IMT 573: Problem Set 8

Prediction - Solutions

YOUR NAME HERE

Due: August 15, 2021

Collaborators:

Instructions: Before beginning this assignment, please ensure you have access to R and RStudio; this can be on your own personal computer or on the IMT 573 R Studio Cloud.

1. Download the `08_ps_prediction.rmd` file from Canvas or save a copy to your local directory on RStudio Cloud. Supply your solutions to the assignment by editing `08_ps_prediction.rmd`.
2. Replace the "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. Be sure that each visualization adds value to your written explanation; avoid redundancy – you do no need four different visualizations of the same pattern.
4. Collaboration on problem sets is fun and useful, and we encourage it, but each student must turn in an individual write-up in their own words as well as code/work that is their own. Regardless of whether you work with others, what you turn in must be your own work; this includes code and interpretation of results. The names of all collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
5. All materials and resources that you use (with the exception of lecture slides) must be appropriately referenced within your assignment.
6. Remember partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. Students are *strongly* encouraged to attempt each question and to document their reasoning process even if they cannot find the correct answer. If you would like to include R code to show this process, but it does not run without errors you can do so with the `eval=FALSE` option as follows:

```
a + b # these object don't exist
# if you run this on its own it with give an error
```

7. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly when you click `Knit`, download and rename the knitted PDF file to `ps7_YourLastName_YourFirstName.pdf`, and submit the PDF file on Canvas.

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

```r
# Load standard libraries
library(tidyverse)
library(gridExtra)
library(MASS)
library(pROC)
library(arm)
library(randomForest)
library(Metrics)
```

Data: In this problem set we will use the `flights` dataset used previously in class. The flights dataset (via the the *nycflights13* library) contains information on flight delays and weather.

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.

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

Solution: When building a statistical model, we want to be able to explore and assess the model performance. In most cases, especially when the primary task is prediction, we want to measure model performance on new (unseen) data. Using a portion of the data, a training dataset, to build a model (determine its parameters) allows us to hold out some data for validation. You can and should explore performance on the training data as well, however, we do not want to overfit the data the model has seen and be poor at predicting new cases. A training/test set split thus allows us to measure performance on unseen data.

Problem 2: Predictions with a continuous output variable

(a) 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. **Solution:** Code to load in the data and perform the join:

```r
# Load data
library(nycflights13)
flightData <- flights
weatherData <- weather
flightDataAll <- merge(flightData, weatherData, by = c('origin', 'year', 'month',
                                                       'day', 'hour', 'time_hour'))
```

(b) 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. **Solution:** Code to perform the required manipulations:

```r
colsToKeep <- c('dep_delay', 'origin', 'dep_time', 'temp', 'wind_speed', 'precip', 'visib')
flightData <- flightDataAll[,colsToKeep]
```

```
flightData <- na.omit(flightData)
#str(data) # explore data structure
```

(c) 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. **Solution:** The following code constructs the training and test sets.

```
# Split into a training set and a test set
set.seed(12345)
num_train <- round(.8*nrow(flightData))
trainInds <- sample(1:nrow(flightData), num_train)
train <- flightData[trainInds,]
test <- flightData[-trainInds,]
```

(d) 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? **Solution:** Code to build the model and make predictions:

```
model <- lm(dep_delay ~., train)
summary(model)
```

```
##
## Call:
## lm(formula = dep_delay ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -109.40  -18.72   -8.13    2.80 1306.04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.9525820  0.4879098  -8.101 5.47e-16 ***
## originJFK   -4.7705896  0.1832798 -26.029  < 2e-16 ***
## originLGA   -4.3649850  0.1849759 -23.598  < 2e-16 ***
## dep_time     0.0211211  0.0001563 135.092  < 2e-16 ***
## temp         0.1152762  0.0042792  26.939  < 2e-16 ***
## wind_speed   0.2792450  0.0140674  19.851  < 2e-16 ***
## precip      67.1543246  2.6364260  25.472  < 2e-16 ***
## visib       -2.0404005  0.0404595 -50.431  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.36 on 261510 degrees of freedom
## Multiple R-squared:  0.08767,    Adjusted R-squared:  0.08764
## F-statistic:  3590 on 7 and 261510 DF,  p-value: < 2.2e-16
```

```
preds <- predict(model, train)
rmse(train$dep_delay, preds)
```

```
## [1] 38.3598
```

```
predsTest <- predict(model, test)
rmse(test$dep_delay, predsTest)
```

```
## [1] 38.56386
```

(e) 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? **Solution:** Here, the RMSE can be improved upon by adding a few choice variables. Note that randomization and variable selection can greatly impact these results. We would expect our training error to decrease as we have a more complex model (i.e. add more variables). However, the performance on our test set may not ultimately increase by much.

```
colsToKeep <- c('dep_delay', 'origin', 'dep_time', 'sched_dep_time',
                'temp', 'humid', 'dewp', 'wind_speed', 'precip', 'visib')
flightData <- flightDataAll[,colsToKeep]

train <- flightData[trainInds,]
test <- flightData[-trainInds,]

train <- na.omit(train)
test <- na.omit(test)

modelUpdated <- lm(dep_delay ~., train)
summary(modelUpdated)
```

```
##
## Call:
## lm(formula = dep_delay ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -82.82  -16.76   -8.07    1.97 1315.12
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -5.161e+01  2.166e+00 -23.828  < 2e-16 ***
## originJFK      -4.969e+00  1.817e-01 -27.339  < 2e-16 ***
## originLGA      -3.085e+00  1.864e-01 -16.552  < 2e-16 ***
## dep_time        6.227e-02  5.107e-04 121.933  < 2e-16 ***
## sched_dep_time -4.330e-02  5.348e-04 -80.965  < 2e-16 ***
## temp            4.443e-01  4.300e-02  10.332  < 2e-16 ***
## humid           4.813e-01  2.286e-02  21.058  < 2e-16 ***
## dewp           -3.812e-01  4.624e-02  -8.244  < 2e-16 ***
## wind_speed      4.395e-01  1.420e-02  30.952  < 2e-16 ***
## precip          4.636e+01  2.698e+00  17.183  < 2e-16 ***
## visib          -2.232e-01  5.163e-02  -4.323 1.54e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.63 on 255076 degrees of freedom
## Multiple R-squared:  0.1262, Adjusted R-squared:  0.1261
## F-statistic:  3683 on 10 and 255076 DF,  p-value: < 2.2e-16
```

```
preds <- predict(model, train)
rmse(train$dep_delay, preds)
```

```
## [1] 38.431
```

```
predsTest <- predict(model, test)
rmse(test$dep_delay, predsTest)
```

```
## [1] 38.29288
```

EXTRA CREDIT: Predictions with a categorical output (classification)

In this problem 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.

(a) 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. **Solution:** The following code constructs the training and test sets.

```
titanic_data <- read.csv('../../labs/titanic.csv')

# Split into a training set and a test set
set.seed(12345)
num_train <- round(.8*nrow(titanic_data))
trainInds <- sample(1:nrow(titanic_data), num_train)
train <- titanic_data[trainInds,]
test <- titanic_data[-trainInds,]
```

(b) Fit the model described above (i.e. one that only takes into account socioeconomic status) using the `glm` function in R. **Solution:** We choose to use passenger class to measure socioeconomic status. Passenger class (measured as a categorical variable) denotes the level of "status" of the ticket holder; higher socioeconomic status groups are more likely to have the resources necessary to purchase a 1st class ticket.

```
# Fit a logistic regression model for survival
survival.glm <- glm(survived ~ factor(pclass),
            data=train, family=binomial())
summary(survival.glm) # Model summary
```

```
##
## Call:
## glm(formula = survived ~ factor(pclass), family = binomial(),
##     data = train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3979  -0.7390  -0.7390   0.9719   1.6921
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.5047     0.1273   3.966 7.31e-05 ***
## factor(pclass)2  -0.8084     0.1868  -4.328 1.50e-05 ***
## factor(pclass)3  -1.6633     0.1610 -10.329  < 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: 1384.7  on 1046  degrees of freedom
## Residual deviance: 1268.3  on 1044  degrees of freedom
## AIC: 1274.3
##
## Number of Fisher Scoring iterations: 4
```

```
# Fit a logistic regression model for survival
# Using a bayesian fitting approach, just to illustrate the function
survival.bayesglm <- bayesglm(survived ~ factor(pclass),
               data=train, family=binomial())
summary(survival.bayesglm) # Model summary
```

```
##
## Call:
## bayesglm(formula = survived ~ factor(pclass), family = binomial(),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3936  -0.7401  -0.7401   0.9756   1.6904
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.4951     0.1266   3.911 9.19e-05 ***
## factor(pclass)2  -0.7945     0.1856  -4.281 1.86e-05 ***
## factor(pclass)3  -1.6500     0.1602 -10.302  < 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: 1384.7  on 1046  degrees of freedom
## Residual deviance: 1268.3  on 1044  degrees of freedom
## AIC: 1274.3
##
## Number of Fisher Scoring iterations: 5
```

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!

(c) What might you conclude based on this model about the probability of survival for lower class passengers? **Solution:** From the model, we see that the probability of survival decreases as passenger class increases, thus lower class passengers have lower probabilities of survival. Remember that first class is this highest passenger class, followed by second and finally third. This effect can be seen in the negative coefficient of `pclass` - 2 and `pclass` - 3. Moreover, the effect size is larger for `pclass` - 3.

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

(d) Predict the survival of passengers for each observation in your test set using the model fit in Problem 2. Save these predictions as `yhat`. **Solution:** The following code predicts probability of survival for the test data.

```
# Get predicted probabilities
yhat <- predict(survival.glm, newdata=test, type="response")
```

```
# Construct confusion matrix
table(test$survived, yhat>.5)
```

(e) 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.

```
##
##     FALSE TRUE
##   0   130   24
##   1    72   36
```

Solution: From the confusion matrix, we see that the model makes 20 false positives, cases where the true value is negative but the model predicts positive. These false positives are cases where we predicted that passengers survive but they did not.

(f) Using the `roc` function, plot the ROC curve for this model. Discuss what you find. **Solution:** The below code allows you to plot the ROC curve for this classification model, as seen in Figure 1. Recall, the area under this curve (AUC of the ROC) provides an overall measure of fit of the model. Therefore, we would like to see the curve as close to the upper left hand corner of the plot. Here we see that the area under the ROC curve is 0.683.

```
# Plot the ROC curve
roc(test$survived, yhat)
```

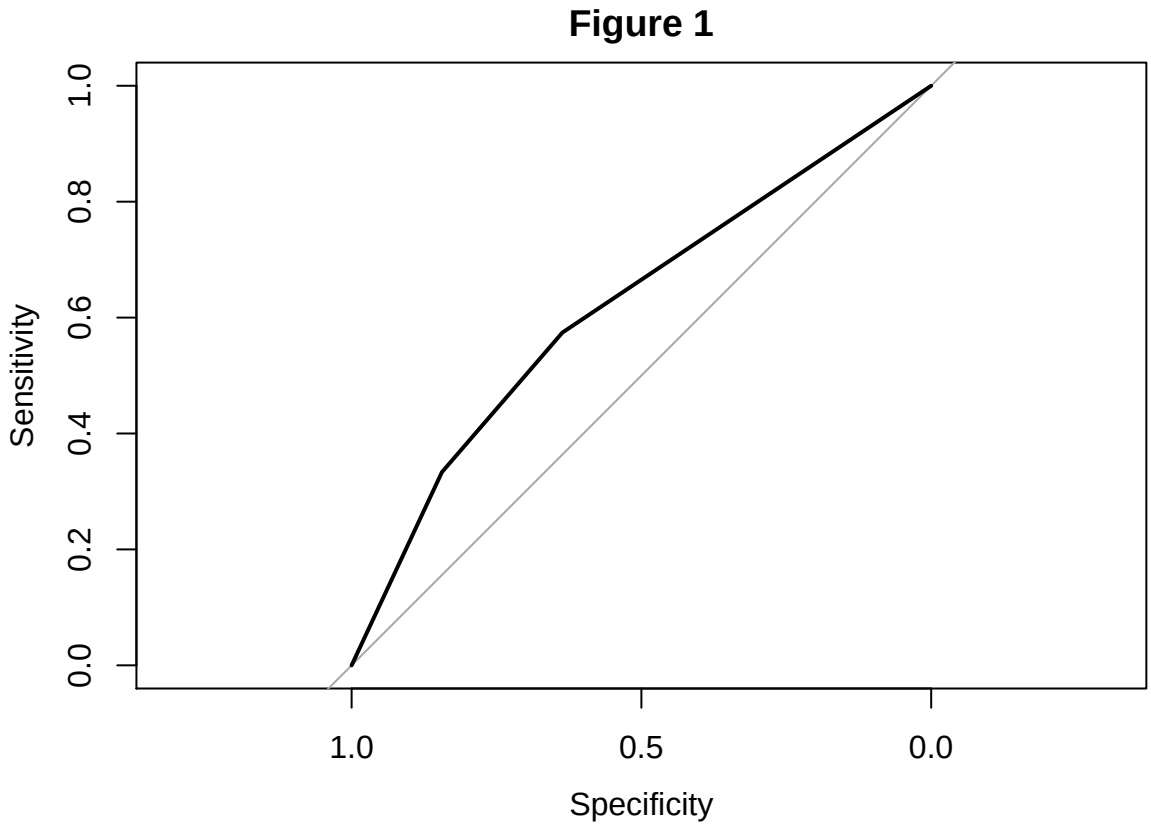
```
## Setting levels: control = 0, case = 1
```

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

```
##
## Call:
## roc.default(response = test$survived, predictor = yhat)
##
## Data: yhat in 154 controls (test$survived 0) < 108 cases (test$survived 1).
## Area under the curve: 0.6211
```

```
plot(roc(test$survived, yhat), main="Figure 1")
```

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



(g) 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?

Solution: Title, as described in this problem, captures both gender and age together. Titles distinguish between males and females, as well as young and old. We might suspect that both age and gender may play a role in how survival resources - such as spots in lifeboats - are allocated to passengers. Therefore title maybe be helpful in predicting survival of the Titanic passengers.

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

Solution: We construct a new variable and add it to the 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")
}
```

```r
# Add this feature to the dataset
train$title <- factor(sapply(train$name, getTitles))
test$title <- factor(sapply(test$name, getTitles))
```

```r
# Fit a new logistic regression model now including title
survival.glm.new <- glm(survived ~ factor(pclass) + factor(title),
                data=train, family=binomial())
summary(survival.glm.new) # Model summary
```

(h) 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?

```
##
## Call:
## glm(formula = survived ~ factor(pclass) + factor(title), family = binomial(),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2128  -0.6448  -0.3982   0.6772   2.2691
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            1.6920     0.3619   4.675 2.93e-06 ***
## factor(pclass)2       -1.0018     0.2273  -4.409 1.04e-05 ***
## factor(pclass)3       -2.0318     0.2065  -9.838  < 2e-16 ***
## factor(title)Miss      0.4893     0.3480   1.406  0.15975
## factor(title)Mr.      -2.1552     0.3413  -6.316 2.69e-10 ***
## factor(title)Mrs.      0.6658     0.3711   1.794  0.07276 .
## factor(title)Nothing  -1.6979     0.5430  -3.127  0.00177 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1384.67  on 1046  degrees of freedom
## Residual deviance:  964.94  on 1040  degrees of freedom
## AIC: 978.94
##
## Number of Fisher Scoring iterations: 5
```

Solution: One way to compare models is to look at the AIC values reported for each. Recall, AIC is a measure of the relative quality of a model. AIC deals with the trade-off between the goodness of fit of the model and the complexity of the model. We see that this new model has a lower AIC value than the previous, therefore, we can conclude this new predictor variable did improve the model.

```r
# Get predicted probability of survival from the new model
yhat2 <- predict(survival.glm.new, newdata=test, type="response")
```

```r
# Look at confusion matrix
table(test$survived, yhat2>.5)
```

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

```
##
##     FALSE TRUE
##   0   132   22
##   1    29   79
```

Solution: There are number of ways to consider goodness of fit for logistic regression models. These methods include classification tables (i.e. confusion matrices), ROC curves, and others. We will focus on these for this problem set. A confusion table shows that correct predictions are higher than errors, however, the rate of false negatives is high. We could also compare the ROC curves for this model with our previous models, we see that this model has a better fit.