INFX 573 Problem Set 8 - Prediction

Divya Gaurav Tripathi

Due: Tuesday, November 26, 2019

Collaborators: Aditya Harish Nayak, Sejal Khatri

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(dplyr)
library(pROC)
library(arm)
library(randomForest)
library(Metrics)
```

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.

We do it because we can train our statistical models on training set and evaluate its performance on test set. This can tell how are model is behaving while predicting the dependent variable for unseen data. It can help us determine if our model is underfitting or overfitting.

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)
```

```
flightsdata <- data(flights) #loading the flights data
weatherdata <- data(weather) #loading the weather data

str(flights)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    336776 obs. of  19 variables:
##  $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
```

```
##  $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr  "UA" "UA" "AA" "B6" ...
##  $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr  "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr  "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr  "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num  1400 1416 1089 1576 762 ...
##  $ hour          : num  5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num  15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

```
str(weather)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    26115 obs. of  15 variables:
##  $ origin    : chr  "EWR" "EWR" "EWR" "EWR" ...
##  $ year      : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hour      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ temp      : num  39 39 39 39.9 39 ...
##  $ dewp      : num  26.1 27 28 28 28 ...
##  $ humid     : num  59.4 61.6 64.4 62.2 64.4 ...
##  $ wind_dir  : num  270 250 240 250 260 240 240 250 260 260 ...
##  $ wind_speed: num  10.36 8.06 11.51 12.66 12.66 ...
##  $ wind_gust : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ precip    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pressure  : num  1012 1012 1012 1012 1012 ...
##  $ visib     : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ time_hour : POSIXct, format: "2013-01-01 01:00:00" "2013-01-01 02:00:00" ...
```

```
flights_weather <- inner_join(flights, weather, by = c("origin" = "origin", "time_hour" = "time_hour"))

str(flights_weather)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    335220 obs. of  32 variables:
##  $ year.x        : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month.x       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day.x         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr  "UA" "UA" "AA" "B6" ...
##  $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr  "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr  "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr  "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num  1400 1416 1089 1576 762 ...
##  $ hour.x        : num  5 5 5 5 6 5 6 6 6 6 ...
```

```
##  $ minute        : num  15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
##  $ year.y        : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month.y       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day.y         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hour.y        : int  5 5 5 5 6 5 6 6 6 6 ...
##  $ temp          : num  39 39.9 39 39 39.9 ...
##  $ dewp          : num  28 25 27 27 25 ...
##  $ humid         : num  64.4 54.8 61.6 61.6 54.8 ...
##  $ wind_dir      : num  260 250 260 260 260 260 240 260 260 260 ...
##  $ wind_speed    : num  12.7 15 15 15 16.1 ...
##  $ wind_gust     : num  NA 21.9 NA NA 23 ...
##  $ precip        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pressure      : num  1012 1011 1012 1012 1012 ...
##  $ visib         : num  10 10 10 10 10 10 10 10 10 10 ...
```

```
#inner join of flights and weather data based on origin and time_hour
```

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.

```
flights_weather_rc <- flights_weather %>%
 dplyr::select(dep_delay,origin,dep_time,temp,wind_speed,precip,visib)
#rc in flights_weather_rc means relevant columns

#We selected only relevant columns and stored the result in flights_weather_rc

flightsweather_cld <-na.omit(flights_weather_rc)

#cld stands for cleaned data in flightsweather_cld

#We removed all the na values and saved result in flightsweather_cld
```

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.

```
set.seed(1)
#setting random seed

train <- sample(nrow(flightsweather_cld),nrow(flightsweather_cld)*0.8)
#taking 80% as training data in train,
#we selected index of rows

#str(traindataindex)

test <- flightsweather_cld[-train,]
#We created test data and saved in test,
#we selected all those rows from flightsweather_cleaneddata
#which were not in train
```

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?

```
lm.fit <- lm(dep_delay~.-dep_delay, data= flightsweather_cld, subset = train)
# We fitted a linear model for dep_delay depending on
```

```
#all variables except itself, we took the train subset from data frame flightsweather_cleaneddata

predictions_depdelay <- predict(lm.fit,flightsweather_cld)
#We used the predict() function to take the predictions from our linear model

mse_train <- mean((flightsweather_cld$dep_delay - predictions_depdelay)[train]^2)
#I found the Mean Squared Error on the training set

rmse_train <- sqrt(mse_train)
#I found the root mean square error of training data

mse_test <- mean((flightsweather_cld$dep_delay - predictions_depdelay)[-train]^2)
#I found the mean squared error for test data, - train means rows not in train

rmse_test <- sqrt(mse_test)
#I found the root mean square error for test data
```

The RMSE for train = 38.4187 The RMSE for test data = 38.3268

It is unexpected as I was expecting RMSE of train data to be lesser.

We get the Root Mean Square Error when we take square root of the mean of squares of residuals. We can use it as a measure of the spread of the y values about the predicted y value. We find that the RMSE of test data is only slightly less than that of train data. Yes it is expected as our training and test data is from same large data frame. It means our model is doing slightly better on test data than our train data.

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?

We would expand our model by adding more variables to the existing model.

```
lm.fit_expand <- update(lm.fit, .~.+carrier+humid,data=flights_weather)

#We included carrier and humid as more variables from the
#flights_weathee dataframe which we got from joining flights and weather data frames

lm.fit_expand
```

```
##
## Call:
## lm(formula = dep_delay ~ origin + dep_time + temp + wind_speed +
##     precip + visib + carrier + humid, data = flights_weather,
##     subset = train)
##
## Coefficients:
## (Intercept)    originJFK    originLGA     dep_time         temp
##   -39.63278     -3.54549     -0.89877      0.02271      0.09398
##  wind_speed       precip        visib    carrierAA    carrierAS
##     0.44445     52.73831     -0.43917     -4.44182     -8.01087
##   carrierB6    carrierDL    carrierEV    carrierF9    carrierFL
##    -1.80421     -4.99392      4.64761      5.85256      3.35610
```

```
##   carrierHA    carrierMQ    carrierOO    carrierUA    carrierUS
##    -2.45575     -4.00352     -4.62109     -2.67816     -8.16959
##   carrierVX    carrierWN    carrierYV        humid
##     0.12385      4.73776      0.47429      0.31524
```

```
flights_weather_av <- flights_weather_rc <- flights_weather %>%
 dplyr::select(dep_delay,origin,dep_time,temp,wind_speed,precip,visib,carrier,humid)

#av stands for additional variables in flights_weather_av

#We added carrier and humid variables and saved in flights_weather_av

flightsweather_av_cld <- na.omit(flights_weather_av)

#av_cld stands for additional variables cleaned data
#We removed the na values and saved in flightsweather_av_cld


percent_originaldata <-
nrow(flightsweather_av_cld)/ nrow(flightsweather_cld)*100

percent_originaldata
```

```
## [1] 100
```

```
#We found that the data retained is 100% with these variables
```

We added another variables carrier and pressure. We would now create our training and test sets.

```
set.seed(1)
#setting random seed as 1 which we had set earlier to get similar row indices

train_av <- sample(nrow(flightsweather_av_cld),nrow(flightsweather_av_cld)*0.8)
#av stands for additional variables in train_av
#We are taking 80% as training data in train_av_cld, we have selected index of rows


test_av <- flightsweather_av_cld[-train,]
#av stands for additional variables in test_av
#We created test data and saved in test_av, we have selected all those rows from flightsweather_av_cld
```

We would now use the predict function and find RMSE for our training and test data.

```
predictions_depdelay_av <- predict(lm.fit_expand,flightsweather_av_cld)
#av stands for additional variables in predictions_depdelay_av
#We used the predict() function to take the predictions from our linear model,
#we have used the lm.fit_expand model which we had fitted for
#more variables by expanding the previous model

mse_train_av <- mean((flightsweather_av_cld$dep_delay - predictions_depdelay_av)[train_av]^2)
#I found the Mean Squared Error on the training set with additional variables

rmse_train_av <- sqrt(mse_train_av)
#I found the root mean square error of training data

mse_test_av <- mean((flightsweather_av_cld$dep_delay - predictions_depdelay_av)[-train_av]^2)
#I found the mean squared error for test data, - train means rows not in train
```

```
rmse_test_av <- sqrt(mse_test_av)
#I found the root mean square error for test data
```

When we have fitted our data with more variables (carrier and pressure), the RMSE of training data = 37.95033; and the RMSE of test data is 37.86247

When we had lesser variables: The RMSE for train was 38.4187 The RMSE for test data was 38.3268

This tells that the RMSE for training data and test data have decreased from earlier. It tells that our model is fitting better for the training data and is also doing better for predicting departure delay from unseen data.

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.

```
titanic_data <- read.csv('titanic.csv')
#We loaded the data set

titanic_data$pclass = as.factor(titanic_data$pclass)
#As we are going to use pclass as the socioeconomic class, we converted it to factor
#pclass =1 : upper class
#pclass = 2: middle class
#pclass = 3: lower class

#str(titanic_data)

set.seed(1)
#We set the random seed

titanic_train_index <- sample(nrow(titanic_data),nrow(titanic_data)*0.8)
#We took th row numbers of 80% of data as the training data

titanic_train =titanic_data[titanic_train_index,]
#We created the train data from training row numbers

titanic_test=titanic_data[-titanic_train_index,]
#We created the test data from removing the row numbers of training data
```

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.

```
#For socioeconomic status we should take class of passenger(pclass)
reg.model = glm(survived ~pclass, family = binomial, data = titanic_train)

#We fitted a logistic regression model for survived based on pclass for training data
```

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

```
summary(reg.model)
```

```
##
## Call:
```

```
## glm(formula = survived ~ pclass, family = binomial, data = titanic_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4345  -0.7671  -0.7671   0.9405   1.6535
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.5866     0.1287   4.559 5.14e-06 ***
## pclass2      -0.8716     0.1876  -4.646 3.38e-06 ***
## pclass3      -1.6594     0.1609 -10.316  < 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: 1399.2  on 1046  degrees of freedom
## Residual deviance: 1283.5  on 1044  degrees of freedom
## AIC: 1289.5
##
## Number of Fisher Scoring iterations: 4
```

```
#plot(reg.model)
```

This model tells that passenger's class is significant from its p value. So the survival of a passenger is dependant on his socioeconomic class.

The survival probability for passengers of 3rd class(lower class) is the least from the p values.

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(reg.model, titanic_test)
#We found predictions based on our model and saved the result as yhat

#yhat
```

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.

```
yhat_classify = ifelse(yhat >0.5, 1, 0)
#Every value in yhat > 0.5 is classified as 1, and every value in yhat< 0.5 is classified as 0
table(yhat_classify)
```

```
## yhat_classify
##   0   1
## 202  60
```

```
#yhat_classify has 202 values of 0 and 60 values of 1

table(yhat_classify, titanic_test$survived)
```

```
##
## yhat_classify   0   1
##             0 140  62
##             1  29  31
```

```
#This table gives the value of predicted and actual survived and not survived
```

The number of false positives on test data = 29. This means that our logistic regression model has predicted survival for those 29 pasengers, who did not survive in reality.

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

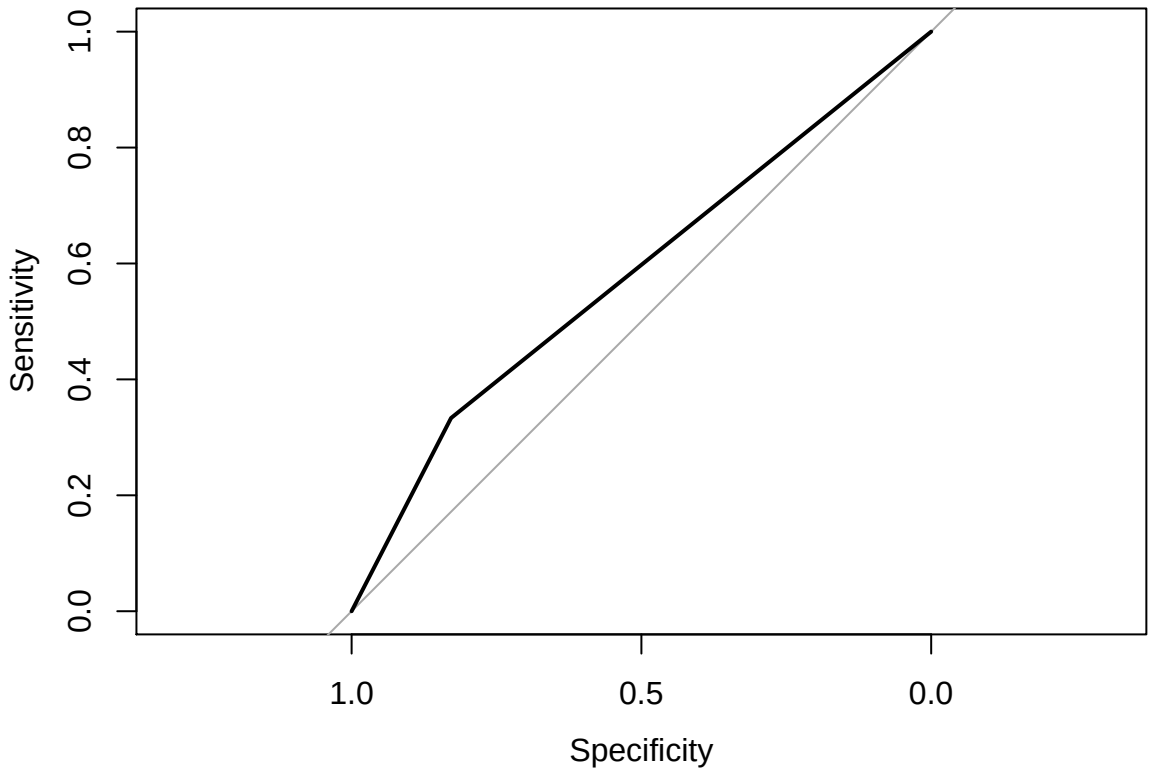
```
roc_output <-roc(titanic_test$survived,yhat_classify)
```

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

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

```
#We utilised the roc function, actual result, predicted result to plot sensitivity and specificity

plot(roc_output, print.cutoffs.at = seq(0,1,0.1))
```




```
#summary(roc_output)
```

sensitivity = true positive/ actual total positives specificity = true negatives/ actual total negatives

The threshold level is used to determine the tradeoff betwenn sensitivity ans specificty. We can find that as sensitivity increases, the specificity decreases.

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.

```
# 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")
}
```

Let us add another column for title of every passenger.

```
titanicdata_title <- titanic_data

#We copied titanic_data into another dataframe titanicdata_title


num_rows =  nrow(titanic_data)

for (i in 1:num_rows) {

  titanicdata_title$title[i] = getTitles(titanic_data$name[i] )
}

#We added another column title having title of every passenger
```

This can tell us if there is a difference between survival rates of adult males and females and child males and females. It might tell us if they were treated differently in giving life boats etc.

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?

Let us again divide our data into training set(80%) and test set(20%), then fit a logistic regression model.

```
set.seed(1)
#We set the random seed

title_train_index <- sample(nrow(titanicdata_title),nrow(titanicdata_title)*0.8)
#We took th row numbers of 80% of data as the training data

title_train = titanicdata_title[title_train_index,]
#We created the train data from training row numbers

title_test= titanicdata_title[-title_train_index,]
#We created the test data from
#removing the row numbers of training data

reg2.model = glm(survived ~pclass + title, family = binomial, data = title_train)
#Our second model has pclass and title
#as predictors for survival.
#We fitted a regularised model (reg2.model) on training data


summary(reg2.model)
```

```
##
## Call:
## glm(formula = survived ~ pclass + title, family = binomial, data = title_train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2923  -0.6442  -0.4140   0.6449   2.2357
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.7660     0.3482   5.071 3.95e-07 ***
## pclass2       -1.0876     0.2291  -4.747 2.06e-06 ***
## pclass3       -2.0342     0.2071  -9.821  < 2e-16 ***
## titleMiss      0.4097     0.3309   1.238 0.215618
## titleMr.      -2.1454     0.3251  -6.599 4.15e-11 ***
## titleMrs.      0.7863     0.3602   2.183 0.029053 *
## titleNothing  -1.8133     0.5340  -3.396 0.000684 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1399.2  on 1046  degrees of freedom
## Residual deviance:  976.2  on 1040  degrees of freedom
## AIC: 990.2
##
## Number of Fisher Scoring iterations: 4
```

We find frpm the p value that several of the titles are statistically significant. We find that the AIC of this model = 990.2. The AIC of previous model was 1289.5. As we havea few more statistically predictors, and the AIC has reduced, we can say that our model has improved.

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

We have to answer about when misclassification occur. Let us calculate misclassification and then do an exploratory analysis of misclassification cases.

```
reg2predictions <- predict(reg2.model, title_test)
#We found predictions for test data set based on our model and saved the result as reg2predictions

reg2pred_classify = ifelse(reg2predictions >0.5, 1, 0)
#We classified our predictions as 0 or 1 based on threshold level of 0.5


title_test$prediction = reg2pred_classify
#We added the predictions coulumn for corresponding rows in test data

#reg2pred_classify

#str(titanicdata_title)
```

Let us make the predicted vs actual table to calculate precison and misclassification.

```
table(reg2pred_classify)
```

```
## reg2pred_classify
##   0   1
## 201  61
```

```
#yhat_classify has 201 values of 0 and 61 values of 1

table(reg2pred_classify, title_test$survived)
```

```
##
## reg2pred_classify   0   1
##                 0 158  43
##                 1  11  50
```

```
#This table gives the value of predicted and actual survived and not survived
```

accuracy = (true positive + true negative)/ total examples = (158+50)/(158+43+11+50) = 208/262 misclassification = 1- accuracy = 1- 0.7938931 = 0.206 In a random prediction we would have considered accuracy of 0.5 and misclassification of 0.5. Our accuracy is higher and mislassification is lowerr than that, so we can say that the overall fit of the model is good.

We would create another dataframe for misclassification, i.e. when actual and predicted are different. We are just doing this to analyse if there is any pattern for misclassification.

```
titanic_misclassified <- subset(title_test, title_test$survived != title_test$prediction)

#titanic_misclassified has all thr cases for incorrect predictions
#titanic_misclassified
```

There are 54 cases of misclassification of test data. Our predictors were passenger class and title. We eyeballed the data and observe that most of the misclassified are class 3 passengers. Other than this, at least in the test data set, there is no pattern of when misclassification occurs.

We did an exploratory analyzsis of misclassification. Let us also calculate misclassification.

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

```
yhat2 <- predict(reg2.model, title_test)
#yhat 2 is the prediction on test data

yhat2_classify = ifelse(yhat2 >0.5, 1, 0)
#We classified yhat2 based on threshold of 0.5
```

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`.

```
title_train_factor <- title_train
#We took a copy of title_train in title_train_factor data frame

title_train_factor$title = as.factor(title_train$title)
#We have to put title as factor for using random forest,
#so we changed title as factor

randomforest.fit <- randomForest(as.factor(survived) ~ pclass + title,
                      data=title_train_factor,
                      importance=TRUE,
```

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

```
                      ntree=100)

#We fitted a random forest

#str(title_train)
#title_train %>% distinct(title)

title_test$title <- as.factor(title_test$title)
#We changed title column of title_test set as factor,
#so that predict function can run

yhat3 <- predict(randomforest.fit, title_test)



summary(yhat3)
```

```
##   0   1
## 206  56
```

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`.

For building another random forest model, we would add two variables- sex(to tell if male or female had any impact) and sibsp(to tell if somebody's siblings or spouses had any impact on survival, a person might have given higher priority to saving their family member).

```
randomforest.fit <- randomForest(as.factor(survived) ~ pclass + title + sex + sibsp,
                      data=title_train_factor,
                      importance=TRUE,
                      ntree=100)
#We made another random forest model with pclass,title,sex, sibsp as predictors

yhat4 <- predict(randomforest.fit,title_test)
#yhat4 is the predicted values for test data
```

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.

We would plot the ROC curves for all the models- from yhat to yat4.

Let us plot for yhat. We had fit a logistic regression model based on socioeconomic class. We had found the predictions and saved it as yhat. Then we had used a threshold of 0.5 to classify these predictions and saved the result as yhat_classify.

```
roc_yhat <-roc(titanic_test$survived,yhat_classify)
```

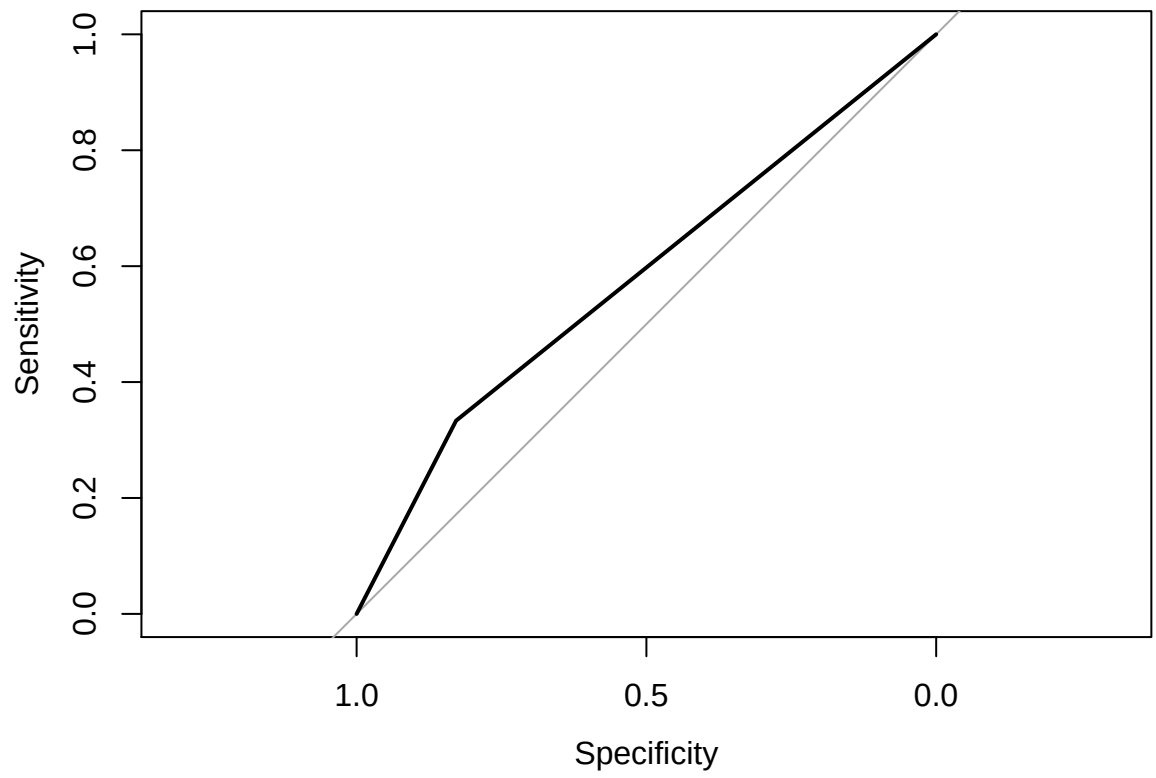
```
## Setting levels: control = 0, case = 1
```

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

```
#We utilised the roc function, actual result, predicted result to plot sensitivity and specificity
auc_yhat <- auc(titanic_test$survived,yhat_classify)
#We found area under the ROC curve using the auc function
auc_yhat
```

```
## [1] 0.5808679
```

```
plot(roc_output, print.cutoffs.at = seq(0,1,0.1))
```




```
#summary(roc_output)
```

Let us plot for yhat2. We had added title besides pclass and fit logistic regression model. We had saved the result as yhat2. We had then used a threshold of 0.5 to classify yhat2 and saved the result as yhat2_classify.

```
roc_yhat2 <-roc(title_test$survived,yhat2_classify)
```

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

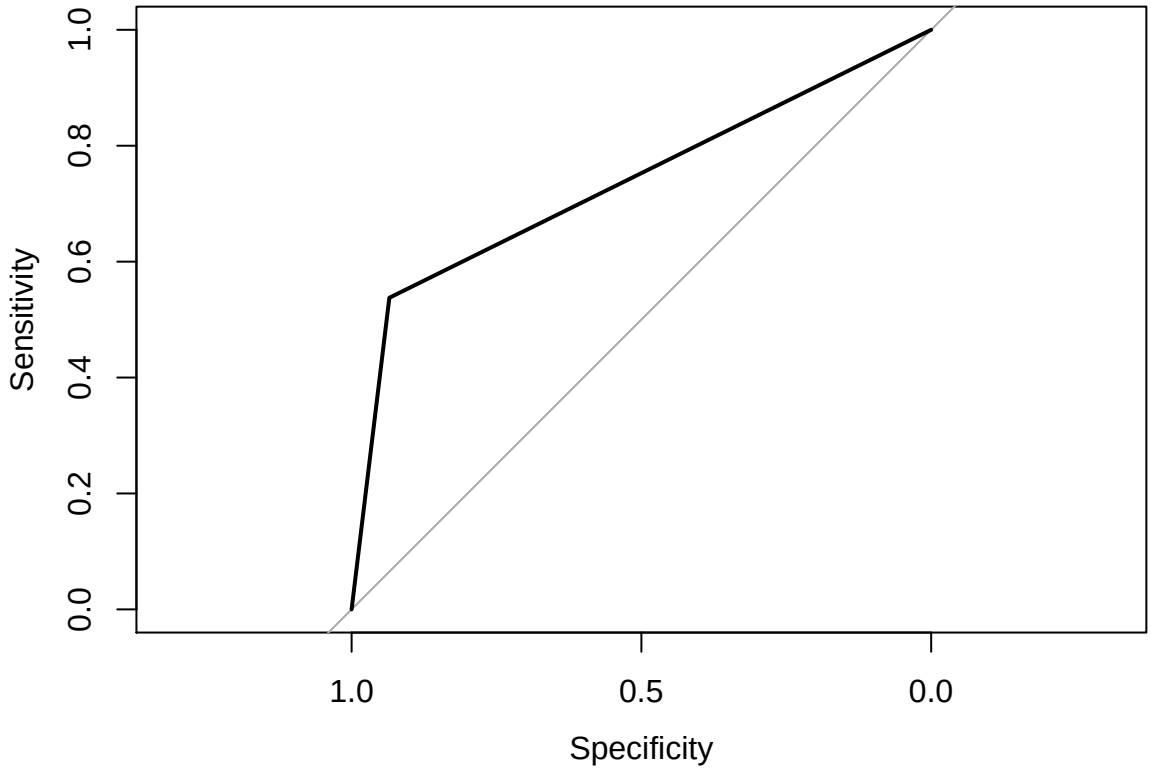
```
## Setting direction: controls < cases
```

```
#We utilised the roc function, actual result, predicted result to plot sensitivity and specificity

auc_yhat2 <- auc(title_test$survived,yhat2_classify)
#We found area under the ROC curve using the auc function
auc_yhat2
```

```
## [1] 0.7362728
```

```
plot(roc_yhat2, print.cutoffs.at = seq(0,1,0.1))
```




```
#summary(roc_output)
```

Let us plot for yhat3. We had got this from random forest model using socioeconpmic class and title as predictors.

```
yhat3 = as.numeric(yhat3)
#I was getting error in ROC function because yhat3 was factor,
#so I changed it to numeric

roc_yhat3 <-roc(title_test$survived,yhat3)
```

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

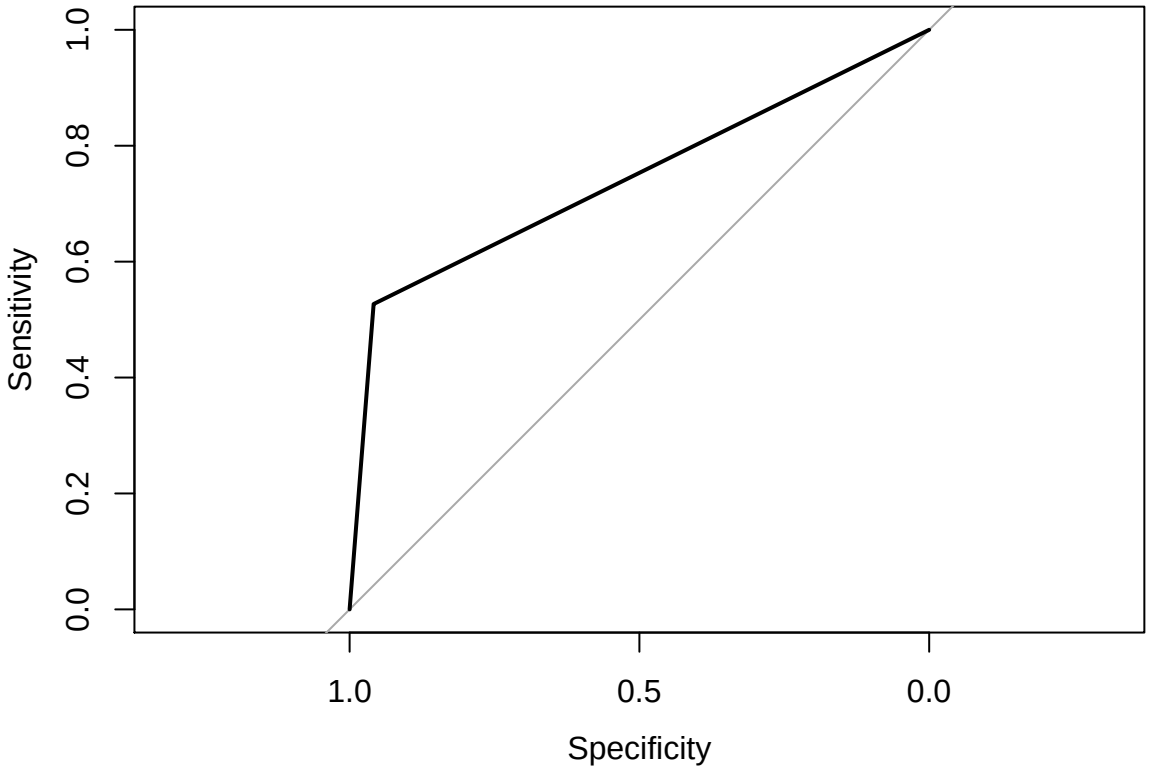
```
## Setting direction: controls < cases
```

```
#We utilised the roc function, actual result, predicted result to plot
#sensitivity and specificity

auc_yhat3 <- auc(title_test$survived,yhat3)
#We found area under the ROC curve using the auc function
auc_yhat3
```

```
## [1] 0.7427308
```

```
plot(roc_yhat3, print.cutoffs.at = seq(0,1,0.1))
```




```
#summary(roc_output)
```

Let us plot for yhat4. We had got this from random forest model using socioeconpmic class, title, sex, sibsp as predictors.

```
yhat4 = as.numeric(yhat4)
#I was getting error in ROC function because yhat4 was factor, so I changed it to numeric

roc_yhat4 <-roc(title_test$survived,yhat4)
```

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

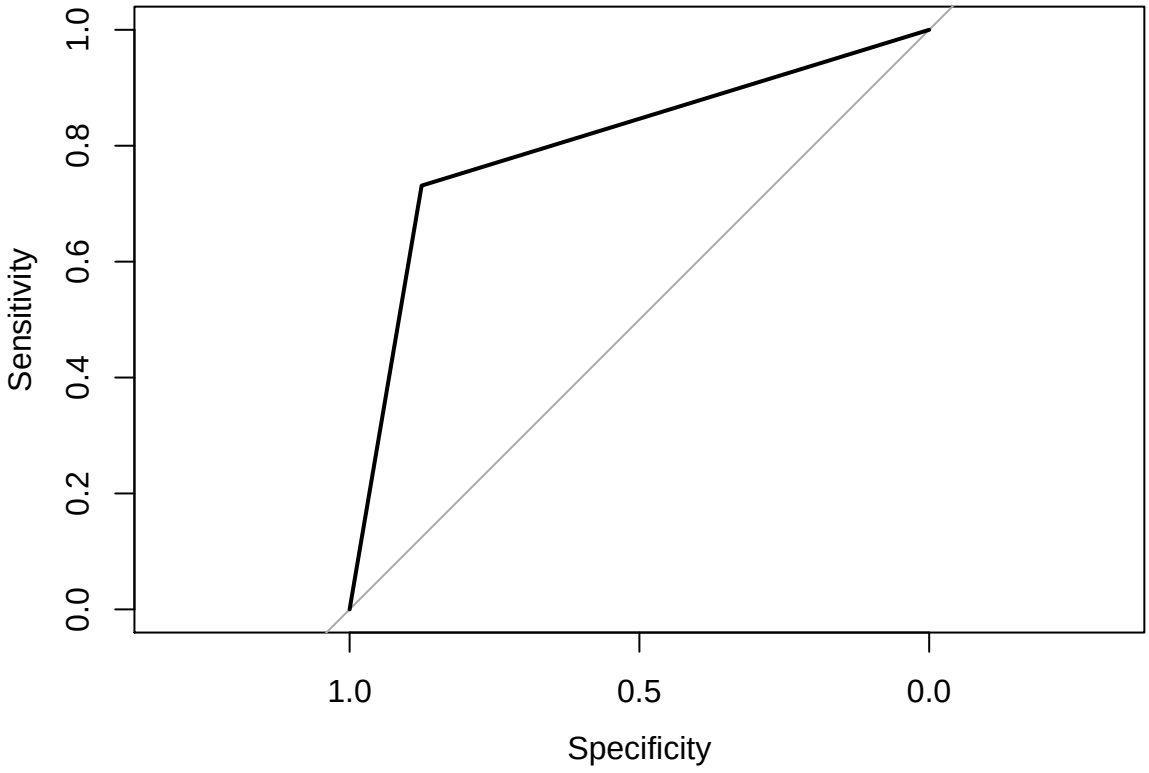
```
## Setting direction: controls < cases
```

```
#We utilised the roc function, actual result, predicted result to plot sensitivity and specificity

auc_yhat4 <- auc(title_test$survived,yhat4)
auc_yhat4
```

```
## [1] 0.8034612
```

```
plot(roc_yhat4, print.cutoffs.at = seq(0,1,0.1))
```




```
#summary(roc_output)
```

In an ROC curve, the sensitivity increases when specificity decreases. It helps us in selecting different threshold levels. AUC is the area under the ROC curve. We can see that the area under ROC curve(auc_yhat4) for yhat4 is the highest. So yhat4 is the best for predicting the results.