Week_04_Rui_Peng.R

raypeng

2025-04-05

```r
#Step 1: Read in the Data
#Read the data into R
library(rpart) #to use decision tree
library(rpart.plot) #display the decision tree
library(ROCR) #print and see how acurate it is

PATH = "/Users/raypeng/Documents/IS 5213 Data science and big data/HMEQ_Scrubbed"
FILE_NAME = "HMEQ_Scrubbed.csv"

INFILE = paste(PATH, FILE_NAME, sep = "/")

setwd(PATH)
df = read.csv(FILE_NAME)

#List the structure of the data (str)
str(df)
```

```
## 'data.frame':    5960 obs. of  29 variables:
##  $ TARGET_BAD_FLAG    : int  1 1 1 1 0 1 1 1 1 1 ...
##  $ TARGET_LOSS_AMT    : int  641 1109 767 1425 0 335 1841 373 1217 1523 ...
##  $ LOAN               : int  1100 1300 1500 1500 1700 1700 1800 1800 2000 2000 ...
##  $ IMP_MORTDUE        : num  25860 70053 13500 65000 97800 ...
##  $ M_MORTDUE          : int  0 0 0 1 0 0 0 0 0 1 ...
##  $ IMP_VALUE          : num  39025 68400 16700 89000 112000 ...
##  $ M_VALUE            : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_YOJ            : num  10.5 7 4 7 3 9 5 11 3 16 ...
##  $ M_YOJ              : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_DEROG          : int  0 0 0 1 0 0 3 0 0 0 ...
##  $ M_DEROG            : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_DELINQ         : int  0 2 0 1 0 0 2 0 2 0 ...
##  $ M_DELINQ           : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_CLAGE          : num  94.4 121.8 149.5 174 93.3 ...
##  $ M_CLAGE            : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_NINQ           : int  1 0 1 1 0 1 1 0 1 0 ...
##  $ M_NINQ             : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_CLNO           : int  9 14 10 20 14 8 17 8 12 13 ...
##  $ M_CLNO             : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ IMP_DEBTINC        : num  35 35 35 35 35 ...
##  $ M_DEBTINC          : int  1 1 1 1 1 0 1 0 1 1 ...
##  $ FLAG.Job.Mgr       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FLAG.Job.Office    : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ FLAG.Job.Other     : int  1 1 1 0 0 1 1 1 1 0 ...
```

```
##  $ FLAG.Job.ProfExe   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FLAG.Job.Sales     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ FLAG.Job.Self      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FLAG.Reason.DebtCon: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FLAG.Reason.HomeImp: int  1 1 1 0 1 1 1 1 1 1 ...
```

```
#Execute a summary of the data
summary(df)
```

```
##  TARGET_BAD_FLAG  TARGET_LOSS_AMT      LOAN        IMP_MORTDUE
##  Min.   :0.0000   Min.   :    0   Min.   : 1100   Min.   :  2063
##  1st Qu.:0.0000   1st Qu.:    0   1st Qu.:11100   1st Qu.: 48139
##  Median :0.0000   Median :    0   Median :16300   Median : 65000
##  Mean   :0.1995   Mean   : 2676   Mean   :18608   Mean   : 72999
##  3rd Qu.:0.0000   3rd Qu.:    0   3rd Qu.:23300   3rd Qu.: 88200
##  Max.   :1.0000   Max.   :78987   Max.   :89900   Max.   :399550
##    M_MORTDUE         IMP_VALUE         M_VALUE          IMP_YOJ
##  Min.   :0.00000   Min.   :  8000   Min.   :0.00000   Min.   : 0.000
##  1st Qu.:0.00000   1st Qu.: 66490   1st Qu.:0.00000   1st Qu.: 3.000
##  Median :0.00000   Median : 89000   Median :0.00000   Median : 7.000
##  Mean   :0.08691   Mean   :101536   Mean   :0.01879   Mean   : 8.756
##  3rd Qu.:0.00000   3rd Qu.:119005   3rd Qu.:0.00000   3rd Qu.:12.000
##  Max.   :1.00000   Max.   :855909   Max.   :1.00000   Max.   :41.000
##      M_YOJ           IMP_DEROG         M_DEROG         IMP_DELINQ
##  Min.   :0.00000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.000
##  Median :0.00000   Median : 0.0000   Median :0.0000   Median : 0.000
##  Mean   :0.08641   Mean   : 0.3431   Mean   :0.1188   Mean   : 0.503
##  3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 1.000
##  Max.   :1.00000   Max.   :10.0000   Max.   :1.0000   Max.   :15.000
##     M_DELINQ         IMP_CLAGE         M_CLAGE          IMP_NINQ
##  Min.   :0.00000   Min.   :   0.0   Min.   :0.00000   Min.   : 0.00
##  1st Qu.:0.00000   1st Qu.: 117.4   1st Qu.:0.00000   1st Qu.: 0.00
##  Median :0.00000   Median : 174.0   Median :0.00000   Median : 1.00
##  Mean   :0.09732   Mean   : 179.5   Mean   :0.05168   Mean   : 1.17
##  3rd Qu.:0.00000   3rd Qu.: 227.1   3rd Qu.:0.00000   3rd Qu.: 2.00
##  Max.   :1.00000   Max.   :1168.2   Max.   :1.00000   Max.   :17.00
##      M_NINQ           IMP_CLNO         M_CLNO         IMP_DEBTINC
##  Min.   :0.00000   Min.   : 0.00   Min.   :0.00000   Min.   :  0.5245
##  1st Qu.:0.00000   1st Qu.:15.00   1st Qu.:0.00000   1st Qu.: 30.7632
##  Median :0.00000   Median :20.00   Median :0.00000   Median : 35.0000
##  Mean   :0.08557   Mean   :21.25   Mean   :0.03725   Mean   : 34.0393
##  3rd Qu.:0.00000   3rd Qu.:26.00   3rd Qu.:0.00000   3rd Qu.: 37.9499
##  Max.   :1.00000   Max.   :71.00   Max.   :1.00000   Max.   :203.3122
##    M_DEBTINC       FLAG.Job.Mgr    FLAG.Job.Office  FLAG.Job.Other
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.2126   Mean   :0.1287   Mean   :0.1591   Mean   :0.4007
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  FLAG.Job.ProfExe FLAG.Job.Sales    FLAG.Job.Self     FLAG.Reason.DebtCon
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
```

```
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :1.0000
##  Mean   :0.2141   Mean   :0.01829   Mean   :0.03238   Mean   :0.6591
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  FLAG.Reason.HomeImp
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2987
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

```
#Print the first six records
head(df)
```

```
##   TARGET_BAD_FLAG TARGET_LOSS_AMT LOAN IMP_MORTDUE M_MORTDUE IMP_VALUE M_VALUE
## 1               1             641 1100       25860         0     39025       0
## 2               1            1109 1300       70053         0     68400       0
## 3               1             767 1500       13500         0     16700       0
## 4               1            1425 1500       65000         1     89000       1
## 5               0               0 1700       97800         0    112000       0
## 6               1             335 1700       30548         0     40320       0
##   IMP_YOJ M_YOJ IMP_DEROG M_DEROG IMP_DELINQ M_DELINQ IMP_CLAGE M_CLAGE
## 1    10.5     0         0       0          0        0  94.36667       0
## 2     7.0     0         0       0          2        0 121.83333       0
## 3     4.0     0         0       0          0        0 149.46667       0
## 4     7.0     1         1       1          1        1 174.00000       1
## 5     3.0     0         0       0          0        0  93.33333       0
## 6     9.0     0         0       0          0        0 101.46600       0
##   IMP_NINQ M_NINQ IMP_CLNO M_CLNO IMP_DEBTINC M_DEBTINC FLAG.Job.Mgr
## 1        1      0        9      0    35.00000         1            0
## 2        0      0       14      0    35.00000         1            0
## 3        1      0       10      0    35.00000         1            0
## 4        1      1       20      1    35.00000         1            0
## 5        0      0       14      0    35.00000         1            0
## 6        1      0        8      0    37.11361         0            0
##   FLAG.Job.Office FLAG.Job.Other FLAG.Job.ProfExe FLAG.Job.Sales FLAG.Job.Self
## 1               0              1                0              0             0
## 2               0              1                0              0             0
## 3               0              1                0              0             0
## 4               0              0                0              0             0
## 5               1              0                0              0             0
## 6               0              1                0              0             0
##   FLAG.Reason.DebtCon FLAG.Reason.HomeImp
## 1                   0                   1
## 2                   0                   1
## 3                   0                   1
## 4                   0                   0
## 5                   0                   1
## 6                   0                   1
```

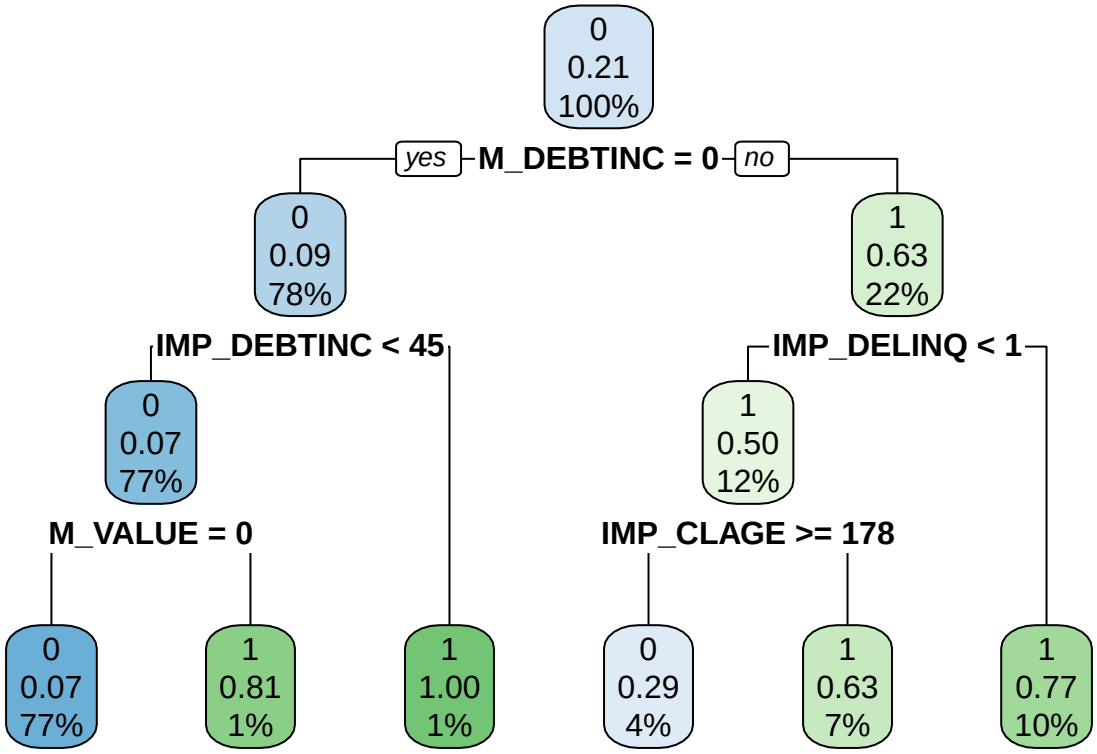
```
#Step 2: Classification Decision Tree
```

```r
#Using the code discussed in the lecture, split the data into training and testing data sets.
#Use the rpart library to predict the variable TARGET_BAD_FLAG
df_flag = df

#Do not use TARGET_LOSS_AMT to predict TARGET_BAD_FLAG.
df_flag$TARGET_LOSS_AMT = NULL
head(df_flag)
```

```
##   TARGET_BAD_FLAG LOAN IMP_MORTDUE M_MORTDUE IMP_VALUE M_VALUE IMP_YOJ M_YOJ
## 1               1 1100       25860         0     39025       0    10.5     0
## 2               1 1300       70053         0     68400       0     7.0     0
## 3               1 1500       13500         0     16700       0     4.0     0
## 4               1 1500       65000         1     89000       1     7.0     1
## 5               0 1700       97800         0    112000       0     3.0     0
## 6               1 1700       30548         0     40320       0     9.0     0
##   IMP_DEROG M_DEROG IMP_DELINQ M_DELINQ IMP_CLAGE M_CLAGE IMP_NINQ M_NINQ
## 1         0       0          0        0  94.36667       0        1      0
## 2         0       0          2        0 121.83333       0        0      0
## 3         0       0          0        0 149.46667       0        1      0
## 4         1       1          1        1 174.00000       1        1      1
## 5         0       0          0        0  93.33333       0        0      0
## 6         0       0          0        0 101.46600       0        1      0
##   IMP_CLNO M_CLNO IMP_DEBTINC M_DEBTINC FLAG.Job.Mgr FLAG.Job.Office
## 1        9      0    35.00000         1            0               0
## 2       14      0    35.00000         1            0               0
## 3       10      0    35.00000         1            0               0
## 4       20      1    35.00000         1            0               0
## 5       14      0    35.00000         1            0               1
## 6        8      0    37.11361         0            0               0
##   FLAG.Job.Other FLAG.Job.ProfExe FLAG.Job.Sales FLAG.Job.Self
## 1              1                0              0             0
## 2              1                0              0             0
## 3              1                0              0             0
## 4              0                0              0             0
## 5              0                0              0             0
## 6              1                0              0             0
##   FLAG.Reason.DebtCon FLAG.Reason.HomeImp
## 1                   0                  1
## 2                   0                  1
## 3                   0                  1
## 4                   0                  0
## 5                   0                  1
## 6                   0                  1
```

```r
FLAG = sample( c(TRUE, FALSE), nrow(df_flag), replace = TRUE, prob = c(0.8, 0.2))
df_train = df_flag[FLAG, ]
df_test = df_flag[!FLAG, ]

#Develop two decision trees, one using Gini and the other using Entropy using the training and testing
#All other parameters such as tree depth are up to you.
tr_set = rpart.control( maxdepth =10 )
t1G = rpart( data = df_train, TARGET_BAD_FLAG ~ .,
             control = tr_set, method = "class", parms = list(split = 'gini'))
```

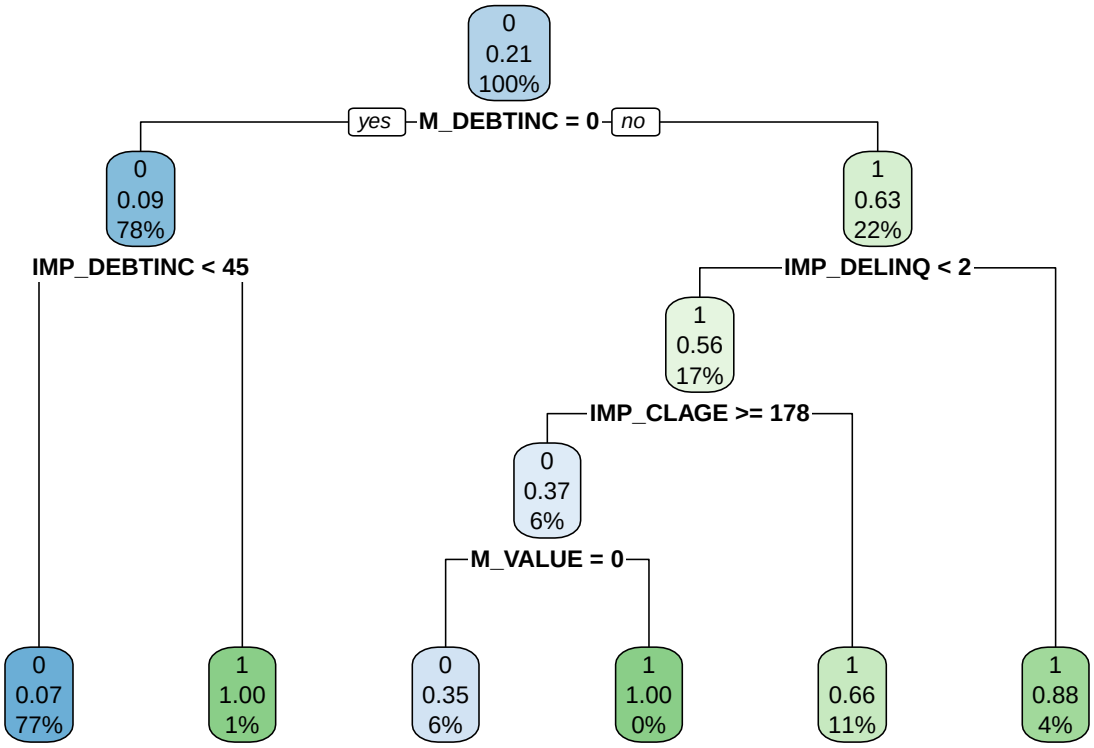
```
t1E = rpart( data = df_train, TARGET_BAD_FLAG ~ .,
             control = tr_set, method = "class", parms = list(split = 'information'))

#Plot both decision trees
#List the important variables for both trees
rpart.plot( t1G )
```




```
t1G$variable.importance
```

```
##   M_DEBTINC IMP_DEBTINC  IMP_DELINQ     M_VALUE   IMP_CLAGE        LOAN
##  464.661471  112.938963   51.926777   43.555321   30.469848   22.025452
##   IMP_DEROG     M_DEROG IMP_MORTDUE    M_DELINQ    IMP_CLNO      M_NINQ
##   19.587669    7.921540    6.810888    6.563561    6.420346    5.205583
##     IMP_YOJ   IMP_VALUE
##    3.449417    3.305691
```

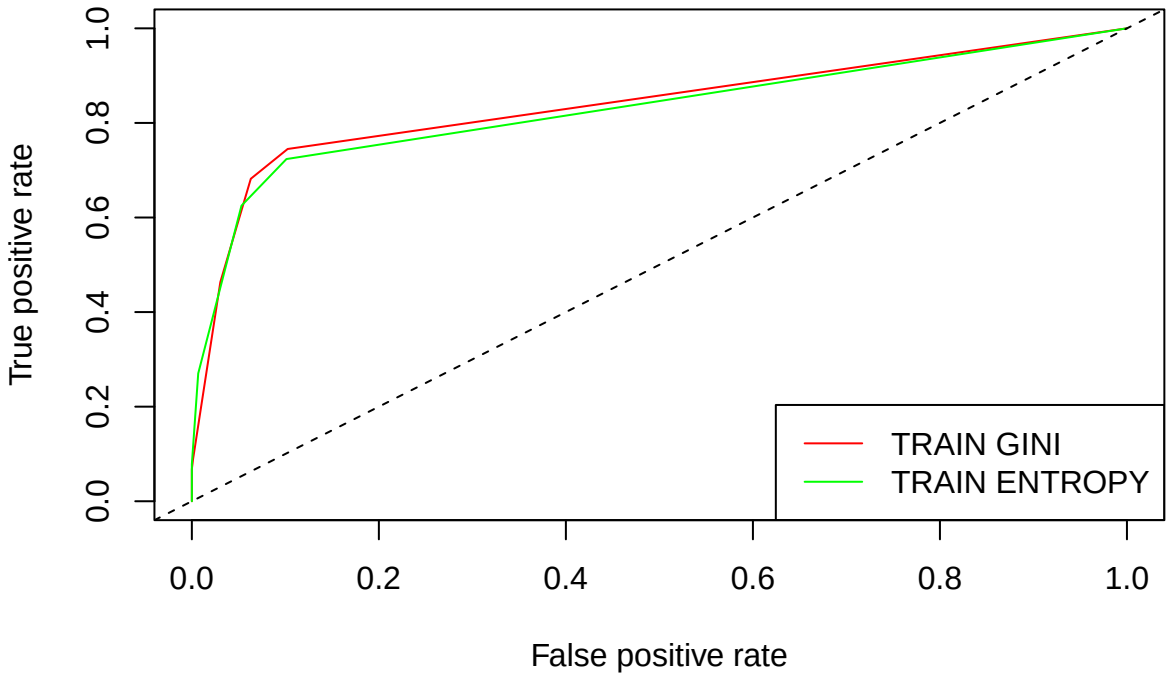
```
rpart.plot( t1E )
```




```
t1E$variable.importance
```

```
##   M_DEBTINC IMP_DEBTINC  IMP_DELINQ   IMP_CLAGE     M_VALUE        LOAN
##  612.835011  167.719280   63.177522   33.333233   30.508164   28.038709
##   IMP_DEROG IMP_MORTDUE    IMP_CLNO     IMP_YOJ   IMP_VALUE
##   10.906729    8.018814    2.955603    2.725719    2.528728
```

```
#Using the training data set, create a ROC curve for both trees
pG = predict( t1G, df_train )
pG2 = prediction( pG[,2], df_train$TARGET_BAD_FLAG )
pG3 = performance( pG2, "tpr", "fpr" )

pE = predict( t1E, df_train )
pE2 = prediction( pE[,2], df_train$TARGET_BAD_FLAG )
pE3 = performance( pE2, "tpr", "fpr" )

plot( pG3, col = "red" )
plot( pE3, col = "green", add = TRUE )
abline( 0,1, lty =2 )
legend( "bottomright", c("TRAIN GINI", "TRAIN ENTROPY"),
        col = c("red", "green"), bty = "y", lty =1)
```




```
aucG = performance( pG2, "auc" )@y.values
aucE = performance( pE2, "auc" )@y.values

print( paste("TRAIN AUC GINI = ", aucG) )
```

```
## [1] "TRAIN AUC GINI =  0.837997956799064"
```

```
print( paste("TRAIN AUC ENTROPY = ", aucE) )
```

```
## [1] "TRAIN AUC ENTROPY =  0.828926379614558"
```

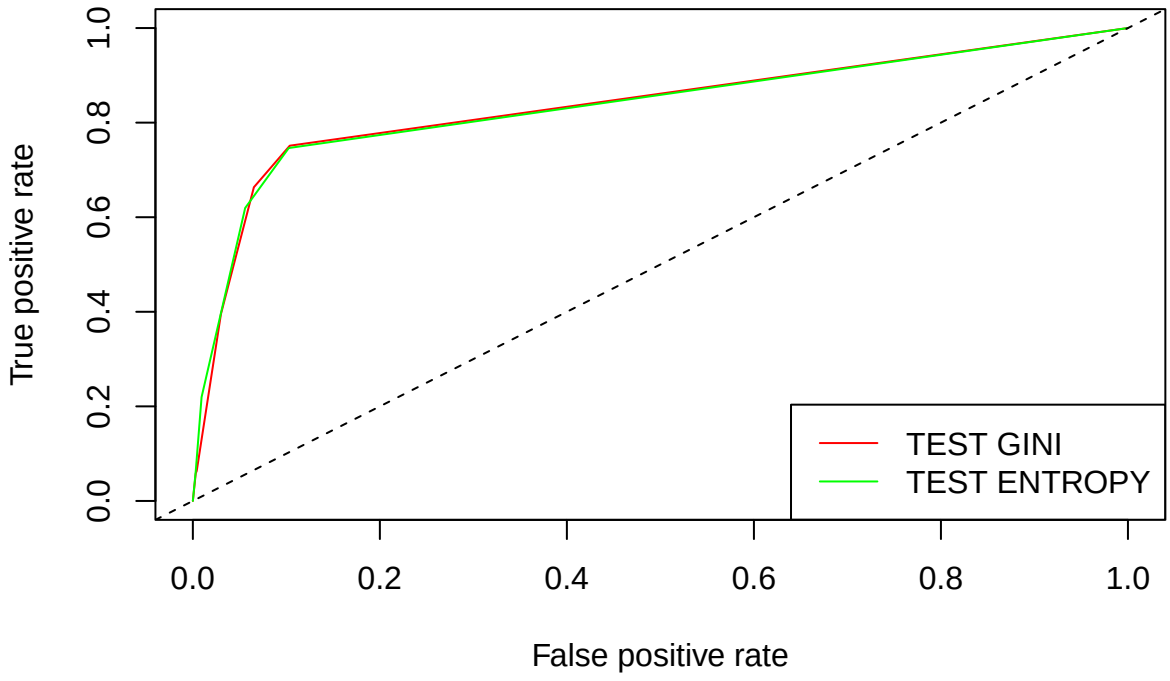
```
fG = predict( t1G, df_train, type = "class" )
fE = predict( t1E, df_train, type = "class" )

table( fG, df_train$TARGET_BAD_FLAG )
```

```
##
## fG     0    1
##   0 3566  313
##   1  240  671
```

```
table( fE, df_train$TARGET_BAD_FLAG )
```

```
##
## fE     0    1
##   0 3604  369
##   1  202  615
```

```
#Using the testing data set, create a ROC curve for both trees
pG = predict( t1G, df_test )
pG2 = prediction( pG[,2], df_test$TARGET_BAD_FLAG )
pG3 = performance( pG2, "tpr", "fpr" )

pE = predict( t1E, df_test )
pE2 = prediction( pE[,2], df_test$TARGET_BAD_FLAG )
pE3 = performance( pE2, "tpr", "fpr" )

plot( pG3, col = "red" )
plot( pE3, col = "green", add = TRUE )
abline( 0,1, lty =2 )
legend( "bottomright", c("TEST GINI", "TEST ENTROPY"),
        col = c("red", "green"), bty = "y", lty =1)
```




```
aucG = performance( pG2, "auc" )@y.values
aucE = performance( pE2, "auc" )@y.values

print( paste("TEST AUC GINI = ", aucG) )
```

```
## [1] "TEST AUC GINI =  0.836734487552129"
```

```r
print( paste("TEST AUC ENTROPY = ", aucE) )
```

```
## [1] "TEST AUC ENTROPY =  0.835978769114116"
```

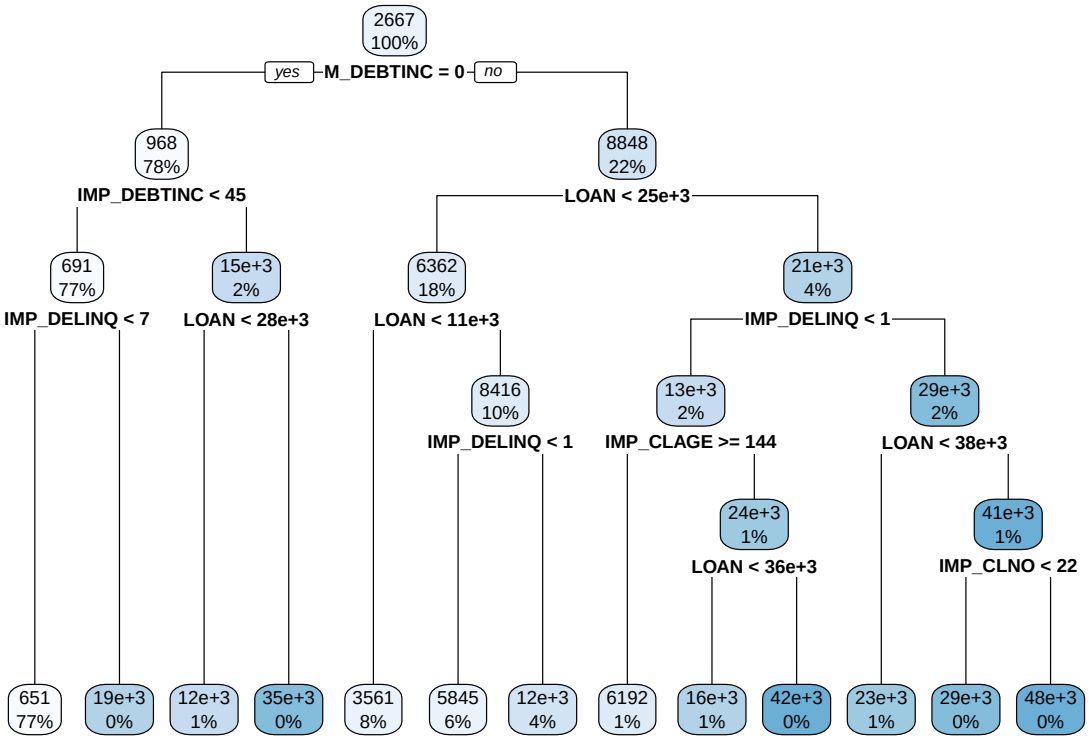
```r
fG = predict( t1G, df_test, type = "class" )
fE = predict( t1E, df_test, type = "class" )

table( fG, df_test$TARGET_BAD_FLAG )
```

```
##
## fG    0   1
##   0 902  69
##   1  63 136
```

```r
table( fE, df_test$TARGET_BAD_FLAG )
```

```
##
## fE    0   1
##   0 911  78
##   1  54 127
```

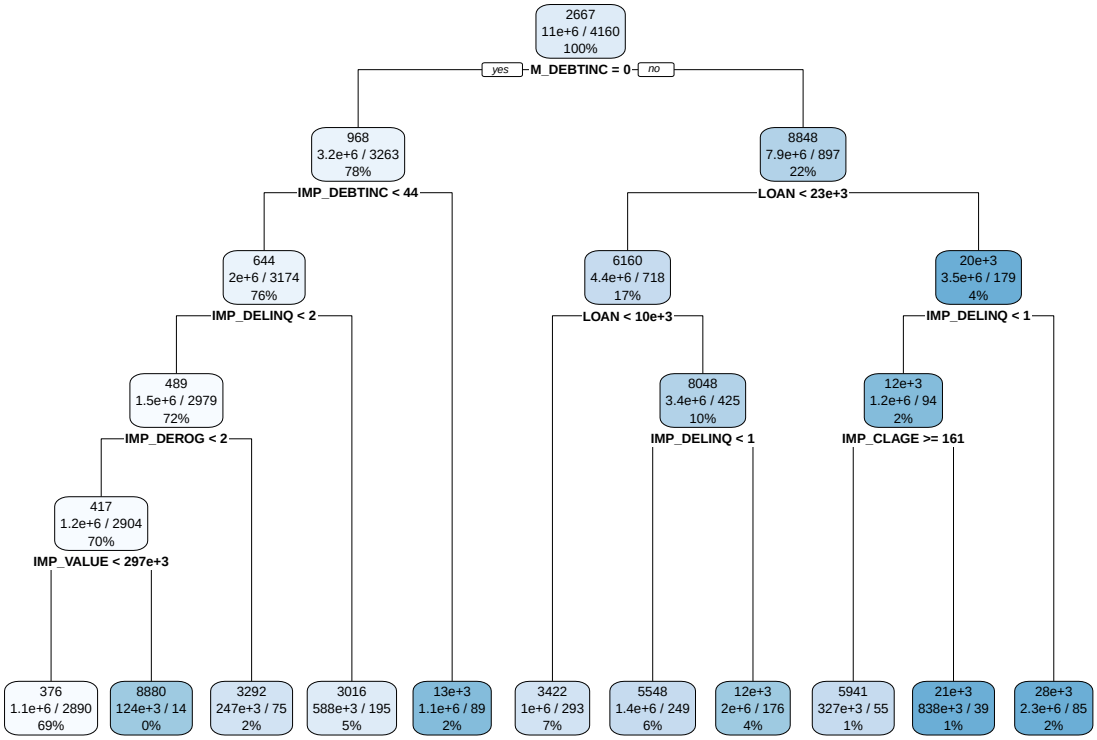
```r
#Write a brief summary of the decision trees discussing whether or not the trees are are optimal, overf
#The trees are optimal. From the training and test data, we can see that both ROC are above the dash li
#Both Gini and Entropy perform well.

#Rerun with different training and testing data at least three times.
#Determine which of the two models performed better and why you believe this
#I believe the Gini model is better after running the three times of training and test datasets.
#The Gini line is always above the line of entropy
#the auc of gini is around 0.85/0.84/0.86 while the auc of entropy is maller: 0.84/0.83/0.85

#Step 3: Regression Decision Tree
#Using the code discussed in the lecture, split the data into training and testing data sets.
#Use the rpart library to predict the variable TARGET_LOSS_AMT
df_amt = df
#Do not use TARGET_BAD_FLAG to predict TARGET_LOSS_AMT.
df_amt$TARGET_BAD_FLAG = NULL

FLAG = sample( c(TRUE, FALSE), nrow(df_amt), replace = TRUE, prob = c(0.7, 0.3))
df_train = df_amt[FLAG, ]
df_test = df_amt[!FLAG, ]

tr_set = rpart.control( maxdepth = 10 )

#Develop two decision trees, one using anova and the other using poisson
#All other parameters such as tree depth are up to you.
#Plot both decision trees
#List the important variables for both trees
t1a = rpart( data = df_train, TARGET_LOSS_AMT ~ ., control = tr_set, method = "anova" )
rpart.plot( t1a )
```




```
t1a$variable.importance
```

```
##                LOAN           M_DEBTINC           IMP_DELINQ         IMP_DEBTINC
##          46808720633         43679127893         18123397111         12774425817
##           IMP_VALUE             IMP_CLNO           IMP_CLAGE          IMP_MORTDUE
##          8623606946          8336244156          8203419604          8122593399
##            IMP_DEROG              M_VALUE FLAG.Reason.HomeImp              M_DEROG
##          4724706096          3851853441          2410606361          1042615601
## FLAG.Reason.DebtCon             IMP_YOJ            M_DELINQ               M_NINQ
##            897319515           843686265           838625592           611970026
##             IMP_NINQ             M_CLNO        FLAG.Job.Mgr
##            479893625           475976687           396941391
```

```
t1p = rpart( data = df_train, TARGET_LOSS_AMT ~ ., control = tr_set, method = "poisson" )
rpart.plot( t1p )
```




```
t1p$variable.importance
```

```
##          M_DEBTINC       IMP_DEBTINC              LOAN          IMP_DELINQ
##        12628595.91         4044579.20         3623156.22         2383614.85
##          IMP_VALUE          IMP_DEROG       IMP_MORTDUE            M_VALUE
##        1224052.25          858229.10          676485.88          597500.27
##          IMP_CLAGE           IMP_CLNO FLAG.Reason.HomeImp FLAG.Reason.DebtCon
##          455436.89          278187.39          146371.42          130847.18
##            M_DEROG           M_DELINQ             M_NINQ             M_CLNO
##          127408.70          101406.92           75405.15           59804.08
##            IMP_YOJ
##           13386.20
```

```
#Using the training data set, calculate the Root Mean Square Error (RMSE) for both trees
p1a = predict( t1a, df_train )
RMSE1a = sqrt ( mean( ( df_train$TARGET_LOSS_AMT - p1a )^2 ))

p1p = predict( t1p, df_train )
RMSE1p = sqrt ( mean( ( df_train$TARGET_LOSS_AMT - p1p )^2 ))

print( paste( "TRAIN RMSE ANOVA =", RMSE1a) )
```

```
## [1] "TRAIN RMSE ANOVA = 4659.97444808421"
```

```r
print( paste( "TRAIN RMSE POISSON =", RMSE1p) )
```

```
## [1] "TRAIN RMSE POISSON = 5083.97012780899"
```

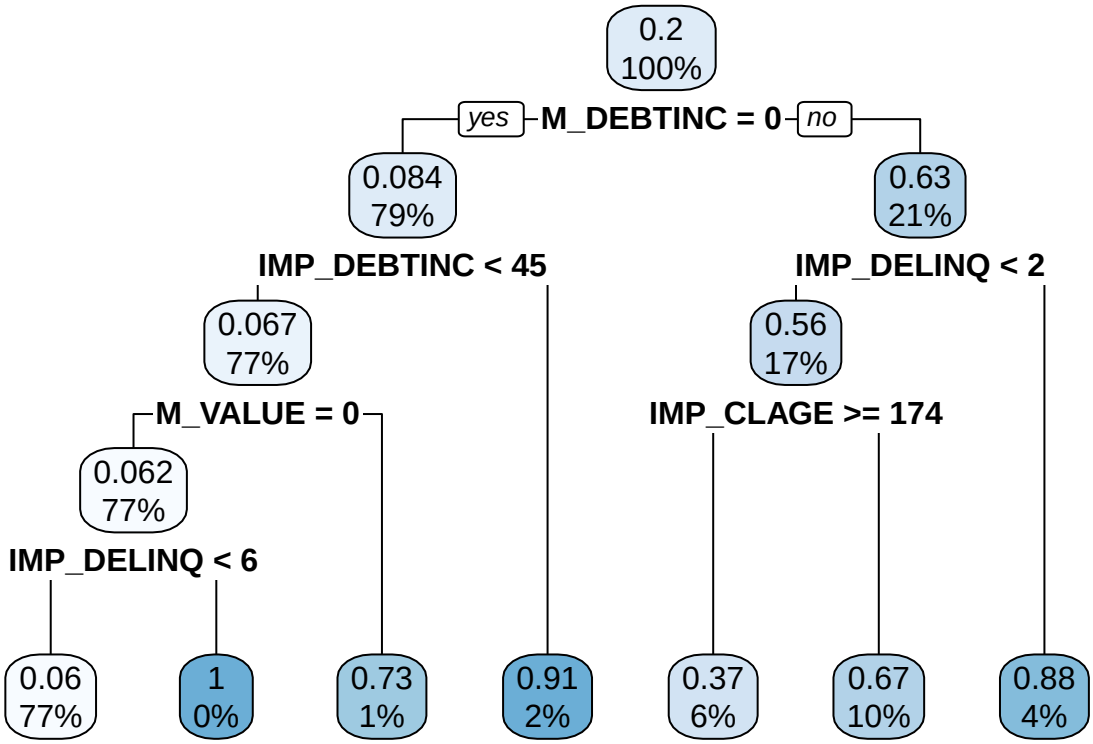
```r
#Using the testing data set, calculate the Root Mean Square Error (RMSE) for both trees
p1a = predict( t1a, df_test )
RMSE1a = sqrt ( mean( ( df_test$TARGET_LOSS_AMT - p1a )^2 ))

p1p = predict( t1p, df_test )
RMSE1p = sqrt ( mean( ( df_test$TARGET_LOSS_AMT - p1p )^2 ))

print( paste( "TEST RMSE ANOVA =", RMSE1a) )
```

```
## [1] "TEST RMSE ANOVA = 5433.3717476806"
```

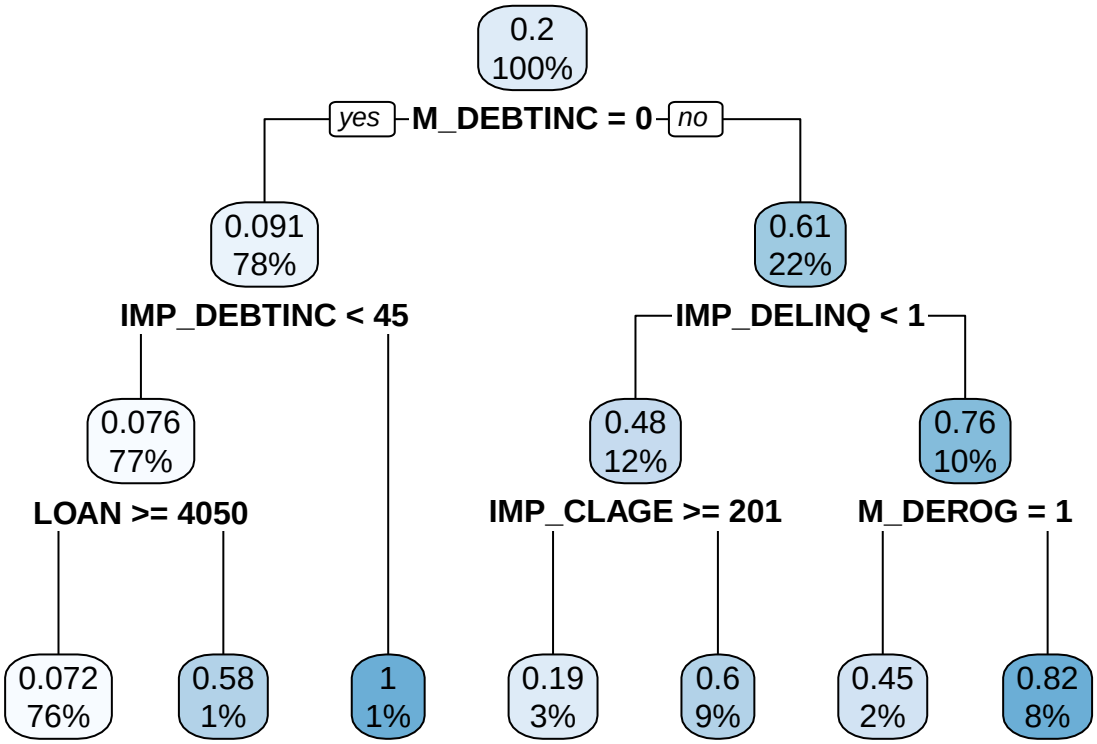
```r
print( paste( "TEST RMSE POISSON =", RMSE1p) )
```

```
## [1] "TEST RMSE POISSON = 5717.11433482924"
```

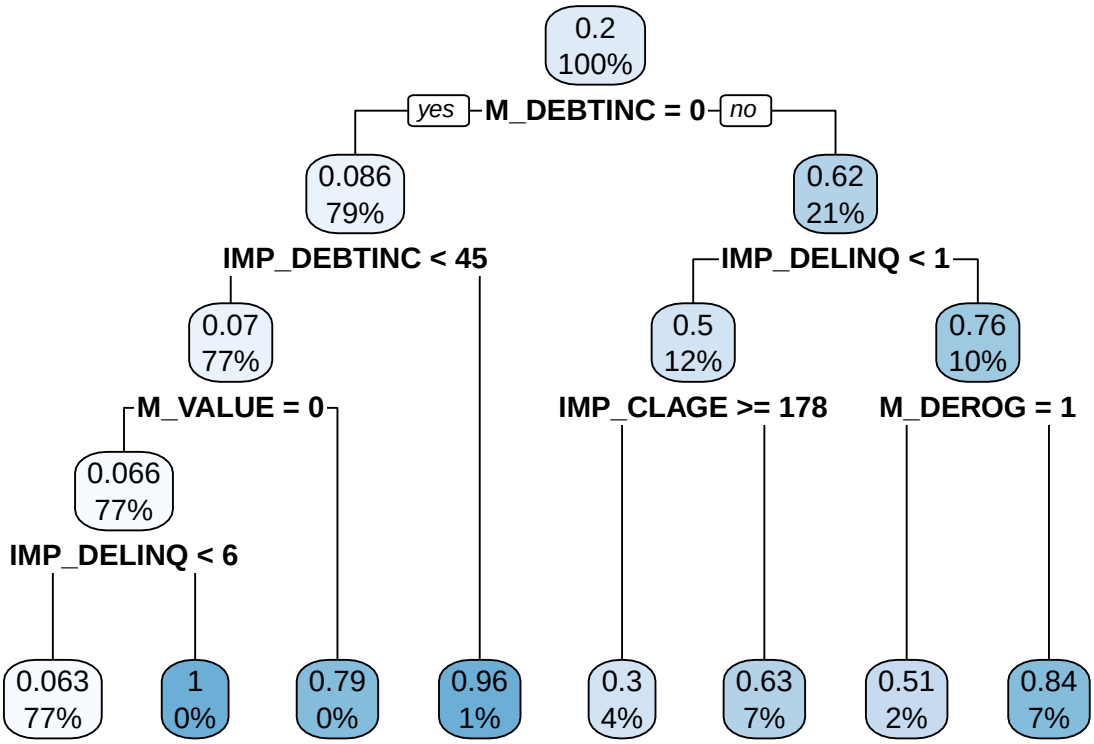
```r
#Write a brief summary of the decision trees discussing whether or not the trees are are optimal, overf
#Maybe the trees are bit underfitting because the RMSEs are larger than the mean.

#Rerun with different training and testing data at least three times.
#Determine which of the two models performed better and why you believe this
#Anova tree is better model
#since the RMSE of test data is always smaller than the poisson one.

#Step 4: Probability / Severity Model Decision Tree (Push Yourself!)
#Using the code discussed in the lecture, split the data into training and testing data sets.
#Use the rpart library to predict the variable TARGET_BAD_FLAG
df_flag = df
df_flag$TARGET_LOSS_AMT = NULL

FLAG = sample( c(TRUE, FALSE), nrow(df_flag), replace = TRUE, prob = c(0.7, 0.3))
df_train = df_flag[FLAG, ]
df_test = df_flag[!FLAG, ]

tr_set = rpart.control( maxdepth = 10 )

#train data
t2_f_train = rpart( data = df_train, TARGET_BAD_FLAG ~ ., control = tr_set )
rpart.plot( t2_f_train )
```




```
p2_f_train = predict( t2_f_train, df )


#test data
t2_f_test = rpart( data = df_test, TARGET_BAD_FLAG ~ ., control = tr_set )
rpart.plot( t2_f_test )
```




```
p2_f_test = predict( t2_f_test, df )

#all data
t2_f_all = rpart( data = df_flag, TARGET_BAD_FLAG ~ ., control = tr_set )
rpart.plot( t2_f_all )
```




```
p2_f_all = predict( t2_f_all, df )

head(p2_f_train)
```

```
##         1         2         3         4         5         6
## 0.66742597 0.88172043 0.66742597 0.66742597 0.66742597 0.05951643
```

```
head(p2_f_test)
```

```
##         1         2         3         4         5         6
## 0.5973154 0.8239437 0.5973154 0.4516129 0.5973154 0.5833333
```

```
head(p2_f_all)
```

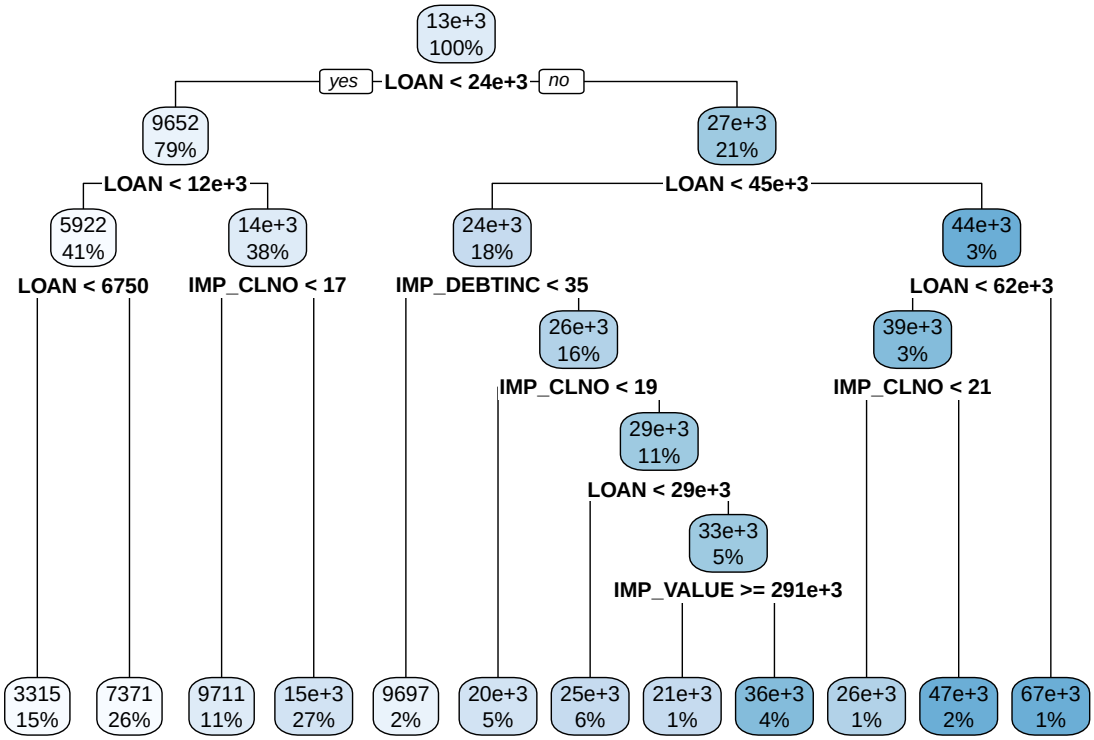
```
##         1         2         3         4         5         6
## 0.63084112 0.83710407 0.63084112 0.50769231 0.63084112 0.06344345
```

```
#Use the rpart library to predict the variable TARGET_LOSS_AMT using only records where TARGET_BAD_FLAG
df_amt_2 = subset( df, TARGET_BAD_FLAG == 1)
df_amt_2$TARGET_BAD_FLAG = NULL
head(df_amt_2)
```

```
##   TARGET_LOSS_AMT LOAN IMP_MORTDUE M_MORTDUE IMP_VALUE M_VALUE IMP_YOJ M_YOJ
```

```
## 1             641 1100       25860         0     39025       0    10.5     0
## 2            1109 1300       70053         0     68400       0     7.0     0
## 3             767 1500       13500         0     16700       0     4.0     0
## 4            1425 1500       65000         1     89000       1     7.0     1
## 6             335 1700       30548         0     40320       0     9.0     0
## 7            1841 1800       48649         0     57037       0     5.0     0
##   IMP_DEROG M_DEROG IMP_DELINQ M_DELINQ IMP_CLAGE M_CLAGE IMP_NINQ M_NINQ
## 1         0       0          0        0  94.36667       0        1      0
## 2         0       0          2        0 121.83333       0        0      0
## 3         0       0          0        0 149.46667       0        1      0
## 4         1       1          1        1 174.00000       1        1      1
## 6         0       0          0        0 101.46600       0        1      0
## 7         3       0          2        0  77.10000       0        1      0
##   IMP_CLNO M_CLNO IMP_DEBTINC M_DEBTINC FLAG.Job.Mgr FLAG.Job.Office
## 1        9      0    35.00000         1            0               0
## 2       14      0    35.00000         1            0               0
## 3       10      0    35.00000         1            0               0
## 4       20      1    35.00000         1            0               0
## 6        8      0    37.11361         0            0               0
## 7       17      0    35.00000         1            0               0
##   FLAG.Job.Other FLAG.Job.ProfExe FLAG.Job.Sales FLAG.Job.Self
## 1              1                0              0             0
## 2              1                0              0             0
## 3              1                0              0             0
## 4              0                0              0             0
## 6              1                0              0             0
## 7              1                0              0             0
##   FLAG.Reason.DebtCon FLAG.Reason.HomeImp
## 1                   0                   1
## 2                   0                   1
## 3                   0                   1
## 4                   0                   0
## 6                   0                   1
## 7                   0                   1
```

```
t2_a = rpart( data = df_amt_2, TARGET_LOSS_AMT ~ .,
              control = tr_set, method = "anova" )
rpart.plot( t2_a )
```




```
p2_a = predict ( t2_a, df )

#List the important variables for both trees
t2_f_all$variable.importance
```

```
##   M_DEBTINC IMP_DEBTINC  IMP_DELINQ     M_VALUE   IMP_CLAGE     M_DEROG
## 285.0105051  64.2695360  38.6857592  25.6672429  18.0381475  15.6708280
##        LOAN   IMP_DEROG    M_DELINQ      M_NINQ      M_CLNO     M_CLAGE
##  12.8228373  11.2507816  10.3565554   8.5221495   6.9916002   4.8633155
##   IMP_VALUE     IMP_YOJ    IMP_CLNO IMP_MORTDUE       M_YOJ
##   4.2755103   2.1618753   1.4187307   0.8107033   0.2515508
```

```
t2_a$variable.importance
```

```
##               LOAN          IMP_VALUE        IMP_MORTDUE           IMP_CLNO
##         97537912195         16788808741         12468593613         11071238709
##         IMP_DEBTINC FLAG.Reason.HomeImp FLAG.Reason.DebtCon           IMP_NINQ
##          8436327252          4166142650          3945210843          2770249169
##           IMP_DELINQ           IMP_CLAGE           M_MORTDUE             IMP_YOJ
##          2167739871          2052513664          1321525422           742530790
##      FLAG.Job.Other     FLAG.Job.ProfExe             M_CLAGE           IMP_DEROG
##           306916520           175320382           140520186            88840773
```

```r
#Using your models, predict the probability of default and the loss given default.
#Multiply the two values together for each record.
p2 = p2_f_all * p2_a
head( p2 )
```

```
##         1         2         3         4         5         6
## 2091.4450 2775.2742 2091.4450 1683.1663 2091.4450  210.3358
```

```r
#Calculate the RMSE value for the Probability / Severity model.
RMSE2 = sqrt( mean( (df$TARGET_LOSS_AMT - p2 )^2 ))

print(RMSE1a)
```

```
## [1] 5433.372
```

```r
print(RMSE1p)
```

```
## [1] 5717.114
```

```r
print(RMSE2)
```

```
## [1] 4867.296
```

```r
#Rerun at least three times to be assured that the model is optimal and not over fit or under fit.
#Comment on how this model compares to using the model from Step 3. Which one would your recommend usin
#Step 4 is recommended with lower RMSE: 4867. On step 3, RMSE1a and RMSE1p were larger.
```