Binary Logit Model

Lunjing Yuan

Monday, 2/8/2021

Read in the SCU reunion data

```
rm(list=ls())       # Remove anything that might currently be in memory so we
can start fresh.
library(data.table) # Load the data.table package.
library(MASS)       # Load the MASS package

reuniondata <- fread('Reuniondata_inclass.csv')
```

You can check out the variables names using

```
names(reuniondata)
```

```
##  [1] "CaseNum"       "RYCohort"      "donatesum"     "SpouseAlum"
##  [5] "SportsAlum"    "UGAlumAwards"  "OtherUGAct"    "EverAssigned"
##  [9] "BoardMember"   "ChildAlum"     "ParentAlum"    "SiblingAlum"
## [13] "GradDegree"    "TotalReunions" "OnePlusEvents" "Years Lapsed"
```

The interpretation of each variable is in the file Reuniondata_inclass.xlsx, in a separate sheet. Please check that sheet to understand the meaning of each variable.

Here, if you prefer using data.frame instead of data.table, that is fine as well.

We first created the Y variable, based on the variable "donatesum", indicating the "total number of donations". As our first order interest is whether someone donate or not, rather than how much they donate, we decided to create a choice variable, as Y for the binary logit model, and added the variable to the last column in the `reuniondata` datatable.

```
reuniondata[,Choice:=as.numeric(donatesum>0),]
dim(reuniondata)
```

```
## [1] 7120   17
```

As a starter, we estimate the binary logit model using all the other variables as X variable. To do that, I first created a data frame, that contains the choice variable, and all the X variables.

```
d = data.frame(y=reuniondata[,17],reuniondata[,4:16])
```

Before using the data, I first split the data into training set with the first 3000 data points and testing data with the rest.

```
idTrn = 1:3000  # row index for the training data
idTst = !(1:nrow(reuniondata) %in% idTrn)  # row index for the testing data
```

```r
blTrn_basic = glm(Choice~ ., data=d, family="binomial", subset = idTrn)
#using only training data. Note that the "." after "~" meaning using all the
rest of the data in data frame "d" except the variable "Choice".
#This is why we created the "d" dataframe, so that we do not need to list all
the other variables we need in the model.
#but you can always use the following command, if you do not want to use "."

#blTrn_basic = glm(Choice~
SpouseAlum+SportsAlum+UGAlumAwards+OtherUGAct+EverAssigned+BoardMember+
#ChildAlum+ParentAlum+SiblingAlum+GradDegree+TotalReunions+OnePlusEvents+
#Years.Lapsed, data=d, family="binomial", subset = idTrn)
#for running additional models, you may find it easier to base your code on
this code.

summary(blTrn_basic)
```

```
##
## Call:
## glm(formula = Choice ~ ., family = "binomial", data = d, subset = idTrn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5560  -0.4260  -0.1524   0.6076   3.0297
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.46706    0.11973  12.253   <2e-16 ***
## SpouseAlum    -0.07978    0.15029  -0.531   0.5955
## SportsAlum     0.20724    0.17451   1.188   0.2350
## UGAlumAwards  -0.26376    0.30443  -0.866   0.3863
## OtherUGAct     0.23864    0.13865   1.721   0.0852 .
## EverAssigned   0.48600    0.20585   2.361   0.0182 *
## BoardMember    0.87495    0.24204   3.615   0.0003 ***
## ChildAlum      0.10904    0.16783   0.650   0.5159
## ParentAlum    -0.09957    0.25480  -0.391   0.6960
## SiblingAlum   -0.12994    0.17832  -0.729   0.4662
## GradDegree    -0.01880    0.17261  -0.109   0.9132
## TotalReunions  0.04072    0.02346   1.736   0.0826 .
## OnePlusEvents  0.25212    0.11967   2.107   0.0351 *
## Years.Lapsed  -0.12326    0.00518 -23.796   <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: 4132.0  on 2999  degrees of freedom
## Residual deviance: 2162.4  on 2986  degrees of freedom
## AIC: 2190.4
```

```
##
## Number of Fisher Scoring iterations: 6
```

Q1: Among these results, the variable with the highest t-value is Years.Lapsed, indicating the "number of years elapsed since last donation". Its t-value has an absolute value that is much higher than the other t-values. That is usually a sign of a problem. What could be the problem here? Think about the model estimation is trying to match the data (values in Y).

The Years.Lapsed variable has a much higher absolute t-value compared to the other variables. This is likely because most independent variables are binary, taking on values of just 0 or 1. However, Years.Lapsed represents the total number of years since the alumnus/alumna last donated to SCU, with 48 indicating no prior donation history. The continuous, numeric nature of this variable, with a wide range of values, contributes to its high t-value. More importantly, including Years.Lapsed could negatively impact the model in two key ways. First, it could distort the overall p-value for the regression, making it seem more statistically significant than it should be. Second, it could obscure the true relationships between valid, useful predictor variables and the response. Problematic variables like Years.Lapsed can impair the regression both mathematically and interpretively. Careful examination of this variable's role is warranted.

In summary, the exceptionally high t-value for Years.Lapsed stems from its continuous nature and wide range of values. Its inclusion risks reducing the validity and interpretability of the regression results. Assessing whether it should be retained requires further analysis.

Q2: Now let's re-estimate the model using some different specifications that you can come up with. Estimate at least 2 more models. I have couple of suggestions:

- Alternative model 1: drop the varaible "Years.Lapsed"

```
d1 = data.frame(y=reuniondata[,17],reuniondata[,4:15])

blTrn_basic_dropY = glm(Choice~
SpouseAlum+SportsAlum+UGAlumAwards+OtherUGAct+EverAssigned+BoardMember
+ChildAlum+ParentAlum+SiblingAlum+GradDegree+TotalReunions+OnePlusEvents,
data=d1, family="binomial", subset = idTrn)

summary(blTrn_basic_dropY)

##
## Call:
## glm(formula = Choice ~ SpouseAlum + SportsAlum + UGAlumAwards +
##     OtherUGAct + EverAssigned + BoardMember + ChildAlum + ParentAlum +
```

```
##     SiblingAlum + GradDegree + TotalReunions + OnePlusEvents,
##     family = "binomial", data = d1, subset = idTrn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8625  -0.7815  -0.6959   0.9670   1.7532
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.2946139  0.0655677 -19.745  < 2e-16 ***
## SpouseAlum     0.3532552  0.1257433   2.809  0.00496 **
## SportsAlum     0.2649784  0.1421800   1.864  0.06237 .
## UGAlumAwards   0.1405653  0.2660173   0.528  0.59722
## OtherUGAct     0.2158645  0.1102997   1.957  0.05034 .
## EverAssigned   1.1211566  0.1788975   6.267 3.68e-10 ***
## BoardMember    1.0591800  0.1999722   5.297 1.18e-07 ***
## ChildAlum      0.4408550  0.1442430   3.056  0.00224 **
## ParentAlum     0.0147849  0.2187874   0.068  0.94612
## SiblingAlum    0.1907491  0.1495007   1.276  0.20199
## GradDegree     0.0006086  0.1373371   0.004  0.99646
## TotalReunions  0.1678999  0.0211556   7.936 2.08e-15 ***
## OnePlusEvents  1.0372973  0.0909258  11.408  < 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: 4132.0  on 2999  degrees of freedom
## Residual deviance: 3340.5  on 2987  degrees of freedom
## AIC: 3366.5
##
## Number of Fisher Scoring iterations: 5
```

- Alternative model 2: You can consider combining some of the variables to create a new dummy variable, for example

```
famAlumdata = reuniondata[,.(SpouseAlum,ChildAlum,ParentAlum,SiblingAlum)]
FamilyAlum = as.numeric(rowSums(famAlumdata)>0)

reuniondata[,FamilyAlum:=as.numeric(rowSums(famAlumdata)>0),]
d2 = data.frame(y=reuniondata[,17],reuniondata[,4:18])


blTrn_basic_combine = glm(Choice~
SportsAlum+UGAlumAwards+OtherUGAct+EverAssigned+BoardMember+GradDegree+TotalR
eunions
+OnePlusEvents+Years.Lapsed+FamilyAlum, data=d2, family="binomial", subset =
idTrn)

summary(blTrn_basic_combine)
```

```
##
## Call:
## glm(formula = Choice ~ SportsAlum + UGAlumAwards + OtherUGAct +
##     EverAssigned + BoardMember + GradDegree + TotalReunions +
##     OnePlusEvents + Years.Lapsed + FamilyAlum, family = "binomial",
##     data = d2, subset = idTrn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4975  -0.4256  -0.1523   0.6066   3.0078
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.463978   0.120657  12.133  < 2e-16 ***
## SportsAlum     0.203551   0.174682   1.165 0.243910
## UGAlumAwards  -0.253150   0.304799  -0.831 0.406230
## OtherUGAct     0.240744   0.138586   1.737 0.082362 .
## EverAssigned   0.489689   0.205822   2.379 0.017351 *
## BoardMember    0.881310   0.241500   3.649 0.000263 ***
## GradDegree    -0.010376   0.172323  -0.060 0.951988
## TotalReunions  0.041137   0.023341   1.762 0.077994 .
## OnePlusEvents  0.258722   0.119444   2.166 0.030307 *
## Years.Lapsed  -0.123222   0.005162 -23.871  < 2e-16 ***
## FamilyAlum    -0.061688   0.116912  -0.528 0.597749
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4132.0  on 2999  degrees of freedom
## Residual deviance: 2163.5  on 2989  degrees of freedom
## AIC: 2185.5
##
## Number of Fisher Scoring iterations: 6
```

- Alternative model 3: Based on the variable `RYCohort`, you can create a variable to calculate the number of years since graduation, then add this variable to the model, and see whether the estimation results improve. When estimating these other models, use the second `glm()` function, and then adding or dropping variables

```
yearsgrad = 2014-reuniondata[,RYCohort]
reuniondata[,yearsgrad:=2014-reuniondata[,RYCohort],]
d3 = data.frame(y=reuniondata[,17],reuniondata[,4:19])


blTrn_basic_yearsgrad = glm(Choice~ yearsgrad+SpouseAlum+SportsAlum
+UGAlumAwards+OtherUGAct+EverAssigned+BoardMember+ChildAlum+ParentAlum+Siblin
gAlum+GradDegree
+TotalReunions+OnePlusEvents+Years.Lapsed, data=d3, family="binomial", subset
= idTrn)
```

```r
summary(blTrn_basic_yearsgrad)
```

```
##
## Call:
## glm(formula = Choice ~ yearsgrad + SpouseAlum + SportsAlum +
##     UGAlumAwards + OtherUGAct + EverAssigned + BoardMember +
##     ChildAlum + ParentAlum + SiblingAlum + GradDegree + TotalReunions +
##     OnePlusEvents + Years.Lapsed, family = "binomial", data = d3,
##     subset = idTrn)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.91057  -0.36633  -0.09768   0.56178   2.43885
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.444015   0.378210  -6.462 1.03e-10 ***
## yearsgrad      0.099503   0.009401  10.585  < 2e-16 ***
## SpouseAlum     0.133796   0.153183   0.873  0.38243
## SportsAlum     0.049299   0.179193   0.275  0.78323
## UGAlumAwards  -0.390778   0.314620  -1.242  0.21421
## OtherUGAct     0.066238   0.145531   0.455  0.64900
## EverAssigned   0.488754   0.211144   2.315  0.02062 *
## BoardMember    0.640112   0.243083   2.633  0.00846 **
## ChildAlum     -0.101036   0.170471  -0.593  0.55339
## ParentAlum     0.129080   0.263204   0.490  0.62384
## SiblingAlum    0.003609   0.182722   0.020  0.98424
## GradDegree    -0.016605   0.178237  -0.093  0.92578
## TotalReunions  0.038803   0.024286   1.598  0.11010
## OnePlusEvents  0.155516   0.124247   1.252  0.21069
## Years.Lapsed  -0.134328   0.005604 -23.972  < 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: 4132.0  on 2999  degrees of freedom
## Residual deviance: 2035.2  on 2985  degrees of freedom
## AIC: 2065.2
##
## Number of Fisher Scoring iterations: 6
```

Q3: Now let's try out-of-sample test on the testing data set. Using each of the three models you estimated above, calculate the following metrics for each model.

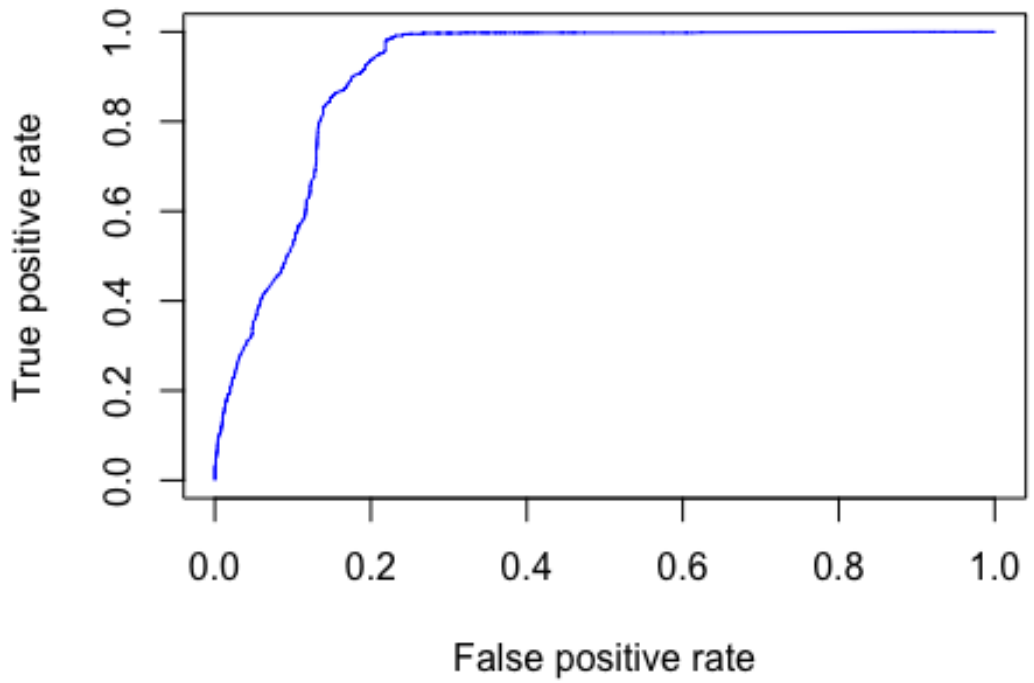
- ln-likelihood value on the testing data.
- Plot the ROC and calculate the AUC values, using the data from the lecture note.

To do that for your basic model for ln-liklihood, you can use the following code

```r
yActual = d[idTst,1] #get the actual value for the choice variable
predTst_basic = predict(blTrn_basic, d[idTst,], type="response")
#use the model results in blTrn_basic, to predict the probability of Y=1 for
each data point in the testing data set

lnlike_basic = sum(log(predTst_basic*yActual+(1-predTst_basic)*(1-yActual)))
#using the predicted probability that Y=1, and the actual data, can calculate
the ln-likelihood for all the data points
lnlike_basic
```

```
## [1] -1404.022
```

```r
# Plot the ROC and calculate the AUC values for basic model.
# install.packages("ROCR")
library(ROCR)
pred <- prediction(predTst_basic,yActual)
perf <- performance(pred,"tpr","fpr")
plot(perf,col='blue')
```

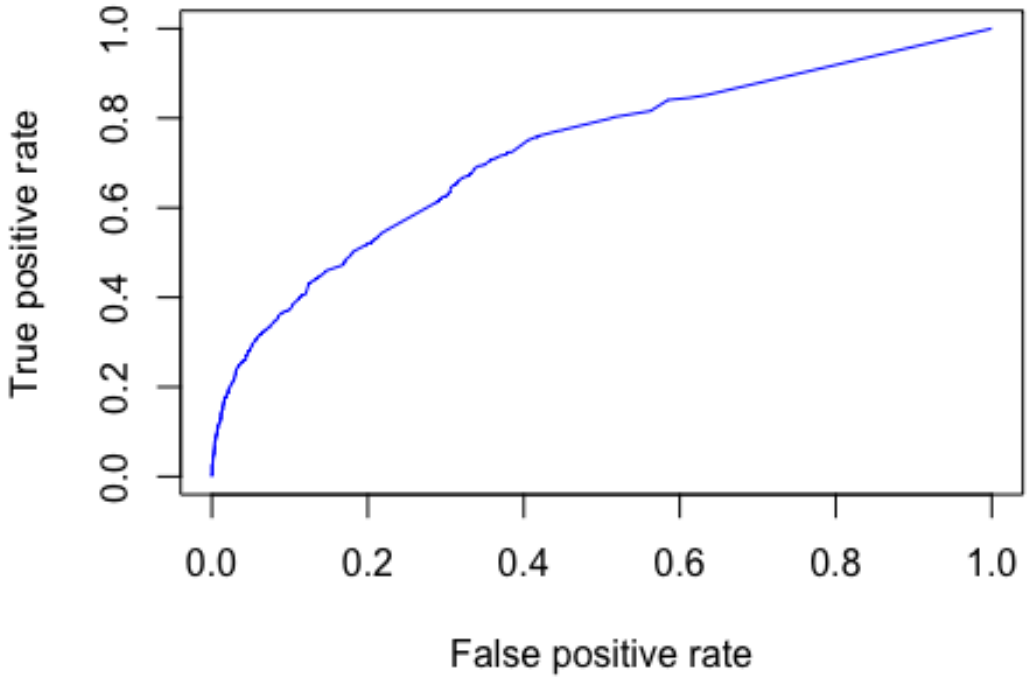



```r
#  AUC value
perf <- performance(pred,measure="auc")
print(paste("AUC= ", perf@y.values[[1]]))
```

```
## [1] "AUC=  0.910019907028007"
```

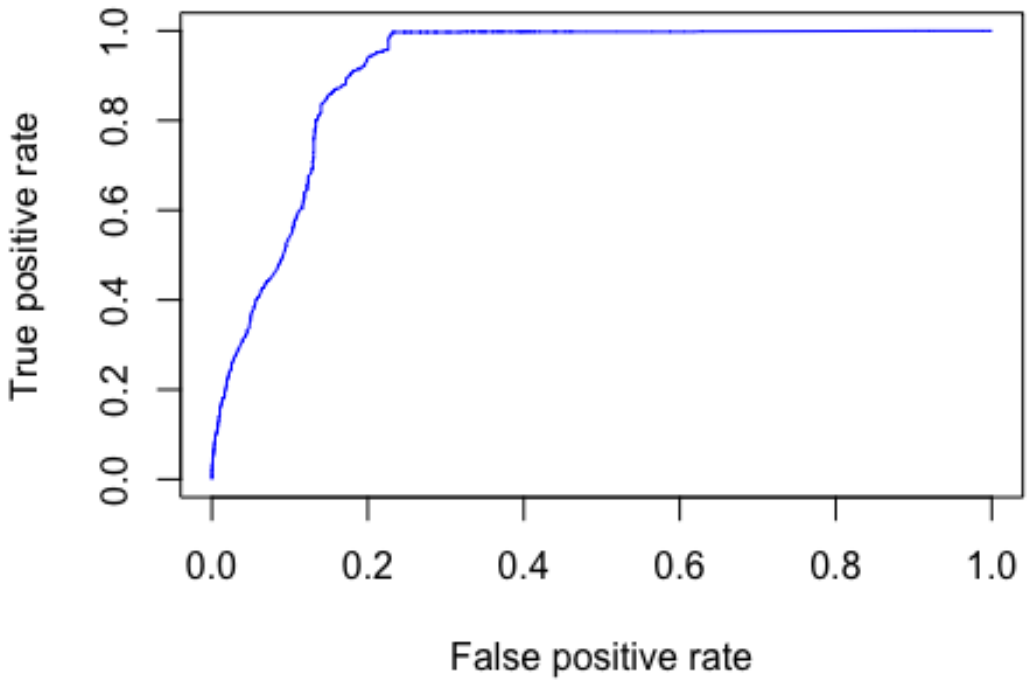
Repeat the above code for your two additional models, and compare the results and conclude which model is best for the out-of-sample test? Comment on the parts that you do not like about the chosen model.

```
# Alternative model 1

yActual = d[idTst,1] #get the actual value for the choice variable
predTst_basic_a1 = predict(blTrn_basic_dropY, d1[idTst,], type="response")
#use the model results in blTrn_basic, to predict the probability of Y=1 for
each data point in the testing data set

lnlike_basic_a1 = sum(log(predTst_basic_a1*yActual+(1-predTst_basic_a1)*(1-
yActual)))
#using the predicted probability that Y=1, and the actual data, can calculate
the ln-likelihood for all the data points
lnlike_basic_a1

## [1] -2284.386

# Plot the ROC and calculate the AUC values for model 1.
# install.packages("ROCR")
library(ROCR)
pred1 <- prediction(predTst_basic_a1,yActual)
perf1 <- performance(pred1,"tpr","fpr")
plot(perf1,col='blue')
```




```
# AUC value for model 1
perf1 <- performance(pred1,measure="auc")
print(paste("AUC_m1= ", perf1@y.values[[1]]))

## [1] "AUC_m1=  0.725454639909634"

# Alternative model 2

yActual = d[idTst,1] #get the actual value for the choice variable
predTst_basic_a2 = predict(blTrn_basic_combine, d2[idTst,], type="response")
#use the model results in blTrn_basic, to predict the probability of Y=1 for
each data point in the testing data set

lnlike_basic_a2 = sum(log(predTst_basic_a2*yActual+(1-predTst_basic_a2)*(1-
yActual)))
#using the predicted probability that Y=1, and the actual data, can calculate
the ln-likelihood for all the data points
lnlike_basic_a2

## [1] -1403.629

# Plot the ROC and calculate the AUC values for model 2.
# install.packages("ROCR")
library(ROCR)
```

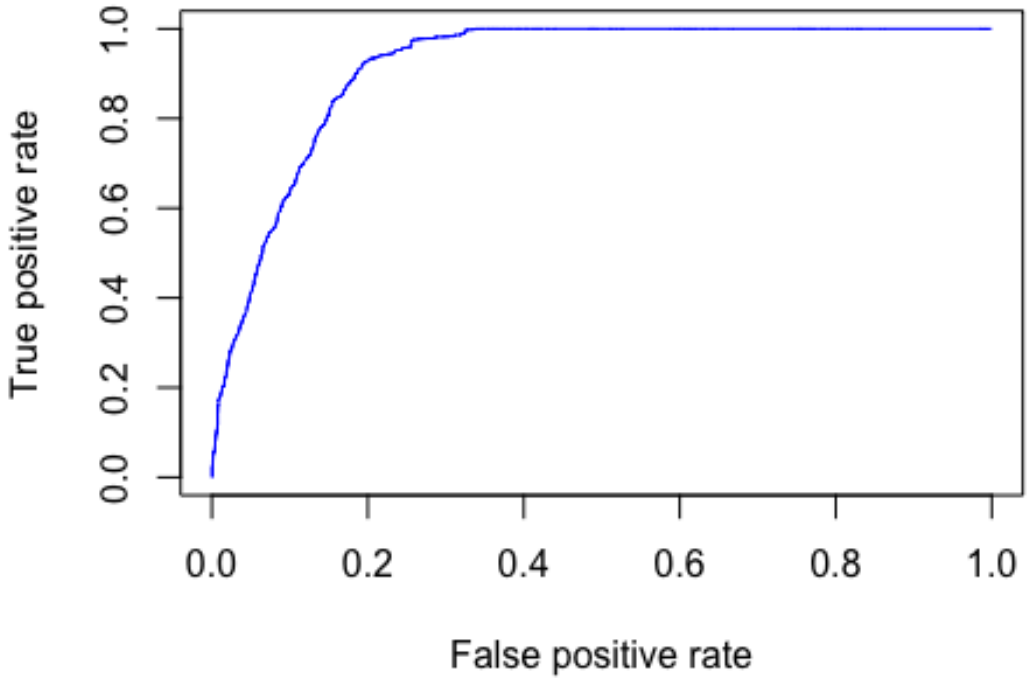
```
pred2 <- prediction(predTst_basic_a2,yActual)
perf2 <- performance(pred2,"tpr","fpr")
plot(perf2,col='blue')
```




```
# AUC value for model 2
perf2 <- performance(pred2,measure="auc")
print(paste("AUC_m2= ", perf2@y.values[[1]]))

## [1] "AUC_m2=  0.911100663631066"

# Alternative model 3

yActual = d[idTst,1] #get the actual value for the choice variable
predTst_basic_a3 = predict(blTrn_basic_yearsgrad, d3[idTst,],
type="response")
#use the model results in blTrn_basic, to predict the probability of Y=1 for
each data point in the testing data set

lnlike_basic_a3 = sum(log(predTst_basic_a3*yActual+(1-predTst_basic_a3)*(1-
yActual)))
#using the predicted probability that Y=1, and the actual data, can calculate
the ln-likelihood for all the data points
lnlike_basic_a3
```

```
## [1] -1702.858
```

```
# Plot the ROC and calculate the AUC values for model 3.
# install.packages("ROCR")
library(ROCR)
pred3 <- prediction(predTst_basic_a3,yActual)
perf3 <- performance(pred3,"tpr","fpr")
plot(perf3,col='blue')
```




```
# AUC value for model 3
perf3 <- performance(pred3,measure="auc")
print(paste("AUC_3= ", perf3@y.values[[1]]))
```

```
## [1] "AUC_3=  0.915569885255118"
```

#which model is best for the out-of-sample test? Comment on the parts that you do not like about the chosen model.

(1) Our goal is to maximize the log-likelihood, where higher values indicate a better model fit. Among the models tested, Model 2 has the highest log-likelihood value of -1403.629, making it the best fitting model for the data.

(2) A limitation of Model 2 is that it combines the four previous alumni variables (SpouseAlum, ChildAlum, ParentAlum, SiblingAlum) into one new dummy variable called FamilyAlum. While this improved model fit, it also reduced the specificity of information we have about each type of alumnus relationship. By consolidating these variables, we lose granular insights into how having a spouse, child, parent or sibling as an alumnus may differently influence donation likelihood. For certain analytical needs, retaining the separated variables could provide useful details that the combined FamilyAlum variable masks.

In summary, Model 2 improves overall fit but reduces specificity in exchange. This illustrates a common tradeoff in modeling between parsimony and insight. Further analysis of variable importance could help determine if the detailed alumni variables should be retained moving forward.