Predic422-CharityProject Part 3

Artur Mrozowski

May 28, 2017

Load packages required for this code.

```
# Load packages required for this code.
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.2.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(lift)
```

```
## Warning: package 'lift' was built under R version 3.2.5
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.2.2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.2.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.3
```

Exercise 1 Read Data from CSV File

```r
charityData = read.csv(file.choose(),na.strings=c("NA"," "))
```

Convert categorical variables to factors

```r
charityData$DONR = as.factor(charityData$DONR)
charityData$HOME = as.factor(charityData$HOME)
charityData$HINC = as.factor(charityData$HINC)
```

Rename the dataset to classData for clarity. Remove charityData from R session environment

```r
classData = charityData
#classData=charityData[charityData$DONR == "1",]
rm(charityData)
```

```r
## Check for Missing Values
which(sapply(classData,anyNA))
```

```
##   HOME   HINC GENDER
##      5      6      7
```

```r
# HOME - Make a level 0 and code missing values as 0
levels(classData$HOME) = c(levels(classData$HOME),"0")
classData$HOME[is.na(classData$HOME)] = "0"
table(classData$HOME,useNA="ifany")
```

```
##
##     0     1
## 23899 46972
```

```r
# HINC - Make a level 0 and code missing values as 0
levels(classData$HINC) = c(levels(classData$HINC),"0")
classData$HINC[is.na(classData$HINC)] = "0"
table(classData$HINC,useNA="ifany")
```

```
##
##     1     2     3     4     5     6     7     0
##  7084 10616  7189 10983 13454  6770  6657  8118
```

```r
# GENDER - Assign A, J, and NA to category U
idxMF = classData$GENDER %in% c("M","F")
classData$GENDER[!idxMF] = "U"
classData$GENDER = factor(classData$GENDER)
table(classData$GENDER)
```

```
##
##     F     M     U
## 38183 30494  2194
```

Part B - Derived or Transformed Variables(Optional)

Part C - Re-categorize Variables

```r
# Separate RFA Values (R = recency, F = frequency, A = amount)
separateRFA = function(xData,varName)
{
  bytes = c("R","F","A")
  newVarNames = paste(varName,bytes, sep="_")

  for (ii in 1:length(bytes)) # Loop over 1 to 3 (corresponding to R, F, and A)
  {
    # Find the unique values for current byte
    byteVals = unique(substr(levels(xData[,varName]),ii,ii))

    for (jj in 1:length(byteVals)) # Loop over unique byte values
    {
      rowIdx = substr(xData[,varName],ii,ii) == byteVals[jj]
      xData[rowIdx,newVarNames[ii]] = byteVals[jj]
    }

    xData[,newVarNames[ii]] = factor(xData[,newVarNames[ii]])
  }

  return(xData)
}
```

```r
# Apply separateRFA to the variables RFA_96 and check results.

classData = separateRFA(classData,"RFA_96")
#table(classData$RFA_96,classData$RFA_96_R)
#table(classData$RFA_96,classData$RFA_96_F)
#table(classData$RFA_96,classData$RFA_96_A)
```

Part D - Drop Variables

```r
dropIdx = which(names(classData) %in% c("DAMT","RFA_96"))

# Drop the variables indicated by dropIdx.
classData2 = classData[,-dropIdx]
names(classData2)   # check that the result is as expected
```

```
##  [1] "ID"       "DONR"     "AGE"      "HOME"     "HINC"     "GENDER"
##  [7] "MEDAGE"   "MEDPPH"   "MEDHVAL"  "MEDINC"   "MEDEDUC"  "NUMPROM"
## [13] "NUMPRM12" "RAMNTALL" "NGIFTALL" "MAXRAMNT" "LASTGIFT" "TDON"
## [19] "RFA_96_R" "RFA_96_F" "RFA_96_A"
```

Exercise 3 Dataset Partitioning

```r
# Specify the fraction of data to use in the hold-out test.
testFraction = 0.25
set.seed(123)
```

```r
# Sample training subset indices.

trainIdx = sample(nrow(classData2),size=(1-testFraction)*nrow(classData2),
                  replace=FALSE)
```

Exercise 4 Model Fitting

```r
glm.fit=glm(DONR~ AGE+MEDAGE+MEDHVAL+MEDINC+MEDEDUC+NUMPROM+MAXRAMNT +MEDINC+MEDEDUC+ NUMPROM+NUMPRM12+I
backwards=step(glm.fit,trace=0)
formula(backwards)
```

```
## DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM + NUMPRM12 +
##     RAMNTALL + TDON + RFA_96_F + RFA_96_A
```

```r
summary(backwards)
```

```
##
## Call:
## glm(formula = DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM +
##     NUMPRM12 + RAMNTALL + TDON + RFA_96_F + RFA_96_A, family = binomial,
##     data = classData2, subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8107  -0.3530  -0.3018  -0.2631   2.8164
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.054e+01  8.444e+01  -0.125  0.90065
## AGE         -2.944e-03  1.287e-03  -2.288  0.02213 *
## MEDAGE       5.556e-03  2.587e-03   2.147  0.03176 *
## MEDHVAL      1.125e-04  2.696e-05   4.172 3.01e-05 ***
## MEDINC       2.724e-04  1.585e-04   1.719  0.08564 .
## NUMPROM      4.338e-03  1.344e-03   3.227  0.00125 **
## NUMPRM12    -1.507e-02  5.820e-03  -2.590  0.00959 **
## RAMNTALL     4.148e-04  2.156e-04   1.924  0.05437 .
## TDON        -3.755e-02  6.083e-03  -6.172 6.73e-10 ***
## RFA_96_F2    2.108e-01  5.299e-02   3.977 6.97e-05 ***
## RFA_96_F3    2.863e-01  6.283e-02   4.556 5.21e-06 ***
## RFA_96_F4    3.846e-01  7.097e-02   5.419 6.00e-08 ***
## RFA_96_AC    7.983e+00  8.445e+01   0.095  0.92469
## RFA_96_AD    8.146e+00  8.444e+01   0.096  0.92315
## RFA_96_AE    7.960e+00  8.444e+01   0.094  0.92490
## RFA_96_AF    7.764e+00  8.444e+01   0.092  0.92674
## RFA_96_AG    7.408e+00  8.444e+01   0.088  0.93009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 20869  on 53136  degrees of freedom
## AIC: 20903
##
## Number of Fisher Scoring iterations: 9
```

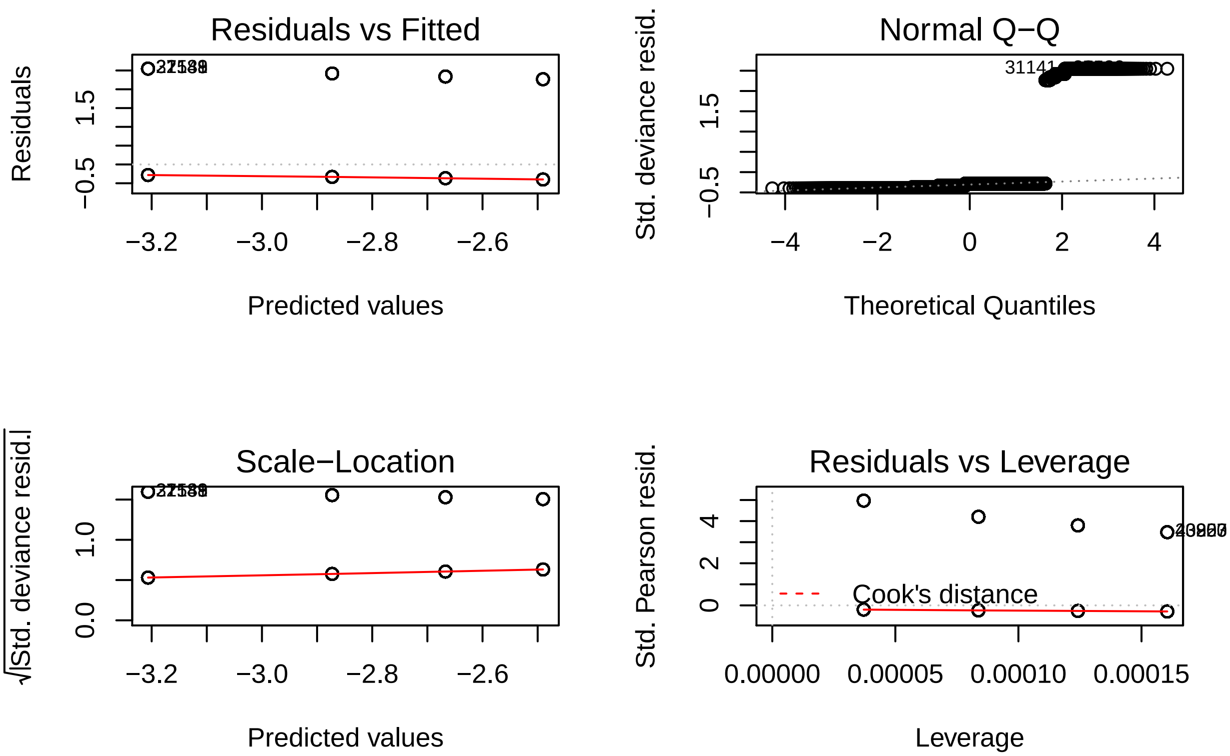
Part A - Simple Logistic Regression

One of the variables with considerable significance is RFA_96_R. I will now fit logistic regression using that variable.

```r
#modelA1 = glm(DONR ~ MAXRAMNT,data=classData2,subset=trainIdx,family=binomial)
modelA1 = glm(DONR ~ RFA_96_F,data=classData2,subset=trainIdx,family=binomial)
summary(modelA1)
```

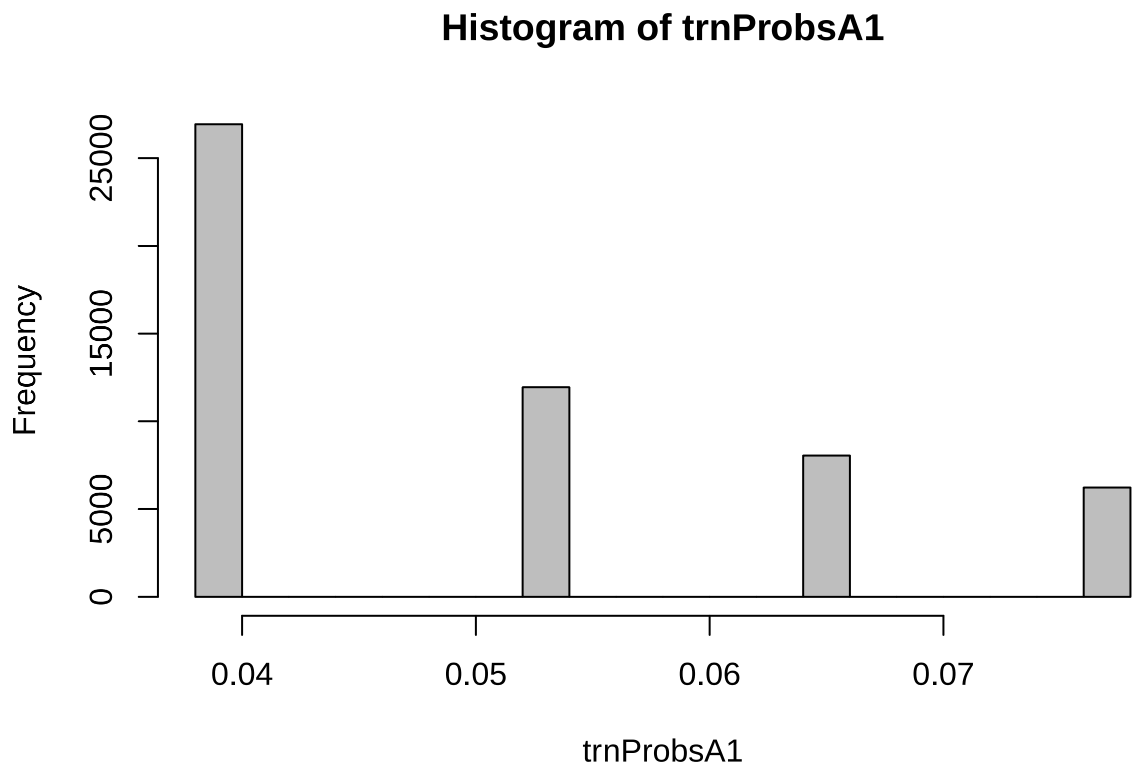
```
##
## Call:
## glm(formula = DONR ~ RFA_96_F, family = binomial, data = classData2,
##     subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3991  -0.3317  -0.2818  -0.2818   2.5480
##
## Coefficients:
##             Estimate Std. Error  z value Pr(>|z|)
## (Intercept) -3.20659    0.03151 -101.766  < 2e-16 ***
## RFA_96_F2    0.33393    0.05144    6.491  8.5e-11 ***
## RFA_96_F3    0.53939    0.05512    9.787  < 2e-16 ***
## RFA_96_F4    0.71628    0.05712   12.539  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 21083  on 53149  degrees of freedom
## AIC: 21091
##
## Number of Fisher Scoring iterations: 6
```

```r
par(mfrow=c(2,2))
plot(modelA1)
```

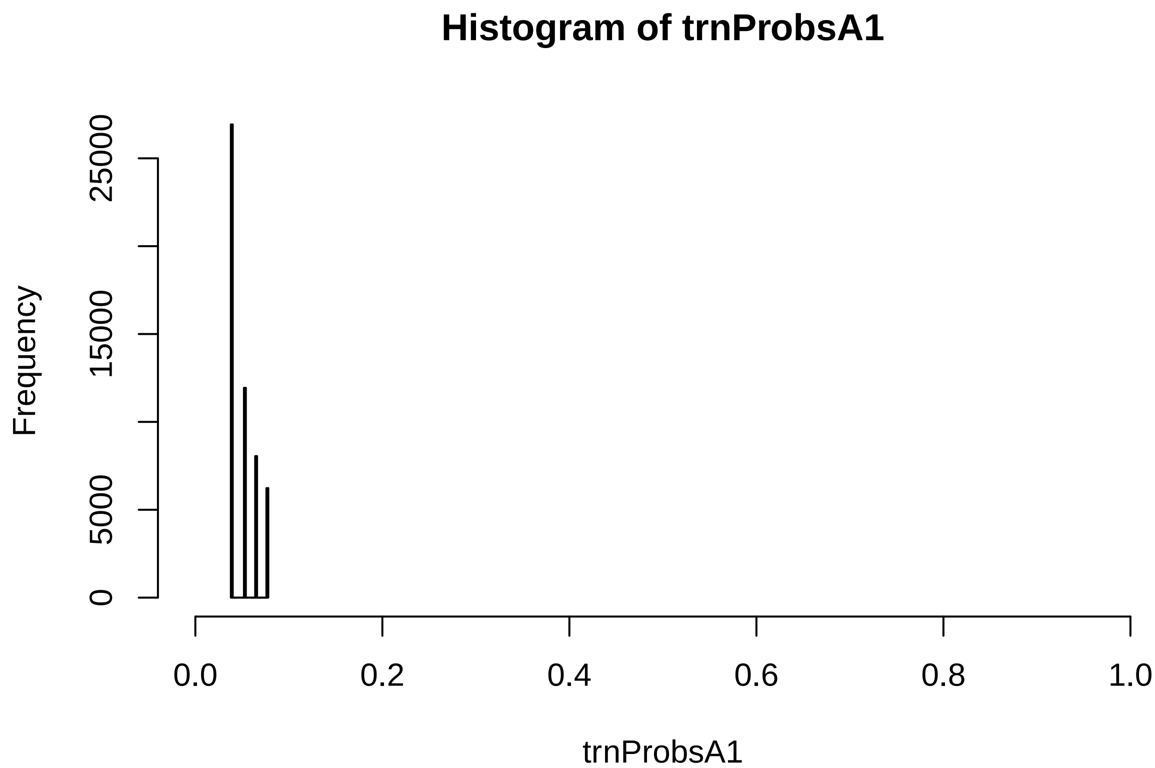



```r
par(mfrow=c(1,1))
```

```r
trnProbsA1 = predict(modelA1,type="response")
hist(trnProbsA1,col="gray")    # Note that scores are distributed around 0.05.
```



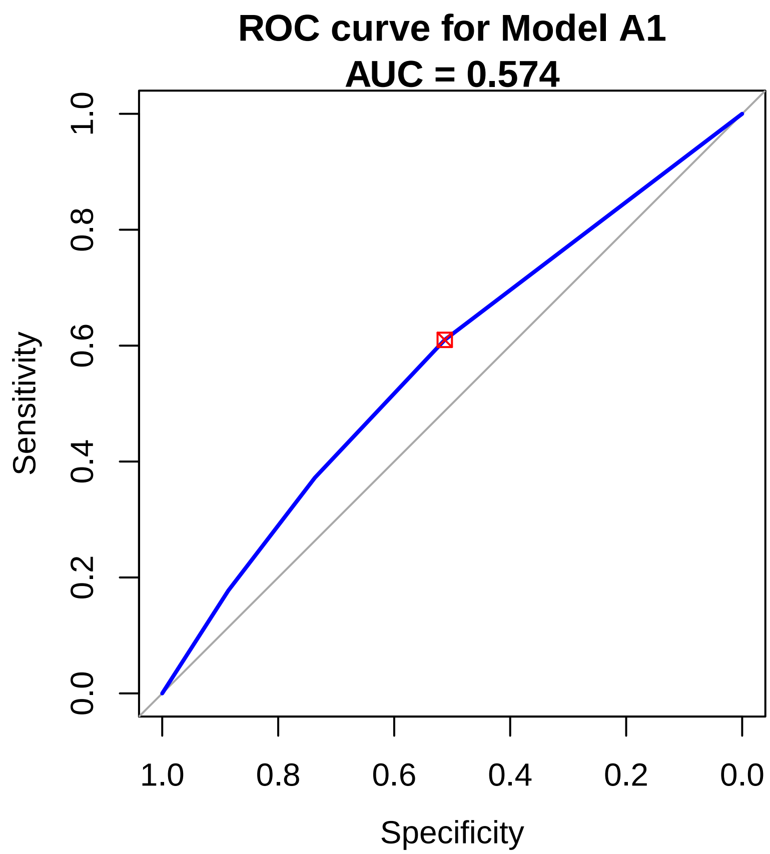

```
hist(trnProbsA1,col="gray",xlim=c(0,1))    # Rescale to make obvious.
```



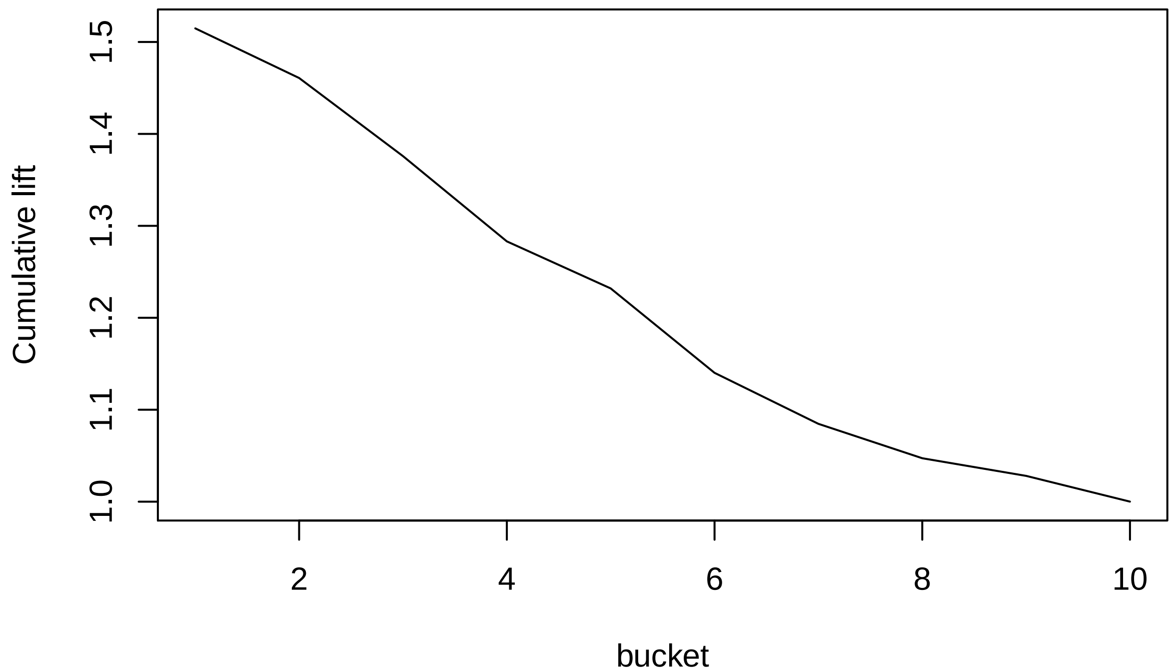

```
# Classification: ROC Curve for Model A1 - Use methods from pROC package.
rocA1 = roc(response=classData2$DONR[trainIdx],predictor=trnProbsA1)
par(pty="s")  # sets plotting region to a square, useful for ROC curves
# Use par(pty="m") to return to default of rectangular plotting region.
plot(rocA1,col="blue",
     main=paste("ROC curve for Model A1\nAUC = ",round(rocA1$auc,digits=3),sep=""))
```

```
##
## Call:
## roc.default(response = classData2$DONR[trainIdx], predictor = trnProbsA1)
##
## Data: trnProbsA1 in 50466 controls (classData2$DONR[trainIdx] 0) < 2687 cases (classData2$DONR[trainI
## Area under the curve: 0.5739
```

```
par(pty="m")
# Classification: Determine "optimal" threshold.
dist01 = sqrt((rocA1$specificities-1)^2 + (rocA1$sensitivities-1)^2)
optIdxA1 = which.min(dist01)  # index corresponding to minimum distance
threshA1 = rocA1$thresholds[optIdxA1]  # threshold corresponding to min. distance
points(rocA1$specificities[optIdxA1],rocA1$sensitivities[optIdxA1],col="red",pch=7)
```




```r
# Ranking: Generate lift chart on training subset and measure top-decile lift.
plotLift(trnProbsA1,classData2$DONR[trainIdx])
```




```
TopDecileLift(trnProbsA1,classData2$DONR[trainIdx])
```

```
## [1] 1.515
```

Part B - Linear Discriminant Analysis

I will use caret package in order to choose the best 4 variables for LDA. Let's see how it works

```
c_1 <- trainControl(method = "none")
maxvar     <-(4)
direction <-"forward"
tune_1     <-data.frame(maxvar,direction)
tr <- train(DONR~., data=classData2, method = "stepLDA", trControl=c_1, tuneGrid=tune_1)
```

```
## Loading required package: klaR

## Warning: package 'klaR' was built under R version 3.2.5

##  `stepwise classification', using 10-fold cross-validated correctness rate of method lda'.

## 70871 observations of 36 variables in 2 classes; direction: forward

## stop criterion: assemble 4 best variables.
```

```
## correctness rate: 0.94868;  in: "ID";  variables (1): ID
## correctness rate: 0.94868;  in: "AGE";  variables (2): ID, AGE
## correctness rate: 0.94868;  in: "HOME1";  variables (3): ID, AGE, HOME1
## correctness rate: 0.94868;  in: "HINC2";  variables (4): ID, AGE, HOME1, HINC2
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        7.00        1.89
```

It is hard to choose any variable because the model seemingly have greate performance. 95% correct values. The number of TP equals to zero so the results are not very helpful. The number of donors is very low so rather pessimistic assumption that nobody will give anything generates high score, no matter what variable is used. I choose AGE for the model but it could by anything.

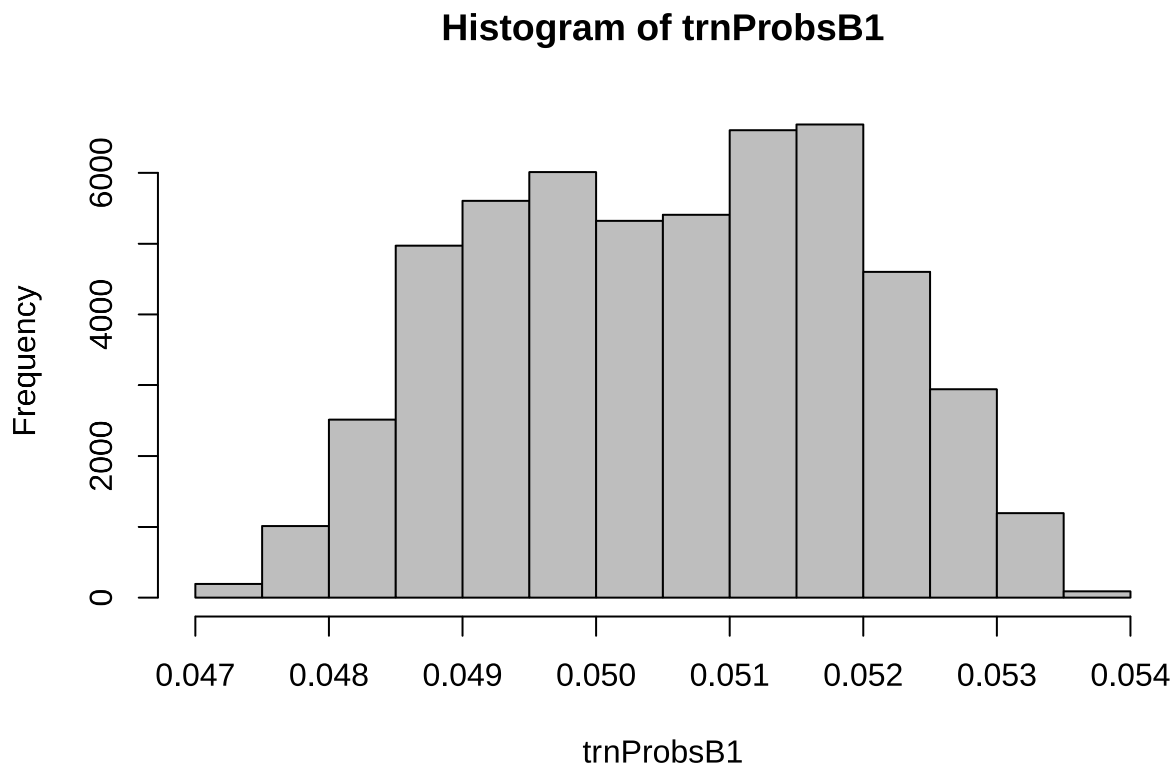
```
modelB1 = lda(DONR~ AGE, data=classData2,
              subset=trainIdx)
```

```
modelB1
```

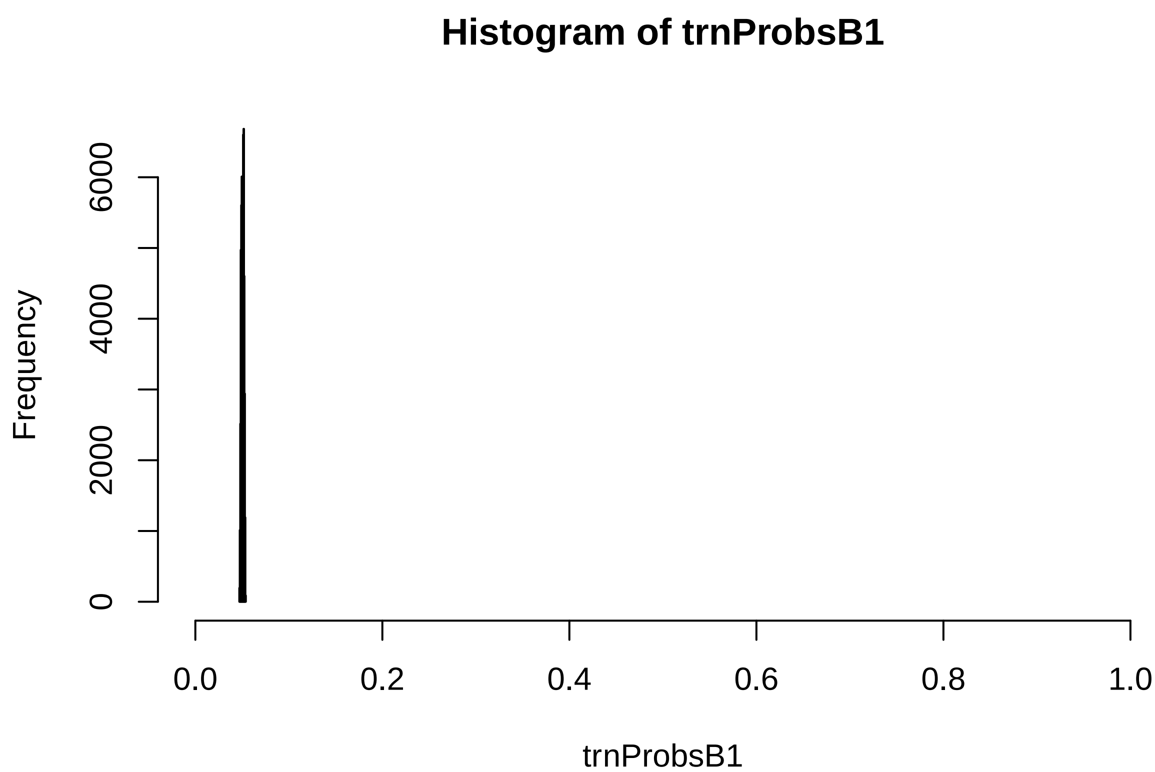
```
## Call:
## lda(DONR ~ AGE, data = classData2, subset = trainIdx)
##
## Prior probabilities of groups:
##          0          1
## 0.94944782 0.05055218
##
## Group means:
##        AGE
## 0 61.70616
## 1 62.18683
##
## Coefficients of linear discriminants:
##            LD1
## AGE 0.06006677
```

```
predB1 = predict(modelB1,classData2[trainIdx,])
trnProbsB1 = predB1$posterior[,2]   # column 2 corresponds to Pr(DONR = 1)
```

```
# Similar to modelA1, we explore the probabilities and build a ROC curve
# for modelB1.
hist(trnProbsB1,col="gray")   # Note that scores are distributed around 0.05.
```



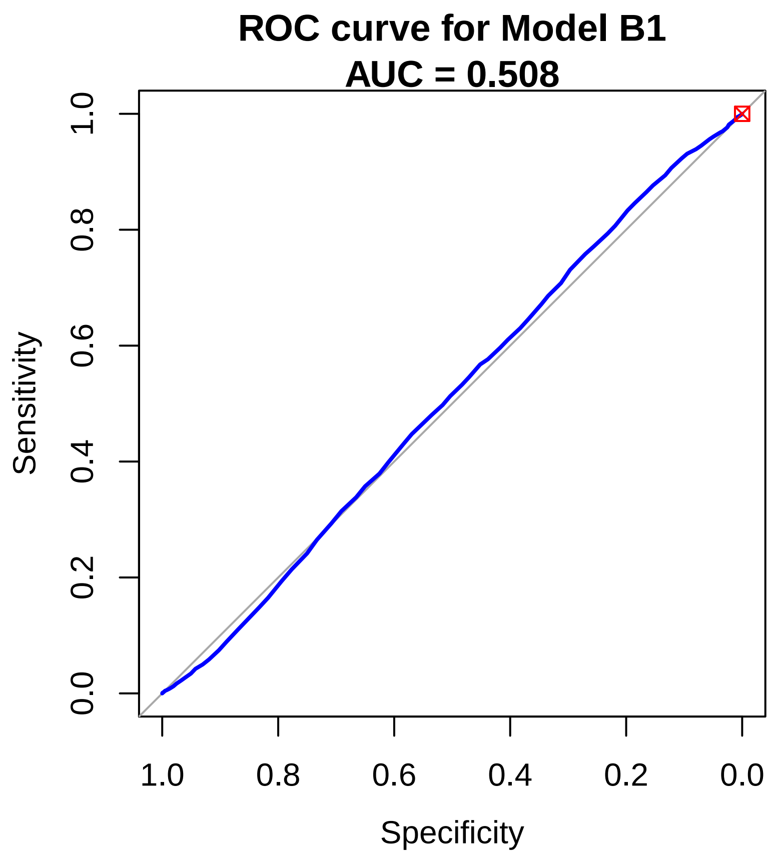

```
hist(trnProbsB1,col="gray",xlim=c(0,1))    # Rescale to make obvious.
```




```
# Classification: ROC Curve for Model B1 - Use methods from pROC package.
rocB1 = roc(response=classData2$DONR[trainIdx],predictor=trnProbsB1)
par(pty="s")  # sets plotting region to a square, useful for ROC curves
# Use par(pty="m") to return to default of rectangular plotting region.
plot(rocB1,col="blue",
     main=paste("ROC curve for Model B1\nAUC = ",round(rocB1$auc,digits=3),sep=""))
```

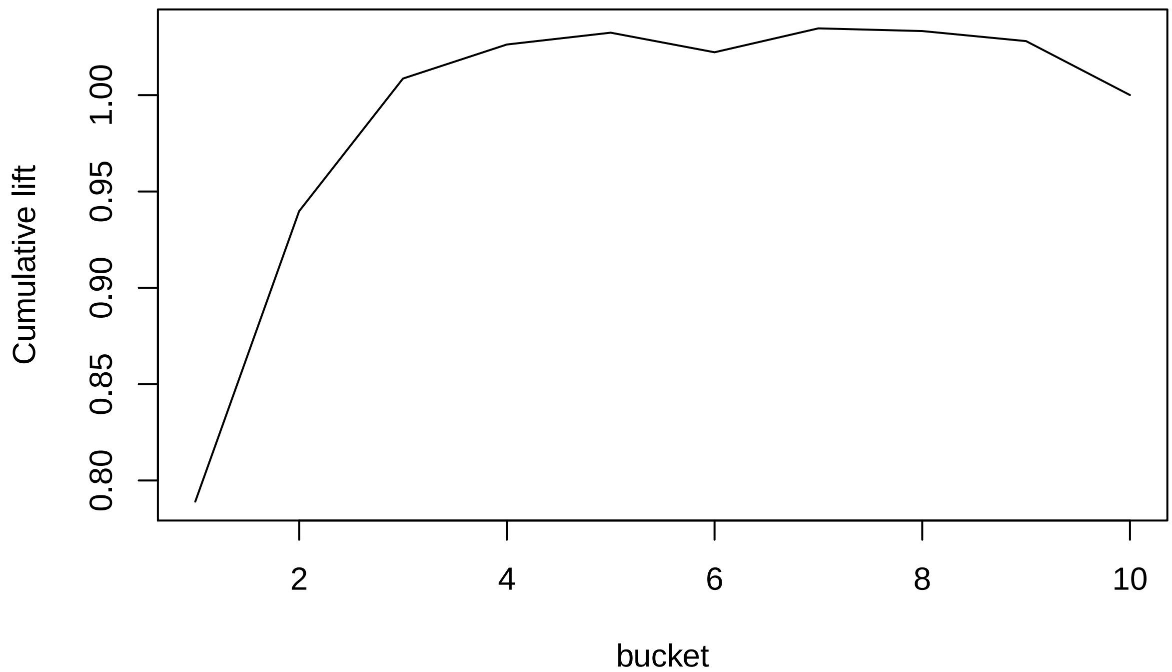
```
##
## Call:
## roc.default(response = classData2$DONR[trainIdx], predictor = trnProbsB1)
##
## Data: trnProbsB1 in 50466 controls (classData2$DONR[trainIdx] 0) < 2687 cases (classData2$DONR[trainI
## Area under the curve: 0.5083
```

```
par(pty="m")
# Classification: Determine "optimal" threshold.
#dist01 = sqrt((rocB1$specificities-1)^2 + (rocB1$sensitivities-1)^2)
dist01=  sqrt((0.68*(rocA1$specificities-1))^2 + (15.62*(rocA1$sensitivities-1)^2))
optIdxB1 = which.min(dist01)  # index corresponding to minimum distance
threshB1 = rocB1$thresholds[optIdxB1]  # threshold corresponding to min. distance
points(rocB1$specificities[optIdxB1],rocB1$sensitivities[optIdxB1],col="red",pch=7)
```



Let's incorporacte relative weights in optimal threshold

```
# Ranking: Generate lift chart on training subset and measure top-decile lift.
plotLift(trnProbsB1,classData2$DONR[trainIdx])
```




```
TopDecileLift(trnProbsB1,classData2$DONR[trainIdx])
```

```
## [1] 0.789
```

Part C - Tree-Based Models

```
fullTree = rpart(DONR ~  NGIFTALL + MAXRAMNT + LASTGIFT + TDON,
                data=classData2,subset=trainIdx,method="class",
                parms=list(split="gini",loss=matrix(c(0,15.62,0.68,0),nrow=2,ncol=2)))
```

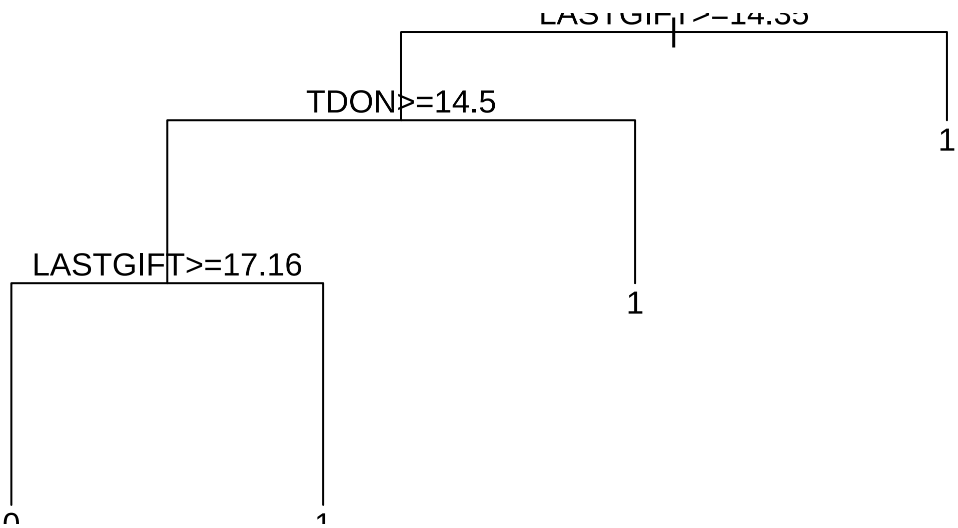
```
summary(fullTree)
```

```
## Call:
## rpart(formula = DONR ~ NGIFTALL + MAXRAMNT + LASTGIFT + TDON,
##     data = classData2, subset = trainIdx, method = "class", parms = list(split = "gini",
##         loss = matrix(c(0, 15.62, 0.68, 0), nrow = 2, ncol = 2)))
##   n= 53153
##
##           CP nsplit rel error   xerror       xstd
## 1 0.02041619      0 1.0000000 22.97059 0.02299019
## 2 0.01000000      3 0.9387514 14.15807 0.05164075
##
## Variable importance
## LASTGIFT MAXRAMNT NGIFTALL     TDON
```

```
##      46       33       11       11
##
## Node number 1: 53153 observations,    complexity param=0.02041619
##   predicted class=1  expected loss=0.6456245  P(node) =1
##     class counts: 50466  2687
##    probabilities: 0.949 0.051
##   left son=2 (33560 obs) right son=3 (19593 obs)
##   Primary splits:
##       LASTGIFT < 14.35 to the right, improve=419.0145, (0 missing)
##       MAXRAMNT < 14.5  to the right, improve=367.7113, (0 missing)
##       NGIFTALL < 5.5   to the left,  improve=288.8737, (0 missing)
##       TDON     < 17.5  to the right, improve=140.7408, (0 missing)
##   Surrogate splits:
##       MAXRAMNT < 14.5  to the right, agree=0.887, adj=0.693, (0 split)
##       NGIFTALL < 11.5  to the left,  agree=0.735, adj=0.282, (0 split)
##       TDON     < 15.5  to the right, agree=0.650, adj=0.050, (0 split)
##
## Node number 2: 33560 observations,    complexity param=0.02041619
##   predicted class=0  expected loss=0.6404386  P(node) =0.6313849
##     class counts: 32184  1376
##    probabilities: 0.959 0.041
##   left son=4 (32420 obs) right son=5 (1140 obs)
##   Primary splits:
##       TDON     < 14.5  to the right, improve=94.33774, (0 missing)
##       NGIFTALL < 6.5   to the left,  improve=73.09901, (0 missing)
##       LASTGIFT < 17.16 to the right, improve=70.47206, (0 missing)
##       MAXRAMNT < 17.16 to the right, improve=35.23428, (0 missing)
##   Surrogate splits:
##       NGIFTALL < 66.5  to the left,  agree=0.966, adj=0.002, (0 split)
##       LASTGIFT < 457.5 to the left,  agree=0.966, adj=0.001, (0 split)
##
## Node number 3: 19593 observations
##   predicted class=1  expected loss=0.6345001  P(node) =0.3686151
##     class counts: 18282  1311
##    probabilities: 0.933 0.067
##
## Node number 4: 32420 observations,    complexity param=0.02041619
##   predicted class=0  expected loss=0.6186329  P(node) =0.6099374
##     class counts: 31136  1284
##    probabilities: 0.960 0.040
##   left son=8 (20182 obs) right son=9 (12238 obs)
##   Primary splits:
##       LASTGIFT < 17.16 to the right, improve=72.68119, (0 missing)
##       NGIFTALL < 4.5   to the left,  improve=52.17868, (0 missing)
##       MAXRAMNT < 24.25 to the right, improve=46.15657, (0 missing)
##       TDON     < 24.5  to the right, improve=33.46467, (0 missing)
##   Surrogate splits:
##       MAXRAMNT < 17.16 to the right, agree=0.939, adj=0.838, (0 split)
##
## Node number 5: 1140 observations
##   predicted class=1  expected loss=0.6251228  P(node) =0.02144752
##     class counts:  1048    92
##    probabilities: 0.919 0.081
##
```

```
## Node number 8: 20182 observations
##   predicted class=0  expected loss=0.5518313  P(node) =0.3796963
##     class counts: 19469   713
##    probabilities: 0.965 0.035
##
## Node number 9: 12238 observations
##   predicted class=1  expected loss=0.6482726  P(node) =0.230241
##     class counts: 11667   571
##    probabilities: 0.953 0.047
```

```r
plot(fullTree)
text(fullTree)
```




```r
# Prune the tree
printcp(fullTree)
```

```
##
## Classification tree:
## rpart(formula = DONR ~ NGIFTALL + MAXRAMNT + LASTGIFT + TDON,
##     data = classData2, subset = trainIdx, method = "class", parms = list(split = "gini",
##         loss = matrix(c(0, 15.62, 0.68, 0), nrow = 2, ncol = 2)))
##
## Variables actually used in tree construction:
## [1] LASTGIFT TDON
##
```

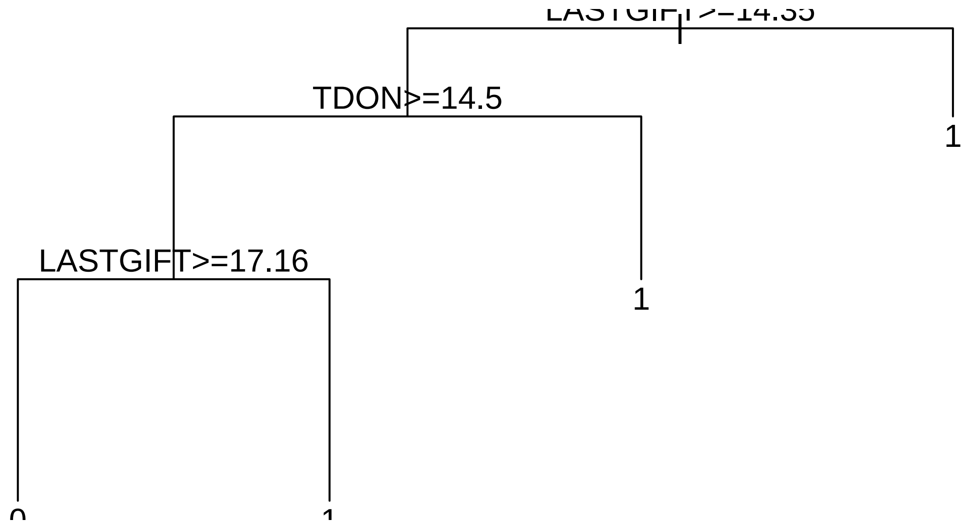
```
## Root node error: 34317/53153 = 0.64562
##
## n= 53153
##
##         CP nsplit rel error xerror     xstd
## 1 0.020416      0   1.00000 22.971 0.022990
## 2 0.010000      3   0.93875 14.158 0.051641
```

```r
cpBest = fullTree$cptable[which.min(fullTree$cptable[,"xerror"]),"CP"]
modelC1 = prune(fullTree,cp=cpBest) # In this case, the optimal tree is the unpruned tree
summary(modelC1)
```

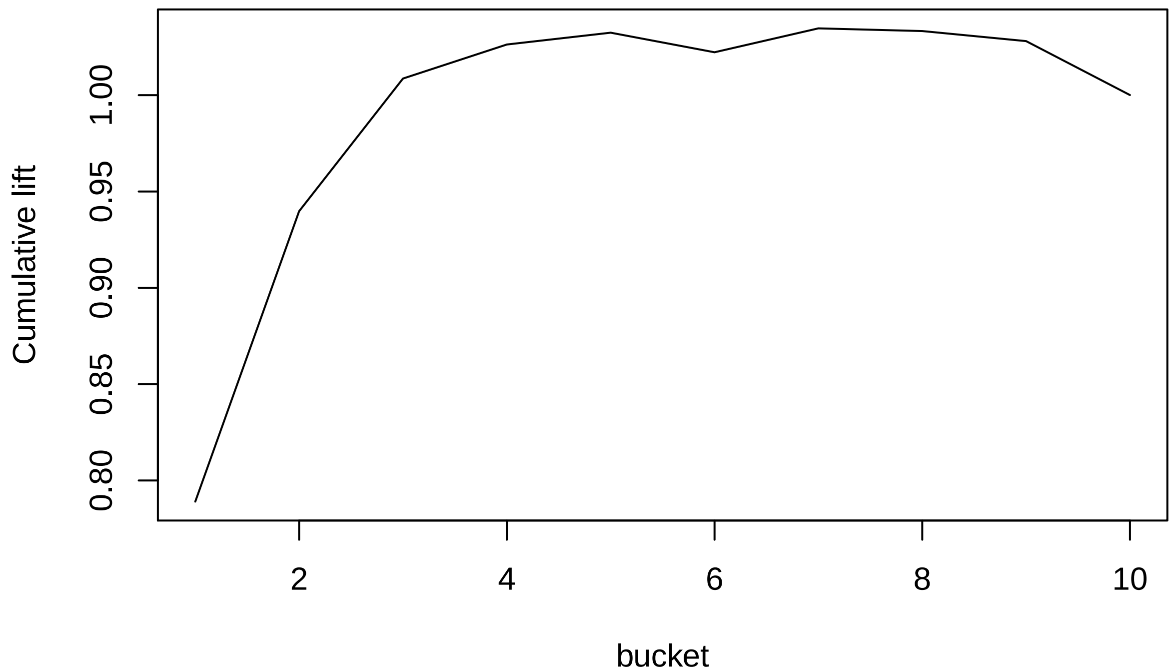
```
## Call:
## rpart(formula = DONR ~ NGIFTALL + MAXRAMNT + LASTGIFT + TDON,
##     data = classData2, subset = trainIdx, method = "class", parms = list(split = "gini",
##         loss = matrix(c(0, 15.62, 0.68, 0), nrow = 2, ncol = 2)))
##   n= 53153
##
##           CP nsplit rel error   xerror       xstd
## 1 0.02041619      0 1.0000000 22.97059 0.02299019
## 2 0.01000000      3 0.9387514 14.15807 0.05164075
##
## Variable importance
## LASTGIFT MAXRAMNT NGIFTALL     TDON
##       46       33       11       11
##
## Node number 1: 53153 observations,    complexity param=0.02041619
##   predicted class=1  expected loss=0.6456245  P(node) =1
##     class counts: 50466  2687
##    probabilities: 0.949 0.051
##   left son=2 (33560 obs) right son=3 (19593 obs)
##   Primary splits:
##       LASTGIFT < 14.35 to the right, improve=419.0145, (0 missing)
##       MAXRAMNT < 14.5  to the right, improve=367.7113, (0 missing)
##       NGIFTALL < 5.5   to the left,  improve=288.8737, (0 missing)
##       TDON     < 17.5  to the right, improve=140.7408, (0 missing)
##   Surrogate splits:
##       MAXRAMNT < 14.5  to the right, agree=0.887, adj=0.693, (0 split)
##       NGIFTALL < 11.5  to the left,  agree=0.735, adj=0.282, (0 split)
##       TDON     < 15.5  to the right, agree=0.650, adj=0.050, (0 split)
##
## Node number 2: 33560 observations,    complexity param=0.02041619
##   predicted class=0  expected loss=0.6404386  P(node) =0.6313849
##     class counts: 32184  1376
##    probabilities: 0.959 0.041
##   left son=4 (32420 obs) right son=5 (1140 obs)
##   Primary splits:
##       TDON     < 14.5  to the right, improve=94.33774, (0 missing)
##       NGIFTALL < 6.5   to the left,  improve=73.09901, (0 missing)
##       LASTGIFT < 17.16 to the right, improve=70.47206, (0 missing)
##       MAXRAMNT < 17.16 to the right, improve=35.23428, (0 missing)
##   Surrogate splits:
##       NGIFTALL < 66.5  to the left,  agree=0.966, adj=0.002, (0 split)
##       LASTGIFT < 457.5 to the left,  agree=0.966, adj=0.001, (0 split)
```

```
##
## Node number 3: 19593 observations
##   predicted class=1  expected loss=0.6345001  P(node) =0.3686151
##     class counts: 18282  1311
##    probabilities: 0.933 0.067
##
## Node number 4: 32420 observations,    complexity param=0.02041619
##   predicted class=0  expected loss=0.6186329  P(node) =0.6099374
##     class counts: 31136  1284
##    probabilities: 0.960 0.040
##   left son=8 (20182 obs) right son=9 (12238 obs)
##   Primary splits:
##       LASTGIFT < 17.16 to the right, improve=72.68119, (0 missing)
##       NGIFTALL < 4.5   to the left,  improve=52.17868, (0 missing)
##       MAXRAMNT < 24.25 to the right, improve=46.15657, (0 missing)
##       TDON     < 24.5  to the right, improve=33.46467, (0 missing)
##   Surrogate splits:
##       MAXRAMNT < 17.16 to the right, agree=0.939, adj=0.838, (0 split)
##
## Node number 5: 1140 observations
##   predicted class=1  expected loss=0.6251228  P(node) =0.02144752
##     class counts:  1048    92
##    probabilities: 0.919 0.081
##
## Node number 8: 20182 observations
##   predicted class=0  expected loss=0.5518313  P(node) =0.3796963
##     class counts: 19469   713
##    probabilities: 0.965 0.035
##
## Node number 9: 12238 observations
##   predicted class=1  expected loss=0.6482726  P(node) =0.230241
##     class counts: 11667   571
##    probabilities: 0.953 0.047
```

```r
plot(modelC1)
text(modelC1,pretty=0)
```




```r
# Ranking: Generate lift chart on training subset and measure top-decile lift.
trnProbsC1 = predict(modelC1,newdata=classData2[trainIdx,],type="prob")[,2]
plotLift(trnProbsB1,classData2$DONR[trainIdx])
```




```r
TopDecileLift(trnProbsB1,classData2$DONR[trainIdx])
```

```
## [1] 0.789
```

Part D - Model of your choice.

I will use the best model as selected in the stepwise backwards selection for logistic regression.

```r
backwards$formula
```

```
## DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM + NUMPRM12 +
##     RAMNTALL + TDON + RFA_96_F + RFA_96_A
```

```r
summary(backwards)
```

```
##
## Call:
## glm(formula = DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM +
##     NUMPRM12 + RAMNTALL + TDON + RFA_96_F + RFA_96_A, family = binomial,
##     data = classData2, subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8107  -0.3530  -0.3018  -0.2631   2.8164
```

```
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.054e+01  8.444e+01  -0.125  0.90065
## AGE         -2.944e-03  1.287e-03  -2.288  0.02213 *
## MEDAGE       5.556e-03  2.587e-03   2.147  0.03176 *
## MEDHVAL      1.125e-04  2.696e-05   4.172 3.01e-05 ***
## MEDINC       2.724e-04  1.585e-04   1.719  0.08564 .
## NUMPROM      4.338e-03  1.344e-03   3.227  0.00125 **
## NUMPRM12    -1.507e-02  5.820e-03  -2.590  0.00959 **
## RAMNTALL     4.148e-04  2.156e-04   1.924  0.05437 .
## TDON        -3.755e-02  6.083e-03  -6.172 6.73e-10 ***
## RFA_96_F2    2.108e-01  5.299e-02   3.977 6.97e-05 ***
## RFA_96_F3    2.863e-01  6.283e-02   4.556 5.21e-06 ***
## RFA_96_F4    3.846e-01  7.097e-02   5.419 6.00e-08 ***
## RFA_96_AC    7.983e+00  8.445e+01   0.095  0.92469
## RFA_96_AD    8.146e+00  8.444e+01   0.096  0.92315
## RFA_96_AE    7.960e+00  8.444e+01   0.094  0.92490
## RFA_96_AF    7.764e+00  8.444e+01   0.092  0.92674
## RFA_96_AG    7.408e+00  8.444e+01   0.088  0.93009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 20869  on 53136  degrees of freedom
## AIC: 20903
##
## Number of Fisher Scoring iterations: 9
```

I would be inclined to drop RFA_96_A

```
modelD1=glm(DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM + NUMPRM12 + RAMNTALL + TDON + RFA_96_F ,da
modelD1.probs=predict(modelD1,type="response")
head(modelD1.probs)
```

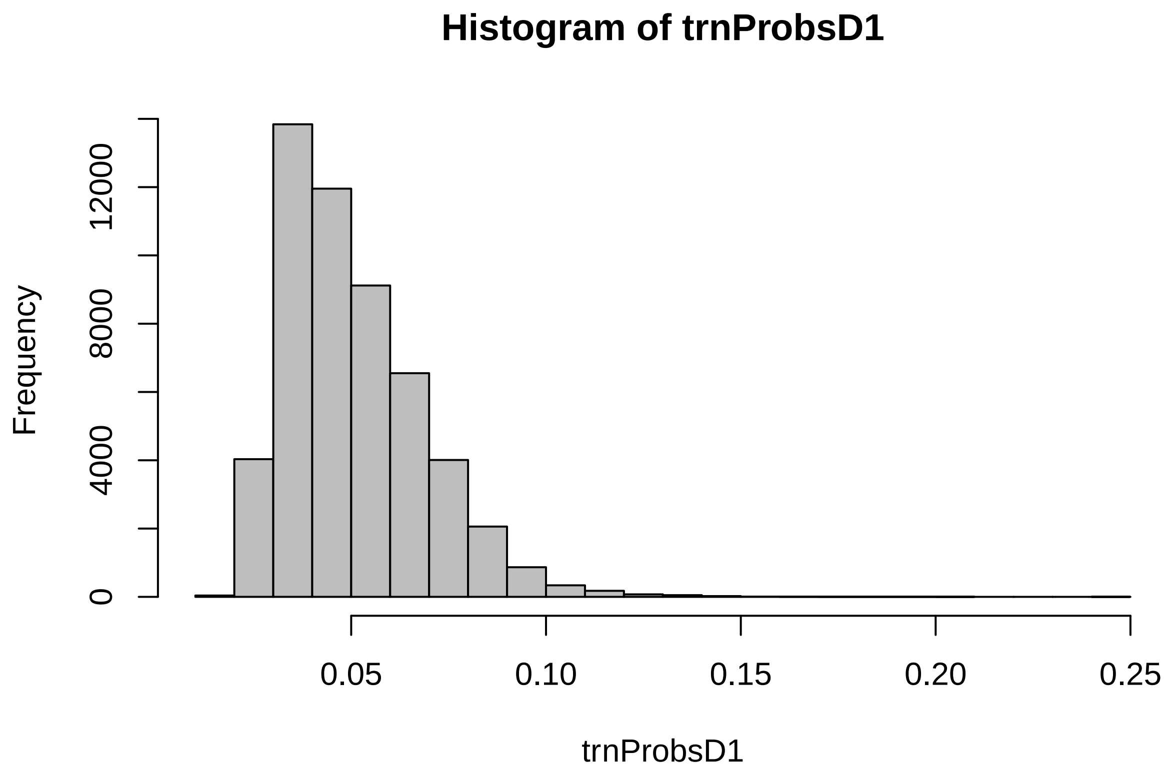
```
##      20381      55868      28984      62578      66649       3229
## 0.03404121 0.05709598 0.04060962 0.03661652 0.06402728 0.06390135
```

```
summary(modelD1)
```

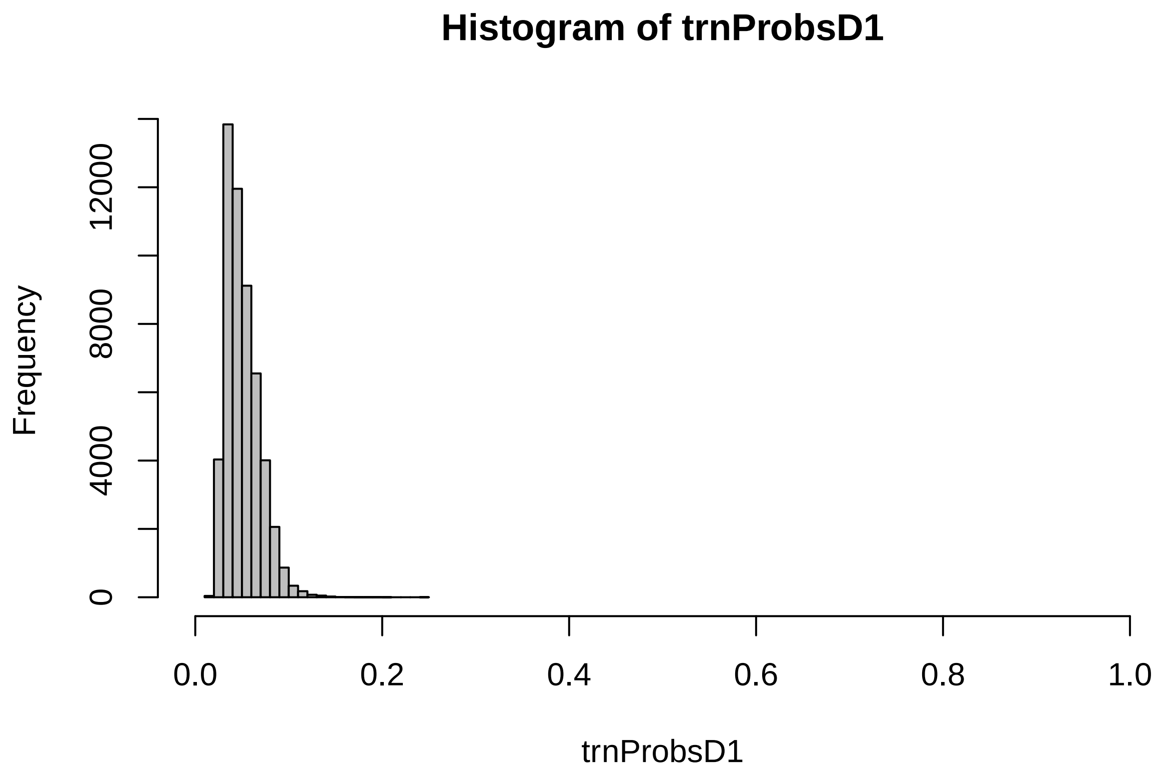
```
##
## Call:
## glm(formula = DONR ~ AGE + MEDAGE + MEDHVAL + MEDINC + NUMPROM +
##     NUMPRM12 + RAMNTALL + TDON + RFA_96_F, family = binomial,
##     data = classData2, subset = trainIdx)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7571  -0.3509  -0.3050  -0.2668   2.7987
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -2.7162542  0.2046606 -13.272  < 2e-16 ***
## AGE         -0.0026013  0.0012853  -2.024   0.0430 *  
## MEDAGE       0.0055501  0.0025776   2.153   0.0313 *  
## MEDHVAL      0.0001047  0.0000269   3.893 9.92e-05 ***
## MEDINC       0.0002368  0.0001582   1.497   0.1344    
## NUMPROM      0.0084732  0.0013363   6.341 2.28e-10 ***
## NUMPRM12    -0.0249265  0.0057344  -4.347 1.38e-05 ***
## RAMNTALL    -0.0003878  0.0002520  -1.539   0.1239    
## TDON        -0.0429197  0.0060382  -7.108 1.18e-12 ***
## RFA_96_F2    0.2569568  0.0523205   4.911 9.05e-07 ***
## RFA_96_F3    0.4660134  0.0562316   8.287  < 2e-16 ***
## RFA_96_F4    0.6400990  0.0586569  10.913  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 21276  on 53152  degrees of freedom
## Residual deviance: 20942  on 53141  degrees of freedom
## AIC: 20966
## 
## Number of Fisher Scoring iterations: 6
```

```r
trnProbsD1 = predict(modelD1,type="response")
hist(trnProbsD1,col="gray")   # Note that scores are distributed around 0.05.
```




```
hist(trnProbsD1,col="gray",xlim=c(0,1))    # Rescale to make obvious.
```

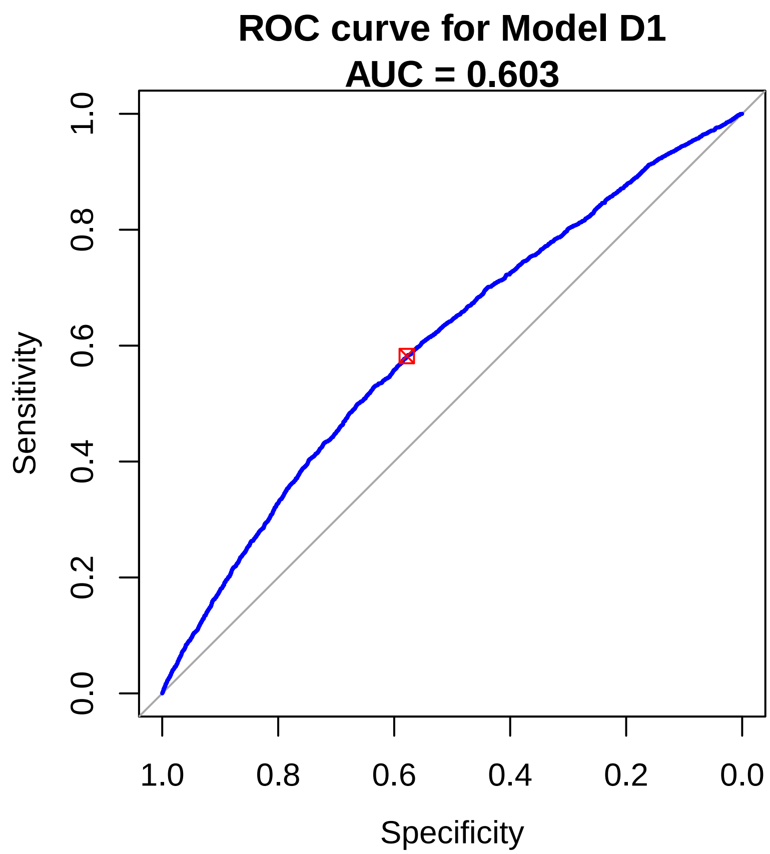



```r
# Classification: ROC Curve for Model D1 - Use methods from pROC package.
rocD1 = roc(response=classData2$DONR[trainIdx],predictor=trnProbsD1)
par(pty="s")  # sets plotting region to a square, useful for ROC curves
# Use par(pty="m") to return to default of rectangular plotting region.
plot(rocD1,col="blue",
     main=paste("ROC curve for Model D1\nAUC = ",round(rocD1$auc,digits=3),sep=""))
```

```
##
## Call:
## roc.default(response = classData2$DONR[trainIdx], predictor = trnProbsD1)
##
## Data: trnProbsD1 in 50466 controls (classData2$DONR[trainIdx] 0) < 2687 cases (classData2$DONR[trainI
## Area under the curve: 0.6027
```

```r
par(pty="m")
```

```r
dist01 = sqrt((rocD1$specificities-1)^2 + (rocD1$sensitivities-1)^2)
optIdxD1 = which.min(dist01)  # index corresponding to minimum distance
threshD1 = rocD1$thresholds[optIdxD1]  # threshold corresponding to min. distance
points(rocD1$specificities[optIdxD1],rocD1$sensitivities[optIdxD1],col="red",pch=7)
```



Exercise 5 Model Validation

```
assignClass = function(probVals,threshVal)
{
  predVals = rep(0,length(probVals))
  predVals[probVals > threshVal] = 1
  predVals = factor(predVals)

  return(predVals)
}

calcMetrics = function(targetVals,predVals)
{
  confMat = table(targetVals,predVals,dnn=c("Target","Predicted"))

  classResults = list(
    confMat = confMat,
    TPrate = round(confMat[2,2] / sum(confMat[2,]),digits=4),
    FPrate = round(confMat[1,2] / sum(confMat[1,]),digits=4),
    accuracy = round(mean(targetVals == predVals),digits=2),
    topDecileLift = TopDecileLift(predVals,targetVals)
  )

  return(classResults)
}
```

```r
calcResults = function(model,modelLabel,dataSet,trainIdx,threshVal=NULL)
{
  if (!is.null(threshVal) & "glm" %in% class(model)) {
    # Predict for glm models
    probVals = predict(model,dataSet,type="response")
    predVals = assignClass(probVals,threshVal)
  } else if (length(intersect(class(model),c("tree","rpart","randomForest")) > 0)) {
    # Predict for tree, rpart, randomForest models
    predVals = predict(model,dataSet,type="class")
  } else if (length(intersect(class(model),c("lda")) > 0)) {
    # Predict for lda models
    predVals = predict(model,dataSet)$class
  } else if (length(intersect(class(model),c("svm")) > 0)) {
    # Predict for svm models
    predVals = predict(model,dataSet)
  }

  results = list(
    name = modelLabel,
    train = calcMetrics(classData2$DONR[trainIdx],predVals[trainIdx]),
    test = calcMetrics(classData2$DONR[-trainIdx],predVals[-trainIdx])
  )

  return(results)
}
```

You can also embed plots, for example:

```r
nModels = 4 # Number of models you fit. I fit 3 models in this sample code.
naTmp = rep(NA,nModels) # Code short-hand.
nanTmp = rep(NaN,nModels)
modelMetrics = data.frame(
  Model = naTmp,
  Train.Accuracy = nanTmp, Train.TP = nanTmp, Train.FP = nanTmp, Train.Lift = nanTmp,
  Test.Accuracy = nanTmp, Test.TP = nanTmp, Test.FP = nanTmp, Test.Lift = nanTmp
  )
```

```r
resultsA1 = calcResults(modelA1,"A1",classData2,trainIdx,threshA1)
print(resultsA1$test$confMat)
```

```
##       Predicted
## Target    0    1
##      0 8570 8198
##      1  368  582
```

```r
modelMetrics[1,] = c(resultsA1$name,
                     resultsA1$train$accuracy,resultsA1$train$TPrate,resultsA1$train$FPrate,resultsA1$tr
                     resultsA1$test$accuracy,resultsA1$test$TPrate,resultsA1$test$FPrate,resultsA1$test$

resultsB1 = calcResults(modelB1,"B1",classData2,trainIdx,threshB1)
print(resultsB1$test$confMat)
```

```
##       Predicted
## Target     0     1
##      0 16768     0
##      1   950     0
```

```
modelMetrics[2,] = c(resultsB1$name,
                     resultsB1$train$accuracy,resultsB1$train$TPrate,resultsB1$train$FPrate,resultsB1$tr
                     resultsB1$test$accuracy,resultsB1$test$TPrate,resultsB1$test$FPrate,resultsB1$test$

resultsC1 = calcResults(modelC1,"C1",classData2,trainIdx)
print(resultsC1$test$confMat)
```

```
##       Predicted
## Target     0     1
##      0  6379 10389
##      1   279   671
```

```
modelMetrics[3,] = c(resultsC1$name,
                     resultsC1$train$accuracy,resultsC1$train$TPrate,resultsC1$train$FPrate,resultsC1$tr
                     resultsC1$test$accuracy,resultsC1$test$TPrate,resultsC1$test$FPrate,resultsC1$test$

resultsD1 = calcResults(modelD1,"D1",classData2,trainIdx,threshD1)
print(resultsD1$test$confMat)
```

```
##       Predicted
## Target    0    1
##      0 9727 7041
##      1  406  544
```

```
modelMetrics[4,] = c(resultsD1$name,
                     resultsD1$train$accuracy,resultsD1$train$TPrate,resultsD1$train$FPrate,resultsD1$tr
                     resultsD1$test$accuracy,resultsD1$test$TPrate,resultsD1$test$FPrate,resultsD1$test$
print(modelMetrics)
```

```
##   Model Train.Accuracy Train.TP Train.FP Train.Lift Test.Accuracy Test.TP
## 1    A1           0.52     0.61   0.4872      1.332          0.52  0.6126
## 2    B1           0.95        0        0      1.012          0.95       0
## 3    C1            0.4   0.7346   0.6142      1.202           0.4  0.7063
## 4    D1           0.58   0.5821   0.4217      1.351          0.58  0.5726
##   Test.FP Test.Lift
## 1  0.4889     1.179
## 2       0     1.053
## 3  0.6196     1.126
## 4  0.4199     1.263
```

6 Model Selection a. LDA has the best score but zero true positive, which makes it pretty much useless for the purpose of predicting donors. Probability of anyone becoming a donor is so low that classifying everyone as not-donors yields the highest score.

Random forest has the high number of trup positive but it comes at price of high number of false positives which could potentially make a campaign quite expensive. b. Which leaves us with the logistic regression models, that considering all the aspcts mentioned above, have the best performance. ModelD1 has got the best accuracy and is the model I'd choose.