INFX 573 Problem Set 8 - Prediction

Priyanka Saraf

Due: Tuesday, November 26, 2019

Collaborators: Vishwa Kirti

Instructions:

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

1. Download the `problemset8.rmd` file from Canvas. Open `problemset8.rmd` in RStudio and supply your solutions to the assignment by editing `problemset8.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 `ps8_YourLastName_YourFirstName.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)
library(gridExtra)
library(MASS)
library(pROC)
library(arm)
library(randomForest)
library(Metrics)
library(dplyr)
require(mlr)
```

Data: In this problem set we will use the `flights` and `titanic` datasets used previously in class. The flights dataset (via the the *nycflights13* library) contains information on flight delays and weather. Titanic text file contains data about the survival of passengers aboard the Titanic. Table 1 contains a description of this 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

As part of this assignment, we will evaluate the performance of several statistical learning methods. We will fit our learning models using a set of *training* observations and measure its performance on a set of *test* observations.

1. Discuss the advantages of using a training/test split when evaluating statistical models.

using training sets allows not just for the developments of the model, but also for the validation of those models. It would help serve as an estimator of the performance of the model. It would also help to determine if the model chosen is flawed.

Predictions with a continuous output variable

2. Load in the flights dataset. Join the flights data to the weather data based on the departure location, date, and hour of the flight. Exclude data entries which cannot be joined to weather data. Copy the joined data so we can refer to it later.

```
# Load data
library(nycflights13)
dim(flights)
```

```
## [1] 336776     19
```

```
dim(weather)
```

```
## [1] 26115    15
```

```
#join data

weatherflights <- merge(flights,weather,by=c("time_hour","origin"))
```

```r
#View(weatherflights)
dim(weatherflights)
```

```
## [1] 335220     32
```

3. From the joined data, keep only the following columns as we build our first model: departure delay, origin, departure time, temperature, wind speed, precipitation, and visibility. Omit observations that do not have all of these variables present.

```r
subsetfw <- subset(weatherflights,select =c(dep_delay,dep_time,origin,temp,wind_speed,precip,visib))
 #View(subsetfw)
 unique(is.na(subsetfw))
```

```
##        dep_delay dep_time origin  temp wind_speed precip visib
## 1          FALSE    FALSE  FALSE FALSE      FALSE  FALSE FALSE
## 29          TRUE     TRUE  FALSE FALSE      FALSE  FALSE FALSE
## 76790      FALSE    FALSE  FALSE FALSE       TRUE  FALSE FALSE
## 215733     FALSE    FALSE  FALSE  TRUE      FALSE  FALSE FALSE
```

```r
 subsetfw <- na.omit(subsetfw)
 #no more NA values in the subset data
```

4. 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 and 20% will be in the test set. Remember to set the random seed.

```r
set.seed(345)
sampledata <- sample(seq_len(nrow(subsetfw)),size=floor(.8*nrow(subsetfw)),replace = FALSE)
training <- subsetfw [sampledata,]
testing <- subsetfw[-sampledata,]
```

5. Build a linear regression model to predict departure delay using the subset of variables indicated in (3.). What is the RMSE on the training set? What is the RMSE on the test set? Which is higher and is this expected?

```r
lmtrain <- lm(dep_delay~.,data = training)
summary(lmtrain)
```

```
##
## Call:
## lm(formula = dep_delay ~ ., data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -111.54  -18.77   -8.12    2.90 1306.15
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.0980300  0.4889109  -8.382   <2e-16 ***
## dep_time     0.0214213  0.0001565 136.895   <2e-16 ***
## originJFK   -4.7190776  0.1835695 -25.707   <2e-16 ***
```

```
## originLGA   -4.1102022  0.1851252 -22.202   <2e-16 ***
## temp         0.1180023  0.0042854  27.536   <2e-16 ***
## wind_speed   0.2598897  0.0140674  18.475   <2e-16 ***
## precip      68.6177703  2.6840474  25.565   <2e-16 ***
## visib       -2.0708462  0.0404327 -51.217   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.39 on 261510 degrees of freedom
## Multiple R-squared:  0.08924,    Adjusted R-squared:  0.08922
## F-statistic:  3661 on 7 and 261510 DF,  p-value: < 2.2e-16
```

```r
ptrain <- predict(lmtrain,training)
rmse(training$dep_delay,ptrain)
```

```
## [1] 38.39427
```

```r
ptest <- predict(lmtrain,testing)
rmse(testing$dep_delay,ptest)
```

```
## [1] 38.42507
```

Calculating the values on training and testing datasets, the value of training rmse is slightly higher than testing rmse. This was not expected, as a good fit model has testing rmse as nearby or lower to training rmse.

6. Now, improve upon these prediction results by including additional variables in your model. Make sure you keep at least 95% of original data (i.e. about 320K observations across both the training and test datasets). Do not include the arrival time, scheduled arrival time, or the arrival delay in your model. Use the same observations as above for the training and test sets (i.e. keep the same rows but add different variables/columns at your discretion). Can you improve upon the training RMSE? Once you have a model that you feel adequately improves the training RMSE, does your model improve the test RMSE? Which variables did you include in your model?

```r
#including dewp in the model
subset_v1 <- subset(weatherflights,select =c(dep_delay,dep_time,origin,temp,wind_speed,precip,visib,dewp
unique(is.na(subset_v1))
```

```
##        dep_delay dep_time origin  temp wind_speed precip visib  dewp
## 1          FALSE    FALSE  FALSE FALSE      FALSE  FALSE FALSE FALSE
## 29          TRUE     TRUE  FALSE FALSE      FALSE  FALSE FALSE FALSE
## 76790      FALSE    FALSE  FALSE FALSE       TRUE  FALSE FALSE FALSE
## 215733     FALSE    FALSE  FALSE  TRUE      FALSE  FALSE FALSE  TRUE
```

```r
subset_v1 <- na.omit(subset_v1)
set.seed(345)
sampledata_v1 <- sample(seq_len(nrow(subset_v1)),size=floor(.8*nrow(subset_v1)),replace = FALSE)
training_v1 <- subset_v1 [sampledata_v1,]
testing_v1 <- subset_v1[-sampledata_v1,]

lmtrain_v1 <- lm(dep_delay~.,data = training_v1)
summary(lmtrain_v1)
```

```
##
## Call:
## lm(formula = dep_delay ~ ., data = training_v1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -97.10  -17.81   -7.55    2.92 1306.51
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.036e+01  4.950e-01  -20.93   <2e-16 ***
## dep_time     2.301e-02  1.573e-04  146.30   <2e-16 ***
## originJFK   -6.120e+00  1.835e-01  -33.36   <2e-16 ***
## originLGA   -3.700e+00  1.838e-01  -20.13   <2e-16 ***
## temp        -4.766e-01  1.026e-02  -46.47   <2e-16 ***
## wind_speed   4.341e-01  1.422e-02   30.52   <2e-16 ***
## precip       5.573e+01  2.671e+00   20.86   <2e-16 ***
## visib       -8.894e-01  4.420e-02  -20.12   <2e-16 ***
## dewp         6.148e-01  9.649e-03   63.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.1 on 261509 degrees of freedom
## Multiple R-squared:  0.1032, Adjusted R-squared:  0.1031
## F-statistic:  3760 on 8 and 261509 DF,  p-value: < 2.2e-16
```

```r
ptrain_v1 <- predict(lmtrain_v1,training_v1)
rmse(training_v1$dep_delay,ptrain_v1)
```

```
## [1] 38.0997
```

```r
ptest_v1 <- predict(lmtrain_v1,testing_v1)
rmse(training_v1$dep_delay,ptest_v1)
```

```
## Warning in actual - predicted: longer object length is not a multiple of
## shorter object length
```

```
## [1] 42.31144
```

```r
#including dewp and  air_time in the model
subset_v2 <- subset(weatherflights,select =c(dep_delay,dep_time,origin,temp,wind_speed,precip,visib,air_
unique(is.na(subset_v2))
```

```
##        dep_delay dep_time origin  temp wind_speed precip visib air_time
## 1          FALSE    FALSE  FALSE FALSE      FALSE  FALSE FALSE    FALSE
## 29          TRUE     TRUE  FALSE FALSE      FALSE  FALSE FALSE     TRUE
## 375        FALSE    FALSE  FALSE FALSE      FALSE  FALSE FALSE     TRUE
## 76790      FALSE    FALSE  FALSE FALSE       TRUE  FALSE FALSE    FALSE
## 215733     FALSE    FALSE  FALSE  TRUE      FALSE  FALSE FALSE    FALSE
##         dewp
## 1      FALSE
```

```
## 29     FALSE
## 375    FALSE
## 76790  FALSE
## 215733  TRUE
```

```
subset_v2 <- na.omit(subset_v2)
set.seed(345)
sampledata_v2 <- sample(seq_len(nrow(subset_v2)),size=floor(.8*nrow(subset_v2)),replace = FALSE)
training_v2 <- subset_v2 [sampledata_v2,]
testing_v2 <- subset_v2[-sampledata_v2,]

lmtrain_v2 <- lm(dep_delay~.,data = training_v2)
summary(lmtrain_v2)
```

```
##
## Call:
## lm(formula = dep_delay ~ ., data = training_v2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -92.54  -17.69   -7.56    2.92 1310.45
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.3848398  0.5159587  -16.25   <2e-16 ***
## dep_time     0.0225361  0.0001574  143.22   <2e-16 ***
## originJFK   -5.7930707  0.1846033  -31.38   <2e-16 ***
## originLGA   -4.2130205  0.1861027  -22.64   <2e-16 ***
## temp        -0.4719301  0.0102609  -45.99   <2e-16 ***
## wind_speed   0.4294104  0.0142350   30.17   <2e-16 ***
## precip      57.1807436  2.7365869   20.89   <2e-16 ***
## visib       -0.8717482  0.0443609  -19.65   <2e-16 ***
## air_time    -0.0093111  0.0008259  -11.27   <2e-16 ***
## dewp         0.6061781  0.0096536   62.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.03 on 260569 degrees of freedom
## Multiple R-squared:  0.1008, Adjusted R-squared:  0.1008
## F-statistic:  3247 on 9 and 260569 DF,  p-value: < 2.2e-16
```

```
ptrain_v2 <- predict(lmtrain_v2,training_v2)
rmse(training_v2$dep_delay,ptrain_v2)
```

```
## [1] 38.03179
```

```
ptest_v2 <- predict(lmtrain_v2,testing_v2)
rmse(testing_v2$dep_delay,ptest_v2)
```

```
## [1] 37.75614
```

Yes, after including variables "dewp" and "air_time", the training rmse decreases. But the test rmse increases when we just include the dewp. If we also include the air_time, the test rmse decreases too, with the training rmse.

Predictions with a categorical output (classification)

7. Load in the titanic data. 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 and 20% will be in the test set. Remember to set the random seed.

```
getwd()
```

```
## [1] "/home/psaraf30"
```

```
titanic_data <- read.csv('titanic.csv')
#titanic_data <- na.omit(titanic_data)
set.seed(333)
sampletitanic <- sample.int(nrow(titanic_data),size = floor(0.8*nrow(titanic_data)),replace = FALSE)
ttrain <- titanic_data[sampletitanic,]
ttest <- titanic_data[-sampletitanic,]
```

In this problem set our goal is to predict the survival of passengers. First, let's train a logistic regression model for survival that controls for the socioeconomic status of the passenger.

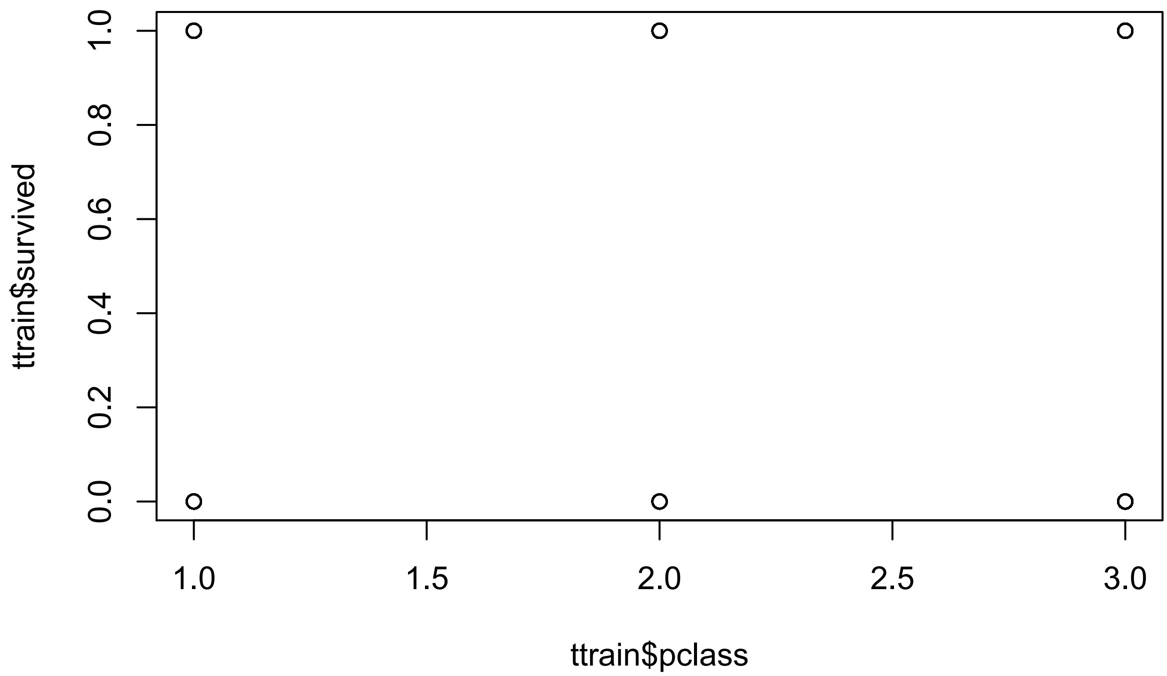
8. Fit the model described above (i.e. one that only takes into account socioeconomic status) using the `glm` function in R.

```
glmfit <- glm(survived~pclass,data = ttrain)
?family
summary(glmfit)
```

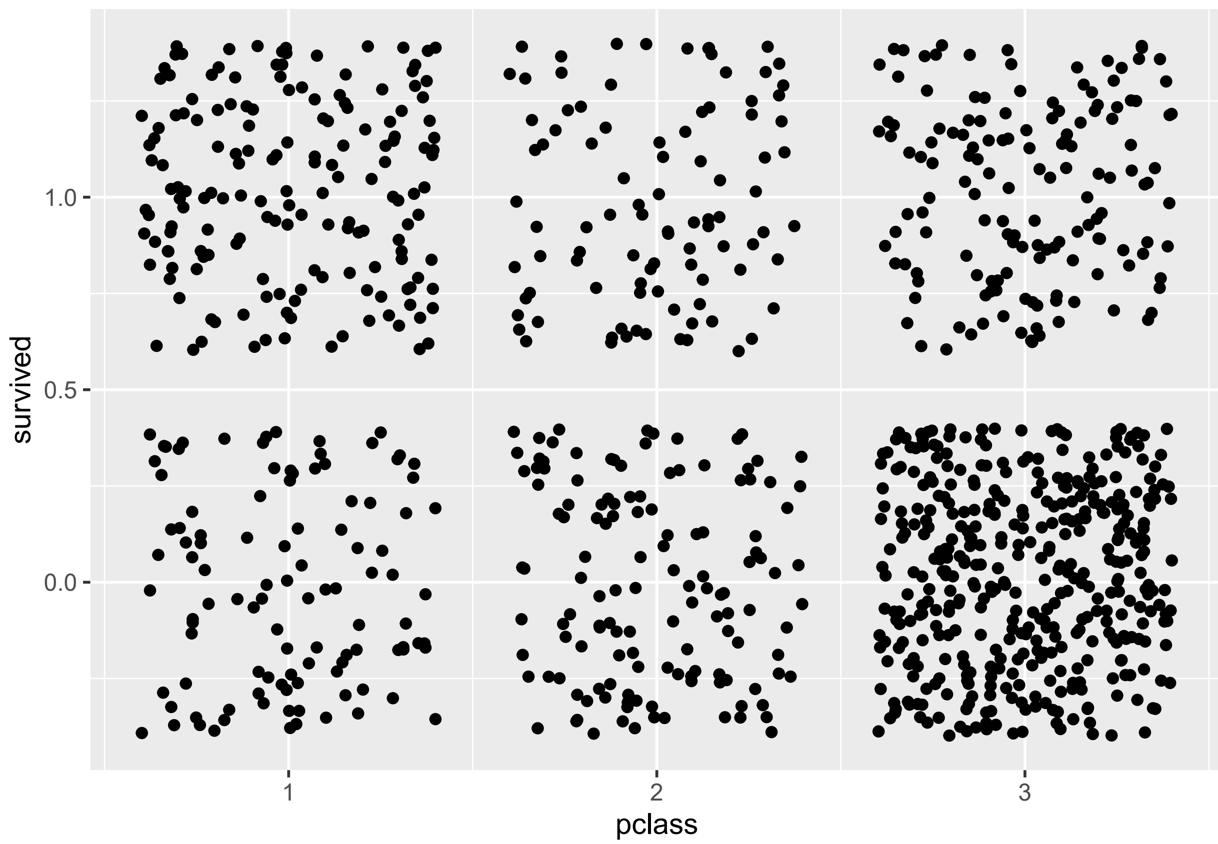
```
##
## Call:
## glm(formula = survived ~ pclass, data = ttrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6000  -0.2636  -0.2636   0.4000   0.7364
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.76814    0.04133  18.585   <2e-16 ***
## pclass      -0.16817    0.01708  -9.844   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2174779)
##
##     Null deviance: 248.34  on 1046  degrees of freedom
## Residual deviance: 227.26  on 1045  degrees of freedom
## AIC: 1377.9
##
## Number of Fisher Scoring iterations: 2
```

9. What might you conclude based on this model about the probability of survival for lower class passengers?

```r
plot(ttrain$pclass,ttrain$survived)
```




```r
ggplot(ttrain,aes(pclass,survived))+geom_jitter()
```



Considering the class of the passengers in the titanic, we see the passengers of pclass = 3 are having the maximum number of datapoints (dense) in terms of death(non survival, "0"), and the maximum number of survival data points is for passenger pclass = 1. Next, let's consider the performance of this model.

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

```
yhat <- predict(glmfit,ttest)
yhatforplot <- predict(glmfit,ttest)
```

11. 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.

```
threshold=0.5
yhat[yhat<threshold]=0
yhat[yhat>=threshold]=1
table(Truth=ttest$survived,Prediction=yhat)
```

```
##       Prediction
## Truth   0   1
##      0 150  17
##      1  59  36
```

```
#View(yhat)
```

Number of false positive on the model we fit are 17, when we create the table.

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

```
rocplot<- roc(survived~yhatforplot,data=ttest)
```

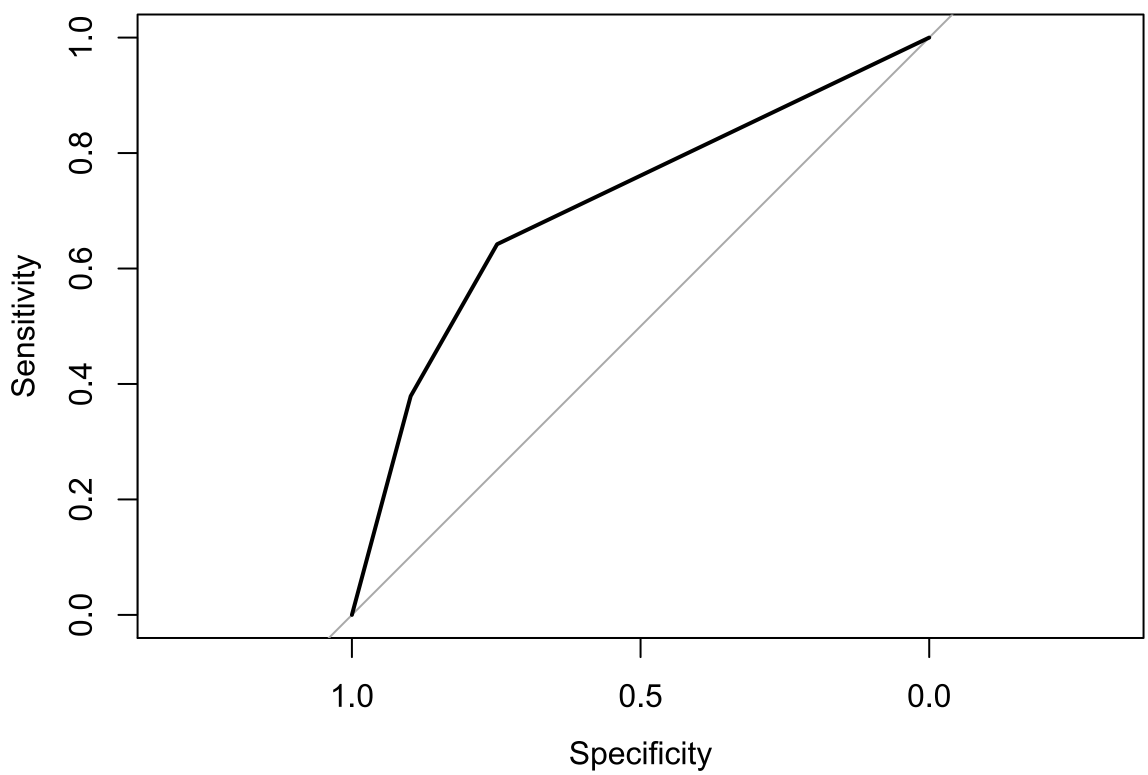
```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
rocplot
```

```
##
## Call:
## roc.formula(formula = survived ~ yhatforplot, data = ttest)
##
## Data: yhatforplot in 167 controls (survived 0) < 95 cases (survived 1).
## Area under the curve: 0.7103
```

```
plot(rocplot)
```



Area under the curve: 0.7103. The greater the curve is from the diagonal, the better it is at distinguish between positives and negatives in general. ###### **13.** Suppose we use the data to construct a new predictor variable based on a passenger's listed title (i.e. Mr., Mrs., Miss., Master). Why might this be an interesting variable to help predict passenger survival?

Use the following custom function to add this predictor to your dataset.

```r
# Making a feature that includes more titles
getTitles <- function(name) {
  for (title in c("Master", "Miss", "Mrs.", "Mr.")) {
    if (grepl(title, name)) {
      return(title)
    }
  }
  return("Nothing")
}
```

The new feature title might help in understanding the age group and gender of the passengers who survived or died. ###### **14.** 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?

```r
#make a new vector coloumn for titles
nametitles <- sapply(ttrain$name, getTitles)
nametitletest <- sapply(ttest$name, getTitles)
#View(nametitles)
#appending to the training titanic dataset
ttrain$name_titles <- nametitles
ttest$name_titles <- nametitletest

traintitanic <- glm(survived~ name_titles+pclass,data=ttrain)
summary(traintitanic)
```

```
##
## Call:
## glm(formula = survived ~ name_titles + pclass, data = ttrain)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.91882  -0.21690  -0.07299   0.22509   0.92701
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.87614    0.06693  13.090  < 2e-16 ***
## name_titlesMiss      0.14978    0.06053   2.474 0.013502 *
## name_titlesMr.      -0.37142    0.05636  -6.590 6.96e-11 ***
## name_titlesMrs.      0.18659    0.06312   2.956 0.003186 **
## name_titlesNothing -0.31966    0.09552  -3.346 0.000848 ***
## pclass              -0.14391    0.01486  -9.686  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1534543)
##
##     Null deviance: 248.34  on 1046  degrees of freedom
## Residual deviance: 159.75  on 1041  degrees of freedom
## AIC: 1016.8
##
## Number of Fisher Scoring iterations: 2
```

```r
predicttitanic <- predict(traintitanic,ttest)
rocplot_new <- roc(survived~predicttitanic,data = ttest)
```

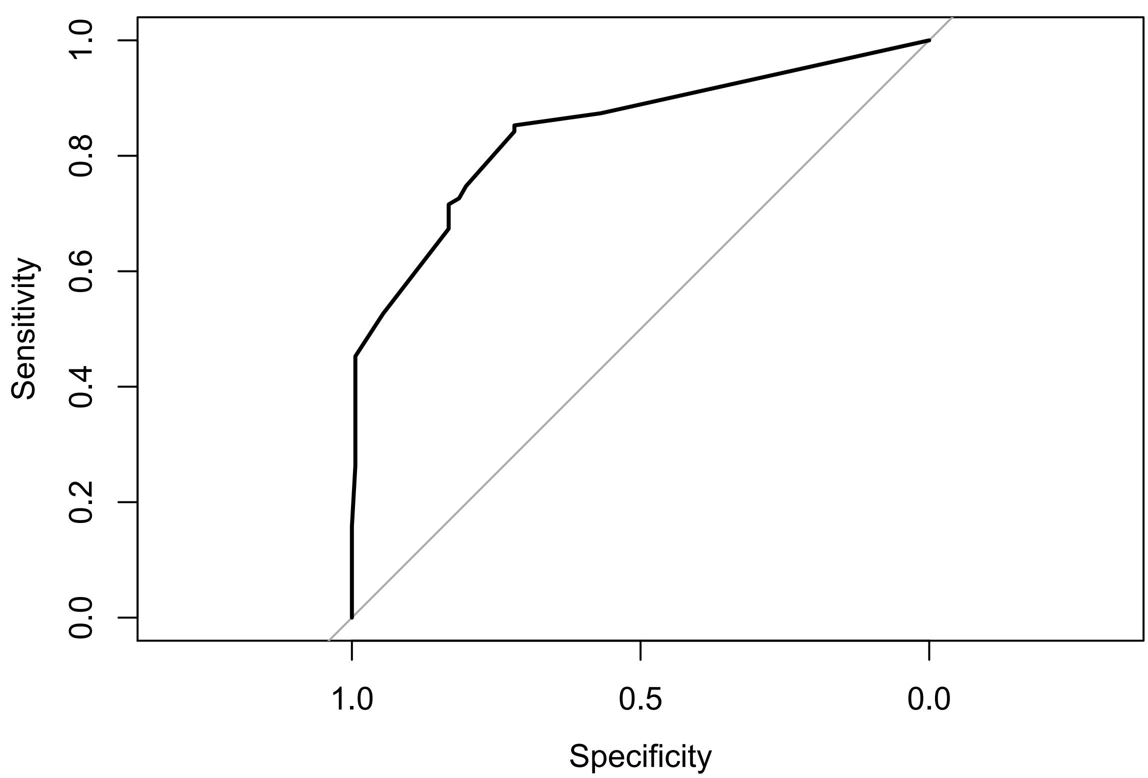
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
rocplot_new
```

```
##
## Call:
## roc.formula(formula = survived ~ predicttitanic, data = ttest)
##
## Data: predicttitanic in 167 controls (survived 0) < 95 cases (survived 1).
## Area under the curve: 0.8435
```

```r
plot(rocplot_new)
```



Area under curve is now 0.8435, which has increased when we included the feature, title. It means the model became better at distinguishing between the negatives and the postitions.

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

```r
#calculating the false positives of the new model with title feature
predicttitanic[predicttitanic<threshold]=0
predicttitanic[predicttitanic>=threshold]=1
#View(predicttitanic)

#calculate the false positives
table(Truth=ttest$survived,Prediction=predicttitanic)
```

```
##      Prediction
## Truth   0   1
##     0 139  28
##     1  27  68
```

It is seen from the table that the number of false positives has increased from 17 to 28 after including the title feature.

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

```r
yhat2 <- predict(traintitanic,ttest)
```

Random forests

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

17. 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`.

```r
typeof(ttrain$name_titles)
```

```
## [1] "character"
```

```r
#ttrain <- filter(ttrain,!grepl("Nothing",ttrain$name_titles))
#View(ttrain)
unique(is.na(ttrain$pclass))
```

```
## [1] FALSE
```

```r
unique(is.na(ttrain$survived))
```

```
## [1] FALSE
```

```r
unique(is.na(ttrain$name_titles))
```

```
## [1] FALSE
```

```r
unique(is.na(ttest$pclass))
```

```
## [1] FALSE
```

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

```r
unique(is.na(ttest$survived))
```

```
## [1] FALSE
```

```r
unique(is.na(ttest$name_titles))
```

```
## [1] FALSE
```

```r
ttrain$name_titles <- as.factor(ttrain$name_titles)
ttest$name_titles <- as.factor(ttest$name_titles)
randomfit <- randomForest(survived~name_titles+pclass,data=ttrain)
```

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

```r
summary(randomfit)
```

```
##                 Length Class  Mode
## call               3   -none- call
## type               1   -none- character
## predicted       1047   -none- numeric
## mse              500   -none- numeric
## rsq              500   -none- numeric
## oob.times       1047   -none- numeric
## importance         2   -none- numeric
## importanceSD       0   -none- NULL
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            11   -none- list
## coefs              0   -none- NULL
## y               1047   -none- numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```r
yhat3 <- predict(randomfit,ttest)
rmse(ttest$survived,yhat3)
```

```
## [1] 0.3656174
```

```r
#view(yhat3)
```

18. Develop your own random forest model (i.e. add/remove variables at your discretion), attempting to improve the model performance. Make predictions for the test set using your new random forest model. Save these predictions as `yhat4`.

```r
ttrain <- na.omit(ttrain)

randomfitnew <- randomForest(survived~pclass+sex+name_titles+sibsp,data=ttrain)
```

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

```r
yhat4 <- predict(randomfitnew,ttest)
rmse(ttest$survived,yhat4)
```

```
## [1] 0.602159
```

19. 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.

```r
rocplot3<- roc(survived~yhat3,data=ttest)
```

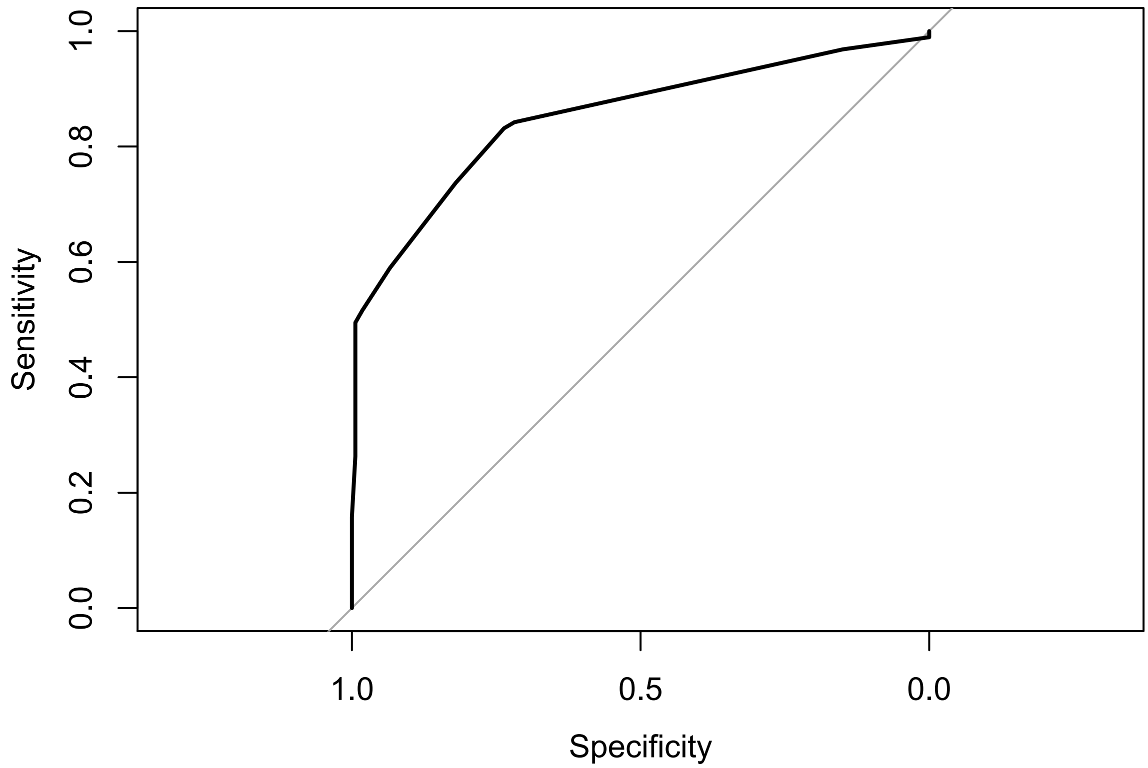
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
rocplot3
```

```
##
## Call:
## roc.formula(formula = survived ~ yhat3, data = ttest)
##
## Data: yhat3 in 167 controls (survived 0) < 95 cases (survived 1).
## Area under the curve: 0.8515
```

```r
plot(rocplot3)
```



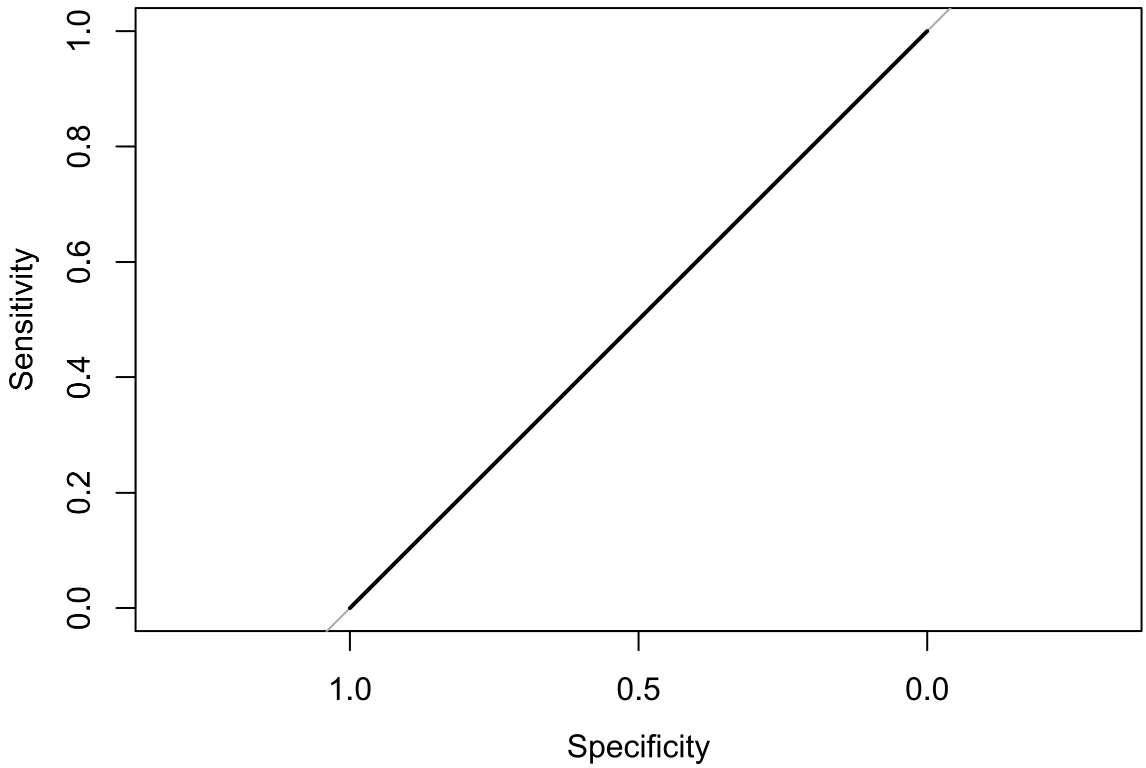

```
rocplot4<- roc(survived~yhat4,data=ttest)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
rocplot4
```

```
##
## Call:
## roc.formula(formula = survived ~ yhat4, data = ttest)
##
## Data: yhat4 in 167 controls (survived 0) < 95 cases (survived 1).
## Area under the curve: 0.5
```

```
plot(rocplot4)
```



The area under the curve for the last question, yhat4 test data,is 0.5, which means that this model is a fail, or a very bad fit. We can even observe from the graph that there is barely some gap between the diagonal and ROC curve for yhat4. Where as the are under the curve for yhat 3 is 0.8, and is considered a good fit.