Rui_Peng_Week_5.R

raypeng

2025-04-12

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

library( randomForest )
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library( gbm )
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

```
SEED = 1
set.seed( SEED )

#Read the data into R

TARGET = "TARGET_BAD_FLAG"

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
```

```r
#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 Models
#Using the code discussed in the lecture, split the data into training and testing data sets.

df_flag = df
df_flag$TARGET_LOSS_AMT = NULL #Do not use TARGET_LOSS_AMT to predict TARGET_BAD_FLAG.

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, ]

dim(df_flag)
```

```
## [1] 5960   28
```

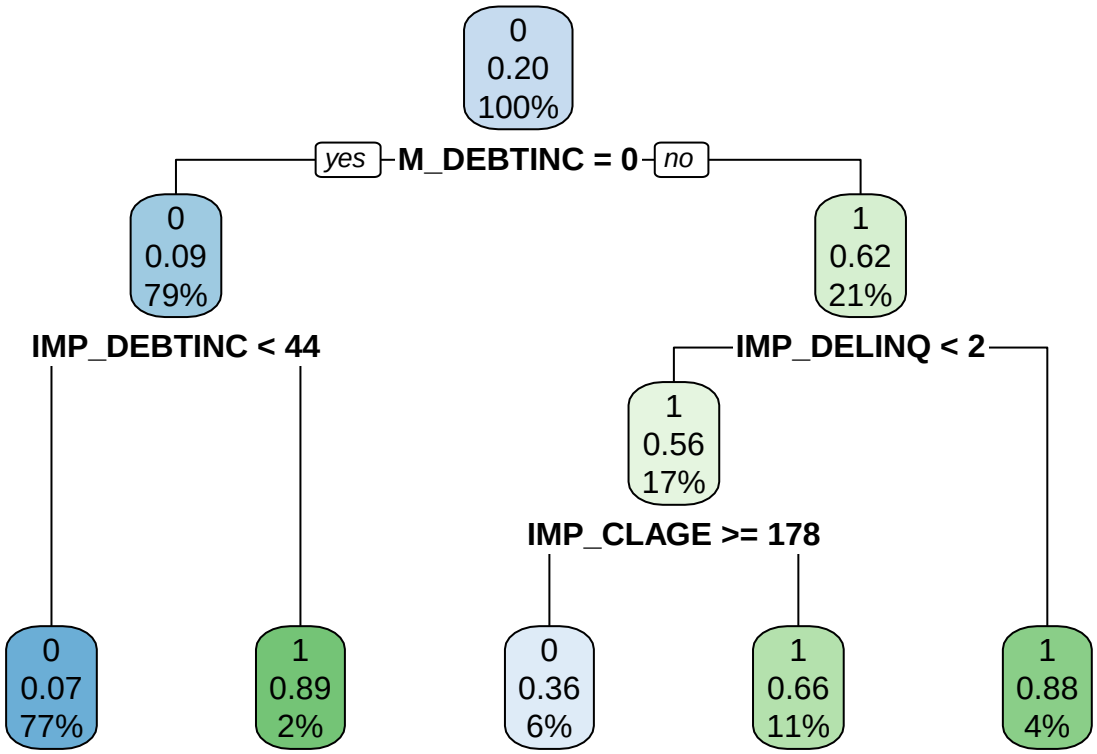
```
dim(df_train)
```

```
## [1] 4142   28
```

```
dim(df_test)
```

```
## [1] 1818   28
```

```
#Create a Decision Tree model using the rpart library to predict the variable TARGET_BAD_FLAG
#Plot the Decision Tree and list the important variables for the tree.

#Decision Tree Model
tr_set = rpart.control( maxdepth = 10 ) #All model parameters such as tree depth are up to you.
tr_model = rpart( data = df_train, TARGET_BAD_FLAG ~ .,
                  control = tr_set, method = "class", parms = list(split = 'information'))
rpart.plot( tr_model )
```




```
tr_model$variable.importance
```

```
##   M_DEBTINC IMP_DEBTINC  IMP_DELINQ   IMP_CLAGE        LOAN     M_VALUE
##  533.397481  134.588883   46.494397   30.749923   24.521888   22.199895
##   IMP_VALUE IMP_MORTDUE    IMP_CLNO     IMP_YOJ
##    7.967967    5.783975    2.459994    2.090995
```

```
pt = predict( tr_model, df_test, type = "prob" )
head( pt )
```

```
##            0          1
## 4  0.3354839 0.66451613
## 6  0.9315112 0.06848885
## 7  0.1206897 0.87931034
## 15 0.3354839 0.66451613
## 17 0.1206897 0.87931034
## 18 0.9315112 0.06848885
```

```
pt2 = prediction( pt[,2], df_test$TARGET_BAD_FLAG )
pt3 = performance( pt2, "tpr", "fpr" )

#Create a Random Forest model using the randomForest library to predict the variable TARGET_BAD_FLAG
#List the important variables for the Random Forest and include the variable importance plot.
```

```r
#Random Forest Model
rf_model = randomForest( data = df_train, TARGET_BAD_FLAG ~ .,
                         ntree = 100, importance = TRUE )
```

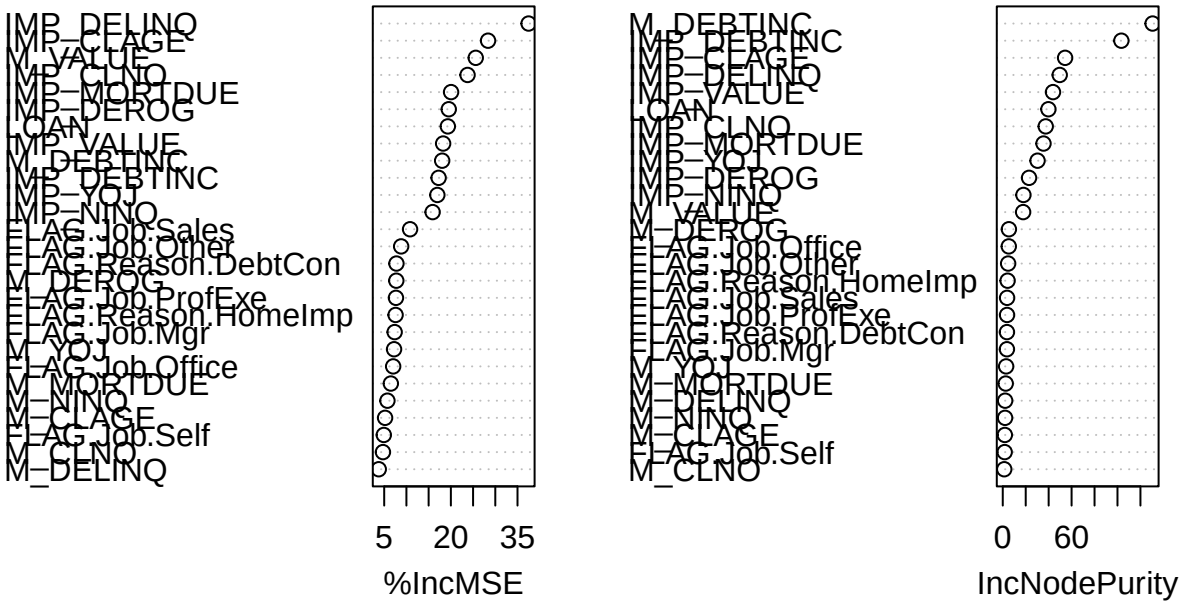
```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```r
importance( rf_model )
```

```
##                       %IncMSE IncNodePurity
## LOAN                19.316810     39.777176
## IMP_MORTDUE         20.052341     35.518005
## M_MORTDUE            6.499260      2.582903
## IMP_VALUE           18.264039     43.787225
## M_VALUE             25.602248     17.743863
## IMP_YOJ             16.963341     30.322241
## M_YOJ                7.220923      2.943678
## IMP_DEROG           19.530010     22.914985
## M_DEROG              7.729186      5.338867
## IMP_DELINQ          37.494775     49.645744
## M_DELINQ             3.780398      2.083958
## IMP_CLAGE           28.371343     54.329981
## M_CLAGE              5.201136      1.794735
## IMP_NINQ            15.901124     17.995586
## M_NINQ               5.727939      2.007916
## IMP_CLNO            23.760935     37.449920
## M_CLNO               4.719829      1.292987
## IMP_DEBTINC         17.246641    103.381548
## M_DEBTINC           18.045350    130.614728
## FLAG.Job.Mgr         7.353261      3.612471
## FLAG.Job.Office      7.020243      5.254487
## FLAG.Job.Other       8.815549      4.689699
## FLAG.Job.ProfExe     7.649276      3.742588
## FLAG.Job.Sales      10.823283      3.779676
## FLAG.Job.Self        4.910753      1.675588
## FLAG.Reason.DebtCon  7.756487      3.643055
## FLAG.Reason.HomeImp  7.565180      4.219131
```

```r
varImpPlot( rf_model )
```

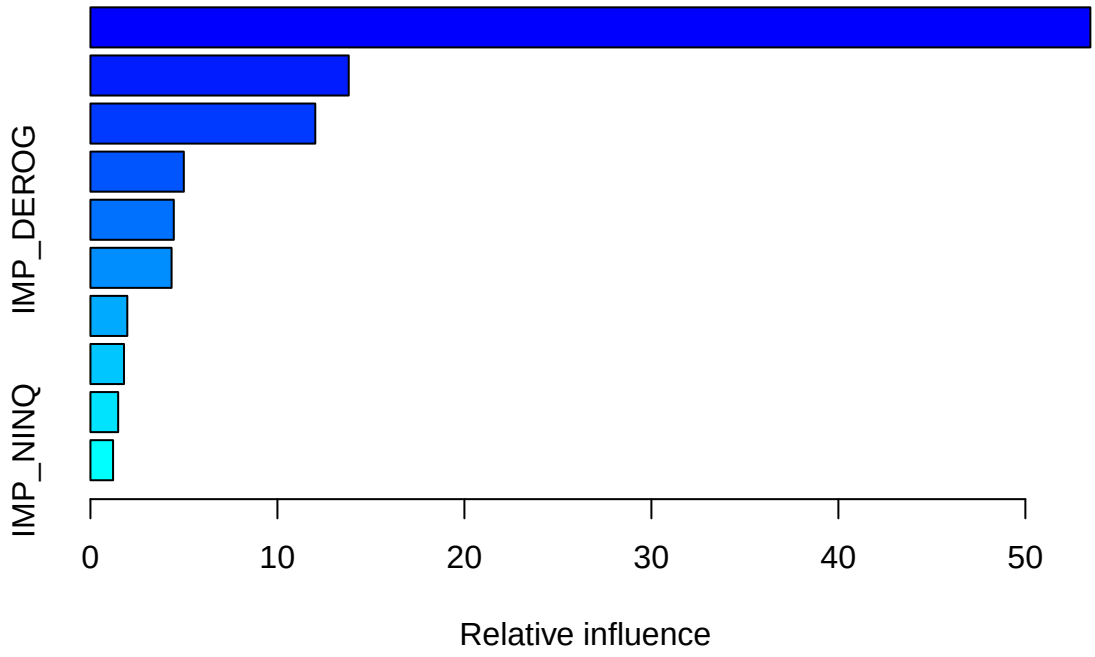
rf_model




```
pr = predict( rf_model, df_test )
head( pr )
```

```
##         4         6         7        15        17        18
## 0.7853333 0.8671667 0.9460000 0.8203333 0.9213333 0.3466667
```

```
pr2 = prediction ( pr, df_test$TARGET_BAD_FLAG )
pr3 = performance( pr2, "tpr", "fpr" )

#Create a Gradient Boosting model using the gbm library to predict the variable TARGET_BAD_FLAG
#List the important variables for the Gradient Boosting model and include the variable importance plot.

#Gradient Boosting Model
gb_model = gbm( data = df_train, TARGET_BAD_FLAG ~ ., n.trees = 100,
                distribution = "bernoulli" )
summary.gbm( gb_model, cBars = 10 )
```




```
##                                  var    rel.inf
## M_DEBTINC                 M_DEBTINC 53.4761627
## IMP_DEBTINC             IMP_DEBTINC 13.8130176
## IMP_DELINQ               IMP_DELINQ 12.0257247
## M_VALUE                     M_VALUE  4.9940416
## IMP_DEROG                 IMP_DEROG  4.4580798
## IMP_CLAGE                 IMP_CLAGE  4.3403733
## IMP_VALUE                 IMP_VALUE  1.9703911
## LOAN                           LOAN  1.7950994
## IMP_CLNO                   IMP_CLNO  1.4838466
## IMP_NINQ                   IMP_NINQ  1.2106400
## FLAG.Job.Sales       FLAG.Job.Sales  0.2176216
## M_DEROG                     M_DEROG  0.2150017
## IMP_MORTDUE             IMP_MORTDUE  0.0000000
## M_MORTDUE                 M_MORTDUE  0.0000000
## IMP_YOJ                     IMP_YOJ  0.0000000
## M_YOJ                         M_YOJ  0.0000000
## M_DELINQ                   M_DELINQ  0.0000000
## M_CLAGE                     M_CLAGE  0.0000000
## M_NINQ                       M_NINQ  0.0000000
## M_CLNO                       M_CLNO  0.0000000
## FLAG.Job.Mgr           FLAG.Job.Mgr  0.0000000
## FLAG.Job.Office     FLAG.Job.Office  0.0000000
## FLAG.Job.Other       FLAG.Job.Other  0.0000000
## FLAG.Job.ProfExe   FLAG.Job.ProfExe  0.0000000
## FLAG.Job.Self         FLAG.Job.Self  0.0000000
```

```
## FLAG.Reason.DebtCon FLAG.Reason.DebtCon  0.0000000
## FLAG.Reason.HomeImp FLAG.Reason.HomeImp  0.0000000
```

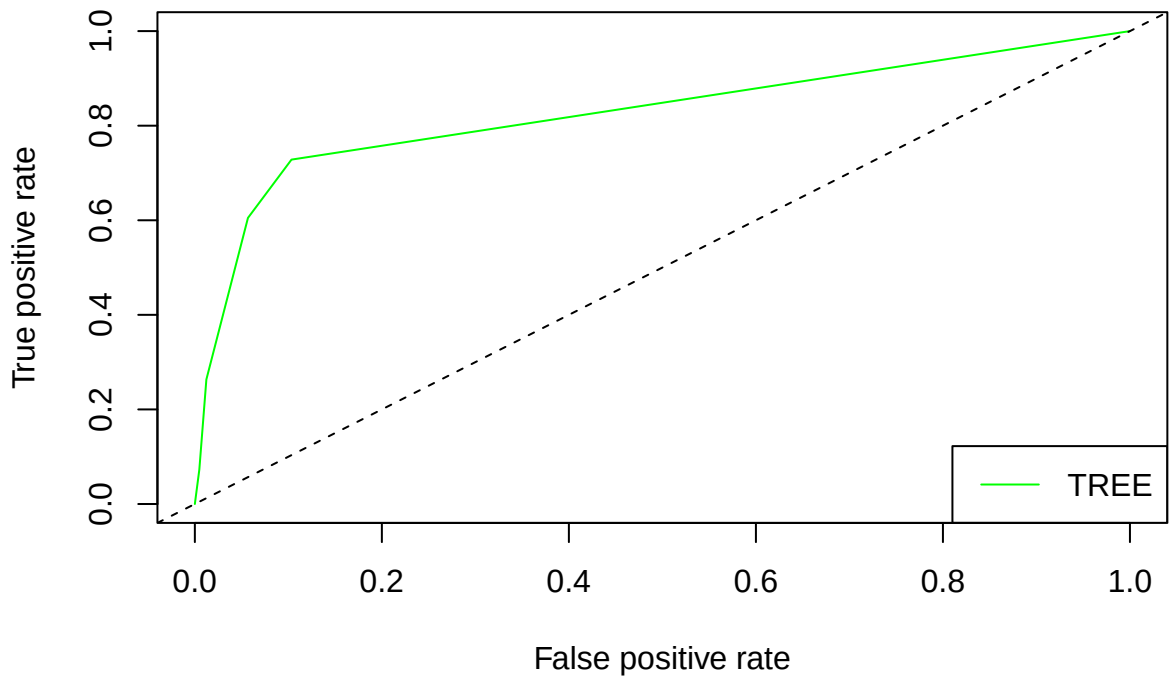
```
pg = predict( gb_model, df_test, type = "response" )
```

```
## Using 100 trees...
```

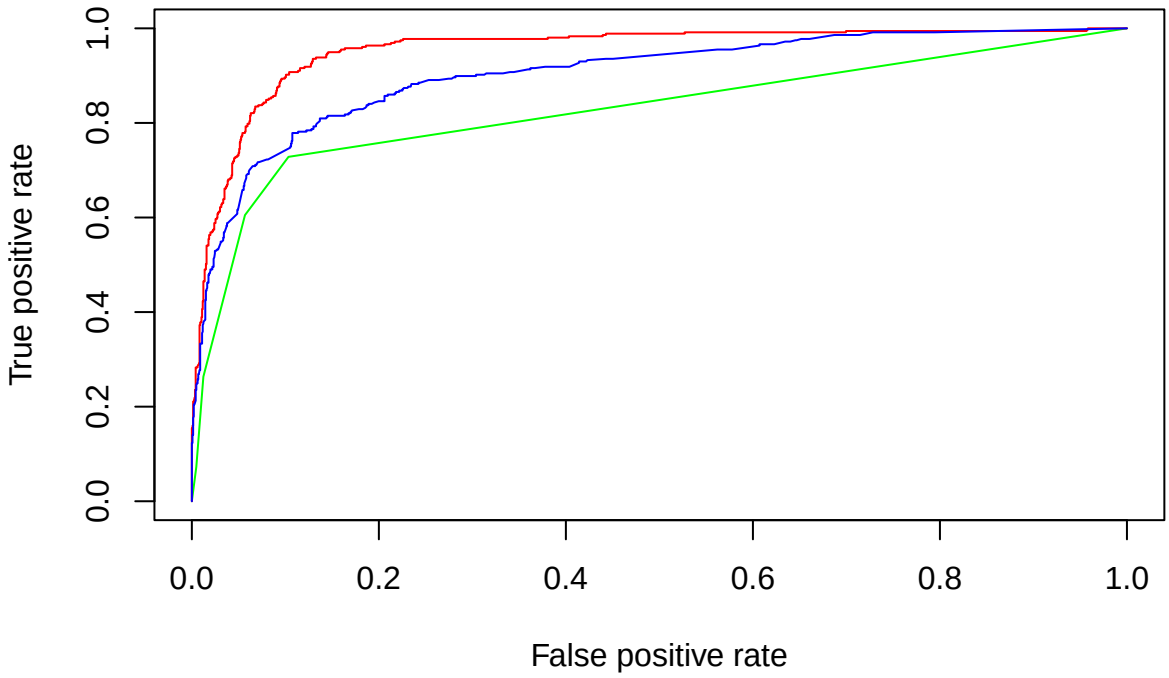
```
head(pg)
```

```
## [1] 0.8970211 0.1571409 0.9489942 0.6561628 0.9719267 0.5539409
```

```
pg2 = prediction( pg, df_test$TARGET_BAD_FLAG )
pg3 = performance( pg2, "tpr", "fpr" )

#Using the testing data set, create a ROC curves for all models. They must all be on the same plot.
plot( pt3, col = "green" )
abline( 0, 1, lty = 2 )
legend( "bottomright", c("TREE"), col = c("green"), bty = "y", lty =1 )
```




```
plot( pt3, col = "green" )
plot( pr3, col = "red", add = TRUE )
plot( pg3, col = "blue", add = TRUE )
```




```
#Display the Area Under the ROC curve (AUC) for all models.
aucT = performance( pt2, "auc" )@y.values
aucR = performance( pr2, "auc" )@y.values
aucG = performance( pg2, "auc" )@y.values

print( paste( "Decision Tree AUC = ", aucT ))
```

```
## [1] "Decision Tree AUC =  0.826618121581281"
```

```
print( paste( "Random Forest AUC = ", aucR ))
```

```
## [1] "Random Forest AUC =  0.95370673936926"
```

```
print( paste( "Gradient Boosting AUC = ", aucG ))
```

```
## [1] "Gradient Boosting AUC =  0.903390103474655"
```

```
#Rerun with different training and testing data at least three times.
#Determine which model performed best and why you believe this.
#Write a brief summary of which model you would recommend using.

#Random Forest Model is the best for this case and it has largest AUC.
#I recommend Random Forest as shown with the red line on the ROC curve chart.
```

```r
#"Decision Tree AUC is around 0.8266"
#"Random Forest AUC is around  0.9537"
#"Gradient Boosting AUC is around  0.9034"


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

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, ]

mean( df_amt$TARGET_LOSS_AMT )
```

```
## [1] 2676.163
```

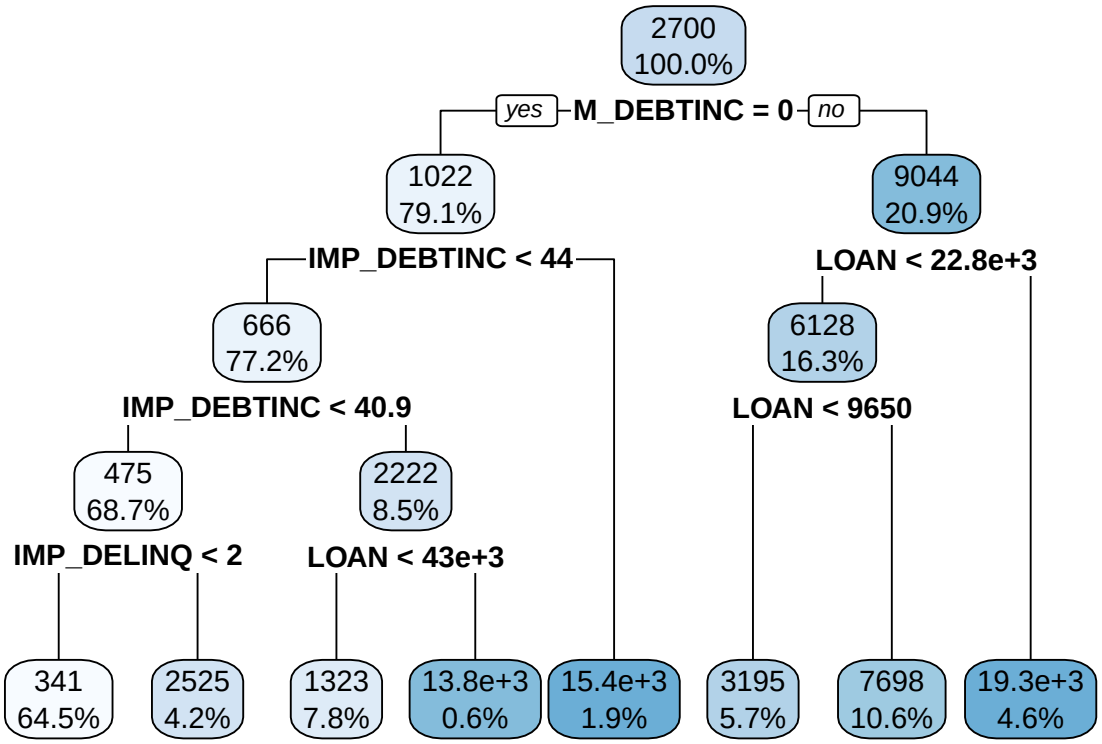
```r
mean( df_train$TARGET_LOSS_AMT )
```

```
## [1] 2699.644
```

```r
mean( df_test$TARGET_LOSS_AMT )
```

```
## [1] 2623.669
```

```r
#Create a Decision Tree model using the rpart library to predict the variable TARGET_LOSS_AMT
#Plot the Decision Tree and list the important variables for the tree.

#Decision Tree Model
tr_set = rpart.control( maxdepth = 10 ) #All model parameters such as tree depth are up to you.
tr_model = rpart( data = df_train, TARGET_LOSS_AMT ~ .,
                  control = tr_set, method = "poisson" )
rpart.plot( tr_model, digits = 3, extra = 100 )
```




```
tr_model$variable.importance
```

```
##           M_DEBTINC         IMP_DEBTINC                LOAN         IMP_DELINQ
##        12363304.653         5716473.850         4157682.841         1119800.042
##           IMP_VALUE             M_VALUE         IMP_MORTDUE          IMP_DEROG
##          512294.607          358980.971          312273.555          215388.583
##             IMP_YOJ FLAG.Reason.HomeImp             IMP_NINQ          IMP_CLAGE
##          103404.145           47981.060           32855.648           12564.283
## FLAG.Reason.DebtCon             IMP_CLNO
##           11995.265            7197.159
```

```
pt = predict( tr_model, df_test )
head(pt)
```

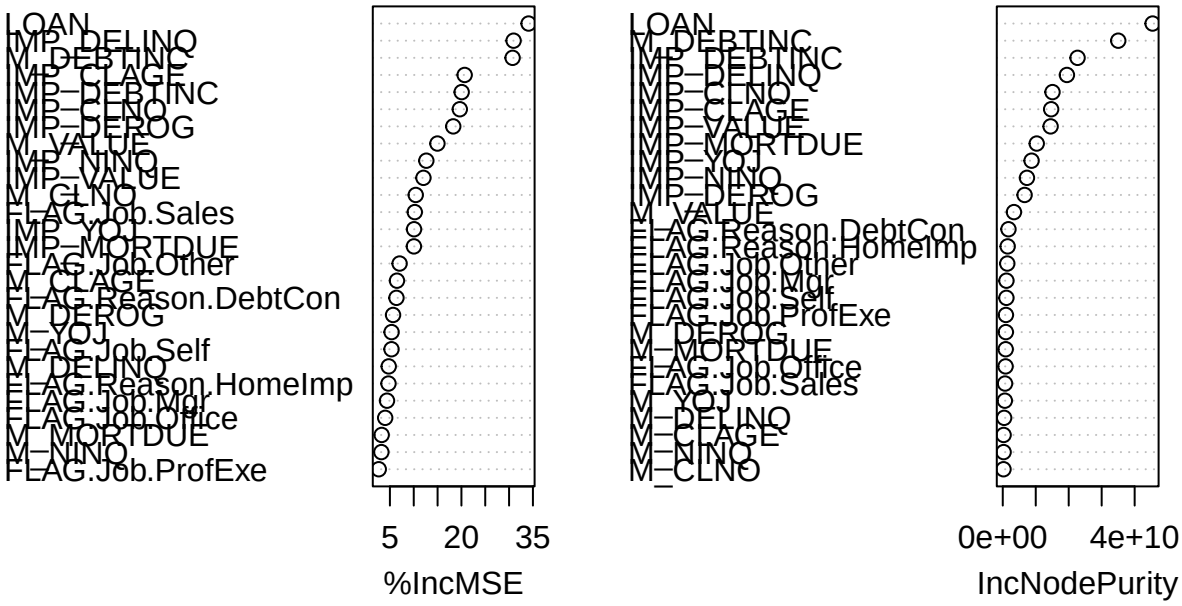
```
##         1         2         6         8         9        10
## 3194.7642 3194.7642  341.1016  341.1016 3194.7642 3194.7642
```

```
RMSEt = sqrt( mean( (df_test$TARGET_LOSS_AMT - pt )^2 ) )

#Create a Random Forest model using the randomForest library to predict the variable TARGET_LOSS_AMT
#List the important variables for the Random Forest and include the variable importance plot.

#Random Forest Model
rf_model = randomForest( data = df_train, TARGET_LOSS_AMT ~ .,
```

```
                        ntree = 200, importance = TRUE )
importance( rf_model )
```

```
##                        %IncMSE IncNodePurity
## LOAN                 34.095576   45449205487
## IMP_MORTDUE          10.031460   10299000629
## M_MORTDUE             3.272255     889087966
## IMP_VALUE            12.031425   14509861931
## M_VALUE              14.966826    3397182762
## IMP_YOJ              10.062998    8741387415
## M_YOJ                 5.350942     631963812
## IMP_DEROG            18.283119    6623629143
## M_DEROG               5.657055     934822320
## IMP_DELINQ           30.946549   19455210412
## M_DELINQ              4.752220     417930848
## IMP_CLAGE            20.679289   14679814958
## M_CLAGE               6.480203     252365511
## IMP_NINQ             12.639399    7331302802
## M_NINQ                3.211669     252070959
## IMP_CLNO             19.663777   15079995783
## M_CLNO               10.417420     223326328
## IMP_DEBTINC          20.055383   22682382302
## M_DEBTINC            30.740130   35032540126
## FLAG.Job.Mgr          4.388293    1114118538
## FLAG.Job.Office       4.010110     856388951
## FLAG.Job.Other        7.036022    1375449784
## FLAG.Job.ProfExe      2.662832     948947638
## FLAG.Job.Sales       10.216173     692099911
## FLAG.Job.Self         5.334734    1101319723
## FLAG.Reason.DebtCon   6.373739    1729522425
## FLAG.Reason.HomeImp   4.702067    1487337399
```

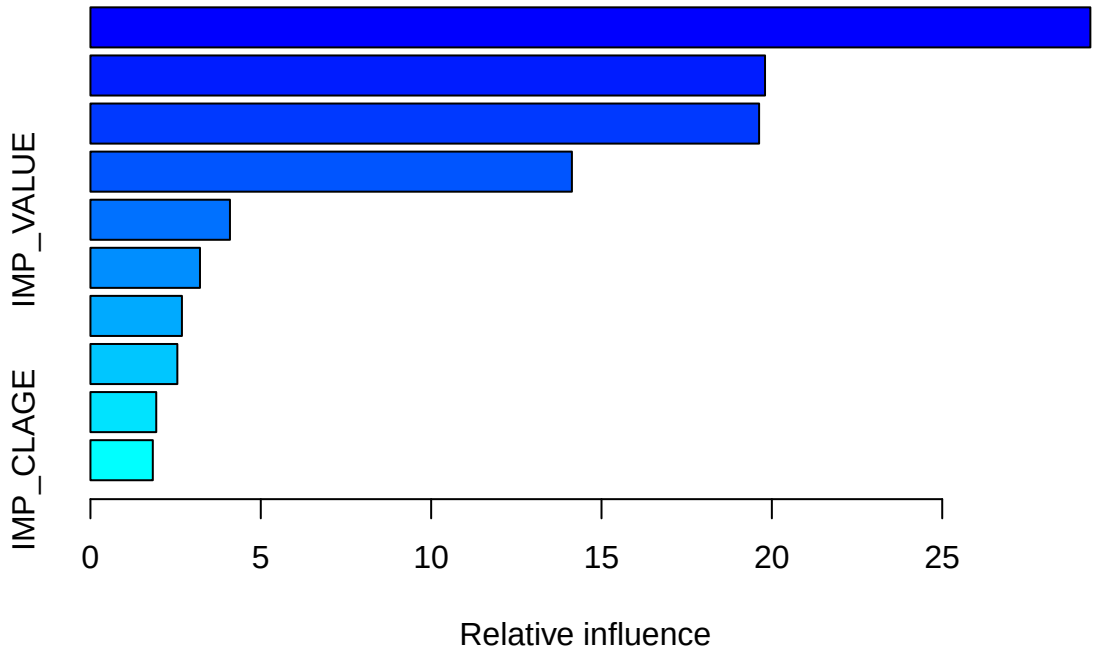
```
varImpPlot( rf_model )
```




```
pr = predict( rf_model, df_test )
head(pr)
```

```
##         1         2         6         8         9        10
## 2177.7672 2577.7647  876.1110  896.7708 2886.8135 2295.3772
```

```
RMSEr = sqrt( mean( (df_test$TARGET_LOSS_AMT - pr )^2 ) )

#Create a Gradient Boosting model using the gbm library to predict the variable TARGET_LOSS_AMT
#List the important variables for the Gradient Boosting model and include the variable importance plot.

#Gradient Boosting Model

gb_model = gbm( data = df_train, TARGET_LOSS_AMT ~ .,
                          n.trees = 200, distribution = "poisson" )
summary.gbm( gb_model, cBars = 10 )
```




```
##                                    var     rel.inf
## M_DEBTINC                   M_DEBTINC 29.34815372
## IMP_DEBTINC                 IMP_DEBTINC 19.79951192
## LOAN                              LOAN 19.62604762
## IMP_DELINQ                  IMP_DELINQ 14.13141517
## IMP_VALUE                   IMP_VALUE  4.09495584
## IMP_DEROG                   IMP_DEROG  3.21537820
## IMP_CLNO                    IMP_CLNO  2.68520122
## M_VALUE                     M_VALUE  2.55105222
## IMP_NINQ                    IMP_NINQ  1.93134692
## IMP_CLAGE                   IMP_CLAGE  1.83083110
## FLAG.Job.Self             FLAG.Job.Self  0.20529953
## FLAG.Reason.HomeImp FLAG.Reason.HomeImp  0.19693045
## FLAG.Job.Sales           FLAG.Job.Sales  0.19309198
## IMP_MORTDUE                 IMP_MORTDUE  0.11048387
## FLAG.Reason.DebtCon FLAG.Reason.DebtCon  0.08030023
## M_MORTDUE                   M_MORTDUE  0.00000000
## IMP_YOJ                     IMP_YOJ  0.00000000
## M_YOJ                       M_YOJ  0.00000000
## M_DEROG                     M_DEROG  0.00000000
## M_DELINQ                    M_DELINQ  0.00000000
## M_CLAGE                     M_CLAGE  0.00000000
## M_NINQ                      M_NINQ  0.00000000
## M_CLNO                      M_CLNO  0.00000000
## FLAG.Job.Mgr              FLAG.Job.Mgr  0.00000000
## FLAG.Job.Office         FLAG.Job.Office  0.00000000
```

```
## FLAG.Job.Other           FLAG.Job.Other  0.00000000
## FLAG.Job.ProfExe       FLAG.Job.ProfExe  0.00000000
```

```
pg = predict( gb_model, df_test, type = "response" )
```

```
## Using 200 trees...
```

```
head(pg)
```

```
## [1] 3057.7739 4763.5344  370.8974  387.9944 3201.9446 3624.5679
```

```
RMSEg = sqrt( mean( (df_test$TARGET_LOSS_AMT - pg )^2 ) )

#Using the testing data set, calculate the Root Mean Square Error (RMSE) for all models.

print( paste( "Decision Tree RMSE =", RMSEt ))
```

```
## [1] "Decision Tree RMSE = 5288.36862187847"
```

```
print( paste( "Random Forest RMSE =", RMSEr ))
```

```
## [1] "Random Forest RMSE = 4201.17775586498"
```

```
print( paste( "Gradient Boosting RMSE =", RMSEg ))
```

```
## [1] "Gradient Boosting RMSE = 5344.21076557694"
```

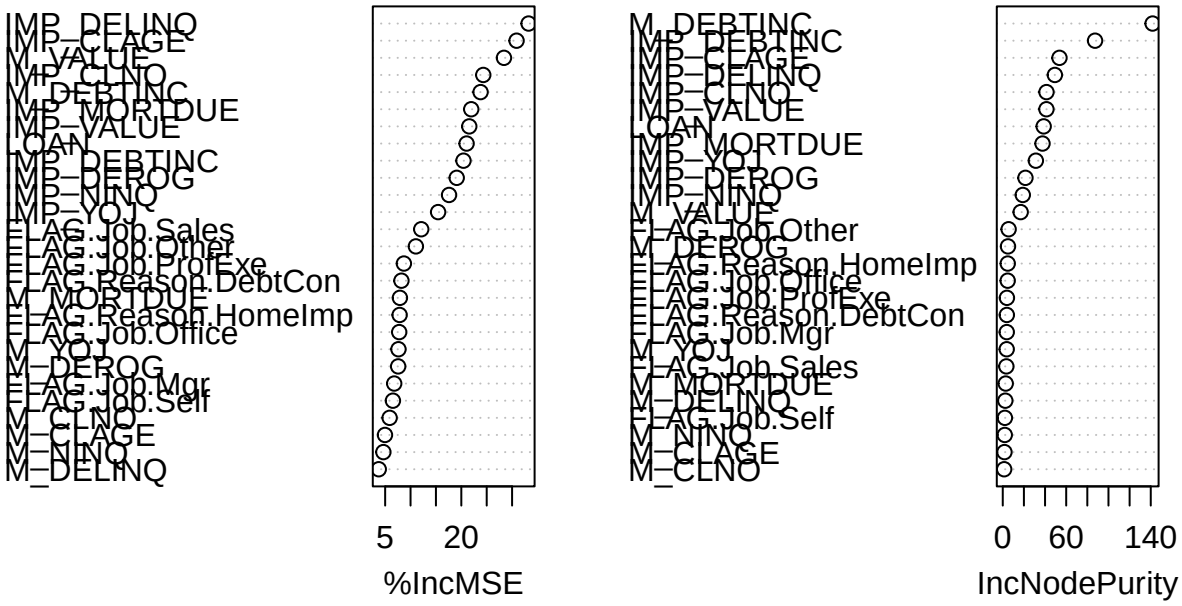
```
#Rerun with different training and testing data at least three times.
#Determine which model performed best and why you believe this.
#Write a brief summary of which model you would recommend using. Note that this is your opinion. There

#The best model is Random Forest one. This has smallest RMSE compared to two others.
#"Decision Tree RMSE is around 5288/5288/5288/5288"
#"Random Forest RMSE is around 4232/4244/4259/4210"
#"Gradient Boosting RMSE is around 5890/5256/5579/6122"

#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 any model from Step 2 in order 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 )

#Random Forest Model 2
rf_model = randomForest( data = df_train, TARGET_BAD_FLAG ~ .,
                         ntree = 100, importance = TRUE )
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
importance( rf_model )
```

```
##                        %IncMSE IncNodePurity
## LOAN                 21.040423     38.702766
## IMP_MORTDUE          21.950699     37.594898
## M_MORTDUE             7.902418      2.803608
## IMP_VALUE            21.562051     41.323318
## M_VALUE              28.395454     16.942045
## IMP_YOJ              15.427740     31.302449
## M_YOJ                 7.618381      3.722218
## IMP_DEROG            19.091163     21.454557
## M_DEROG               7.560507      4.926909
## IMP_DELINQ           33.250516     49.277899
## M_DELINQ              3.739354      2.409904
## IMP_CLAGE            30.879954     53.648308
## M_CLAGE               4.959649      1.615230
## IMP_NINQ             17.582364     18.945744
## M_NINQ                4.645149      1.890865
## IMP_CLNO             24.318657     41.366187
## M_CLNO                5.879180      1.326957
## IMP_DEBTINC          20.422332     87.312831
## M_DEBTINC            23.788028    141.751541
## FLAG.Job.Mgr          6.780649      3.785483
## FLAG.Job.Office       7.766323      4.651602
## FLAG.Job.Other       11.049067      5.533509
## FLAG.Job.ProfExe      8.686985      3.934495
## FLAG.Job.Sales       12.105524      3.407349
## FLAG.Job.Self         6.524638      1.998294
## FLAG.Reason.DebtCon   8.203827      3.840146
## FLAG.Reason.HomeImp   7.851941      4.702795
```

```
varImpPlot( rf_model )
```




```
pr_flag = predict( rf_model, df_test )
head( pr_flag )
```

```
##         1         5        10        11        14        15
## 0.8951667 0.7046667 0.5379444 0.8005000 0.5261667 0.5225000
```

```
pr2_flag = prediction ( pr_flag, df_test$TARGET_BAD_FLAG )
pr3_flag = performance( pr2, "tpr", "fpr" )

#Develop three models to predict the variable TARGET_LOSS_AMT
#using only records where TARGET_BAD_FLAG is 1.

df_amt = subset( df, TARGET_BAD_FLAG == 1)
df_amt$TARGET_BAD_FLAG = NULL
head(df_amt)
```

```
##   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
```

```
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, ]

mean( df_amt$TARGET_LOSS_AMT )
```

```
## [1] 13414.58
```

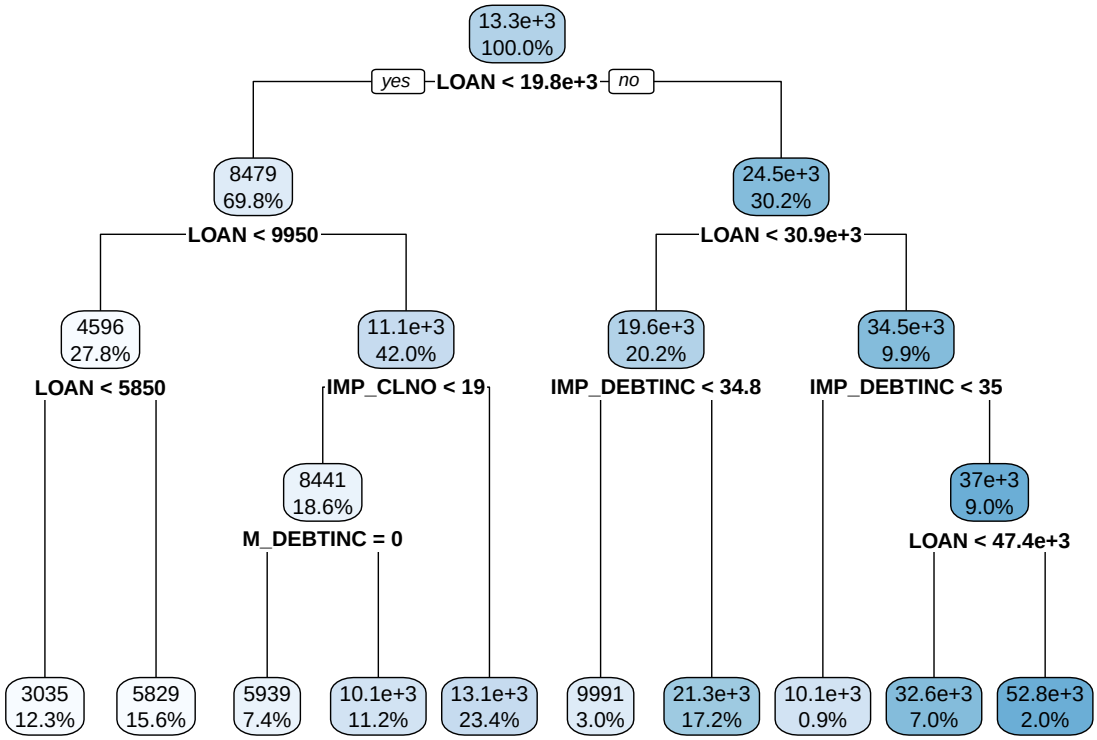
```
mean( df_train$TARGET_LOSS_AMT )
```

```
## [1] 13324.08
```

```
mean( df_test$TARGET_LOSS_AMT )
```

```
## [1] 13646.25
```

```
#Decision Tree Model
tr_set = rpart.control( maxdepth = 10 ) #All model parameters such as tree depth are up to you.
tr_model = rpart( data = df_train, TARGET_LOSS_AMT ~ .,
                  control = tr_set, method = "poisson" )
rpart.plot( tr_model, digits = 3, extra = 100 )
```




```
tr_model$variable.importance
```

```
##                LOAN           IMP_VALUE         IMP_MORTDUE         IMP_DEBTINC
##         4659207.175          1014609.754           748066.552           522019.727
## FLAG.Reason.HomeImp             IMP_CLNO FLAG.Reason.DebtCon             IMP_YOJ
##           205419.878           191413.190           153664.227            97503.803
##            M_DEBTINC       FLAG.Job.Self            IMP_CLAGE            M_MORTDUE
##            81015.181            73346.770            71800.737            54411.434
##           IMP_DELINQ                M_NINQ
##            21573.088             3857.866
```

```
pt = predict( tr_model, df_test )
head(pt)
```

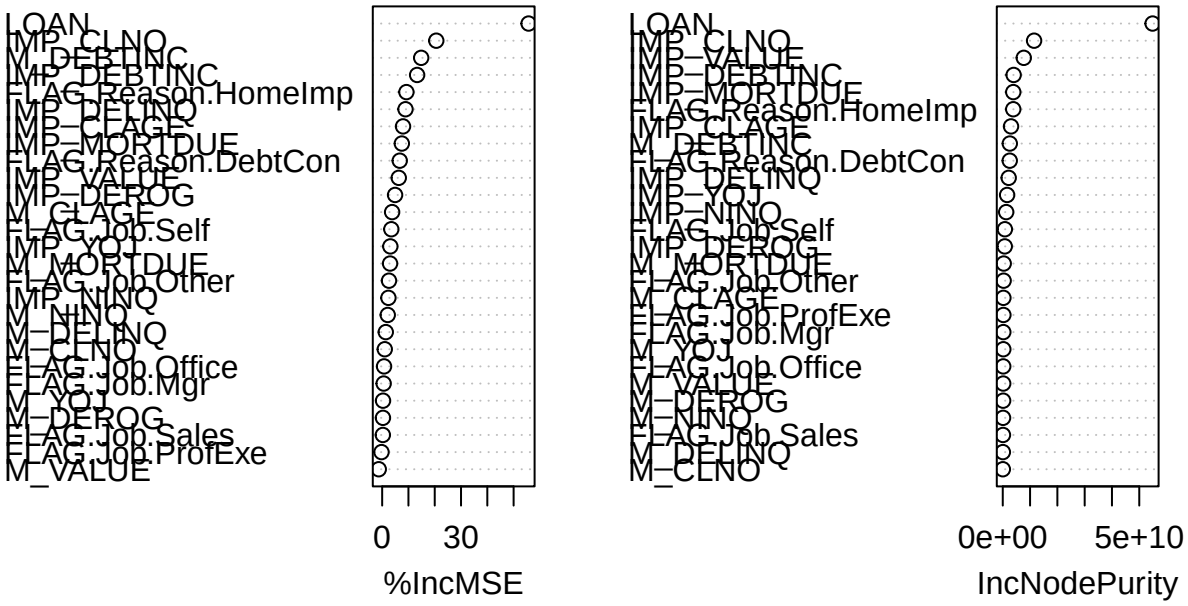
```
##        2        3        9       12       17       22
## 3034.769 3034.769 3034.769 3034.769 3034.769 3034.769
```

```
RMSEt = sqrt( mean( (df_test$TARGET_LOSS_AMT - pt )^2 ) )

#Random Forest Model
rf_model = randomForest( data = df_train, TARGET_LOSS_AMT ~ .,
                         ntree = 200, importance = TRUE )
importance( rf_model )
```

```
##                         %IncMSE IncNodePurity
## LOAN                 55.7003967   54884464841
## IMP_MORTDUE           7.4788166    3846126221
## M_MORTDUE             3.0110547     350989439
## IMP_VALUE             6.3139431    7695152610
## M_VALUE              -1.2896546     164673644
## IMP_YOJ               3.1128929    1599100009
## M_YOJ                 0.3250612     203786955
## IMP_DEROG             4.9098658     725458208
## M_DEROG               0.3023181     135883088
## IMP_DELINQ            8.8790367    2171496767
## M_DELINQ              1.3767135      39168805
## IMP_CLAGE             7.9354776    3020919598
## M_CLAGE               3.7930337     230988044
## IMP_NINQ              2.3772898    1252439275
## M_NINQ                2.1232401     105280917
## IMP_CLNO             20.5806120   11477290253
## M_CLNO                0.9838603      23292415
## IMP_DEBTINC          13.2995524    3982781808
## M_DEBTINC            14.8483233    2577892600
## FLAG.Job.Mgr          0.5773085     205258078
## FLAG.Job.Office       0.7166137     173599011
## FLAG.Job.Other        2.5968940     322514907
## FLAG.Job.ProfExe     -0.2498893     230611598
## FLAG.Job.Sales        0.2547774      66564285
## FLAG.Job.Self         3.4888510     770817837
## FLAG.Reason.DebtCon   6.6913670    2558685307
## FLAG.Reason.HomeImp   9.2288979    3838439976
```

```
varImpPlot( rf_model )
```

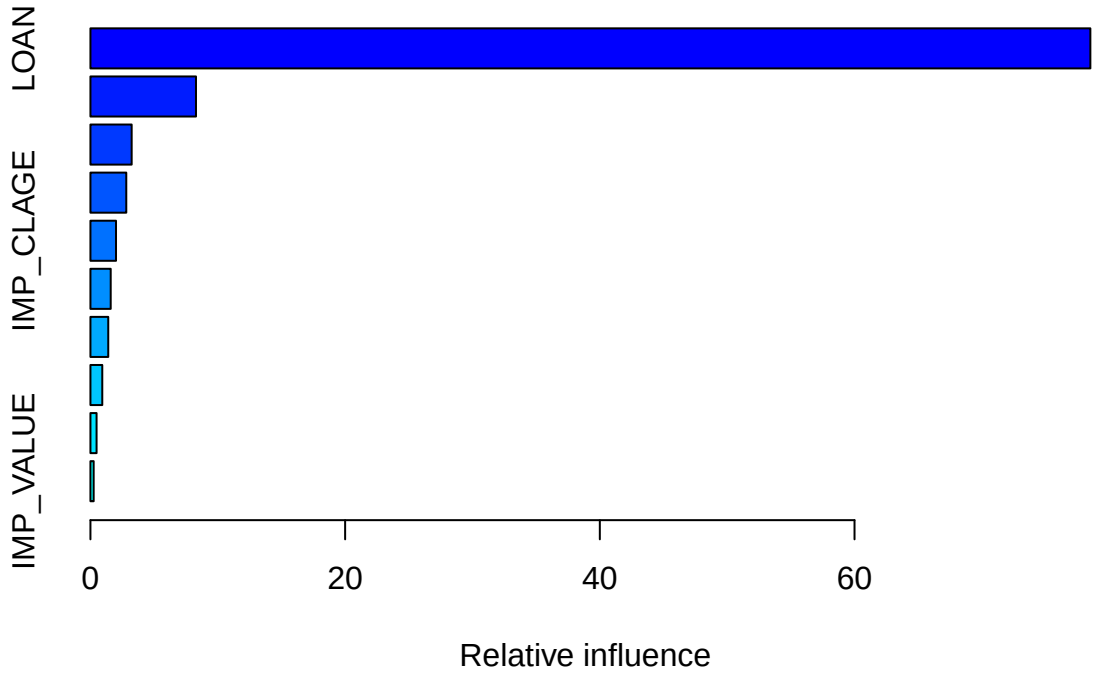
rf_model




```
pr = predict( rf_model, df_test )
head(pr)
```

```
##        2        3        9       12       17       22
## 2485.745 1991.662 2125.837 2090.861 6082.782 2919.759
```

```
RMSEr = sqrt( mean( (df_test$TARGET_LOSS_AMT - pr )^2 ) )

#Gradient Boosting Model
gb_model = gbm( data = df_train, TARGET_LOSS_AMT ~ .,
                n.trees = 200, distribution = "poisson" )
summary.gbm( gb_model, cBars = 10 )
```




```
##                                    var     rel.inf
## LOAN                              LOAN 78.52001190
## IMP_CLNO                      IMP_CLNO  8.28893643
## IMP_DEBTINC                IMP_DEBTINC  3.23636914
## M_DEBTINC                    M_DEBTINC  2.81526363
## IMP_CLAGE                    IMP_CLAGE  2.00912715
## IMP_DELINQ                  IMP_DELINQ  1.59135001
## FLAG.Reason.HomeImp FLAG.Reason.HomeImp  1.39944632
## FLAG.Job.Self             FLAG.Job.Self  0.92936055
## FLAG.Reason.DebtCon FLAG.Reason.DebtCon  0.48169063
## IMP_VALUE                    IMP_VALUE  0.25238137
## IMP_DEROG                    IMP_DEROG  0.16146174
## IMP_YOJ                        IMP_YOJ  0.15687381
## IMP_NINQ                      IMP_NINQ  0.07534070
## IMP_MORTDUE                IMP_MORTDUE  0.04862016
## FLAG.Job.Sales          FLAG.Job.Sales  0.03376646
## M_MORTDUE                    M_MORTDUE  0.00000000
## M_VALUE                        M_VALUE  0.00000000
## M_YOJ                            M_YOJ  0.00000000
## M_DEROG                        M_DEROG  0.00000000
## M_DELINQ                      M_DELINQ  0.00000000
## M_CLAGE                        M_CLAGE  0.00000000
## M_NINQ                          M_NINQ  0.00000000
## M_CLNO                          M_CLNO  0.00000000
## FLAG.Job.Mgr              FLAG.Job.Mgr  0.00000000
## FLAG.Job.Office        FLAG.Job.Office  0.00000000
```

```
## FLAG.Job.Other           FLAG.Job.Other  0.00000000
## FLAG.Job.ProfExe       FLAG.Job.ProfExe  0.00000000
```

```
pg = predict( gb_model, df_test, type = "response" )
```

```
## Using 200 trees...
```

```
head(pg)
```

```
## [1] 3395.791 2765.632 3098.706 2939.846 4762.968 1545.195
```

```
RMSEg = sqrt( mean( (df_test$TARGET_LOSS_AMT - pg )^2 ) )


print( paste( "Decision Tree RMSE =", RMSEt ))
```

```
## [1] "Decision Tree RMSE = 6516.1724031693"
```

```
print( paste( "Random Forest RMSE =", RMSEr ))
```

```
## [1] "Random Forest RMSE = 3575.91158446549"
```

```
print( paste( "Gradient Boosting RMSE =", RMSEg ))
```

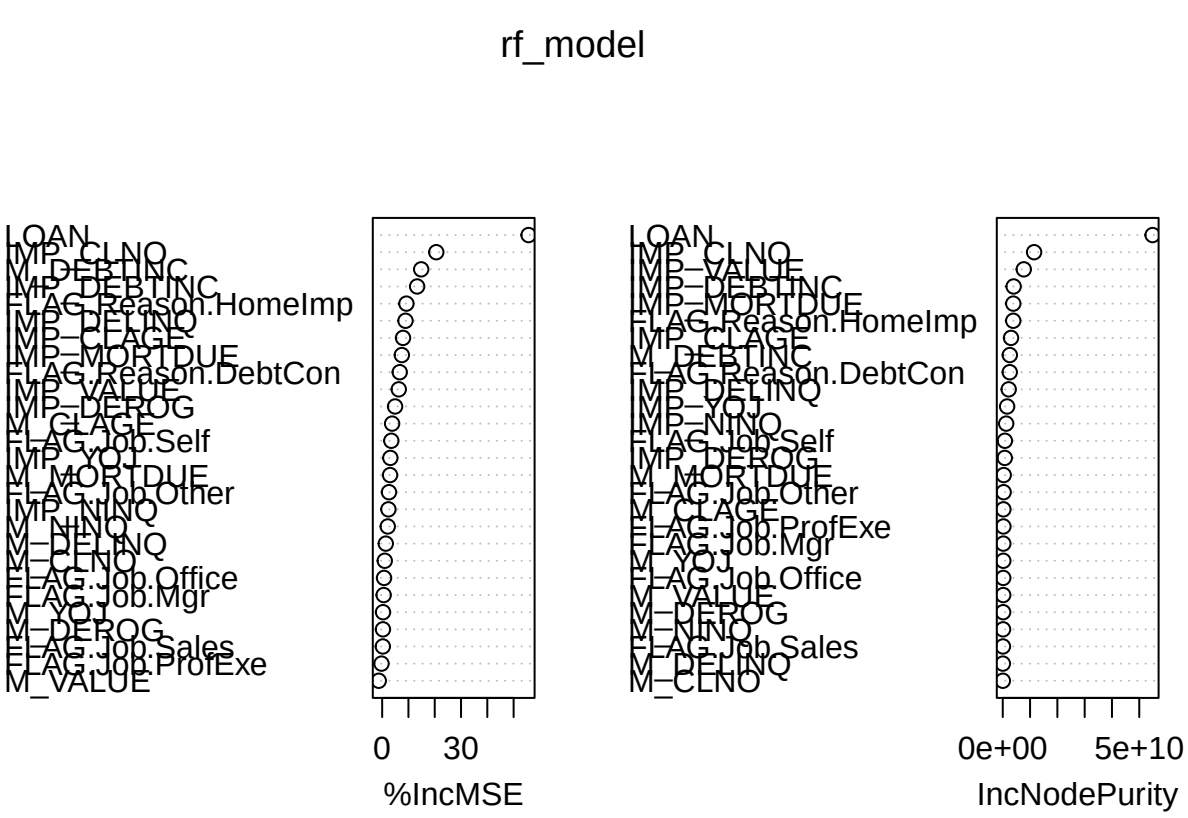
```
## [1] "Gradient Boosting RMSE = 3152.1597415062"
```

```
#Select one of the models to predict damage.
#I would choose Gradient Boosting since it has the smallest RMSE.
#"Decision Tree RMSE = 5326/5752/6632/6389"
#"Random Forest RMSE = 3547/3442/4289/5170"
#"Gradient Boosting RMSE = 3500/3430/3562/4260"

#List the important variables for both models.
importance( rf_model )
```

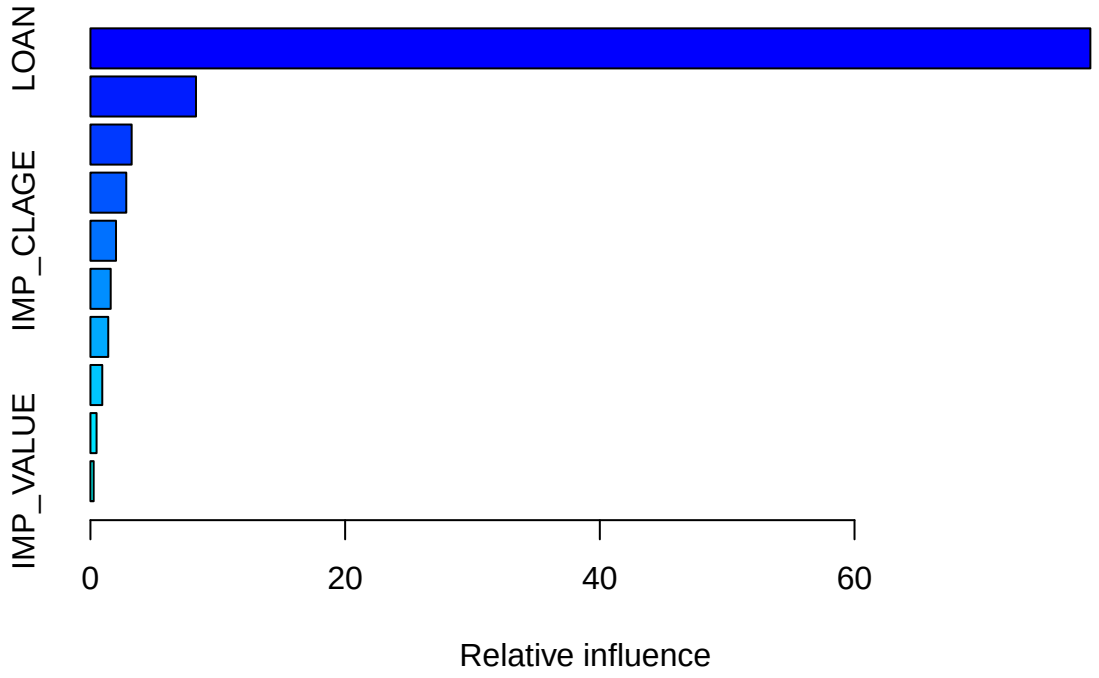
```
##                      %IncMSE IncNodePurity
## LOAN                55.7003967   54884464841
## IMP_MORTDUE          7.4788166    3846126221
## M_MORTDUE            3.0110547     350989439
## IMP_VALUE            6.3139431    7695152610
## M_VALUE             -1.2896546     164673644
## IMP_YOJ              3.1128929    1599100009
## M_YOJ                0.3250612     203786955
## IMP_DEROG            4.9098658     725458208
## M_DEROG              0.3023181     135883088
## IMP_DELINQ           8.8790367    2171496767
## M_DELINQ             1.3767135      39168805
## IMP_CLAGE            7.9354776    3020919598
## M_CLAGE              3.7930337     230988044
```

```
## IMP_NINQ              2.3772898    1252439275
## M_NINQ                2.1232401     105280917
## IMP_CLNO             20.5806120   11477290253
## M_CLNO                0.9838603      23292415
## IMP_DEBTINC          13.2995524    3982781808
## M_DEBTINC            14.8483233    2577892600
## FLAG.Job.Mgr          0.5773085     205258078
## FLAG.Job.Office       0.7166137     173599011
## FLAG.Job.Other        2.5968940     322514907
## FLAG.Job.ProfExe     -0.2498893     230611598
## FLAG.Job.Sales        0.2547774      66564285
## FLAG.Job.Self         3.4888510     770817837
## FLAG.Reason.DebtCon   6.6913670    2558685307
## FLAG.Reason.HomeImp   9.2288979    3838439976
```

```
varImpPlot( rf_model )
```




```
summary.gbm( gb_model, cBars = 10 )
```




```
##                                    var     rel.inf
## LOAN                              LOAN 78.52001190
## IMP_CLNO                      IMP_CLNO  8.28893643
## IMP_DEBTINC                IMP_DEBTINC  3.23636914
## M_DEBTINC                    M_DEBTINC  2.81526363
## IMP_CLAGE                    IMP_CLAGE  2.00912715
## IMP_DELINQ                  IMP_DELINQ  1.59135001
## FLAG.Reason.HomeImp FLAG.Reason.HomeImp  1.39944632
## FLAG.Job.Self             FLAG.Job.Self  0.92936055
## FLAG.Reason.DebtCon FLAG.Reason.DebtCon  0.48169063
## IMP_VALUE                    IMP_VALUE  0.25238137
## IMP_DEROG                    IMP_DEROG  0.16146174
## IMP_YOJ                        IMP_YOJ  0.15687381
## IMP_NINQ                      IMP_NINQ  0.07534070
## IMP_MORTDUE                IMP_MORTDUE  0.04862016
## FLAG.Job.Sales          FLAG.Job.Sales  0.03376646
## M_MORTDUE                    M_MORTDUE  0.00000000
## M_VALUE                        M_VALUE  0.00000000
## M_YOJ                            M_YOJ  0.00000000
## M_DEROG                        M_DEROG  0.00000000
## M_DELINQ                      M_DELINQ  0.00000000
## M_CLAGE                        M_CLAGE  0.00000000
## M_NINQ                          M_NINQ  0.00000000
## M_CLNO                          M_CLNO  0.00000000
## FLAG.Job.Mgr              FLAG.Job.Mgr  0.00000000
## FLAG.Job.Office        FLAG.Job.Office  0.00000000
```

```
## FLAG.Job.Other           FLAG.Job.Other  0.00000000
## FLAG.Job.ProfExe       FLAG.Job.ProfExe  0.00000000
```

```
#Using your models, predict the probability of default and the loss given default.
#Multiply the two values together for each record.
p2 = pr_flag * pg
```

```
## Warning in pr_flag * pg: longer object length is not a multiple of shorter
## object length
```

```
head(p2)
```

```
##         1         5        10        11        14        15
## 3039.7992 1948.8484 1666.9317 2353.3468 2506.1150  807.3645
```

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

```
## Warning in df$TARGET_LOSS_AMT - p2: longer object length is not a multiple of
## shorter object length
```

```
print(RMSE2)
```

```
## [1] 8636.565
```

```
#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

#This Gradient Boosting Model is probably underfitting
#and the RMSE2 is super large (8521/8582/8393/9008).
#I may still recommend the Random Forest Model of Step 3 which has lowest RMSE (around 4200).
```