INFX 573 Problem Set 7 - Prediction

Pierre Augustamar

Due: Tueday, November 29, 2016

Collaborators:

Instructions:

Before beginning this assignment, please ensure you have access to R and RStudio.

1. Download the `problemset7.Rmd` file from Canvas. Open `problemset7.Rmd` in RStudio and supply your solutions to the assignment by editing `problemset7.Rmd`.
2. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
3. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text.
4. Collaboration on problem sets is acceptable, and even encouraged, but each student must turn in an individual write-up in his or her own words and his or her own work. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit PDF`, rename the R Markdown file to `YourLastName_YourFirstName_ps7.Rmd`, knit a PDF and submit the PDF file on Canvas.

Setup:

In this problem set you will need, at minimum, the following R packages.

```
# Load standard libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.3.2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```
library(gridExtra)
library(MASS)
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.3.2
```

```
library(arm)
```

```
## Warning: package 'arm' was built under R version 3.3.2
```

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

```
## Warning: package 'lme4' was built under R version 3.3.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.3.2
```

```
library(dplyr)
library(class)
library(ResourceSelection)
```

```
## Warning: package 'ResourceSelection' was built under R version 3.3.2
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.3.2
```

```
## Warning: package 'gplots' was built under R version 3.3.2
```

Data: In this problem set we will use the `titanic` dataset used previously in class. The Titanic text file contains data about the survival of passengers aboard the Titanic. Table 1 contains a description of this data.

```
# Load data
titanic_data <- read.csv('../labs/titanic.csv')
#str(data) # explore data structure
attach(titanic_data)
```

Variable	Description
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
survived	Survival (0 = No; 1 = Yes)
name	Name
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
ticket	Ticket Number
fare	Passenger Fare
cabin	Cabin
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
boat	Lifeboat
body	Body Identification Number
home.dest	Home/Destination

Table 1: Description of variables in the Titanic Dataset

1. As part of this assignment we will evaluate the performance of a few different statistical learning methods. We will fit a particular statistical learning method on a set of *training* observations and measure its performance on a set of *test* observations.

   (a) Discuss the advantages of using a training/test split when evaluating statistical models. When using a model to make predictions we need to have a way to assess the quality, accuracy and reliability fo the model in question. Test data is used to provid assessment of the reliability of the predictive models. Thus, the generated model is built from how the data is partitioned and the inforamtion gathered from the data.

   (b) Split your data into a *training* and *test* set based on an 80-20 split, in other words, 80% of the observations will be in the training set.

```
#set seed to ensure that results and figures are reproducible.
set.seed(2)
titanic_data$id <- 1:nrow(titanic_data) #set id
# split the observations for 80 percent to be allocated for training set
train <- titanic_data %>% sample_frac(.80)
#create test data for titanic. select data that do not exist in training set.
test <- anti_join(titanic_data, train, by = 'id')
```

   Note generated the train and test data based on examples from: http://stackoverflow.com/questions/17200114/how-to-split-data-into-training-testing-sets-using-sample-function-in-r-program

```
#Investigate each of the data sets and check for any NA values

summary(train) #summary for train data set
```

```
##      pclass          survived                                     name
##  Min.   :1.000   Min.   :0.0000   Connolly, Miss. Kate            :   2
##  1st Qu.:2.000   1st Qu.:0.0000   Abbing, Mr. Anthony             :   1
##  Median :3.000   Median :0.0000   Abbott, Master. Eugene Joseph   :   1
##  Mean   :2.291   Mean   :0.3868   Abbott, Mr. Rossmore Edward     :   1
##  3rd Qu.:3.000   3rd Qu.:1.0000   Abbott, Mrs. Stanton (Rosa Hunt):   1
```

```
##  Max.   :3.000   Max.   :1.0000   Abelseth, Miss. Karen Marie     :   1
##                                   (Other)                         :1040
##      sex           age              sibsp            parch
##  female:370   Min.   : 0.1667   Min.   :0.0000   Min.   :0.0000
##  male  :677   1st Qu.:21.0000   1st Qu.:0.0000   1st Qu.:0.0000
##               Median :28.0000   Median :0.0000   Median :0.0000
##               Mean   :30.0264   Mean   :0.4842   Mean   :0.3887
##               3rd Qu.:39.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##               Max.   :80.0000   Max.   :8.0000   Max.   :9.0000
##               NA's   :210
##       ticket          fare                     cabin     embarked
##  CA. 2343:   9   Min.   :  0.000                 :813    :  2
##  1601    :   7   1st Qu.:  7.925   B57 B59 B63 B66:  4   C:217
##  3101295 :   7   Median : 14.479   B96 B98        :  4   Q: 95
##  CA 2144 :   7   Mean   : 33.966   C22 C26        :  4   S:733
##  347077  :   6   3rd Qu.: 31.387   C23 C25 C27    :  4
##  PC 17608:   6   Max.   :512.329   C78            :  4
##  (Other) :1005   NA's   :1         (Other)        :214
##       boat          body                      home.dest          id
##         :654   Min.   :  1.0                      :458   Min.   :   1.0
##  13     : 32   1st Qu.: 68.0   New York, NY       : 47   1st Qu.: 326.5
##  15     : 32   Median :155.0   London             :  9   Median : 659.0
##  4      : 28   Mean   :157.6   Paris, France      :  9   Mean   : 655.0
##  C      : 28   3rd Qu.:247.0   Montreal, PQ       :  8   3rd Qu.: 986.5
##  14     : 26   Max.   :328.0   Cornwall / Akron, OH:  7   Max.   :1309.0
##  (Other):247   NA's   :952     (Other)            :509
```

```
summary(test)  #summary for test data set
```

```
##      pclass          survived
##  Min.   :1.000   Min.   :0.0000
##  1st Qu.:2.000   1st Qu.:0.0000
##  Median :3.000   Median :0.0000
##  Mean   :2.309   Mean   :0.3626
##  3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :3.000   Max.   :1.0000
##
##                                                 name         sex
##  Abelseth, Mr. Olaus Jorgensen                    :  1   female: 96
##  Ahlin, Mrs. Johan (Johanna Persdotter Larsson):  1   male  :166
##  Aldworth, Mr. Charles Augustus                   :  1
##  Alhomaki, Mr. Ilmari Rudolf                      :  1
##  Ali, Mr. William                                 :  1
##  Allum, Mr. Owen George                           :  1
##  (Other)                                          :256
##       age            sibsp            parch               ticket
##  Min.   : 0.75   Min.   :0.0000   Min.   :0.0000   16966     :  3
##  1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.0000   349909    :  3
##  Median :28.00   Median :0.0000   Median :0.0000   382652    :  3
##  Mean   :29.30   Mean   :0.5573   Mean   :0.3702   W./C. 6608:  3
##  3rd Qu.:38.00   3rd Qu.:1.0000   3rd Qu.:0.0000   113503    :  2
##  Max.   :70.00   Max.   :8.0000   Max.   :5.0000   12749     :  2
##  NA's   :53                                        (Other)   :246
##       fare                cabin     embarked      boat
```

```
##  Min.   :  0.000              :201    :  0          :169
##  1st Qu.:  7.854   E34        :  3   C: 53    C      : 10
##  Median : 13.500   C23 C25 C27:  2   Q: 28    3      :  9
##  Mean   : 30.617   C83        :  2   S:181    10     :  7
##  3rd Qu.: 29.125   F G63      :  2            13     :  7
##  Max.   :263.000   F33        :  2            14     :  7
##                    (Other)    : 50            (Other): 53
##       body                                              home.dest
##  Min.   :  4.00                                              :106
##  1st Qu.: 98.75   New York, NY                               : 17
##  Median :152.50   London                                     :  5
##  Mean   :172.46   Belfast                                    :  3
##  3rd Qu.:277.00   Rotherfield, Sussex, England Essex Co, MA:  3
##  Max.   :327.00   Tuxedo Park, NY                            :  3
##  NA's   :236      (Other)                                    :125
##        id
##  Min.   :   6.0
##  1st Qu.: 331.5
##  Median : 645.0
##  Mean   : 655.0
##  3rd Qu.: 964.8
##  Max.   :1307.0
##
```

2. In this problem set our goal is to predict the survival of passengers. First consider training a logistic regression model for survival that controls for the socioeconomic status of the passenger.

   (a) Fit the model described above using the `glm` function in R.

   Note: I suggested `bayesglm` as well in case the model was unstable (you can see this with extremely large s.e. estimates for the coefficients). Be sure you included `pclass` as a `factor` because it is a categorical variable!

```
#fit a linear regression model in order to predict survival

#Use generalized linear model for the logistic regression
glm.fit = glm(test$survived~factor(test$pclass), data=train, family=binomial)

#generate a summary of the logistic regression for better analysis
summary(glm.fit)
```

```
##
## Call:
## glm(formula = test$survived ~ factor(test$pclass), family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4196  -0.7585  -0.7585   0.9532   1.6651
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)             0.5534     0.2617   2.115   0.0345 *
## factor(test$pclass)2   -1.1925     0.3859  -3.090   0.0020 **
## factor(test$pclass)3   -1.6520     0.3248  -5.086 3.66e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 343.17  on 261  degrees of freedom
## Residual deviance: 315.55  on 259  degrees of freedom
## AIC: 321.55
##
## Number of Fisher Scoring iterations: 4
```

Intercept: passengers have 0.55 chance of survival from the titanic, this will be our baseline. factor(pclass)2: passengers from the 2nd class have 1.19 less chance from surviving the titanic factor(pclass)3: passengers from the 3rd class have 1.65 less chances from surviving the titanic

(b) What might you conclude based on this model about the probability of survival for lower class passengers? Each of the coefficients show a very small p-value where all of them are less than 0.05. Thus, there is a clear association between the socio-economic status of a passenger and the probabiliy of survival. To that end, we can conclude that passengers from the 3rd class have the least possibility of surviving the Titanic.

3. Next, let's consider the performance of this model.

(a) Predict the survival of passengers for each observation in your test set using the model fit in Problem 2. Save these predictions as `yhat`.

```
#get predictions on the test data
yhat=predict(glm.fit, type="response")
yhat[1:10] #load the first 10 results for analysis
```

```
##         1         2         3         4         5         6         7
## 0.6349206 0.6349206 0.6349206 0.6349206 0.6349206 0.6349206 0.6349206
##         8         9        10
## 0.6349206 0.6349206 0.6349206
```

Given that the type="response" option tells R to output probabilities in the form of P(Y=1|x), then the values that were outputed correspond to the probability of a passenger surviving the Titanic. The first 10 probabilities that are shown correspond then to passengers surviving the Titanic. They show a value of 0.63 of predictions that our test data is not too far from the train data.

(b) Use a threshold of 0.5 to classify predictions. What is the number of false positives on the test data? Interpret this in your own words.

```
testsurvived = ifelse(test$survived==1,"Yes","NO")
glm.fit = glm(test$survived~factor(test$pclass), data=test, family=binomial)
glm.probs = predict(glm.fit, type="response")
glm.pred=rep("No", 262) #creates a vector of class predictions
glm.pred[glm.probs>.5]="Yes" #probability of survival based on a threshold of .5 or greater
table(glm.pred, testsurvived) #generates a confusion matrix.
```

```
##         testsurvived
## glm.pred  NO Yes
##      No  144  55
##      Yes  23  40
```

```
(144 + 40)/262  #70% correct prediction
```

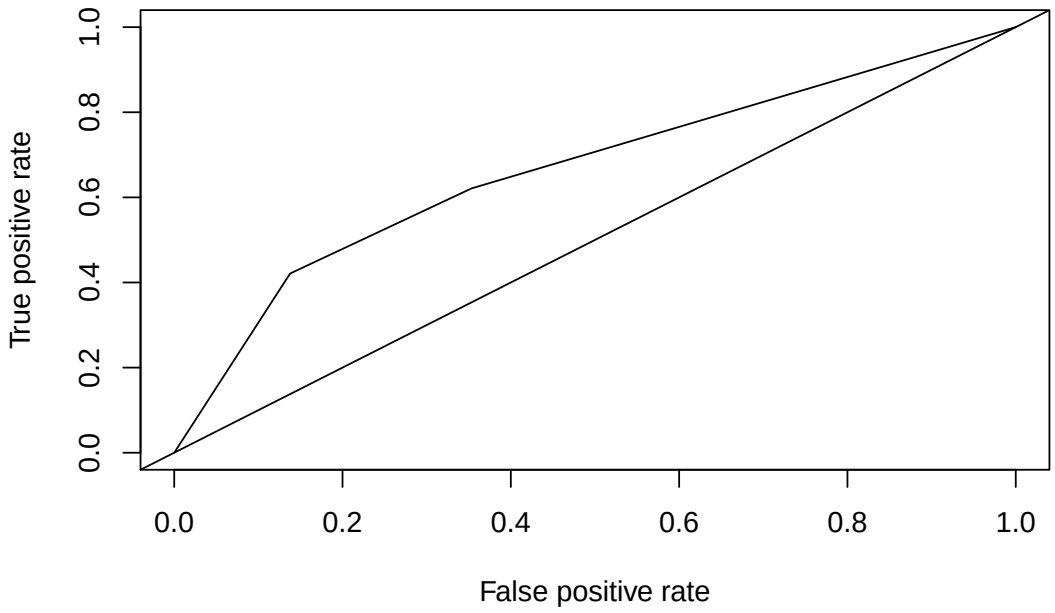
```
## [1] 0.7022901
```

Our test model's accuracy indicates that there were 144 individuals that did not survived and 30 individuals survived, for a total of 144+40 = 184 correct predictions. Based on the above predictions we anticipate a 30% error test rate or 100-70 = 30 or 30%.

The number of false positives for this model would be 55. In other words, there were actually 55 people that did not survive when the test model predicted otherwise. The false positive rate = 55/199 = 0.28

(c) Using the `roc` function, plot the ROC curve for this model. Discuss what you find.

```
pred = prediction(glm.probs,test$survived)
perf = performance(pred,measure="tpr",x.measure = "fpr")
#plot the AUC graph
plot(perf)
#add a diagonal line to the graph
abline(a=0, b= 1)
```




```
#calculating the AUC(Area under the curve)
auc <- performance(pred, measure = "auc")
auc
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
```

```
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.6654901
##
##
## Slot "alpha.values":
## list()
```

The graph shows a classifier that's above the random line, thus, the prediction data that we have used for our model is better than guessing. Though the ROC graph does not have the perfect "hump shaped" curve that's continually increasing, it's a however a respectable graph that slowly moving close to mediocre. We can see that our graph is moving closer to the diagonal line. Ideally, we would want to see a graph that's further away from the diagonal line. We noticed that we are losing in sensitivity (TPR barely reaching 40%).

Also, we have calculated the auc(area under the curve). It shows that Our model has an auc = 0.67. We have gotten a poor AUC as we would want to have an AUC = 1. Thus, the AUC is telling us that some of the test data that we have selected for our model may not be representing the classifier that we are expecting. We may have falsely calculated that did not survived and counted them as survived. The AUC results match the behavior that we are noticing on the graph as the curve is relatively close to the diagonal line.

4. Suppose we use the data to construct a new predictor variable based on a passenger's listed title (i.e. Mr., Mrs., Miss., Master).

   (a) Why might this be an interesting variable to help predict passenger survival? title can be used for possible interesting interesting predictions. For instance, we could simulate the followings: * We could generate a prediction of survival based on those whose title's are rare like the following: Capt, Col, Don, donna, Lady, Sir, the Countess, Dr, Rev, Jonkeer, and Dona * Since there are quite large number of missing 'Age', we can estimate someone's age based on the title. We could generate a prediction of survival based on those whose age were estimated based on the title * We could generate a prediction of survival for title's that matches a female who is possibly a mother or does not have the tittle 'Miss'. We would expect that mother's and child were left out of the boat first. * We could estimate the possibility of survival when adding each title's as coefficient * We could estimate survival when combining the socioeconomic status with the title's of that individual

   (b) Write a function to add this predictor to your dataset.

```
#create a function to return someone's title
nametitle <- function(name){
    result <- gsub('(.*, )|(\\..*)', '',name)
    return(result)
}
```

Note that this code was modeled from: https://www.kaggle.com/mrisdal/titanic/exploring-survival-on-the-titanic/code

   (c) Fit a second logistic regression model including this new feature. Use the `summary` function to look at the model. Did this new feature improve the model?

```
#cleaning and processing title for train and test data set

#convert name from factor to char
train$name = as.character(train$name)
#remove the person's title from the name
```

```r
train$title = nametitle(train$name)
#allocate mlle, ms and mme in the proper buckets
train$title[train$title == 'Mlle'] <-'Miss'
train$title[train$title == 'Ms']   <- 'Miss'
train$title[train$title == 'Mme']  <- 'Mrs'

#convert name from factor to char
test$name = as.character(test$name)
#split out the person's title from the name
test$title = nametitle(test$name)
#allocate mlle, ms and mme in the proper buckets
test$title[test$title == 'Mlle'] <-'Miss'
test$title[test$title == 'Ms']   <- 'Miss'
test$title[test$title == 'Mme']  <- 'Mrs'
```

```r
#fit a linear regression model in order to predict survival based on the person's title

#Use generalized linear model for the logistic regression
glm.fit = glm(as.factor(train$survived)~train$pclass+train$title, data=train, family=binomial)
#generate a summary of the glm for analysis
summary(glm.fit)
```

```
##
## Call:
## glm(formula = as.factor(train$survived) ~ train$pclass + train$title,
##     family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1747  -0.6651  -0.4347   0.6789   2.1938
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.465e+01  1.455e+03  -0.010    0.992
## train$pclass             -9.156e-01  1.005e-01  -9.113   <2e-16 ***
## train$titleCol            1.487e+01  1.455e+03   0.010    0.992
## train$titleDona           3.113e+01  2.058e+03   0.015    0.988
## train$titleDr             1.587e+01  1.455e+03   0.011    0.991
## train$titleJonkheer       4.217e-07  2.058e+03   0.000    1.000
## train$titleLady           3.113e+01  2.058e+03   0.015    0.988
## train$titleMajor          1.557e+01  1.455e+03   0.011    0.991
## train$titleMaster         1.720e+01  1.455e+03   0.012    0.991
## train$titleMiss           1.762e+01  1.455e+03   0.012    0.990
## train$titleMr             1.509e+01  1.455e+03   0.010    0.992
## train$titleMrs            1.783e+01  1.455e+03   0.012    0.990
## train$titleRev            9.156e-01  1.572e+03   0.001    1.000
## train$titleSir            3.113e+01  2.058e+03   0.015    0.988
## train$titlethe Countess   3.113e+01  2.058e+03   0.015    0.988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1397.33  on 1046  degrees of freedom
```

```
## Residual deviance:  986.26  on 1032  degrees of freedom
## AIC: 1016.3
##
## Number of Fisher Scoring iterations: 14
```

When looking at the AIC value for both models, it appears that the new feature did improve the model. For instance, the AIC (Akaike information criterion) for the linear regression containing the pclass in relation to the surviving is at: 1301.6. On the other hand, when adding the title to the model, the AIC decreases to 1016.3. Ideally, the lower the AIC, then the better the model may perform best when used for prediction outside the dataset. Thus, by adding the title to our model, we have a better probabilty to predict who survived the Titanic. Note that AIC should not be used as the sole characteristic for this, but can be used as a starting point.

(d) Comment on the overall fit of this model. For example, you might consider exploring when misclassification occurs.

Coefficients - We noticed that all the coefficients related to title were non-significant ($p > 0.05$), however the coefficient for pclass related factors were significant.

Deviance - We have a deviance of 1397.33 on 1046 degrees of freedom. However, when adding the indepdent variables (pclass and title), the deviance decreased to 985,93 on 1031 degrees of freedom. It was a significant reduction in deviance. The residual deviance was reduced by 411.40 with a loss of 15 degrees of freedom.

fisher scoring - it shows that the model had to go through 14 iterations in order to perform the fit.

AIC - as stated in the previous question/answer, the lower the AIC the better, this model has a lower AIC compare to when the title was not added into the model.

Best fit - to find how well our model fits depends on the difference between the model and the observed data. We calculated the Hosmer-Lemeshow Goodness of Fit to get a better idea.

```
#generating the Hosmer-Lemeshow goodness fit and selecting 10 groups
hoslem.test(train$survived, fitted(glm.fit), g=10)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  train$survived, fitted(glm.fit)
## X-squared = 41.78, df = 8, p-value = 1.489e-06
```

The generated Hosmer-Lemeshow goodness of fit test is based on dividing the sample up according to their predicted probabilities, or risks. Specifically, based on the estimated parameter values. Ideally, small values with large p-values indicate a good fit to the data while large values with p-values below 0.05 indicate a poor fit. Our model does not appear to fit well because we have significance difference between the model and the observed data since the pv-value is under 0.05.

Note that this analysis was modeled agains the following url at: http://www.theanalysisfactor.com/r-glm-model-fit/

Misclassification - looking at the generated confusion matrix calculated in step b from the previous question. It shows the following misclassifcation rate: (FP + FN) / total = 55+23/ 262 = 55.08. Knowing that FP (False Positive) are predictions that should have been false, but were predicted as true. Similarly, FN(False Negative) are predictions that should have been true, but were predicted as false. It shows that our model has a rate of 55.08 of misclassification. We may need to change some of the classfications probability threshold to improve the misclassification rate. In this model we use a threshold of 0.5, thus in order to have a lower misclassification rate, We will need to decrease it to 0.1.

(e) Predict the survival of passengers for each observation in your test data using the new model. Save these predictions as `yhat2`.

```
set.seed(5) #set a seed for reproducibility

yhat2=predict(glm.fit, type="response")
yhat2[1:10] #get the first 10 records
```

```
##         243         919         750         220        1232        1231
## 0.88616756 0.09015142 0.60701110 0.38212093 0.60701110 0.55501303
##         169        1086         609         715
## 0.88616756 0.09015142 0.09015142 0.09015142
```

Given that the type="response" option tells R to output probabilities in the form of P(Y=1|x), then the values that were outputed correspond to the probability of a passenger surviving the Titanic based on socio-economicstatus and title. The first 10 probabilities that are shown correspond then to passengers surviving the Titanic.

5. Another very popular classifier used in data science is called a *random forest*[1].

   (a) Use the `randomForest` function to fit a random forest model with passenger class and title as predictors. Make predictions for the test set using the random forest model. Save these predictions as `yhat3`.

```
set.seed(42) #set a seed value for reproducibility

rf_yhat3 = randomForest(as.factor(test$survived)~test$pclass+ as.factor(test$title), data=trai
# Predict using the test set
yhat3 = predict(rf_yhat3, newdata=test)
```

   ,

   (b) Develop your own random forest model, attempting to improve the model performance. Make predictions for the test set using your new random forest model. Save these predictions as `yhat4`.

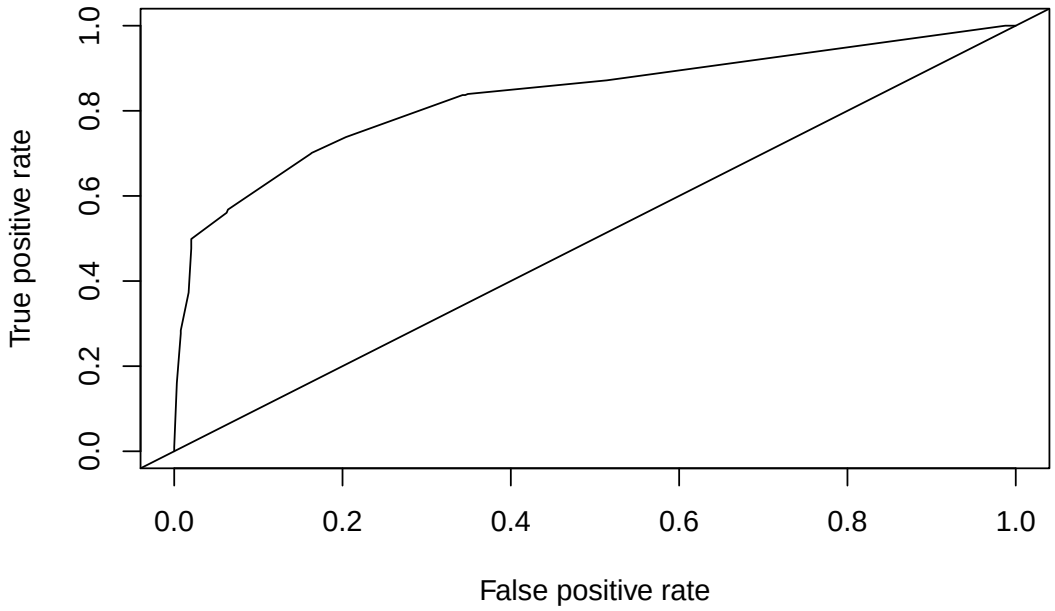
```
set.seed(43) # set the seed for reproducibility
rf_yhat4 = randomForest(as.factor(test$survived)~test$pclass+as.factor(test$title)+test$sex,da
# Predict using the test set
yhat4 = predict(rf_yhat4, newdata=test)
```

   (c) Compare the accuracy of each of the models from this problem set using ROC curves. Comment on which statistical learning method works best for predicting survival of the Titanic passengers.

```
#calculate ROC and AUC for yhat2

glm.probs = predict(glm.fit, type="response")
#calculate the prediction
pred = prediction(glm.probs,train$survived)
#calculate the performance
perf = performance(pred,measure="tpr",x.measure = "fpr")
#ploting the ROC grah
plot(perf)
abline(a=0, b= 1) # create a diagonal line
```

[1] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm




```
#calculating the AUC value
auc <- performance(pred, measure = "auc")
auc
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8324738
##
##
## Slot "alpha.values":
## list()
```

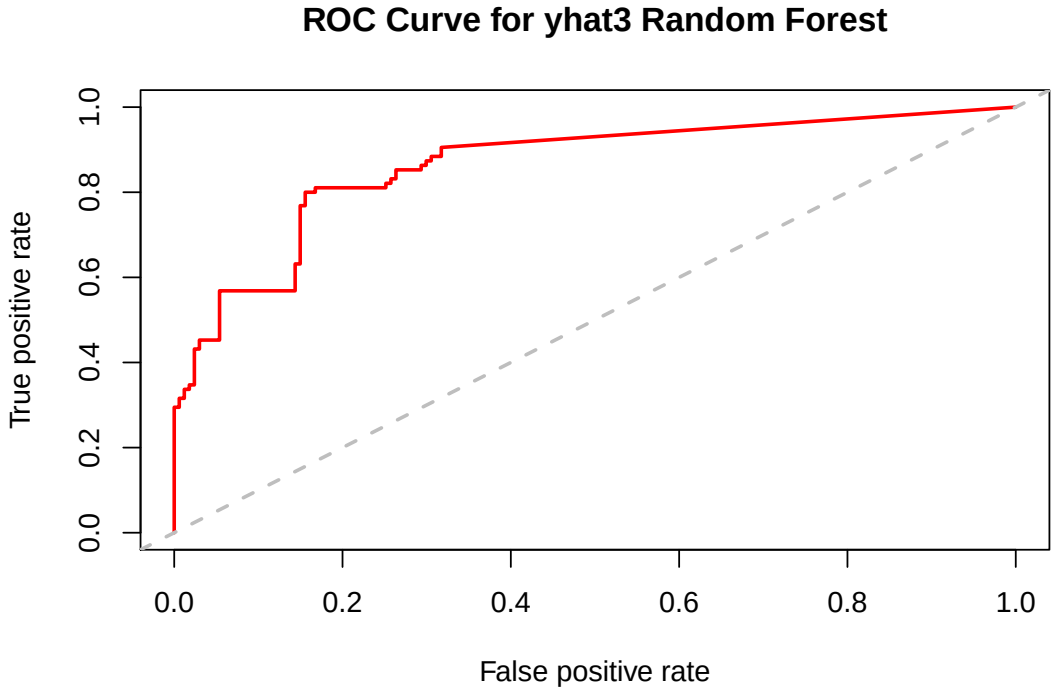
The graph shows a classifier that's above the random line, thus, the prediction data that we have used for our model is better than guessing. Though the ROC graph has a somewhat an acceptable "hump shaped" curve that's continually increasing.However, even though the graph is further away from the diagonal compare to the initial ROC graph generated in the previous question, we would

want to see it a lot more further away from the diagonal line. We noticed that we are losing in sensitivity. TPR is between 40% to 60%.

Also, we have calculated the AUC and for this graph it is at 0.83. AUC is bound between 0 and 1, so this is fairly good. This AUC is telling us that someo of our test data in our model may not be representing the classifier that we are expecting. We may have falsely calculated that did not survived and counted them as survived.

```
#calculate ROC and AUC for yhat3

#get the predictions for yhat3
predictions = as.vector(rf_yhat3$votes[,2])
#generate the prediction of survivers
titanic.rf.pred=prediction(predictions, test$survived)
# generate performance in terms of 0 and 1 or surviving the titanic
titanic.rf.perf = performance(titanic.rf.pred,measure="tpr",x.measure = "fpr")
#plot the curve
plot(titanic.rf.perf,main="ROC Curve for yhat3 Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#compute the area under curve
auc <- performance(titanic.rf.pred,"auc")
auc
```

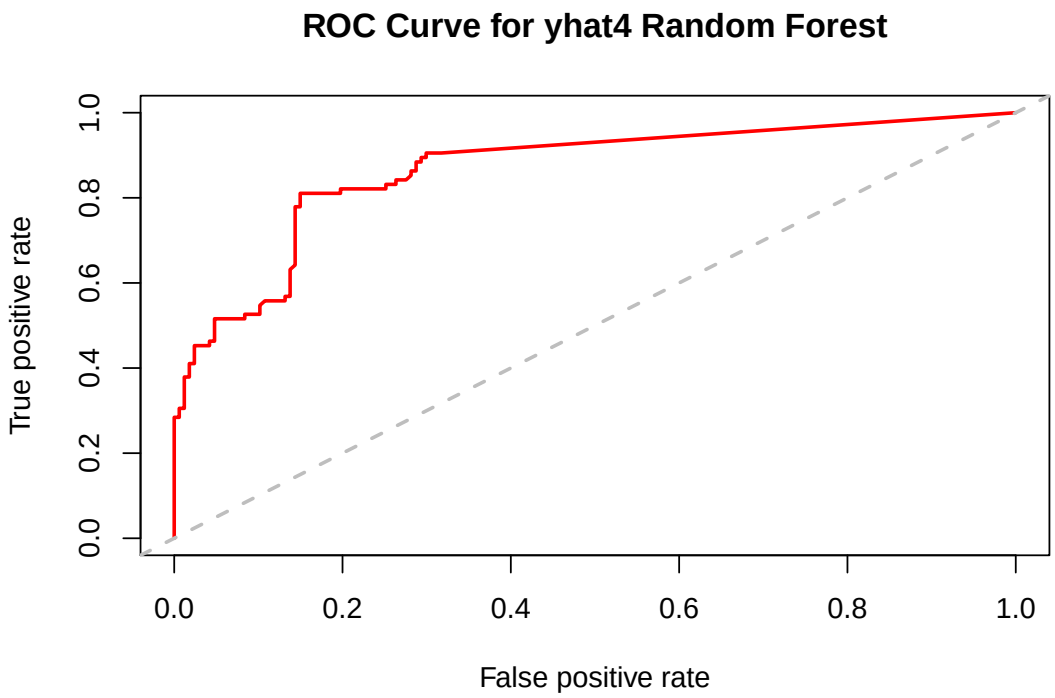
```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
```

```
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.8648598
##
##
## Slot "alpha.values":
## list()
```

The graph shows a classifier that's above the random line, thus, the prediction data that we have used for our model is better than guessing. The ROC graph shows somewhat an acceptable "hump shaped" curve that's continually increasing.However, we noticed multiple times that the lines were dipping. Those dipping might be a sign of irregularity with the data. But, those lines seem to always recover. Ideally, We would want to have a fairly nice curve. We noticed that we are losing in sensitivity and the TPR was around 40%.

Also, we have calculated the AUC and for this graph it is at 0.86. AUC is bound between 0 and 1, so this is fairly good. This AUC is telling us that someo of our test data in our model may not be representing the classifier that we are expecting. We may have falsely calculated that did not survived and counted them as survived.

```
#get the predictions for yhat4
predictions4 = as.vector(rf_yhat4$votes[,2])
#generate the prediction of survivers
titanic.rf.pred4=prediction(predictions4, test$survived)
# generate performance in terms of 0 and 1 or surviving the titanic
titanic.rf.perf4 = performance(titanic.rf.pred4,measure="tpr",x.measure = "fpr")
#plot the curve
plot(titanic.rf.perf4,main="ROC Curve for yhat4 Random Forest",col=2,lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```




```
#compute area under curve
auc <- performance(titanic.rf.pred4,"auc")
auc
```

```
## An object of class "performance"
## Slot "x.name":
## [1] "None"
##
## Slot "y.name":
## [1] "Area under the ROC curve"
##
## Slot "alpha.name":
## [1] "none"
##
## Slot "x.values":
## list()
##
## Slot "y.values":
## [[1]]
## [1] 0.866278
##
##
## Slot "alpha.values":
## list()
```

The graph shows a classifier that's above the random line, thus, the prediction data that we have used for our model is better than guessing. The ROC graph shows somewhat an acceptable "hump shaped" curve that's continually increasing.However, we noticed that some the lines were dipping. It's not as bas as the previous ROC graphs. Those dipping might be a sign of irregularity with

the data. But, those lines always recover. Ideally, We would want to have a fairly nice curve. We noticed that we are losing in sensitivity and the TPR was between 35% and 38%.

Also, we have calculated the AUC and for this graph it is at 0.87. AUC is bound between 0 and 1, so this is pretty good. This AUC is telling us that we have a very small set of our test data in our model that may not be representing the classifier that we are expecting. We may have falsely calculated that did not survived and counted them as survived.

To summarize our analysis, we know that in the context of an ROC curve, the more "up and left" it looks, the larger the AUC will be and thus, the better the classifier is. Based on the results we got from each of the graphs, we can fairly say that the model with the higher AUC will be the best selection. Thus, in this case "yhat4" which is the model were added the "sex" classifier has made an improvement on our prediction.

6. Finally, we will explore a gradient boosted tree model, using the `xgboost` package written by your fellow UW student Tianqi Chen. `xgboost` stands for "Extreme Gradient Boosting", which is state-of-the-art software known for its fast training time and predictive accuracy.

   (a) The XGB algorithm can only handle numeric data, so first we need to convert all categorical variables to a different representation, such as a sparse matrix.

```
library(Matrix)
sparse.matrix.train <- sparse.model.matrix(survived~pclass + sex + title -1, data = train) #co
sparse.matrix.test <-  sparse.model.matrix(survived~pclass + sex + title -1, data = test) #con
output_vector = train$survived #output vector to be predicted
```

   (b) The following code fits a boosted tree model and produces a plot. Run the code and provide an explanation of the resulting plot.

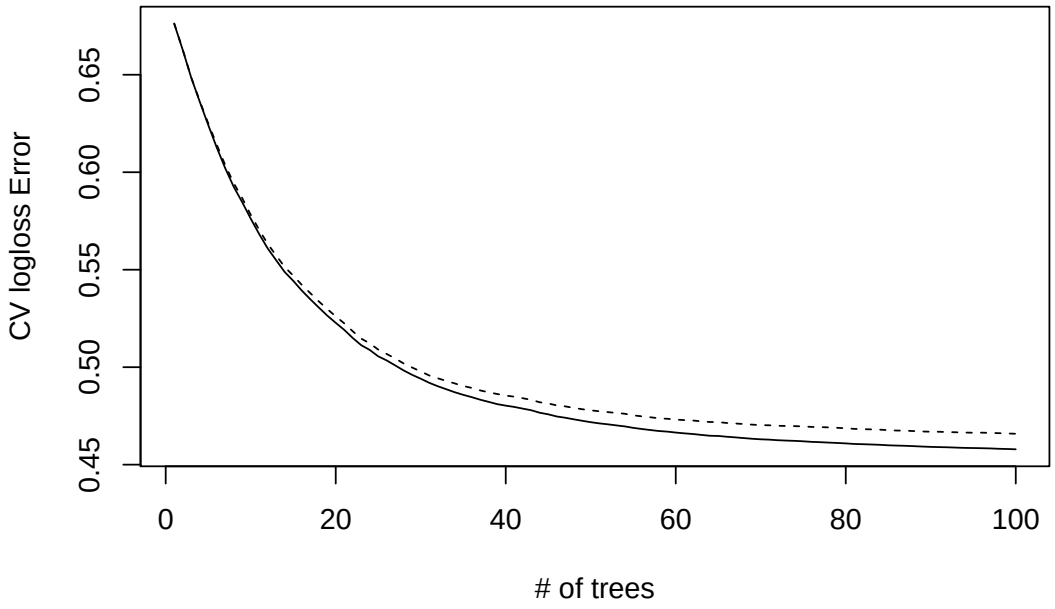
```
xgb.model.one <- xgb.cv(data= sparse.matrix.train,     #train sparse matrix
                        label= output_vector,    #output vector to be predicted
                        eval.metric = 'logloss',     #model minimizes Root Mean Squared Error
                        objective = "reg:logistic", #regression
                        nfold = 10,
                        #tuning parameters
                        max.depth = 3,             #Vary btwn 3-15
                        eta = 0.05,                 #Vary btwn 0.1-0.3
                        nthread = 5,              #Increase this to improve speed
                        subsample= 1,             #Vary btwn 0.8-1
                        colsample_bytree = 0.5,    #Vary btwn 0.3-0.8
                        lambda = 0.5,              #Vary between 0-3
                        alpha = 0.5,               #Vary between 0-3
                        min_child_weight = 3,      #Vary btwn 1-10
                        nround = 100              #Vary btwn 100-3000 based on max.depth, eta,su
                        )
```

```
## [0]  train-logloss:0.676175+0.002062 test-logloss:0.676304+0.002634
## [1]  train-logloss:0.662753+0.003938 test-logloss:0.662928+0.003822
## [2]  train-logloss:0.648508+0.006110 test-logloss:0.648647+0.005345
## [3]  train-logloss:0.636192+0.007081 test-logloss:0.636576+0.007449
## [4]  train-logloss:0.624224+0.006439 test-logloss:0.625090+0.008708
## [5]  train-logloss:0.612705+0.006536 test-logloss:0.614156+0.009509
## [6]  train-logloss:0.602052+0.006903 test-logloss:0.603645+0.010527
## [7]  train-logloss:0.592533+0.006857 test-logloss:0.594430+0.011739
## [8]  train-logloss:0.584619+0.007753 test-logloss:0.586711+0.012651
## [9]  train-logloss:0.576180+0.007282 test-logloss:0.578118+0.012276
## [10] train-logloss:0.568233+0.007157 test-logloss:0.570423+0.013857
```

```
## [11] train-logloss:0.560901+0.007404 test-logloss:0.563238+0.014640
## [12] train-logloss:0.554816+0.008333 test-logloss:0.557243+0.014746
## [13] train-logloss:0.548804+0.008358 test-logloss:0.551395+0.014359
## [14] train-logloss:0.544290+0.007806 test-logloss:0.546957+0.015028
## [15] train-logloss:0.539337+0.008027 test-logloss:0.542240+0.015672
## [16] train-logloss:0.534890+0.008006 test-logloss:0.538017+0.015779
## [17] train-logloss:0.530636+0.007655 test-logloss:0.533788+0.016827
## [18] train-logloss:0.526483+0.007754 test-logloss:0.529852+0.017086
## [19] train-logloss:0.522740+0.007289 test-logloss:0.526048+0.017836
## [20] train-logloss:0.519152+0.007530 test-logloss:0.522492+0.018622
## [21] train-logloss:0.514988+0.007356 test-logloss:0.518425+0.018865
## [22] train-logloss:0.511281+0.006646 test-logloss:0.514637+0.019509
## [23] train-logloss:0.508842+0.006762 test-logloss:0.512228+0.020425
## [24] train-logloss:0.505562+0.006555 test-logloss:0.509021+0.021131
## [25] train-logloss:0.503439+0.006194 test-logloss:0.506767+0.021985
## [26] train-logloss:0.500895+0.006360 test-logloss:0.504474+0.022603
## [27] train-logloss:0.498313+0.006174 test-logloss:0.501895+0.023423
## [28] train-logloss:0.496040+0.005870 test-logloss:0.499779+0.023857
## [29] train-logloss:0.494087+0.005312 test-logloss:0.497828+0.024861
## [30] train-logloss:0.491984+0.005416 test-logloss:0.495739+0.025373
## [31] train-logloss:0.490239+0.005517 test-logloss:0.494311+0.025450
## [32] train-logloss:0.488692+0.005331 test-logloss:0.492926+0.025895
## [33] train-logloss:0.487146+0.005504 test-logloss:0.491670+0.026097
## [34] train-logloss:0.485815+0.005190 test-logloss:0.490360+0.026696
## [35] train-logloss:0.484627+0.005490 test-logloss:0.489353+0.026802
## [36] train-logloss:0.483314+0.005814 test-logloss:0.488081+0.026993
## [37] train-logloss:0.482200+0.005953 test-logloss:0.487100+0.027297
## [38] train-logloss:0.481037+0.005965 test-logloss:0.486236+0.027451
## [39] train-logloss:0.480275+0.005590 test-logloss:0.485403+0.027953
## [40] train-logloss:0.479611+0.005688 test-logloss:0.484880+0.028117
## [41] train-logloss:0.478754+0.005740 test-logloss:0.484083+0.028430
## [42] train-logloss:0.477906+0.005940 test-logloss:0.483247+0.028666
## [43] train-logloss:0.476635+0.005676 test-logloss:0.482002+0.028916
## [44] train-logloss:0.475815+0.005644 test-logloss:0.481386+0.029339
## [45] train-logloss:0.474707+0.005659 test-logloss:0.480393+0.029496
## [46] train-logloss:0.474067+0.005738 test-logloss:0.479901+0.029590
## [47] train-logloss:0.473258+0.005626 test-logloss:0.479080+0.029911
## [48] train-logloss:0.472504+0.005923 test-logloss:0.478436+0.029981
## [49] train-logloss:0.471784+0.005815 test-logloss:0.477856+0.030137
## [50] train-logloss:0.471176+0.006010 test-logloss:0.477382+0.030241
## [51] train-logloss:0.470703+0.006062 test-logloss:0.476952+0.030312
## [52] train-logloss:0.470138+0.006056 test-logloss:0.476549+0.030585
## [53] train-logloss:0.469621+0.005940 test-logloss:0.476120+0.030840
## [54] train-logloss:0.468879+0.005651 test-logloss:0.475303+0.031169
## [55] train-logloss:0.468293+0.005470 test-logloss:0.474786+0.031394
## [56] train-logloss:0.467767+0.005250 test-logloss:0.474323+0.031730
## [57] train-logloss:0.467269+0.005035 test-logloss:0.473788+0.031949
## [58] train-logloss:0.466959+0.005097 test-logloss:0.473524+0.032033
## [59] train-logloss:0.466465+0.005105 test-logloss:0.473140+0.032178
## [60] train-logloss:0.466066+0.005142 test-logloss:0.472848+0.032284
## [61] train-logloss:0.465722+0.005133 test-logloss:0.472654+0.032399
## [62] train-logloss:0.465275+0.005027 test-logloss:0.472203+0.032480
## [63] train-logloss:0.464822+0.005000 test-logloss:0.471875+0.032643
## [64] train-logloss:0.464728+0.004998 test-logloss:0.471836+0.032773
```

```
## [65] train-logloss:0.464322+0.004941 test-logloss:0.471391+0.032856
## [66] train-logloss:0.463997+0.004824 test-logloss:0.471115+0.033078
## [67] train-logloss:0.463635+0.004805 test-logloss:0.470840+0.033254
## [68] train-logloss:0.463280+0.004900 test-logloss:0.470525+0.033293
## [69] train-logloss:0.463009+0.004796 test-logloss:0.470283+0.033486
## [70] train-logloss:0.462834+0.004751 test-logloss:0.470189+0.033601
## [71] train-logloss:0.462534+0.004766 test-logloss:0.469953+0.033706
## [72] train-logloss:0.462352+0.004857 test-logloss:0.469840+0.033732
## [73] train-logloss:0.462201+0.004862 test-logloss:0.469755+0.033745
## [74] train-logloss:0.461984+0.004931 test-logloss:0.469644+0.033747
## [75] train-logloss:0.461684+0.004785 test-logloss:0.469361+0.034004
## [76] train-logloss:0.461529+0.004845 test-logloss:0.469275+0.034007
## [77] train-logloss:0.461321+0.004866 test-logloss:0.469110+0.034062
## [78] train-logloss:0.461120+0.004871 test-logloss:0.468905+0.034212
## [79] train-logloss:0.460913+0.004764 test-logloss:0.468653+0.034453
## [80] train-logloss:0.460614+0.004633 test-logloss:0.468354+0.034628
## [81] train-logloss:0.460499+0.004701 test-logloss:0.468213+0.034637
## [82] train-logloss:0.460342+0.004656 test-logloss:0.468101+0.034709
## [83] train-logloss:0.460180+0.004603 test-logloss:0.467949+0.034822
## [84] train-logloss:0.459930+0.004589 test-logloss:0.467697+0.034974
## [85] train-logloss:0.459790+0.004524 test-logloss:0.467493+0.035089
## [86] train-logloss:0.459664+0.004530 test-logloss:0.467420+0.035146
## [87] train-logloss:0.459489+0.004526 test-logloss:0.467235+0.035198
## [88] train-logloss:0.459287+0.004460 test-logloss:0.467022+0.035386
## [89] train-logloss:0.459109+0.004457 test-logloss:0.466891+0.035534
## [90] train-logloss:0.458998+0.004502 test-logloss:0.466869+0.035591
## [91] train-logloss:0.458901+0.004462 test-logloss:0.466721+0.035689
## [92] train-logloss:0.458749+0.004473 test-logloss:0.466614+0.035743
## [93] train-logloss:0.458597+0.004388 test-logloss:0.466467+0.035862
## [94] train-logloss:0.458527+0.004360 test-logloss:0.466374+0.035919
## [95] train-logloss:0.458438+0.004387 test-logloss:0.466414+0.035923
## [96] train-logloss:0.458327+0.004422 test-logloss:0.466287+0.035901
## [97] train-logloss:0.458145+0.004392 test-logloss:0.466152+0.035967
## [98] train-logloss:0.458027+0.004347 test-logloss:0.466001+0.036093
## [99] train-logloss:0.457888+0.004362 test-logloss:0.465853+0.036074
```

```
plot(data.frame(xgb.model.one)[,1], type='l', col='black', ylab='CV logloss Error', xlab='# of
lines(data.frame(xgb.model.one)[,3], type='l', lty=2, col='black')
```



XGBoost model can evaluate and report on the performance on a test set for the model during training. To that end, the generated plot uses the logloss or binary logarithmic loss evaluation metrics to fit the boosted model tree. The plot shows a logarithmic loss of the XGBOOST model for each evaluation metrics on the train.logloss.mean for the training datasets, and the test.logloss.mean for the test datasets. From looking at the plot, it looks like there is an opportunity to stop the learining perhaps around between 40 to 60 trees.

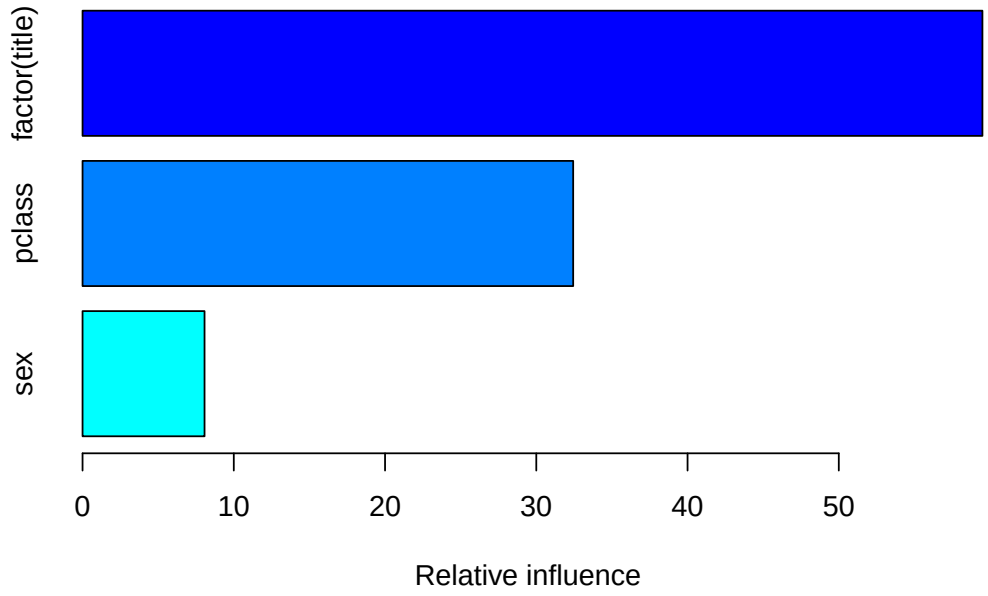
(c) Modify the code to fit a boosted tree model that allows for 8 levels in each tree and uses a learning rate $\eta = .1$. Produce a visualization comparing the two models and explain what you can conclude about the new model. Which model do you prefer and why?

```
library(gbm) # add the gbm library
```

```
## Warning: package 'gbm' was built under R version 3.3.2
## Loading required package: survival
## Warning: package 'survival' was built under R version 3.3.2
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
set.seed(1) # set the seed value for reproducibility
#generate the generalized boost model
gbm_model=gbm(survived~pclass+sex+factor(title),data=train,n.trees=5000,
              interaction.dept=8,distribution="bernoulli", bag.fraction = 0.1)

#generate a summary of the model to get the relative influence
summary(gbm_model)
```

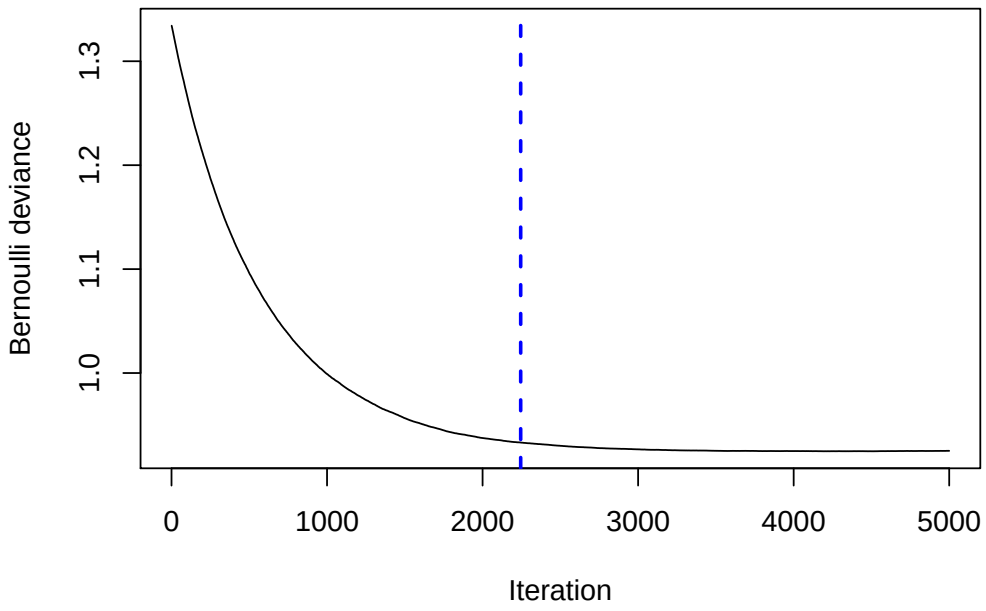



```
##                          var   rel.inf
## factor(title) factor(title) 59.496338
## pclass               pclass 32.438324
## sex                     sex  8.065338
```

```
#generate the performance of the model
gbm.perf(gbm_model)
```

```
## Using OOB method...
```

```
## Warning in gbm.perf(gbm_model): OOB generally underestimates the optimal
## number of iterations although predictive performance is reasonably
## competitive. Using cv.folds>0 when calling gbm usually results in improved
## predictive performance.
```




```
## [1] 2245
```

The plot shows that between 2000 to 3000 trees the curve stops decreasing and started to stabilize. Thus, there is no need to continue training for more trees at the expense of more computation.

When comparing both the xgboost and the gbm modle, I prefer the xgboost model because it controls over-fitting. It shows a better performance as far as running time of generating the model. The gbm model tends to take a long time to run as the cv.fold increases. I even noticed that it bogged down my computer if for instance the cv.fold is over 50. Also, the gbm requires more iterations to come up with a training trees and thus requires more computation time. Having said all that, I do like the relative influence grahp that the gbm model provides. It shows that title has the biggest influence on the gbm object while sex has the lowest influence. Thus, this tells us that creating a gbm model without the "title" classifier will have a better performance. And it also shows that adding the "sex" classifier will improve the performance.