Import the data file

```
### Import dataset
Bank_data <- read.csv(file="/Users/rsklanu/Cart_assignment/PL_XSELL.csv", header=TRUE)
```

Perform univariate analysis and understand data

```
summary(Bank_data)
sapply(Bank_data,function(x){sum(is.na(x))})
str(Bank_data)
```

```
   CUST_ID          TARGET            AGE          GENDER       BALANCE            OCCUPATION
C1      :    1   Min.   :0.0000   Min.   :21.00   F: 5433   Min.   :      0   PROF     :5417
C10     :    1   1st Qu.:0.0000   1st Qu.:32.00   M:14376   1st Qu.:  64754   SAL      :5855
C100    :    1   Median :0.0000   Median :38.00   O:  191   Median : 231676   SELF-EMP:3568
C1000   :    1   Mean   :0.1256   Mean   :38.42             Mean   : 511362   SENP     :5160
C10000  :    1   3rd Qu.:0.0000   3rd Qu.:46.00             3rd Qu.: 653677
C10001  :    1   Max.   :1.0000   Max.   :55.00             Max.   :8960141
(Other):19994
 AGE_BKT          SCR          HOLDING_PERIOD  ACC_TYPE       ACC_OP_DATE    LEN_OF_RLTN_IN_MNTH
<25  :1753   Min.   :100.0   Min.   : 1.00   CA: 4241   11/30/2010:   24   Min.   : 29.0
>50  :3835   1st Qu.:227.0   1st Qu.: 7.00   SA:15750   04-03-2009:   23   1st Qu.: 79.0
26-30:3434   Median :364.0   Median :15.00              7/25/2010 :   22   Median :125.0
31-35:3484   Mean   :448.2   Mean   :14.96              05-06-2013:   21   Mean   :125.2
36-40:2814   3rd Qu.:644.0   3rd Qu.:22.00              02-07-2007:   20   3rd Qu.:172.0
41-45:3267   Max.   :999.0   Max.   :31.00              6/24/2010 :   20   Max.   :221.0
46-50:2483                                              (Other)   :19870
NO_OF_I_CR_TXNS NO_OF_I_DR_TXNS  TOT_NO_OF_I_TXNS NO_OF_BR_CSH_WDL_DR_TXNS NO_OF_ATM_DR_TXNS
Min.   : 0.00   Min.   : 0.000   Min.   :  0.00   Min.   : 0.000          Min.   : 0.000
1st Qu.: 6.00   1st Qu.: 2.000   1st Qu.:  9.00   1st Qu.: 1.000          1st Qu.: 0.000
Median :10.00   Median : 5.000   Median : 14.00   Median : 1.000          Median : 1.000
Mean   :12.55   Mean   : 6.884   Mean   : 18.98   Mean   : 1.883          Mean   : 1.029
3rd Qu.:14.00   3rd Qu.: 7.000   3rd Qu.: 21.00   3rd Qu.: 2.000          3rd Qu.: 1.000
Max.   :75.00   Max.   :74.000   Max.   :149.00   Max.   :15.000          Max.   :25.000

NO_OF_NET_DR_TXNS NO_OF_MOB_DR_TXNS NO_OF_CHQ_DR_TXNS   FLG_HAS_CC       ANT_ATM_DR
Min.   : 0.000    Min.   : 0.0000   Min.   : 0.000    Min.   :0.0000   Min.   :     0
1st Qu.: 0.000    1st Qu.: 0.0000   1st Qu.: 0.000    1st Qu.:0.0000   1st Qu.:     0
Median : 0.000    Median : 0.0000   Median : 2.000    Median :0.0000   Median :  6000
Mean   : 1.172    Mean   : 0.4118   Mean   : 2.138    Mean   :0.3054   Mean   : 10880
3rd Qu.: 1.000    3rd Qu.: 0.0000   3rd Qu.: 4.000    3rd Qu.:1.0000   3rd Qu.: 15800
Max.   :22.000    Max.   :25.0000   Max.   :75.000    Max.   :1.0000   Max.   :799400

ANT_BR_CSH_WDL_DR   ANT_CHQ_DR       ANT_NET_DR       ANT_MOB_DR       ANT_L_DR        FLG_HAS_ANY_CHGS
Min.   :     0    Min.   :      0   Min.   :     0   Min.   :     0   Min.   :      0   Min.   :0.0000
1st Qu.:  2930    1st Qu.:      0   1st Qu.:     0   1st Qu.:     0   1st Qu.: 237926   1st Qu.:0.0000
Median :340150    Median :  23640   Median :     0   Median :     0   Median : 605115   Median :0.0000
Mean   :378474    Mean   : 124528   Mean   :237388   Mean   : 22425   Mean   : 773717   Mean   :0.1106
3rd Qu.:674675    3rd Qu.:  72470   3rd Qu.:473370   3rd Qu.:     0   3rd Qu.:1078327   3rd Qu.:0.0000
Max.   :999998    Max.   :4378648   Max.   :999854   Max.   :199667   Max.   :6514071   Max.   :1.0000
```

```
AMT_OTH_BK_ATM_USG_CHGS AMT_MIN_BAL_NMC_CHGS NO_OF_IW_CHQ_BNC_TXNS NO_OF_OW_CHQ_BNC_TXNS
Min.   :  0.000         Min.   :  0.000       Min.   :0.00000       Min.   :0.0000
1st Qu.:  0.000         1st Qu.:  0.000       1st Qu.:0.00000       1st Qu.:0.0000
Median :  0.000         Median :  0.000       Median :0.00000       Median :0.0000
Mean   :  1.099         Mean   :  1.292       Mean   :0.00275       Mean   :0.0044
3rd Qu.:  0.000         3rd Qu.:  0.000       3rd Qu.:0.00000       3rd Qu.:0.0000
Max.   :250.000         Max.   :170.000       Max.   :2.00000       Max.   :2.0000

AVG_AMT_PER_ATM_TXN AVG_AMT_PER_CSH_WDL_TXN AVG_AMT_PER_CHQ_TXN AVG_AMT_PER_NET_TXN AVG_AMT_PER_MOB_TXN
Min.   :    0       Min.   :     0          Min.   :     0       Min.   :     0       Min.   :     0
1st Qu.:    0       1st Qu.:  1266          1st Qu.:     0       1st Qu.:     0       1st Qu.:     0
Median : 6000       Median :147095          Median :  8545       Median :     0       Median :     0
Mean   : 7489       Mean   :242736          Mean   : 25897       Mean   :179839       Mean   : 28384
3rd Qu.:13500       3rd Qu.:385000          3rd Qu.: 28825       3rd Qu.:257899       3rd Qu.:     0
Max.   :25000       Max.   :999640          Max.   :557842       Max.   :999834       Max.   :199867

FLG_HAS_NOMINEE  FLG_HAS_OLD_LOAN      random
Min.   :0.0000   Min.   :0.0000   Min.   :0.0000114
1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.2481866
Median :1.0000   Median :0.0000   Median :0.5061214
Mean   :0.9812   Mean   :0.4929   Mean   :0.5019330
3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.7535712
Max.   :1.0000   Max.   :1.0000   Max.   :0.9999471
```

```
'data.frame':   20000 obs. of  40 variables:
 $ CUST_ID                : Factor w/ 20000 levels "C1","C10","C100",..: 17699 16532 11027 17984 2363 11747 14115 15556 15215 12434
...
 $ TARGET                 : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AGE                    : int  27 47 40 53 36 42 30 53 42 30 ...
 $ GENDER                 : Factor w/ 3 levels "F","M","O": 2 2 2 2 2 1 2 1 1 2 ...
 $ BALANCE                : num  3314 28489 18217 71729 1671623 ...
 $ OCCUPATION             : Factor w/ 4 levels "PROF","SAL","SELF EMP",..: 3 2 3 2 1 1 1 2 3 1 ...
 $ AGE_BKT                : Factor w/ 7 levels "<25",">50","26-30",..: 3 7 5 2 5 6 3 2 6 3 ...
 $ SCR                    : int  776 324 603 196 157 498 479 562 185 178 ...
 $ HOLDING_PERIOD         : int  30 28 2 13 24 26 14 25 15 13 ...
 $ ACC_TYPE               : Factor w/ 2 levels "CA","SA": 2 2 2 1 2 2 2 1 2 2 ...
 $ ACC_OP_DATE            : Factor w/ 4869 levels "01-01-2000","01-01-2001",..: 3270 1807 3575 994 2861 863 4533 3160 258 335 ...
 $ LEN_OF_RLTN_IN_MNTH    : int  146 104 61 187 185 192 177 99 88 111 ...
 $ NO_OF_L_CR_TXNS        : int  7 8 10 16 20 5 6 14 18 14 ...
 $ NO_OF_L_DR_TXNS        : int  3 2 5 14 1 2 6 3 14 8 ...
 $ TOT_NO_OF_L_TXNS       : int  10 10 15 30 21 7 12 17 32 22 ...
 $ NO_OF_BR_CSH_WDL_DR_TXNS: int  0 0 1 4 1 1 0 3 6 3 ...
 $ NO_OF_ATM_DR_TXNS      : int  1 1 1 2 0 1 1 0 2 1 ...
 $ NO_OF_NET_DR_TXNS      : int  2 1 1 3 0 0 1 0 4 0 ...
 $ NO_OF_MOB_DR_TXNS      : int  0 0 0 1 0 0 0 0 1 0 ...
 $ NO_OF_CHQ_DR_TXNS      : int  0 0 2 4 0 0 4 0 1 4 ...
 $ FLG_HAS_CC             : int  0 0 0 0 0 1 0 0 1 0 ...
 $ AMT_ATM_DR             : int  13100 6600 11200 26100 0 18500 6200 0 35400 18000 ...
 $ AMT_BR_CSH_WDL_DR      : int  0 0 561120 673500 806400 379310 0 945160 708432 869080 ...
 $ AMT_CHQ_DR             : int  0 0 49320 60780 0 0 10680 0 51490 32610 ...
 $ AMT_NET_DR             : num  971557 799811 997570 761506 0 ...
 $ AMT_MOB_DR             : int  0 0 0 71388 0 0 0 0 179332 0 ...
 $ AMT_L_DR               : num  986657 806411 1613210 1573354 806480 ...
 $ FLG_HAS_ANY_CHGS       : int  0 1 1 0 0 0 1 0 0 0 ...
 $ AMT_OTH_BK_ATM_USG_CHGS: int  0 0 0 0 0 0 0 0 0 0 ...
 $ AMT_MIN_BAL_NMC_CHGS   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NO_OF_IW_CHQ_BNC_TXNS  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NO_OF_OW_CHQ_BNC_TXNS  : int  0 0 1 0 0 0 0 0 0 0 ...
 $ AVG_AMT_PER_ATM_TXN    : num  13100 6600 11200 13050 0 ...
 $ AVG_AMT_PER_CSH_WDL_TXN: num  0 0 561120 168350 806480 ...
 $ AVG_AMT_PER_CHQ_TXN    : num  0 0 24660 15195 0 ...
 $ AVG_AMT_PER_NET_TXN    : num  485778 799811 997570 253169 0 ...
 $ AVG_AMT_PER_MOB_TXN    : num  0 0 0 71388 0 ...
 $ FLG_HAS_NOMINEE        : int  1 1 1 1 1 1 0 1 1 0 ...
 $ FLG_HAS_OLD_LOAN       : int  1 0 1 0 0 1 1 1 1 0 ...
 $ random                 : num  1.14e-05 1.11e-04 1.20e-04 1.37e-04 1.74e-04 ...
### Dividing data to train and test
```

```
> sapply(Bank_data,function(x){sum(is.na(x))})
      CUST_ID        TARGET           AGE        GENDER       BALANCE 
            0             0             0             0             0 
   OCCUPATION       AGE_BKT           SCR HOLDING_PERIOD      ACC_TYPE 
            0             0             0             0             0 
  ACC_OP_DATE LEN_OF_RLTN_IN_MNTH NO_OF_L_CR_TXNS NO_OF_L_DR_TXNS TOT_NO_OF_L_TXNS 
            0             0             0             0             0 
NO_OF_BR_CSH_WDL_DR_TXNS NO_OF_ATM_DR_TXNS NO_OF_NET_DR_TXNS NO_OF_MOB_DR_TXNS NO_OF_CHQ_DR_TXNS 
            0             0             0             0             0 
     FLG_HAS_CC    AMT_ATM_DR  AMT_BR_CSH_WDL_DR     AMT_CHQ_DR     AMT_NET_DR 
            0             0             0             0             0 
     AMT_MOB_DR     AMT_L_DR  FLG_HAS_ANY_CHGS AMT_OTH_BK_ATM_USG_CHGS AMT_MIN_BAL_NMC_CHGS 
            0             0             0             0             0 
NO_OF_IW_CHQ_BNC_TXNS NO_OF_OW_CHQ_BNC_TXNS AVG_AMT_PER_ATM_TXN AVG_AMT_PER_CSH_WDL_TXN AVG_AMT_PER_CHQ_TXN 
            0             0             0             0             0 
  AVG_AMT_PER_NET_TXN AVG_AMT_PER_MOB_TXN  FLG_HAS_NOMINEE  FLG_HAS_OLD_LOAN        random 
            0             0             0             0             0 
```

We can observe that there are no rows which have any missing values. And after checking all the data, we have noticed that date is not in date format, we have a column with sequence which must be eliminated and random value column which must be deleted.

We can perform decision tree model below scenario

1. Removing Random and ID column and change date to date data type.
2. Reduce dimensions with co-linearity, scale data and build model.

Removing Random and ID column and change date to date data type.

```
> ### Dividing data to train and test
> Bank_data=Bank_data[c(-1,-40)]
> Bank_data$ACC_OP_DATE = parse_date_time(x = Bank_data$ACC_OP_DATE,orders = c("d m y", "d B Y", "m/d/y"))
> set.seed(50)
> n=nrow(Bank_data)
> trainIndex<-sample(1:n, size = round(0.70*n), replace=FALSE)
> traindata=Bank_data[trainIndex,]
> testdata= Bank_data[-trainIndex,]
> #View(traindata)
> #View(testdata)
> prop.table(table(traindata$TARGET))

    0     1 
0.872 0.128 
> prop.table(table(testdata$TARGET))

   0    1 
0.88 0.12 
> accu=integer();
> accut=integer();
> minbu=integer();
> for (i in seq(1,n,50)){
>   Bank_tree=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =10,minsplit = 10,xval = 10, cp=0.0028),method = 'c
lass')
>   res<-predict(Bank_tree,newdata = traindata,type="class")
>   result<-predict(Bank_tree,newdata = testdata,type="class")
>   xy=table(traindata$TARGET,res)
>   tempo=(xy[[1]]+xy[[4]])/nrow(traindata)
>   accu=c(accu,tempo) ##acc
>   xx=table(testdata$TARGET,result)
>   temp=((xx[[1]]+xx[[4]])/nrow(testdata))
>   accu=c(accu,temp) ##Accuracy
>   minbu=c(minbu,i)
> }
> summary(accu)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.8823  0.8823  0.8823  0.8823  0.8823  0.8823 
> summary(accut)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.8856  0.8856  0.8856  0.8855  0.8856  0.8856 
> plot(accut~minbu,type="l")
> plot(accu~minbu,type="l")
> Bank_tree_final=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =10,minsplit = 10,xval = 10, cp=0.0025),method
= "class")
> prp(Bank_tree_final)
> fancyRpartPlot(Bank_tree_final)
```

Divide data to train and test data.

```
Bank_data=Bank_data[c(-1,-40)]
set.seed(60)
n=nrow(Bank_data)
trainIndex<-sample(1:n, size = round(0.70*n), replace=FALSE)
traindata<-Bank_data[trainIndex,]
testdata<-Bank_data[-trainIndex,]
```

Check the data proportion.

```
> prop.table(table(traindata$TARGET))

    0     1
0.873 0.126
> prop.table(table(testdata$TARGET))

    0     1
0.88 0.11
```

```
prop.table(table(traindata$TARGET))
prop.table(table(testdata$TARGET))
```

Choosing appropriate CP value.

```
> numfold = trainControl(method = "cv", number = 10)
> cpgrid = expand.grid(.cp=seq(0.001,0.005,0.0001))
> cpVal = train(TARGET ~ ., data = traindata, method='rpart', trControl=numfold, tuneGrid=cpgrid)
> cpVal
CART

14000 samples
   37 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 12599, 12601, 12600, 12601, 12600, 12600, ...
Resampling results across tuning parameters:

  cp      Accuracy   Kappa
  0.0010  0.8881431  0.407131235
  0.0011  0.8877856  0.391417372
  0.0012  0.8876428  0.389551426
  0.0013  0.8874287  0.373387312
  0.0014  0.8867858  0.364368734
  0.0015  0.8864288  0.357340525
  0.0016  0.8853573  0.343870405
  0.0017  0.8842856  0.326442554
  0.0018  0.8840714  0.323377724
  0.0019  0.8817142  0.274718651
  0.0020  0.8809999  0.262387548
  0.0021  0.8805002  0.241993567
  0.0022  0.8802859  0.225884366
  0.0023  0.8802141  0.219874023
  0.0024  0.8801428  0.210950162
  0.0025  0.8787142  0.186839316
  0.0026  0.8787142  0.186839316
  0.0027  0.8781430  0.173260077
  0.0028  0.8780713  0.161827141
  0.0029  0.8772147  0.146870160
  0.0030  0.8771431  0.141082978
  0.0031  0.8759997  0.126652428
  0.0032  0.8759997  0.124738748
  0.0033  0.8759997  0.124738748
  0.0034  0.8760711  0.122918313
```

```
  0.0030  0.8771431  0.141882978
  0.0031  0.8759997  0.126652428
  0.0032  0.8759997  0.124708748
  0.0033  0.8759997  0.124708748
  0.0034  0.8760711  0.122918313
  0.0035  0.8758282  0.117114454
  0.0036  0.8758568  0.118089076
  0.0037  0.8758568  0.118089076
  0.0038  0.8744284  0.097575308
  0.0039  0.8744284  0.097575308
  0.0040  0.8740715  0.092400124
  0.0041  0.8738286  0.088728384
  0.0042  0.8740000  0.091368548
  0.0043  0.8740000  0.091368560
  0.0044  0.8737143  0.088905527
  0.0045  0.8727855  0.081395749
  0.0046  0.8727856  0.081531537
  0.0047  0.8721429  0.075483566
  0.0048  0.8721429  0.075483566
  0.0049  0.8721429  0.075483566
  0.0050  0.8717860  0.070514043

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.001.
```

0.001 is the purest form and the decision tree obtained would be over fit. The decision tree obtained using 0.19 is very complex hence 0.20 to 0.28 can be selected.

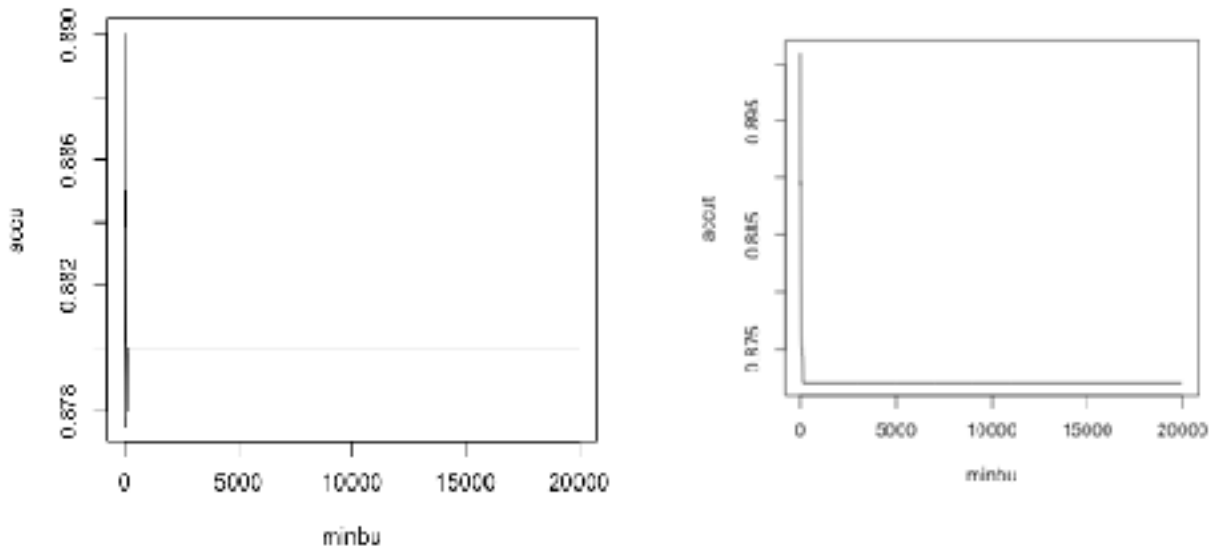
Build decision tree in loop to find the best minimum bucket.

```
accu=integer();
accut=integer();
minbu=integer();
tempsen=integer();
tempsens=integer()
for (i in seq(1,n,10)){
  Bank_tree=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =i,minsplit = 10,xval = 10, cp=0.001),method = "class")
  res<-predict(Bank_tree,newdata = traindata,type="class")
  result<-predict(Bank_tree,newdata = testdata,type="class")
  xy=table(traindata$TARGET,res)
  tempo=((xy[[1]]-xy[[4]])/nrow(traindata))
  tempsen=c(tempsen,xy[[2]]+xy[[4]])
  accut=c(accut,tempo) ##Ac
  xi=table(testdata$TARGET,result)
  temp=((xx[[1]]+xx[[4]])/nrow(testdata))
  tempsens=c(tempsens,xy[[2]]+xy[[4]])
  accu=c(accu,temp) ##Accuracy
  minbu=c(minbu,i)
}
```

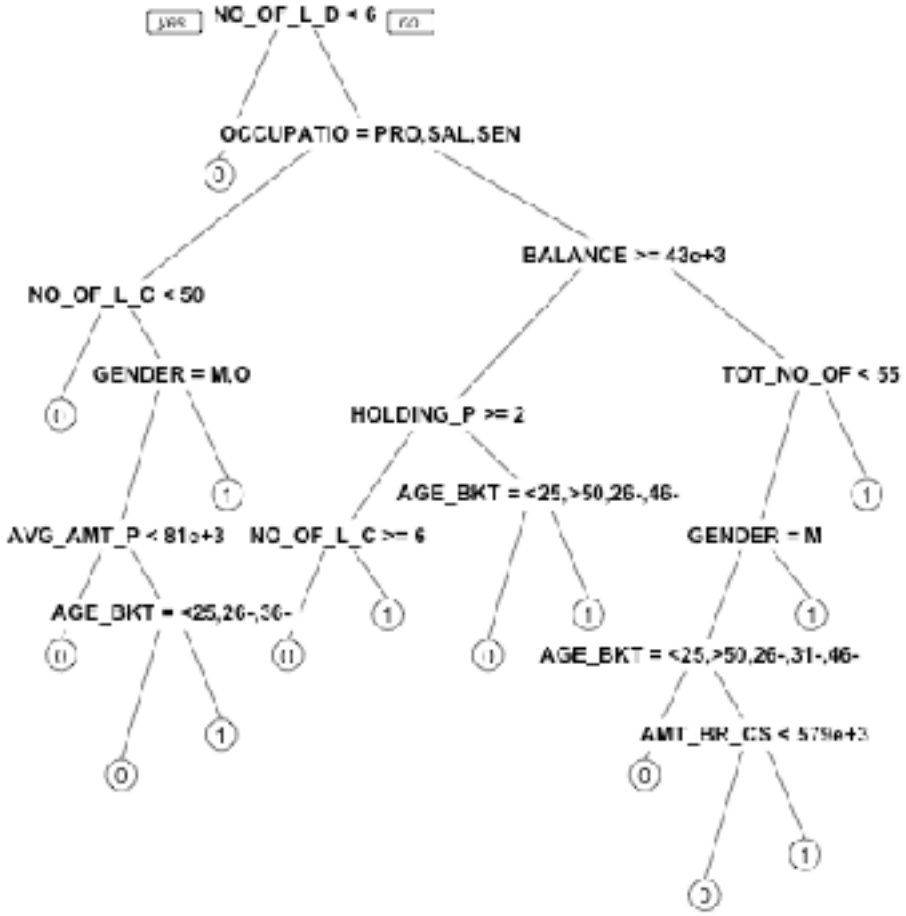
Plot min bucket graph.

```
> summary(accu)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8700  0.8900  0.8850  0.8880  0.8800  0.8900
> summary(accut)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8700  0.8700  0.8700  0.8700  0.8700  0.8850
```

	minbu	accu	accut
1	1	0.8900000	0.9009286
2	11	0.8838333	0.8893571
3	21	0.8850000	0.8896429
4	31	0.8801667	0.8812857
5	41	0.8801667	0.8777857
6	51	0.8791667	0.8761429
7	61	0.8775000	0.8754286
8	71	0.8778333	0.8750714
9	81	0.8778333	0.8750714
10	91	0.8795000	0.8745000
11	101	0.8795000	0.8745000
12	111	0.8798333	0.8742857
13	121	0.8780000	0.8735714
14	131	0.8800000	0.8720000
15	141	0.8800000	0.8720000
16	151	0.8800000	0.8720000
17	161	0.8800000	0.8720000
18	171	0.8800000	0.8720000
19	181	0.8800000	0.8720000
20	191	0.8800000	0.8720000
21	201	0.8800000	0.8720000



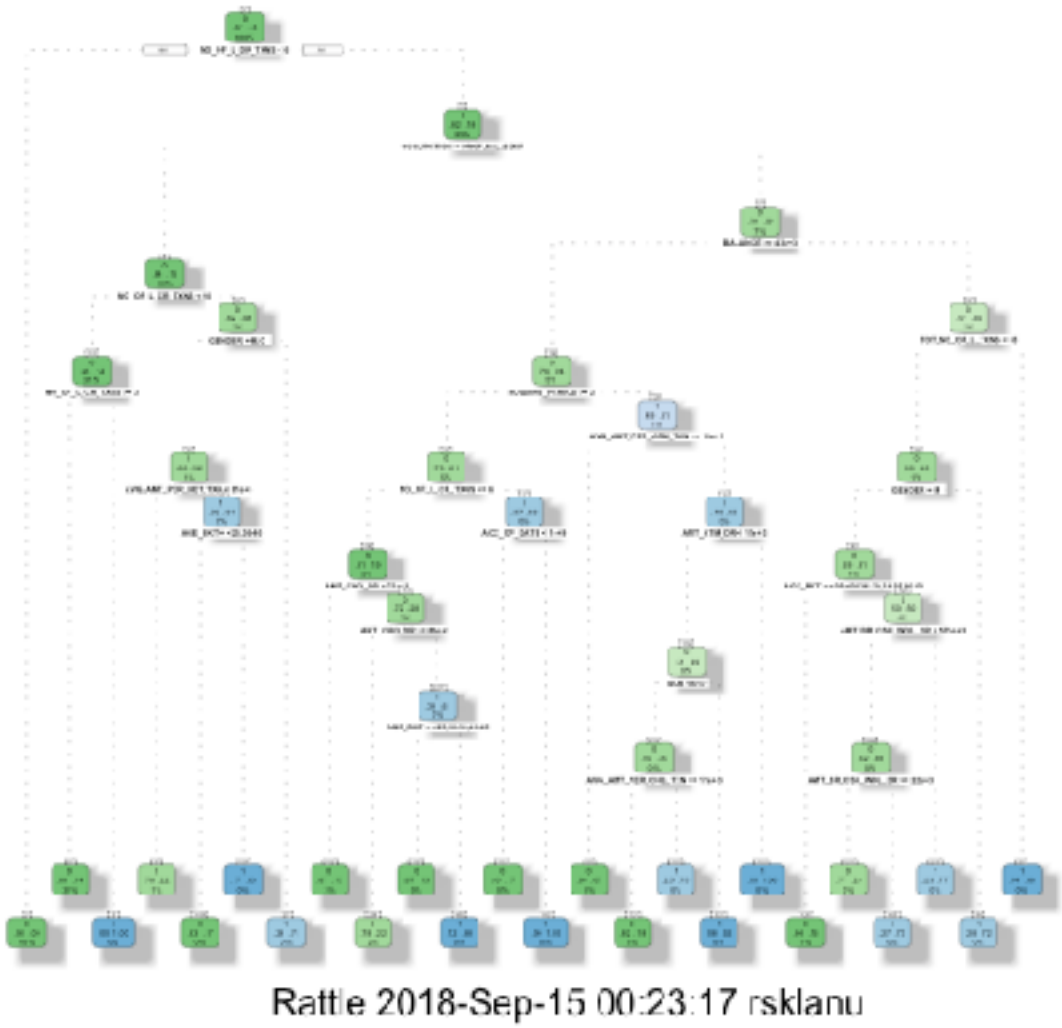
20 can be chosen as min bucket.
Build final model with min bucket with 0.28 CP and 20 as min bucket.



Build tree with min bucket 20 and CP 0.0028.

```
Bank_tree_1 <- rpart("TARGET" ~ ., data = trainData,control=rpart.con.rol(minbucket =10,minsplit = 10,xval = 10, cp=0.0028),method = "class")
```

Visualize the tree



We were able to visualise the above decision tree with the model built.

For the above model even through the accuracy was 88-89%

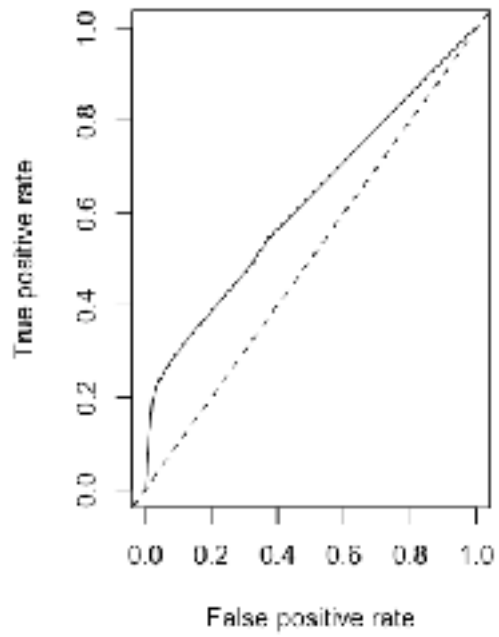
Confusion matrix looks like below.

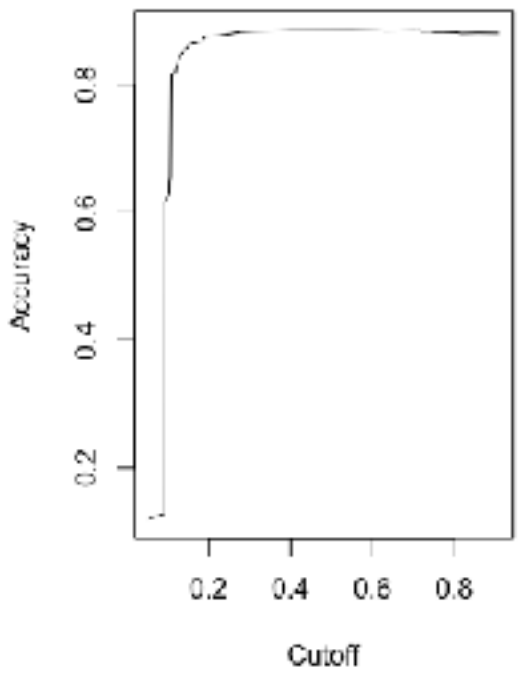
```
> traindata$TARGET=as.factor(traindata$TARGET)
> testdata$TARGET=as.factor(testdata$TARGET)
> Bank_tree_final=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =20,minsplit = 10,xval = 10, cp=0.0028),method = "class")
> res=predict(Bank_tree_final,newdata = traindata,type="class")
> result=predict(Bank_tree_final,newdata = testdata,type="class")
> xy=table(traindata$TARGET,res)
> xx=table(testdata$TARGET,result)
> xy
   res
        0     1
  0 12145    63
  1  1000   854
> xx
   result
        0    1
  0 5210   42
  1  696   94
> Accuracytrain=(xy[1]+xy[4])/nrow(traindata)
> Accuracytest=(xx[1]+xx[4])/nrow(testdata)
> Accuracytrain
[1] 0.8866429
> Accuracytest
[1] 0.8863333
```

Confusion matrix for min bucket 10 and cp=0.2 is given below.

```
> Bank_tree_final=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =20,minsplit = 10,xval = 10, cp=0.002),method = "class")
> res<-predict(Bank_tree_final,newdata = traindata,type="class")
> result<-predict(Bank_tree_final,newdata = testdata,type='class')
> xy=table(traindata$TARGET,res)
> Accuracytrain=(xy[[1]]+xy[[4]])/nrow(traindata)
> xx=table(testdata$TARGET,result)
> Accuracytest=(xx[[1]]+xx[[4]])/nrow(testdata)
> xy
   res
      0    1
  0 12345   65
  1  1688  184
> xx
   result
      0    1
  0 5238   42
  1  636   84
> Accuracytrain
[1] 0.8806429
> Accuracytest
[1] 0.8803333
> Bank_tree_final=rpart(TARGET ~ ., data = traindata,control=rpart.control(minbucket =20,minsplit = 10,xval = 10, cp=0.002),method = "class")
> res<-predict(Bank_tree_final,newdata = traindata,type="class")
> result<-predict(Bank_tree_final,newdata = testdata,type='class')
> xy=table(traindata$TARGET,res)
> Accuracytrain=(xy[[1]]+xy[[4]])/nrow(traindata)
> xx=table(testdata$TARGET,result)
> Accuracytest=(xx[[1]]+xx[[4]])/nrow(testdata)
> xy
   res
      0    1
  0 12060  148
  1  1290  482
> xx
   result
      0    1
  0 5188   92
  1  661  119
> Accuracytrain
[1] 0.8961429
> Accuracytest
[1] 0.8845
```

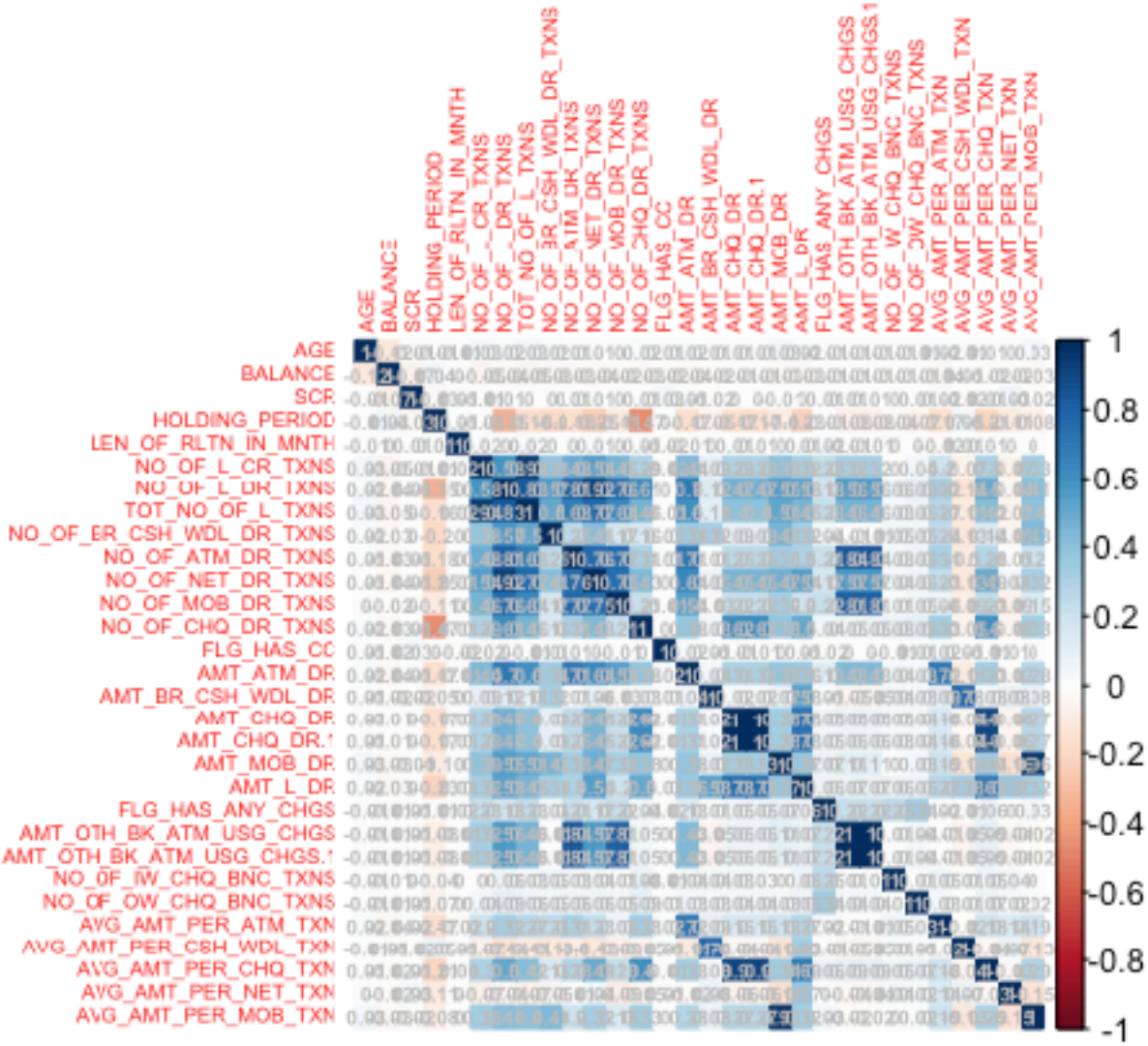
ROC and Accuracy plot looks like below



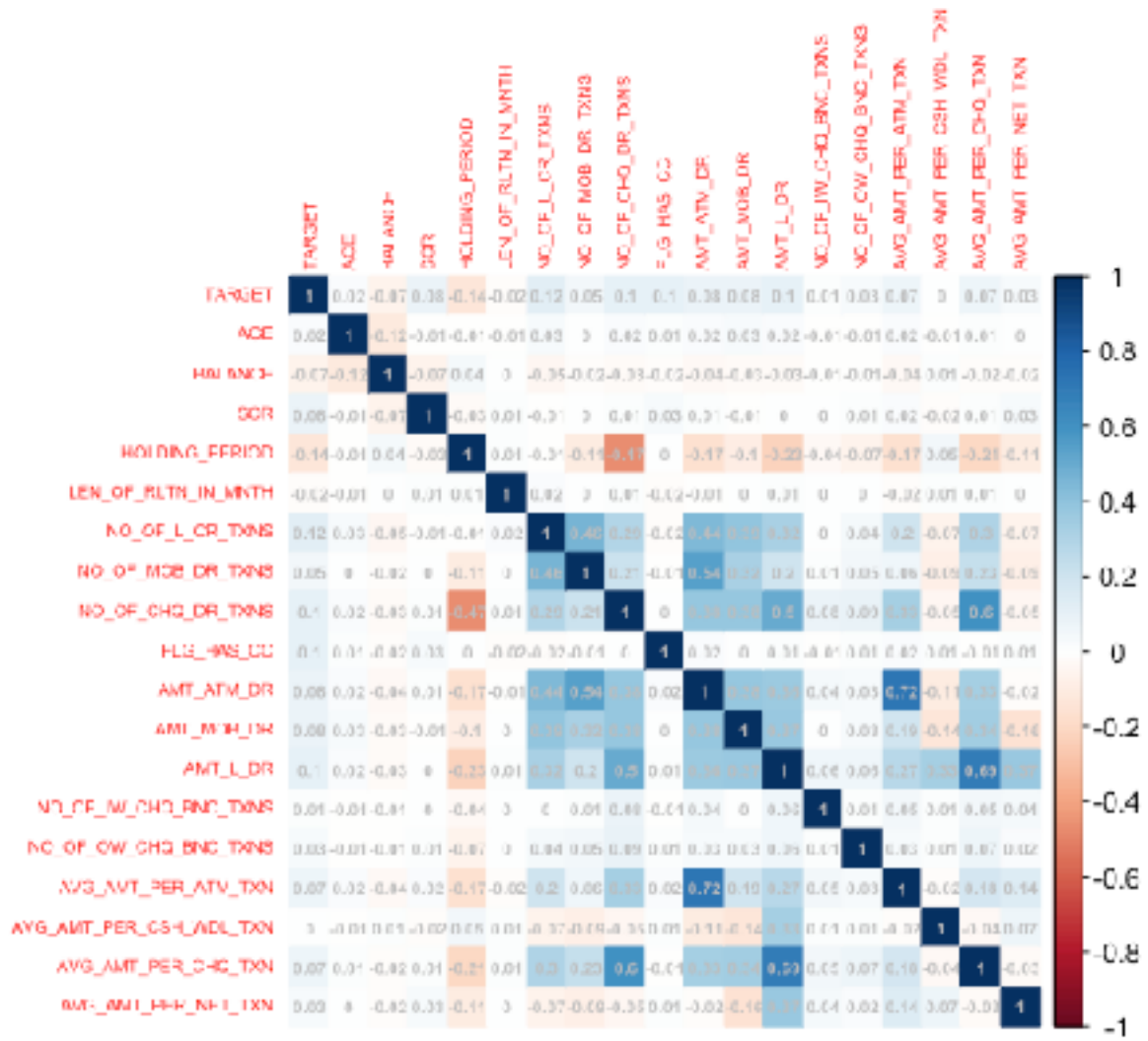


We then tried to reduce collinear dimensions and scaling data, we could arrive at below model

Finding collinearity.



Reduced columns coliniaerity



Reading data from excel and removing ID and random column

```
> Bank_data_scale <- read.csv("C:/Users/oskilaru/Cart_assignment/PL_XSELL.csv", header=TRUE)
> View(Bank_data_scale)
> Bank_data_scale=Bank_data_scale[c(-1,-40)]
```

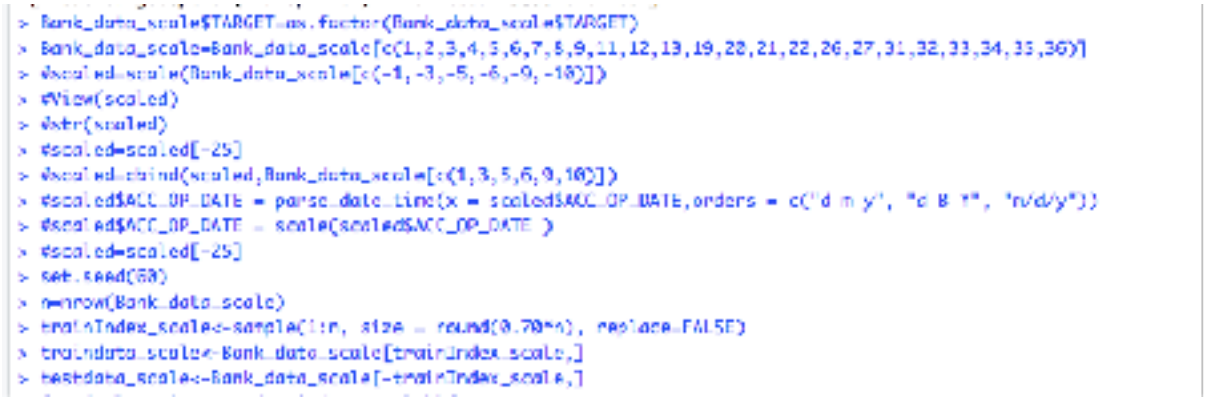
```
> Bank_data_scale$REC_EFF_DATE = parse_date_time(x = Bank_data_scale$REC_EFF_DATE,orders = c("d m y", "d B Y", "m/d/y"))
```

Scaling data after removing all colinear row which provide similar data. Also date has been delete as it gives same info as LEN_OF_RLTN_IN_MNTH.

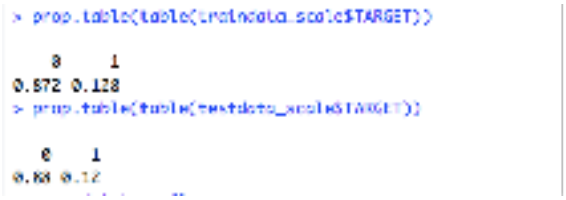
Dividing data to train and test

```
> for (column in names(traindata_scale)){
+
+   if (class(traindata_scale[, column]) != "factor"){
+     traindata_scale[, column] = scale(traindata_scale[, column])
+     traindata_scale[, column] = scale(traindata_scale[, column])
+   }
+
+   if (class(testdata_scale[, column]) != "factor"){
+     testdata_scale[, column] = scale(testdata_scale[, column])
+     testdata_scale[, column] = scale(testdata_scale[, column])
+   }
+
+ }
```

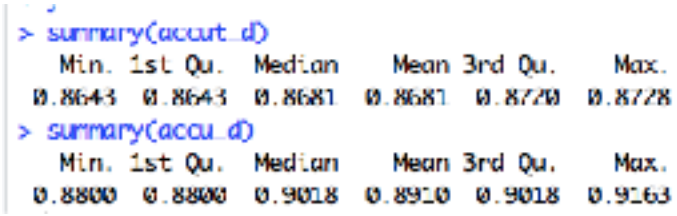
Dividing data for train and test and removing unwanted columns.



Checking proportion and finding min bucket

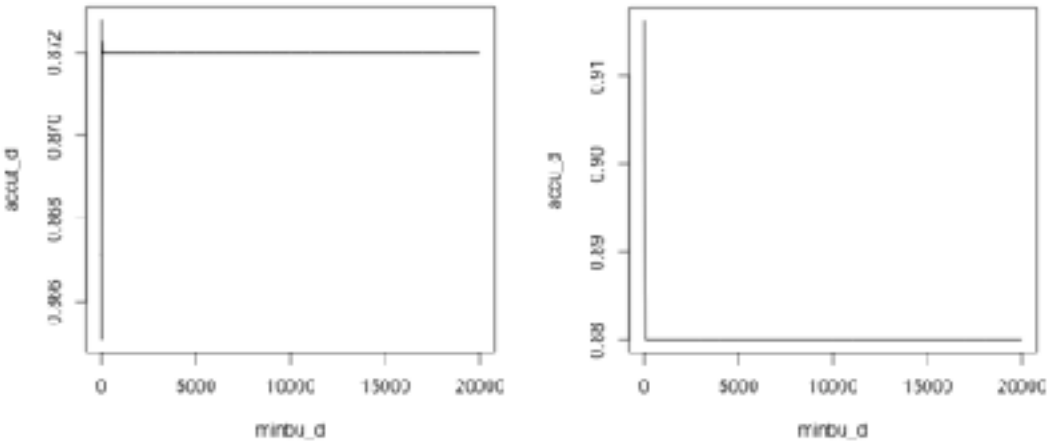


Plot accuracy and find min bucket with max accuracy in both test and train.



When we compare the accuracy and min bucket bets case is overfit model whose bucket size is one. To avoid over fitting and get better accuracy I have chosen bucket size of 20.
Here is where we can arrive at median accuracy of both test and train data.

On the training set we have got approximately 87% but for test we have got 91% accuracy.



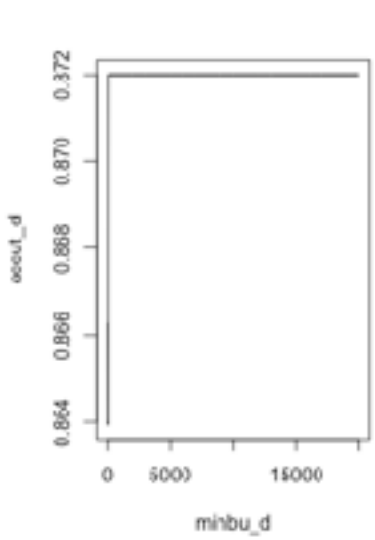
Building final model

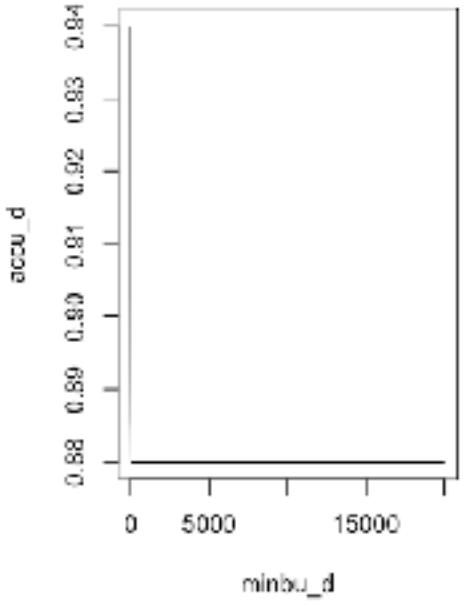
```
> Bank_tree_final_scale=rpart(TARGET ~ ., data = testdata_scale,control=rpart.control(minbucket =20,minsplit = 10,xval = 10, cp=0.002),method = "cla
ss")
```

```
  object 'scaled' not found
> res_d<-predict(Bank_tree_final_scale,newdata = traindata_scale,type="class")
> result_d<-predict(Bank_tree_final_scale,newdata = testdata_scale,type="class")
> xy_d=table(traindata_scale$TARGET,res_d)
> xx_d=table(testdata_scale$TARGET,result_d)
> xy_d
   res_d
        0     1
  0 11892   316
  1  1576   216
> xx_d
   result_d
       0    1
  0 5205   75
  1  573  147
```

Visualising the tree

ROC and AUC curve



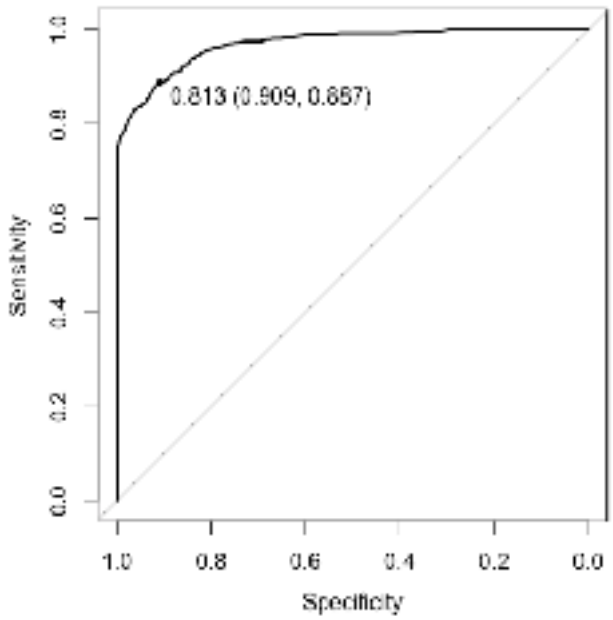


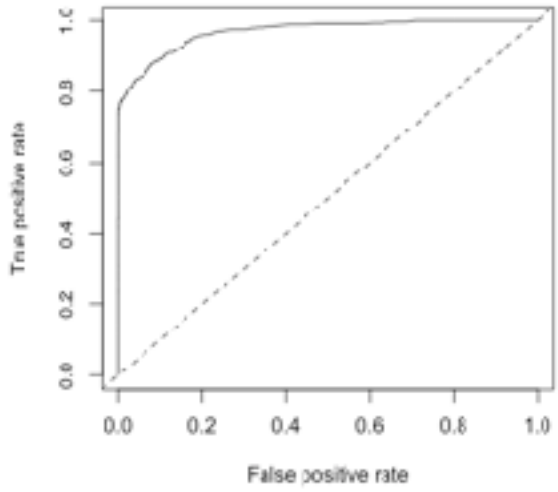
Using Random forest.

```
> Bank_data <- read.csv(file="/Users/rsklonu/CourseAssignment/PL_XSELL.csv", header=TRUE)
> Bank_data=Bank_data[c(-1,-40)]
> Bank_data=Bank_data[c(1,2,3,4,5,6,7,8,9,10,11,12,13,19,20,21,22,26,27,31,32,33,34,35,36)]
> Bank_data$ACC_OP_DATE = parse_date_time(x = Bank_data$ACC_OP_DATE,orders = c("d m y", "d B Y", "m/d/y"))
> set.seed(88)
> nrow(Bank_data)
> trainIndex<-sample(1:n, size = round(0.70*n), replace=FALSE)
> trainBank<-Bank_data[trainIndex,]
> testBank<-Bank_data[-trainIndex,]
> trainBank$TARGET=as.factor(trainBank$TARGET)
> testBank$TARGET=as.factor(testBank$TARGET)
> rf.fit = randomForest(TARGET ~ ., data = trainBank, ntree=500, mtry=10)
> pred.test.rf = predict(rf.fit, newdata = testBank)
> table(pred.test.rf, actual)
```

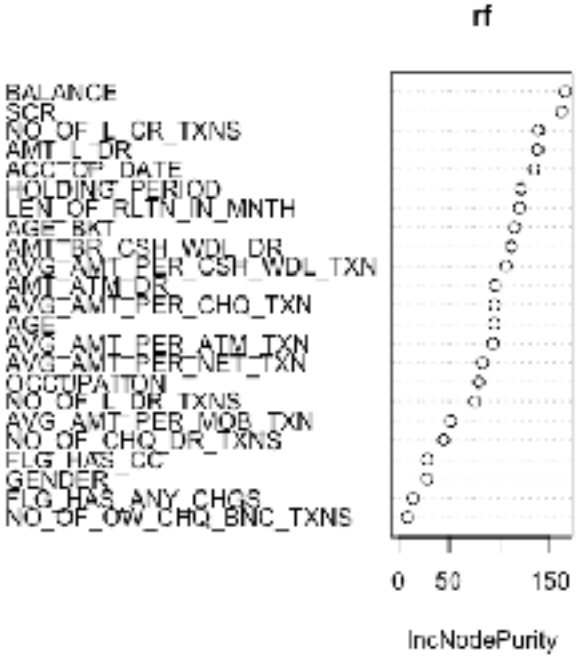
```
> actual = testBank$TARGET
> table(pred.test.rf, actual)
            actual
pred.test.rf    0    1
           0 5268  177
           1   12  543
> confusionMatrix(pred.test.rf, actual)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5268  177
         1   12  543

               Accuracy : 0.9685
                 95% CI : (0.9638, 0.9728)
    No Information Rate : 0.88
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8345
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9977
            Specificity : 0.7544
         Pos Pred Value : 0.9675
         Neg Pred Value : 0.9784
             Prevalence : 0.8800
> precision(pred.test.rf, actual)
[1] 0.9674931
> varImp(rf.fit)
                          Overall
AGE                    140.18335
GENDER                  39.88709
BALANCE                267.02592
OCCUPATION             107.15078
AGE_BKT                155.86445
SCR                    249.48198
HOLDING_PERIOD         187.91698
ACC_TYPE                14.84268
ACC_OP_DATE            191.17475
LEN_OF_RLTN_IN_MNTH    163.43261
NO_OF_L_CR_TXNS        215.43662
NO_OF_L_DR_TXNS        115.25184
NO_OF_CHQ_DR_TXNS       64.48336
FLG_HAS_CC              39.04389
AMT_ATM_DR             135.68878
AMT_BR_CSH_WDL_DR      165.24844
AMT_L_DR               285.75475
FLG_HAS_ANY_CHGS        19.62448
NO_OF_ON_CHQ_BNC_TXNS   13.58133
AVG_AMT_PER_ATM_TXN    133.34562
AVG_AMT_PER_CSH_WDL_TXN 150.00728
AVG_AMT_PER_CHQ_TXN    141.17843
AVG_AMT_PER_NET_TXN    127.38884
AVG_AMT_PER_MOB_TXN     82.44425
> varImpPlot(rf.fit)
> library(pROC)
```

```
> res = predict(rf.fit, newdata = testBank, type = 'prob')
> # Draw ROC curve.
> result.roc <- roc(testBank$TARGET, res[,1])
> plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
> pred.data <- prediction(predict(rf.fit, newdata = testBank,type = "prob")[, 2], testBank$TARGET)
> plot(performance(pred.data, "tpr", "fpr"))
> abline(0, 1, lty = 2)
> #to get threshold and accuracy
> result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy"))
> print(result.coords)
threshold  accuracy
    0.813     0.906
>
```

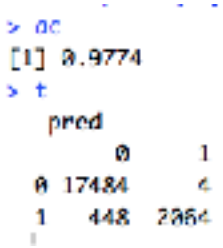




Significant variables in random forest are plotted as below.



Accuracy achieved using random forest model is 98%



Ratio of misclassification is less compared to decision tree model.
The only disadvantage of using random forest is , its a black box approach, we are unable to explain why we classified a customer to defaulter or non defaulter.

I therefore considered top 20 , non related columns from the above graph and tried to draw a decision tree and check if better results can be obtained using decision tree, I obtained the similar result.

Based on the requirement if the customer requires an explanation or not we ca use either of the methods can be used.