (0.00398,0.00797]   53        0       53
5:         [0,0.00398] 1003        0     1003
> train_rank_table
               deciles  cnt cnt_tar1 cnt_tar0
1:           (0.303,1]  348      332       16
2:      (0.0359,0.303]  335        3      332
3:    (0.00797,0.0359]  315        0      315
4:         [0,0.00797] 2490        0     2490
> test_rank_table$rrate = round(test_rank_table$cnt_tar1/test_rank_table$cnt,
4)*100
> test_rank_table
               deciles  cnt cnt_tar1 cnt_tar0 rrate
1:           (0.255,1]  150      143        7 95.33
```

```
2:     (0.0398,0.255]  140         0       140  0.00
3:   (0.00797,0.0398]  148         0       148  0.00
4: (0.00398,0.00797]   53         0        53  0.00
5:       [0,0.00398] 1003         0      1003  0.00
> train_rank_table$rrate = round(train_rank_table$cnt_tar1/train_rank_table$c
nt,4)*100
> train_rank_table
              deciles  cnt cnt_tar1 cnt_tar0 rrate
1:          (0.303,1]  348      332       16  95.4
2:    (0.0359,0.303]  335        3      332   0.9
3: (0.00797,0.0359]  315        0      315   0.0
4:       [0,0.00797] 2490        0     2490   0.0
```

Cumulative response rate

```
> test_rank_table$cum_resp = cumsum(test_rank_table$cnt_tar1)
> train_rank_table$cum_resp = cumsum(train_rank_table$cnt_tar1)
> test_rank_table$cum_non_resp = cumsum(test_rank_table$cnt_tar0)
> train_rank_table$cum_non_resp = cumsum(train_rank_table$cnt_tar0)
> test_rank_table
             deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp
1:          (0.255,1]  150      143        7 95.33      143            7
2:     (0.0398,0.255]  140        0      140  0.00      143          147
3:   (0.00797,0.0398]  148        0      148  0.00      143          295
4: (0.00398,0.00797]   53        0       53  0.00      143          348
5:       [0,0.00398] 1003        0     1003  0.00      143         1351
> train_rank_table
              deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp
1:          (0.303,1]  348      332       16  95.4      332           16
2:    (0.0359,0.303]  335        3      332   0.9      335          348
3: (0.00797,0.0359]  315        0      315   0.0      335          663
4:       [0,0.00797] 2490        0     2490   0.0      335         3153
```

KS

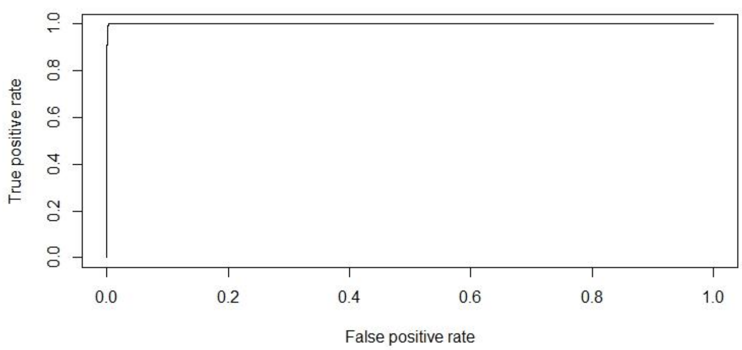
```
> test_rank_table$cum_rel_resp = round(test_rank_table$cum_resp/sum(test_rank
_table$cnt_tar1),4)*100
> test_rank_table$cum_rel_non_resp = round(test_rank_table$cum_non_resp/sum(t
est_rank_table$cnt_tar0),4)*100
> train_rank_table$cum_rel_resp = round(train_rank_table$cum_resp/sum(train_r
ank_table$cnt_tar1),4)*100
> train_rank_table$cum_rel_non_resp = round(train_rank_table$cum_non_resp/sum
(train_rank_table$cnt_tar0),4)*100
> test_rank_table
             deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_r
el_resp
1:          (0.255,1]  150      143        7 95.33      143            7
100
2:     (0.0398,0.255]  140        0      140  0.00      143          147
100
3:   (0.00797,0.0398]  148        0      148  0.00      143          295
100
4: (0.00398,0.00797]   53        0       53  0.00      143          348
100
5:       [0,0.00398] 1003        0     1003  0.00      143         1351
100
   cum_rel_non_resp
1:             0.52
```

```
2:             10.88
3:             21.84
4:             25.76
5:            100.00
> train_rank_table
            deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_re
l_resp
1:        (0.303,1]  348      332       16  95.4      332           16
99.1
2:   (0.0359,0.303]  335        3      332   0.9      335          348
100.0
3: (0.00797,0.0359]  315        0      315   0.0      335          663
100.0
4:      [0,0.00797] 2490        0     2490   0.0      335         3153
100.0
   cum_rel_non_resp
1:             0.51
2:            11.04
3:            21.03
4:           100.00
> test_rank_table$ks = abs(test_rank_table$cum_rel_resp - test_rank_table$cum
_rel_non_resp)
> train_rank_table$ks = abs(train_rank_table$cum_rel_resp - train_rank_table$
cum_rel_non_resp)
> test_rank_table
             deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_r
el_resp
1:         (0.255,1]  150      143        7 95.33      143            7
100
2:    (0.0398,0.255]  140        0      140  0.00      143          147
100
3:  (0.00797,0.0398]  148        0      148  0.00      143          295
100
4: (0.00398,0.00797]   53        0       53  0.00      143          348
100
5:       [0,0.00398] 1003        0     1003  0.00      143         1351
100
   cum_rel_non_resp    ks
1:             0.52 99.48
2:            10.88 89.12
3:            21.84 78.16
4:            25.76 74.24
5:           100.00  0.00
> train_rank_table
            deciles  cnt cnt_tar1 cnt_tar0 rrate cum_resp cum_non_resp cum_re
l_resp
1:        (0.303,1]  348      332       16  95.4      332           16
99.1
2:   (0.0359,0.303]  335        3      332   0.9      335          348
100.0
3: (0.00797,0.0359]  315        0      315   0.0      335          663
100.0
4:      [0,0.00797] 2490        0     2490   0.0      335         3153
100.0
   cum_rel_non_resp    ks
1:             0.51 98.59
2:            11.04 88.96
3:            21.03 78.97
4:           100.00  0.00
```

both data have almost same numbers, and shows were to target exactly

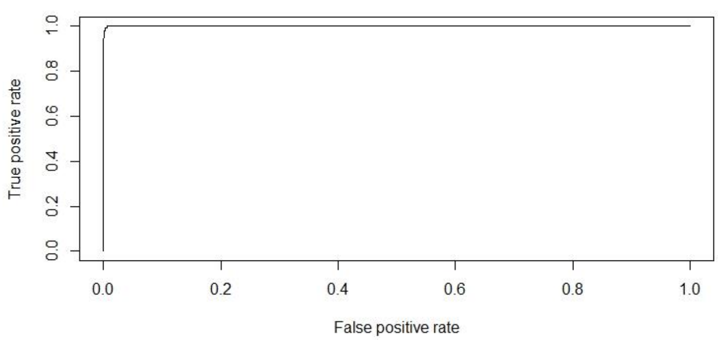
ROCR

```
> library(ROCR)
> library(ineq)
> test_predobj = prediction(test_w_c_p_s$prob1, test_w_c_p_s$`Personal Loan`)
> test_perf = performance(test_predobj, "tpr", "fpr")
> plot(test_perf)
```



it is almost = 1

```
> train_predobj = prediction(train_w_c_p_s$prob1, train_w_c_p_s$`Personal Loa
n`)
> train_perf = performance(train_predobj, "tpr", "fpr")
> plot(train_perf)
```



both are almost identical

```
> test_ks=max(test_perf@y.values[[1]]-test_perf@x.values[[1]])
> test_ks
[1] 0.9970392
> train_ks=max(train_perf@y.values[[1]]-train_perf@x.values[[1]])
> train_ks
[1] 0.993974
```

AUC

```
> train_auc= performance(train_predobj, "auc")
> train_auc= as.numeric(train_auc@y.values)
> train_auc
```

```
[1] 0.9998622
> test_auc= performance(test_predobj, "auc")
> test_auc= as.numeric(test_auc@y.values)
> test_auc
[1] 0.9998887
```

Gini

```
> train_gini = ineq(train_w_c_p_s$prob1, "gini")
> test_gini = ineq(test_w_c_p_s$prob1, "gini")
> train_gini
[1] 0.8853921
> test_gini
[1] 0.8828866
```

Concordance

```
> Concordance(actuals = test_w_c_p_s$`Personal Loan`,predictedScores = test_w
_c_p_s$prob1)
$Concordance
[1] 0.9998861

$Discordance
[1] 0.0001138758

$Tied
[1] 4.676977e-17

$Pairs
[1] 193193

> Concordance(actuals = train_w_c_p_s$`Personal Loan`,predictedScores = train
_w_c_p_s$prob1)
$Concordance
[1] 0.999857

$Discordance
[1] 0.0001429579

$Tied
[1] 2.604796e-17

$Pairs
```

Performance of the **Forrest**

```
> rfpredobjtrain = prediction(train$score , train$`Personal Loan`)
Error in prediction(train$score, train$`Personal Loan`) :
  Number of cross-validation runs must be equal for predictions and labels.
```

I couldn't find the solution

I cant find KS and AUC due to the error

Gini

```
> ineq(train$score, "gini")
[1] 0.4967223
> ineq(test$score, "gini")
[1] 0.4969288
```

concordance

```
> Concordance(actuals = train$`Personal Loan`, predictedScores = train$score)
$Concordance
[1] 0.007168723

$Discordance
[1] 0.9928313

$Tied
[1] 0

$Pairs
[1] 1056255
> Concordance(actuals = test$`Personal Loan`, predictedScores = test$score)
$Concordance
[1] 0.00685325

$Discordance
[1] 0.9931467

$Tied
[1] 0

$Pairs
[1] 193193
```

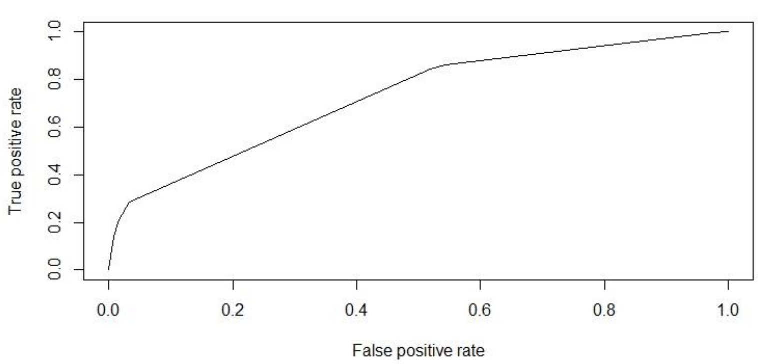
both numbers are close

CHAID performance

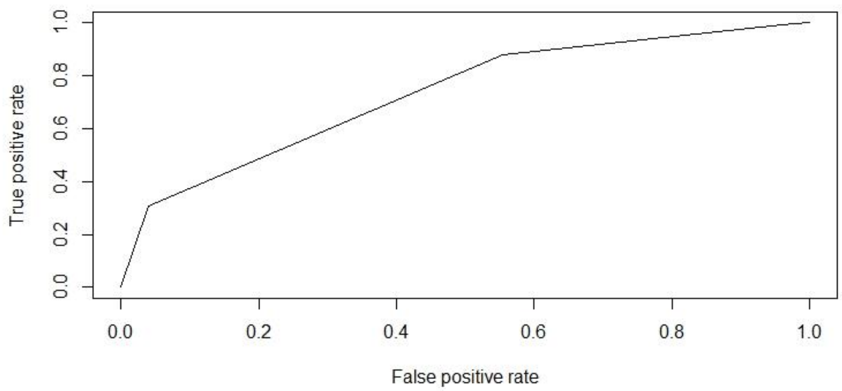
```
> chaid_test_bpl$chaid.pred = predict(chaid_tree_test, data=chaid_test_bpl, t
ype="response")
> chaid_test_bpl$chaid.score = predict(chaid_tree_test, data=chaid_test_bpl,
type="prob")[,"1"]
> chaid_train_bpl$chaid.pred = predict(chaid_tree_train, data=chaid_train_bpl
, type="response")
> chaid_train_bpl$chaid.score = predict(chaid_tree_train, data=chaid_train_bp
l, type="prob")[,"1"]
> chaid_test_cm = table(chaid_test_bpl$`Personal Loan`,chaid_test_bpl$chaid.p
red)
> chaid_train_cm = table(chaid_train_bpl$`Personal Loan`,chaid_train_bpl$chai
d.pred)
> chaid_test_cm

       0    1
  0 1351    0
  1  143    0
> chaid_train_cm

       0    1
  0 3127   26
  1  287   48
(chaid_test_cm[1,2]+chaid_test_cm[2,1])/nrow(chaid_test_bpl)
[1] 0.0957162
> (chaid_train_cm[1,2]+chaid_train_cm[2,1])/nrow(chaid_train_bpl)
[1] 0.08973624
```

error rate is pretty close ### accuracy is 1-

```
> chaidpredobjtrain = prediction(chaid_train_bpl$chaid.score,chaid_train_bpl$
`Personal Loan`)
> chaidpreftrain= performance(chaidpredobjtrain,"tpr","fpr")
> plot(chaidpreftrain)
```




```
> chaidpredobjtest = prediction(chaid_test_bpl$chaid.score,chaid_test_bpl$`Pe
rsonal Loan`)
> chaidpreftest= performance(chaidpredobjtest,"tpr","fpr")
> plot(chaidpreftest)
```



the difference is minimal

#KS + AUC

```
> max(chaidpreftrain@y.values[[1]]-chaidpreftrain@x.values[[1]])
[1] 0.325588
> max(chaidpreftest@y.values[[1]]-chaidpreftest@x.values[[1]])
[1] 0.3267147
> chaid_test_auc=performance(chaidpredobjtest,"auc")
> as.numeric(chaid_test_auc@y.values)
[1] 0.7319028
> chaid_train_auc=performance(chaidpredobjtrain,"auc")
> as.numeric(chaid_train_auc@y.values)
[1] 0.7271871
```

##gini

```
> ineq(chaid_test_bpl$chaid.score, "gini")
[1] 0.4194119
> ineq(chaid_train_bpl$chaid.score, "gini")
[1] 0.4107345
```

it is not pure at all

concordance

```
> Concordance(actuals = chaid_test_bpl$`Personal Loan`,predictedScores = chai
d_test_bpl$chaid.score)
$Concordance
[1] 0.5554653

$Discordance
[1] 0.4445347

$Tied
[1] 5.551115e-17

$Pairs
[1] 193193
```

```
> Concordance(actuals = chaid_train_bpl$`Personal Loan`,predictedScores = cha
id_train_bpl$chaid.score)
$Concordance
[1] 0.5602492

$Discordance
[1] 0.4397508

$Tied
[1] 0

$Pairs
[1] 1056255
```
the numbers are not good , almost 50% is error

Conclusion

Both models performance are good , the bank can make the marketing campaign as per the clusters with prediction error rate below 5% , so it will be effective because we know the target customers.