131HW3

```
titanic <- read.csv('titanic.csv')
set.seed(1008)
titanic$survived<-as.factor(titanic$survived)
titanic$survived <- ordered(titanic$survived, levels = c("Yes","No"))
titanic$pclass<- as.factor(titanic$pclass)
titanic%>%count(survived)
```

Question 1

Split the data, stratifying on the outcome variable, survived. You should choose the proportions to split the data into. Verify that the training and testing data sets have the appropriate number of observations. Take a look at the training data and note any potential issues, such as missing data.

Why is it a good idea to use stratified sampling for this data?

```
titanic_split <- initial_split(titanic, prop = 0.80,
                               strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
nrow(titanic)
```

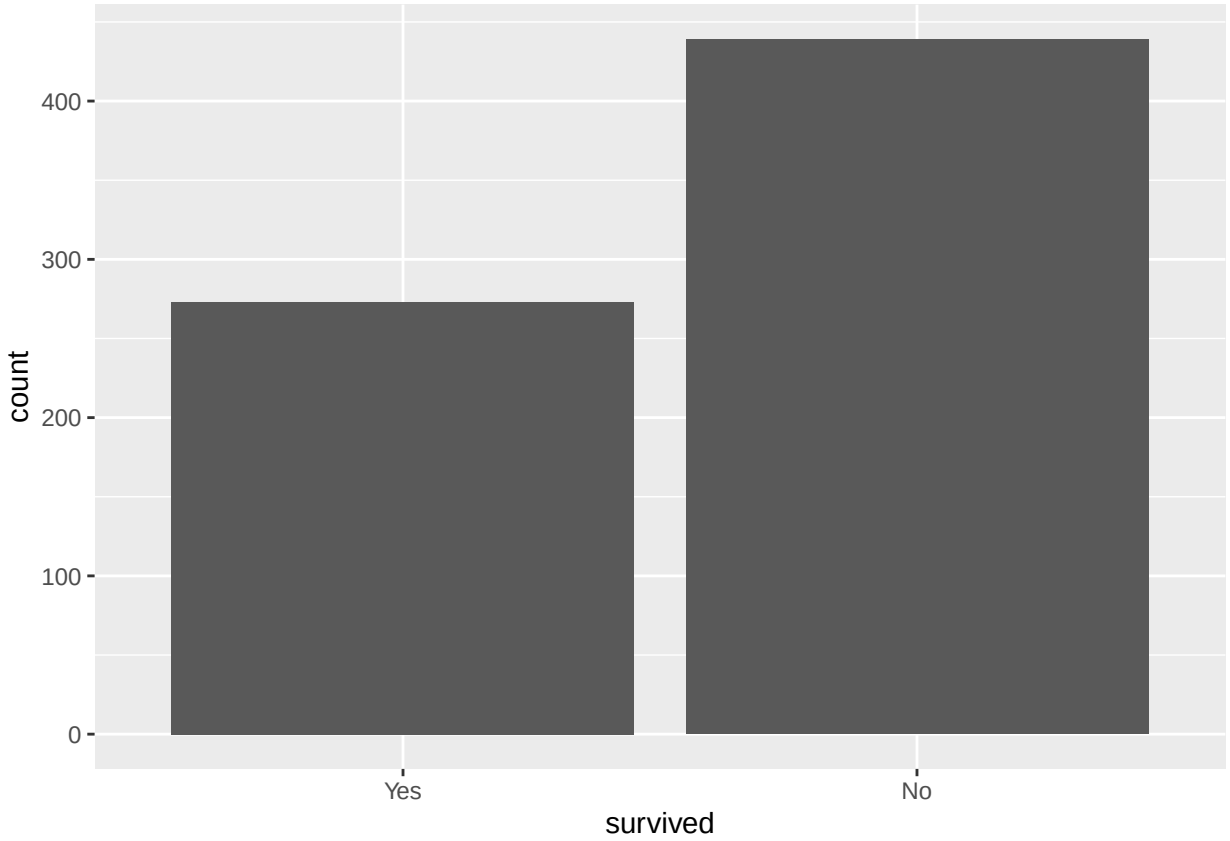
```
## [1] 891
```

There seems to be a lot of missing data in columns such as cabin and age, which may skew our results. Not only does it decrease our pool of usable data, the missing data may also lead to misleading results. For example, perhaps the missing data is more common in a certain demographic than others. Then, the demographic with the missing data is more likely to get underrepresented. It is good to use stratified sampling for this data because it ensures that there is an equal representation of those who did and didn't survive in the testing and training. This is especially important since the dataset we're using isn't extremely large at 891 rows, and a result, there is a higher chance for the amount survived to be underrepresented without stratification.

Question 2

Using the training data set, explore/describe the distribution of the outcome variable survived.

```
titanic_train %>%
  ggplot(aes(x = survived)) +
  geom_bar()
```



From the graph, there seems to be around 275 people that survived and 440 that did not. This slight disparity in those that did and didn't survive is similar to our dataset which is important.

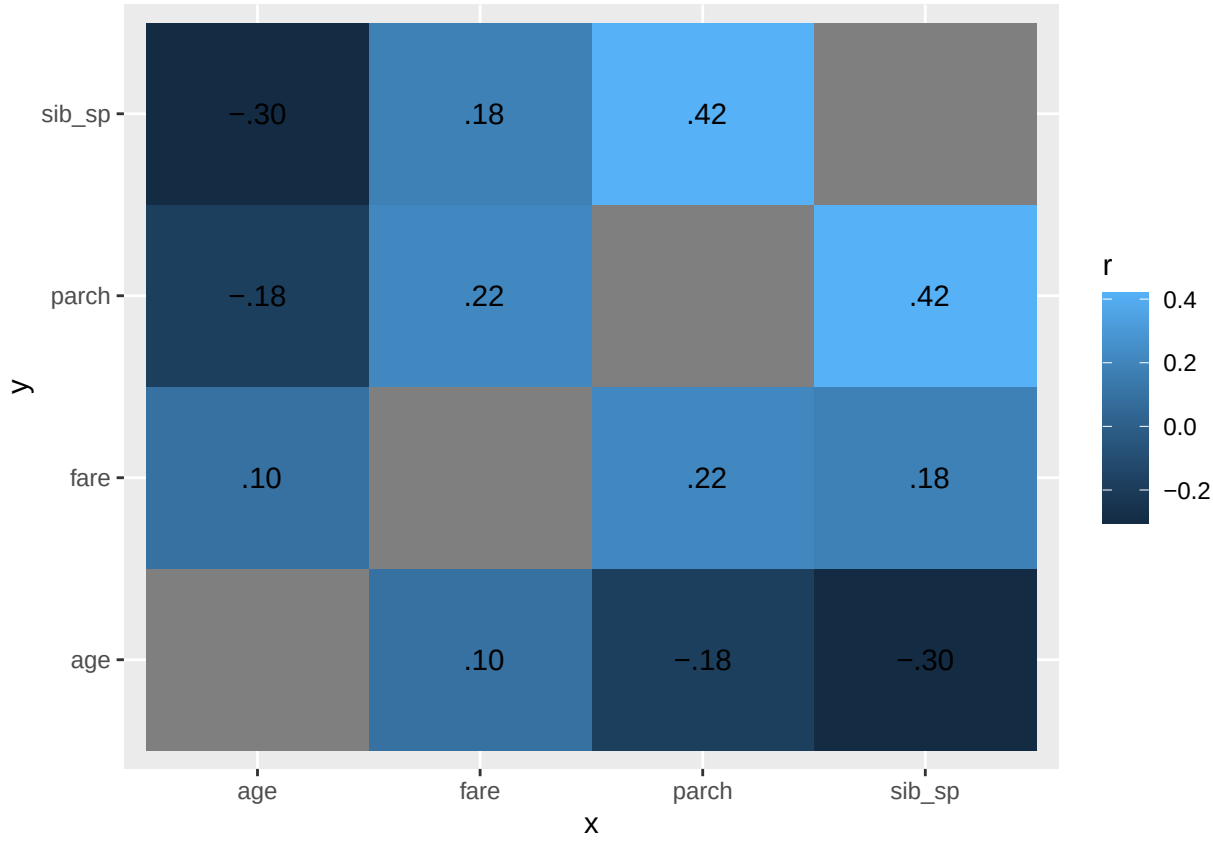
Question 3

Using the training data set, create a correlation matrix of all continuous variables. Create a visualization of the matrix, and describe any patterns you see. Are any predictors correlated with each other? Which ones, and in which direction?

```
titanic_train2 <- titanic_train[,sapply(titanic_train,is.numeric)]
titanic_train3 <- subset(titanic_train2, select = -c(passenger_id)) %>%   correlate()
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
titanic_train3 %>%
  stretch() %>%
  ggplot(aes(x, y, fill = r)) +
  geom_tile() +
  geom_text(aes(label = as.character(fashion(r))))
```



It seems that many of the variables have a very slight positive correlation with fare and age at .10, sib_sp and fare at .18, and parch and fare at .22. The largest, and potentially most important, positive correlation is between sib_sp and parch at .42. The only negative correlations are between parch and age at -.18 and sib_sp and age at -.30.

Question 4

Using the training data, create a recipe predicting the outcome variable survived. Include the following predictors: ticket class, sex, age, number of siblings or spouses aboard, number of parents or children aboard, and passenger fare.

Recall that there were missing values for age. To deal with this, add an imputation step using step_impute_linear(). Next, use step_dummy() to dummy encode categorical predictors. Finally, include interactions between:

```
Sex and passenger fare, and
Age and passenger fare.
```

You'll need to investigate the tidymodels documentation to find the appropriate step functions to use.

```
titanic_recipe <-
  recipe(survived ~ pclass+sex+age+sib_sp+parch+fare,data = titanic_train) %>%
  step_impute_linear(age)%>%
  step_dummy(all_nominal_predictors())  %>%
  step_normalize(all_predictors()) %>%
  step_interact(terms = ~ sex_male:fare + age:fare)
```

Question 5

Specify a logistic regression model for classification using the "glm" engine. Then create a workflow. Add your model and the appropriate recipe. Finally, use fit() to apply your workflow to the training data.

```
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)

log_fit <- fit(log_wkflow, titanic_train)
log_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_normalize()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##     (Intercept)              age           sib_sp            parch
##          0.6746           0.6153           0.5407           0.1222
##            fare        pclass_X2        pclass_X3         sex_male
##         -0.2647           0.4279           1.1602           1.4000
## sex_male_x_fare       fare_x_age
##          0.2667          -0.2233
##
## Degrees of Freedom: 711 Total (i.e. Null);  702 Residual
## Null Deviance:       948
## Residual Deviance: 603.7     AIC: 623.7
```

Question 6

Repeat Question 5, but this time specify a linear discriminant analysis model for classification using the "MASS" engine.

```
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

lda_wkflow <- workflow() %>%
```

```
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)

lda_fit <- fit(lda_wkflow, titanic_train)
lda_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_linear()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_normalize()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## Call:
## lda(..y ~ ., data = data)
##
## Prior probabilities of groups:
##      Yes       No
## 0.383427 0.616573
##
## Group means:
##             age      sib_sp       parch       fare   pclass_X2  pclass_X3
## Yes -0.05047762 -0.06446088  0.07453457  0.3418475  0.12415890 -0.4231571
## No   0.03139041  0.04008615 -0.04635066 -0.2125840 -0.07721043  0.2631478
##       sex_male sex_male_x_fare fare_x_age
## Yes -0.7052732     -0.27146835 0.19853342
## No   0.4385867     -0.09368961 0.02159929
##
## Coefficients of linear discriminants:
##                          LD1
## age              0.352528315
## sib_sp           0.255066211
## parch            0.078127432
## fare            -0.119963534
## pclass_X2        0.259854979
## pclass_X3        0.718019096
## sex_male         1.021778176
## sex_male_x_fare  0.003650702
## fare_x_age      -0.094109332
```

Question 7

Repeat Question 5, but this time specify a quadratic discriminant analysis model for classification using the "MASS" engine.

```r
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)

qda_fit <- fit(qda_wkflow, titanic_train)
qda_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_quad()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_normalize()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## Call:
## qda(..y ~ ., data = data)
##
## Prior probabilities of groups:
##      Yes       No
## 0.383427 0.616573
##
## Group means:
##             age      sib_sp       parch       fare   pclass_X2  pclass_X3
## Yes -0.05047762 -0.06446088  0.07453457  0.3418475  0.12415890 -0.4231571
## No   0.03139041  0.04008615 -0.04635066 -0.2125840 -0.07721043  0.2631478
##       sex_male sex_male_x_fare fare_x_age
## Yes -0.7052732     -0.27146835 0.19853342
## No   0.4385867     -0.09368961 0.02159929
```

Question 8

Repeat Question 5, but this time specify a naive Bayes model for classification using the "klaR" engine. Set the usekernel argument to FALSE.

```r
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)

nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)
```

```r
nb_fit <- fit(nb_wkflow, titanic_train)
nb_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: naive_Bayes()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_normalize()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## $apriori
## grouping
##      Yes       No
## 0.383427 0.616573
##
## $tables
## $tables$age
##            [,1]      [,2]
## Yes -0.05047762 1.0060718
## No   0.03139041 0.9960639
##
## $tables$sib_sp
##            [,1]      [,2]
## Yes -0.06446088 0.6265633
## No   0.04008615 1.1727285
##
## $tables$parch
##            [,1]      [,2]
## Yes  0.07453457 0.9110251
## No  -0.04635066 1.0498851
##
## $tables$fare
##           [,1]      [,2]
## Yes  0.3418475 1.3448232
## No  -0.2125840 0.6180922
##
## $tables$pclass_X2
##            [,1]      [,2]
## Yes  0.12415890 1.0835746
## No  -0.07721043 0.9373163
##
## $tables$pclass_X3
##           [,1]      [,2]
## Yes -0.4231571 0.9571308
## No   0.2631478 0.9345447
##
## $tables$sex_male
```

```
##           [,1]      [,2]
## Yes -0.7052732 0.9506117
## No   0.4385867 0.7478515
##
## $tables$sex_male_x_fare
##            [,1]      [,2]
## Yes -0.27146835 1.5080174
## No  -0.09368961 0.5260978
##
## $tables$fare_x_age
##           [,1]      [,2]
## Yes 0.19853342 0.9859321
## No  0.02159929 0.8088514
##
## ...
## and 1446 more lines.
```

Question 9

Now you've fit four different models to your training data.

Use predict() and bind_cols() to generate predictions using each of these 4 models and your training data. Then use the accuracy metric to assess the performance of each of the four models.

Which model achieved the highest accuracy on the training data?

```
bind_predictions <- bind_cols(predict(log_fit,new_data = titanic_train, type = 'prob'),
predict(lda_fit,new_data = titanic_train, type = 'prob'),
predict(qda_fit,new_data = titanic_train, type = 'prob'),
predict(nb_fit,new_data = titanic_train, type = 'prob'))
```

```
## New names:
## * .pred_Yes -> .pred_Yes...1
## * .pred_No -> .pred_No...2
## * .pred_Yes -> .pred_Yes...3
## * .pred_No -> .pred_No...4
## * .pred_Yes -> .pred_Yes...5
## * ...
```

```
bind_predictions
```

```
## # A tibble: 712 x 8
##    .pred_Yes...1 .pred_No...2 .pred_Yes...3 .pred_No...4 .pred_Yes...5
##            <dbl>        <dbl>         <dbl>        <dbl>         <dbl>
##  1        0.0865        0.913        0.0510        0.949   0.00525
##  2        0.0703        0.930        0.0392        0.961   0.00358
##  3        0.302         0.698        0.231         0.769   0.0553
##  4        0.0769        0.923        0.0535        0.947   0.0000708
##  5        0.144         0.856        0.0808        0.919   0.00809
##  6        0.0196        0.980        0.0113        0.989   0.000252
##  7        0.763         0.237        0.826         0.174   0.580
##  8        0.0464        0.954        0.0362        0.964   0.000000421
##  9        0.481         0.519        0.587         0.413   0.273
```

```
## 10        0.500         0.500        0.629         0.371   0.000564
## # ... with 702 more rows, and 3 more variables: .pred_No...6 <dbl>,
## #   .pred_Yes...7 <dbl>, .pred_No...8 <dbl>
```

```
log_acc <- augment(log_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc <- augment(lda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
qda_acc <- augment(qda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
nb_acc <- augment(nb_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
accuracies <- c(log_acc$.estimate, lda_acc$.estimate,
                nb_acc$.estimate, qda_acc$.estimate)
models <- c("Logistic Regression", "LDA", "Naive Bayes", "QDA")
results <- tibble(accuracies = accuracies, models = models)
```

```
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 4 x 2
##   accuracies models
##        <dbl> <chr>
## 1      0.815 Logistic Regression
## 2      0.812 QDA
## 3      0.803 LDA
## 4      0.802 Naive Bayes
```

The best performing model is Logistic Regression at around 81.5%. However, this is only marginally better than the other methods.

Question 10

Fit the model with the highest training accuracy to the testing data. Report the accuracy of the model on the testing data.

Again using the testing data, create a confusion matrix and visualize it. Plot an ROC curve and calculate the area under it (AUC).

How did the model perform? Compare its training and testing accuracies. If the values differ, why do you think this is so?

```
predict(log_fit, new_data = titanic_test, type = "prob")
```

```
## # A tibble: 179 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.929    0.0707
##  2    0.0969   0.903
##  3    0.553    0.447
##  4    0.894    0.106
##  5    0.828    0.172
```

```
##  6    0.221    0.779
##  7    0.222    0.778
##  8    0.276    0.724
##  9    0.159    0.841
## 10    0.138    0.862
## # ... with 169 more rows
```

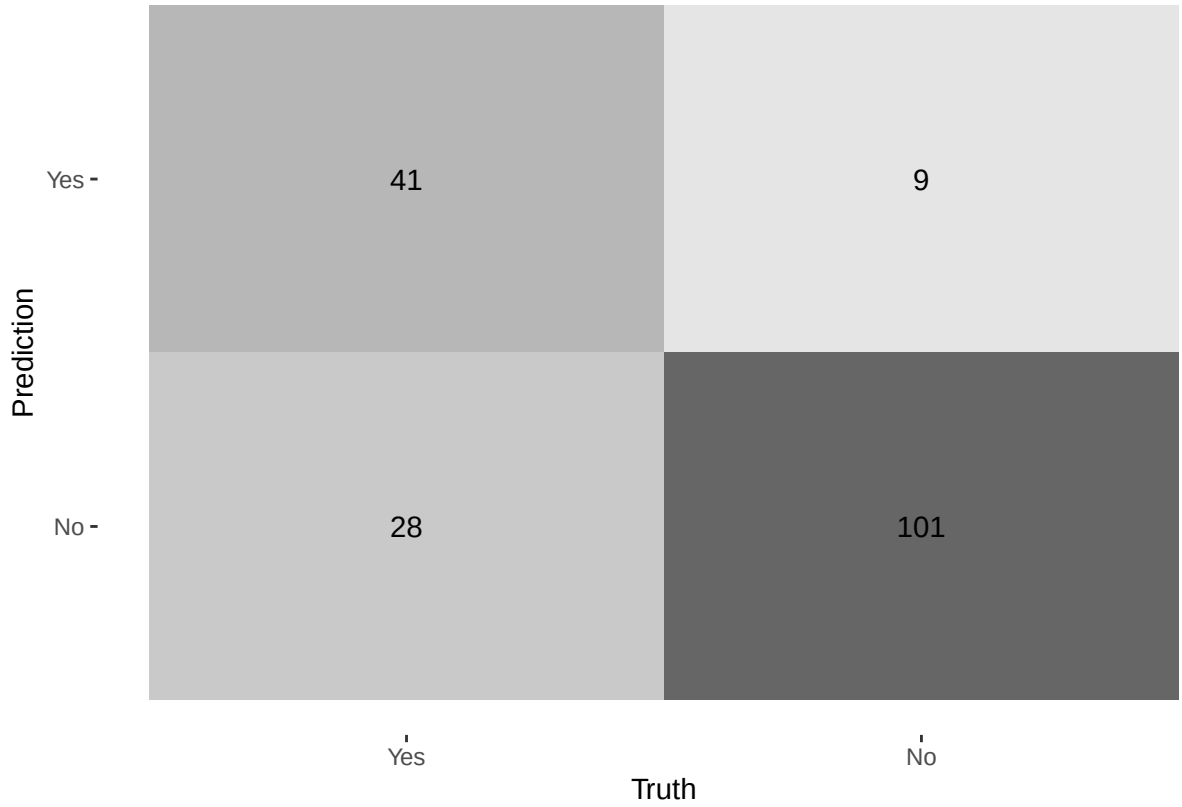
```
log_acc2 <- augment(log_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc2
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.793
```

```
augment(log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes  41   9
##        No   28 101
```

```
augment(log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
augment(log_fit, new_data = titanic_test) %>%
  roc_auc(survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.821
```

The area under the ROC curve is approximately .821. In terms of performance, the model with testing data had a 79.3% accuracy which is just slightly lower than the 81.5% with training data. This is to be expected since the model was created/optimized with the training data, and therefore, it is far more likely to perform better with it.