# Homework 3 Submission

Vardan Martirosyan

2022-10-28

First, we read in the data.

```
dataset <- read.csv("/Users/vardan/Desktop/pstat131/Homework/Homework3/data/titanic.csv")
```

Then, we load in the tidyverse and tidymodels libraries as desired. Additionally, we also load the 'ggplot2' and 'corrplot' libraries, which can help with some of the questions asked of us.

```
library(tidyverse)
library(tidymodels)
library(ggplot2)
library(corrplot)
library(MASS)
library(klaR)
library(discrim)
tidymodels_prefer()
```

We also are told that 'survived' and 'pclass' should be changed to factors, and that when changing 'survived' to a factor, to reorder the factor so that 'Yes' is the first factor. We do this as follows:

```
#Changing them to factors.
dataset$survived <- as.factor(dataset$survived)
dataset$pclass <- as.factor(dataset$pclass)

#Reordering the factor so that 'Yes' is the first factor.
dataset$survived <- relevel(dataset$survived, 'Yes')
```

Finally, we set the seed so that our code can be reproduced.

```
set.seed(69)
```

## Question 1

We are asked to split the data, stratifying on the outcome variable 'survived'. We are asked to choose the proportions to split the data into. We choose 80/20 again, as this has worked out consistently in the past for us in previous homeworks and lab assignments.

```
dataset_split <- initial_split(dataset, prop = 0.80, strata = survived)

dataset_train <- training(dataset_split)
dataset_test <- testing(dataset_split)
```

We then want to verify that the training and testing data sets have an appropriate number of observations. We check their sizes as follows:

```
#First, we check the size of the original dataset.
nrow(dataset)
```

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

```
#Then, we check the size of the training and test datasets.
nrow(dataset_train)
```

```
## [1] 712
```

```
nrow(dataset_test)
```

```
## [1] 179
```

Looking at the training and testing datasets, it looks like they each have a good number of observations for the purposes they serve. We are then asked to take a look at the training data and see if there are any potential issues, such as missing data.

```
head(dataset_train, 15)
```

```
##    passenger_id survived pclass
## 1             1       No      3
## 7             7       No      1
## 8             8       No      3
## 13           13       No      3
## 14           14       No      3
## 17           17       No      3
## 19           19       No      3
## 21           21       No      2
## 27           27       No      3
## 30           30       No      3
## 31           31       No      1
## 34           34       No      2
## 35           35       No      1
## 38           38       No      3
## 39           39       No      3
##                                                    name    sex age sib_sp
## 1                               Braund, Mr. Owen Harris   male  22      1
## 7                               McCarthy, Mr. Timothy J   male  54      0
## 8                         Palsson, Master. Gosta Leonard   male   2      3
## 13                        Saundercock, Mr. William Henry   male  20      0
## 14                           Andersson, Mr. Anders Johan   male  39      1
## 17                                  Rice, Master. Eugene   male   2      4
## 19 Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele) female  31      1
## 21                                  Fynney, Mr. Joseph J   male  35      0
## 27                               Emir, Mr. Farred Chehab   male  NA      0
## 30                                   Todoroff, Mr. Lalio   male  NA      0
## 31                              Uruchurtu, Don. Manuel E   male  40      0
```

```
## 34                                  Wheadon, Mr. Edward H   male  66      0
## 35                                Meyer, Mr. Edgar Joseph   male  28      1
## 38                               Cann, Mr. Ernest Charles   male  21      0
## 39                     Vander Planke, Miss. Augusta Maria female  18      2
##    parch     ticket    fare cabin embarked
## 1      0  A/5 21171  7.2500  <NA>        S
## 7      0      17463 51.8625   E46        S
## 8      1     349909 21.0750  <NA>        S
## 13     0  A/5. 2151  8.0500  <NA>        S
## 14     5     347082 31.2750  <NA>        S
## 17     1     382652 29.1250  <NA>        Q
## 19     0     345763 18.0000  <NA>        S
## 21     0     239865 26.0000  <NA>        S
## 27     0       2631  7.2250  <NA>        C
## 30     0     349216  7.8958  <NA>        S
## 31     0   PC 17601 27.7208  <NA>        C
## 34     0 C.A. 24579 10.5000  <NA>        S
## 35     0   PC 17604 82.1708  <NA>        C
## 38     0 A./5. 2152  8.0500  <NA>        S
## 39     0     345764 18.0000  <NA>        S
```

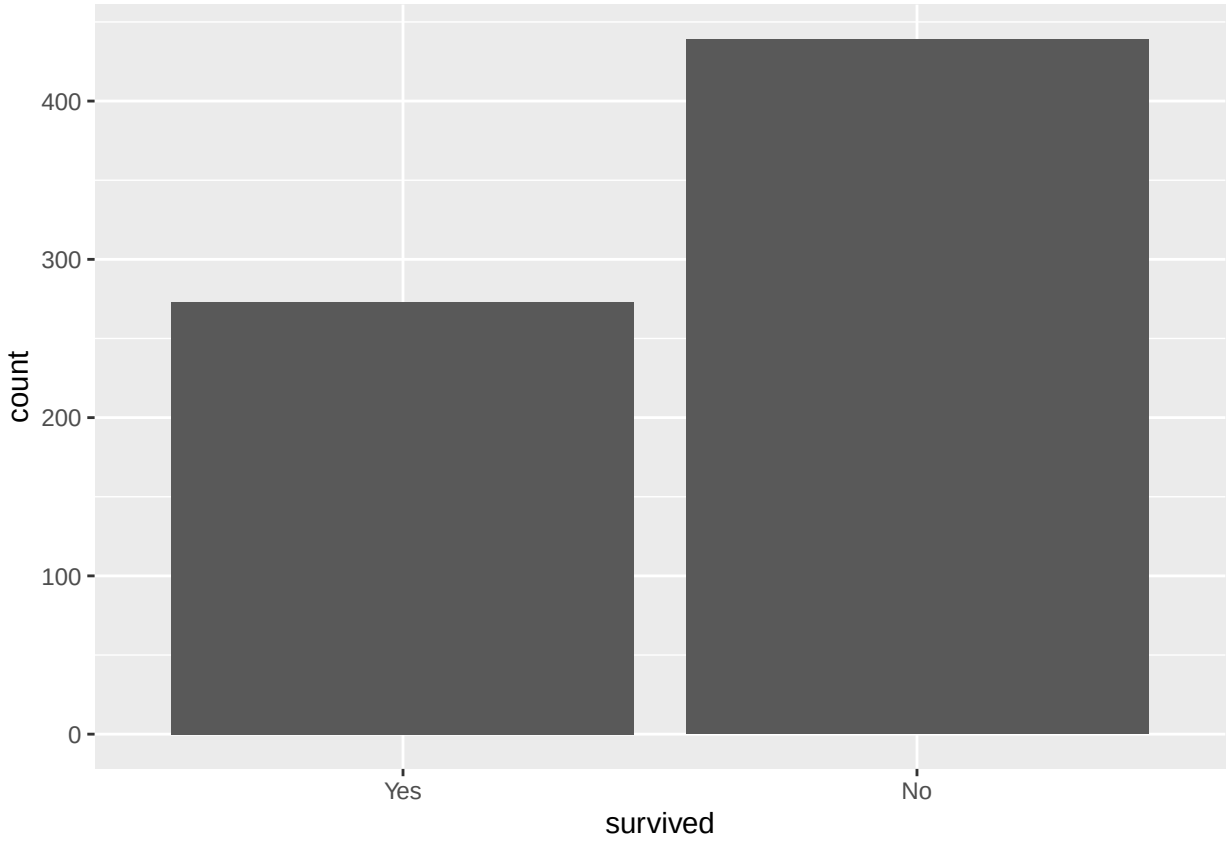
Looking at the data, it seems mostly clean, with the exception of some missing values (as expected). In particular, we see that the Cabin number variable seems to be missing for quite a bit of the observations. In addition, there are some inputs for the 'Age' variable that are also missing. There may be other values that are missing in other columns, but the overall message is clear: this dataset definitely has some missing data, which could impact the overall results of the analysis.

Finally, for this question, we are asked why it might be a good idea to use stratified sampling for this dataset. The reason we use stratified sampling is as follows. The majority of the passangers on the Titanic died, and thus, if we didn't use stratified sampling on the 'survived' variable, then perhaps a larger majority of our training dataset would have had more observations of passenger's who didn't survive. By using stratified sampling, we can guarantee that our training and testing datasets have roughly equal amounts of passengers who passed away and survived.

## Question 2

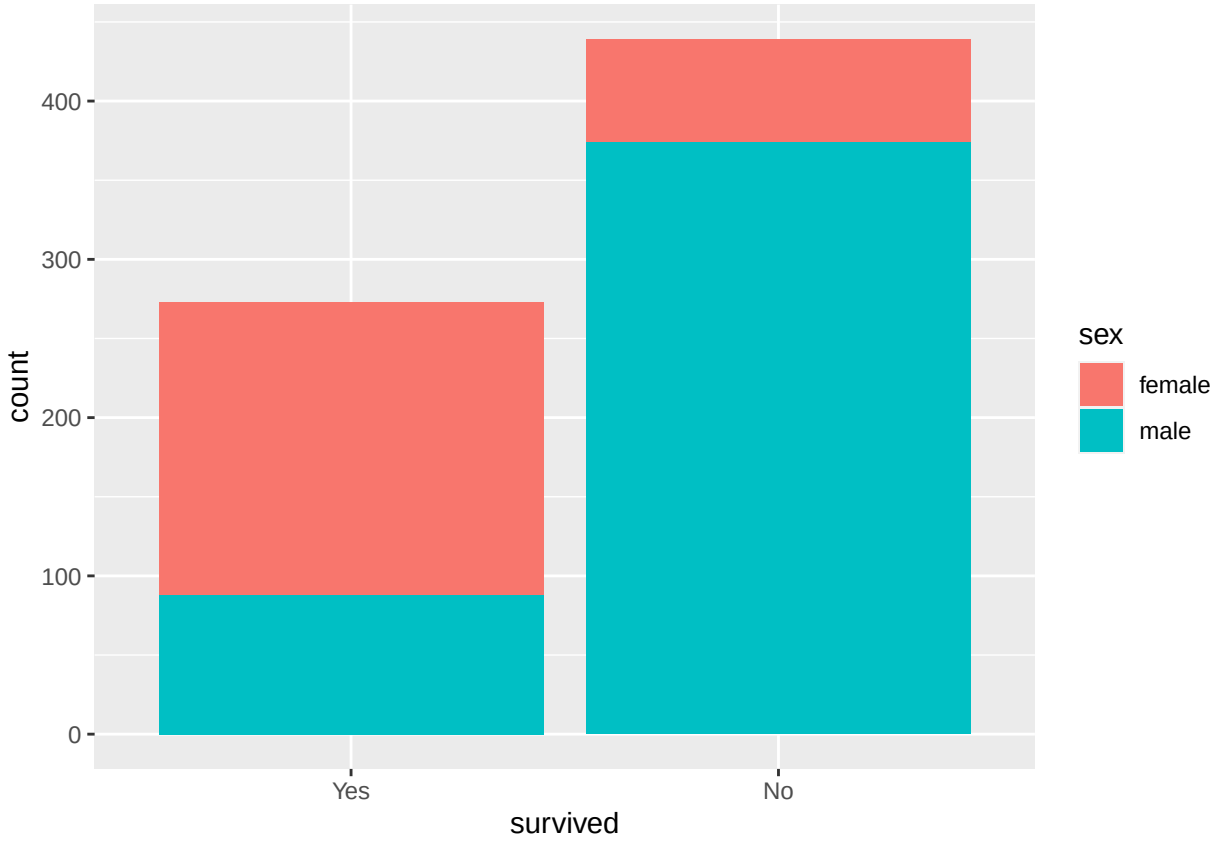
We are asked to explore/describe the distribution of the outcome variable 'survived' with the training dataset. First, let us plot a bar plot of the survived vs. not survived.

```
ggplot(dataset_train, aes(x = survived)) + geom_histogram(stat = 'count')
```




From this, we can see that it looks like more people did not survive, as opposed to did survive. This graph is not very informative though. Let us instead try grouping 'survived' with some other variables that might tell us more about it.

```
ggplot(dataset_train, aes(x = survived, fill = sex)) + geom_bar()
```

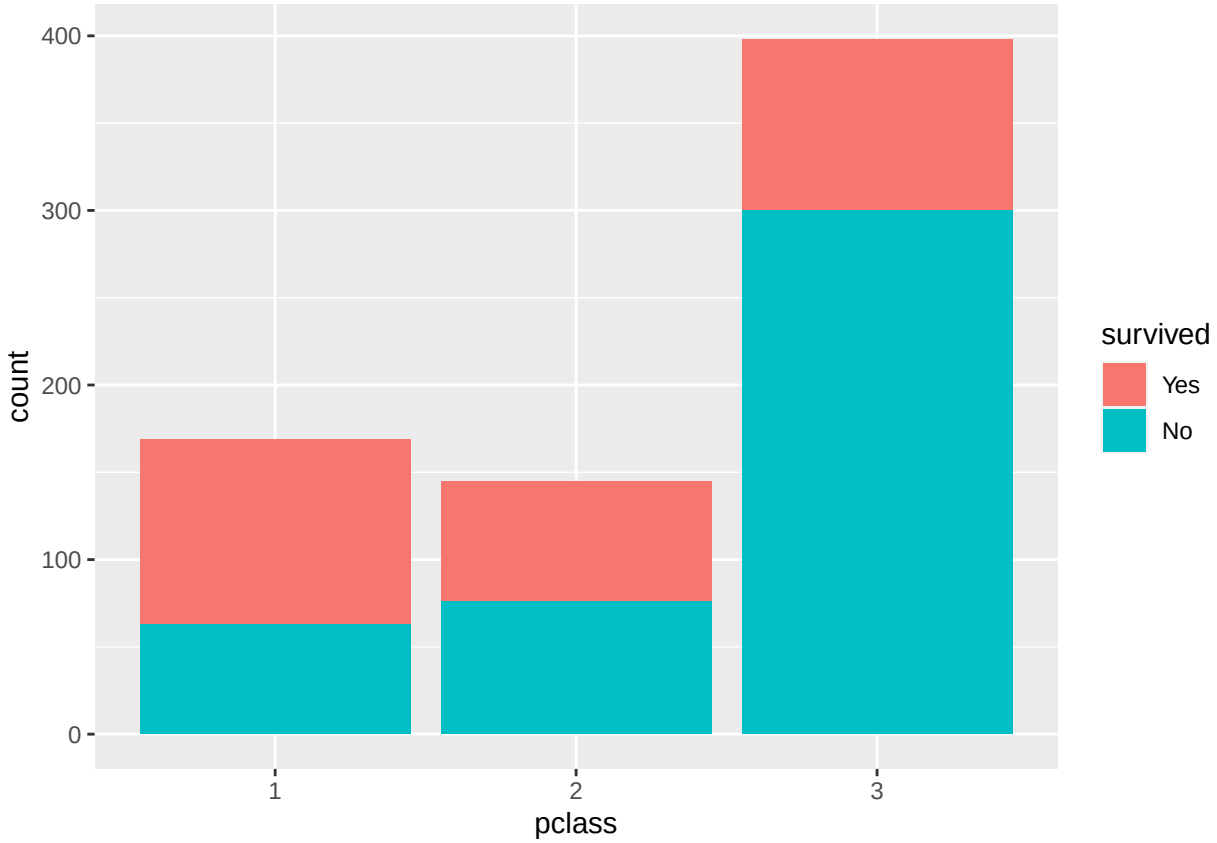



Looking at this, we can see that there seems to be a much higher proportion of females that survived as opposed to males. Let's look at the age distribution, and see the ages that survived.

```
ggplot(dataset_train, aes(x = age, fill = survived)) + geom_histogram(binwidth = 2)
```


Looking at this histogram, we can see that it seems like the majority of the passengers who survived were younger in age: particularly, those below 10 seem to have a higher chance of surviving as opposed to not surviving. Then, let us finally look at the survival rates by class.

```
ggplot(dataset_train, aes(x = pclass, fill = survived)) + geom_bar()
```




Looking at this, it seems like first class passengers had a pretty decent chance of surviving, well over 50 percent. Second class passengers also had a somewhat decent chance of surviving, with around a 50 percent survival rate. Finally, it seems like the third class passengers had a very bad survival rate, with only a little over a quarter of the passengers in the third class surviving.

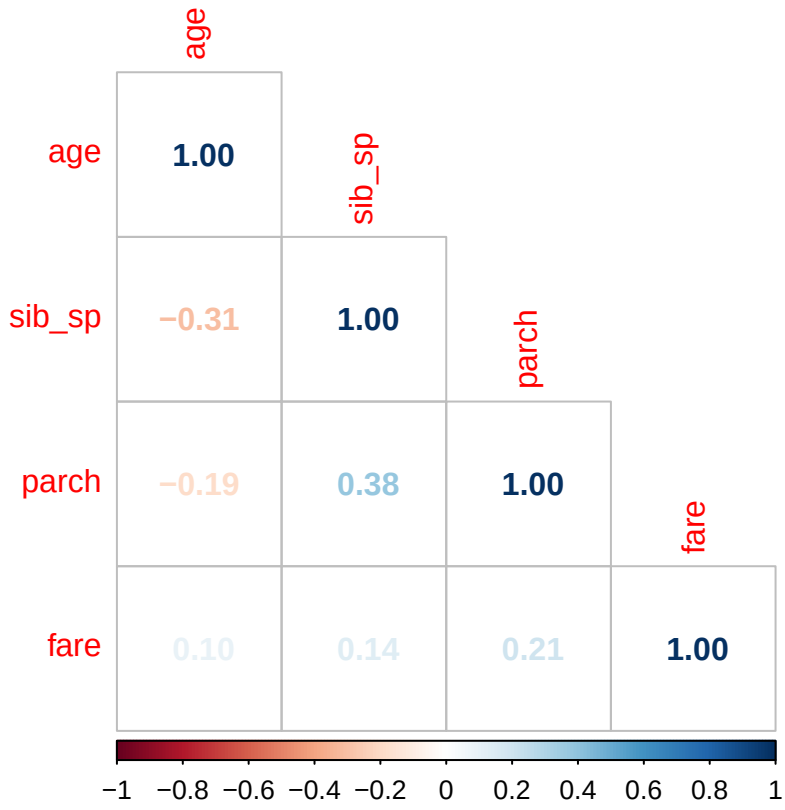
Thus, we have enough information to describe the distribution of survived as follows. It seems that more passengers did not survive on the Titanic, as opposed to survived. It also seems that women were much more likely to survive, as opposed to men. Additionally, younger passengers also had a higher chance of surviving the crash than older passengers. Finally, it seems that first class passengers and second class passengers had a somewhat decent chance of surviving, as opposed to third class passengers. This is the distribution of the 'survived' variable.

## Question 3

Using the training dataset, we are asked to create a correlation matrix of all continuous variables. We state that 'age', 'fare', 'sib_sp', 'parch', 'fare' are continuous variables. We create the correlation matrix as follows:

```
#Assigning all of the non continuous variables as null.
dataset_corr <- dataset
dataset_corr$passenger_id <- NULL
dataset_corr$survived <- NULL
dataset_corr$pclass <- NULL
dataset_corr$name <- NULL
dataset_corr$sex <- NULL
dataset_corr$ticket <- NULL
```

```
dataset_corr$cabin <- NULL
dataset_corr$embarked <- NULL

#Creating the correlation matrix, and visualizing it.
corrplot(cor(dataset_corr, use = "complete.obs"), method = 'number', type = 'lower')
```




Looking at this matrix, we see that age and sib_sp seem to be slightly negatively correlated with one another. Additionally, we see that parch and sib_sp are slightly positively correlated with one another. These are the only two sets of predictors that seem to have some type of meaningful relationship with one another.

## Question 4

We are asked to create a recipe predicting the outcome variable survived, using the predictors ticket class, sex, age, number of siblings or spouses aboard, number of parents or children aboard, and passenger fare. We are asked to do this on the training data. Additionally, we are asked to use step_impute_linear() to account for the fact that there are some missing variables in age. We are also asked to include interactions between sex/passenger fare, and age/passenger fare. We do all of this as follows:

```
#Creating the recipe.
survived_recipe <-
  #Defining the predictors and response variables, and doing it on the training dataset.
  recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = dataset_train) %>%
  #imputing the variable age.
```

```
  step_impute_linear(age) %>%
  #creating dummy variables for all categorical variables.
  step_dummy(all_nominal_predictors()) %>%
  #creating the interaction terms.
  step_interact( ~ starts_with("sex"):fare) %>%
  step_interact( ~ age:fare)
```

## Question 5

We are asked to specify a logistic regression model for classification using the 'glm' engine. Then, we are asked to create a workflow, and to add our model and the appropriate recipe. We are then asked to use fit() to apply our workflow to the training data, and to store the results of fit(). We do all of this as follows:

```
#First, we define the logistic regression model using 'glm'.
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

#We then define the workflow and add the recipe.
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(survived_recipe)

#We then fit the data on the training data.
log_fit <- fit(log_wkflow, dataset_train)

#We then tidy the data and assign it to a variable name.
glm_tidy <- log_fit %>% tidy()

#Printing out the tidied data so we can look at it.
glm_tidy
```

```
## # A tibble: 10 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -4.19      0.621       -6.75  1.51e-11
##  2 age              0.0542    0.0122       4.44  8.85e- 6
##  3 sib_sp           0.401     0.121        3.31  9.19e- 4
##  4 parch            0.0629    0.128        0.490 6.24e- 1
##  5 fare             0.00104   0.00901      0.115 9.08e- 1
##  6 pclass_X2        1.11      0.340        3.26  1.13e- 3
##  7 pclass_X3        2.30      0.353        6.51  7.43e-11
##  8 sex_male         2.45      0.281        8.71  2.96e-18
##  9 sex_male_x_fare  0.00789   0.00683      1.15  2.48e- 1
## 10 age_x_fare      -0.000299  0.000188    -1.59  1.11e- 1
```

## Question 6

We are asked to repeat Question 5, but this time to specify a linear discriminant analysis model for classification using the 'MASS' engine. We do this as follows:

```r
#First, we define the LDA model using 'MASS'.
lda_mod <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

#We then define the workflow and add the recipe.
lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(survived_recipe)

#We then fit the data on the training data.
lda_fit <- fit(lda_wkflow, dataset_train)

#Printing out the coefficients of the linear discriminants.
lda_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_linear()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * 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  sex_male
## Yes 28.58747 0.5091575 0.4871795 46.44966 0.2527473 0.3589744 0.3223443
## No  29.94800 0.5626424 0.3302961 21.60706 0.1731207 0.6833713 0.8519362
##     sex_male_x_fare age_x_fare
## Yes        12.23347  1468.4764
## No         18.20878   662.4589
##
## Coefficients of linear discriminants:
##                             LD1
## age              3.055656e-02
## sib_sp           2.177414e-01
## parch            3.866781e-02
## fare             2.611665e-03
## pclass_X2        7.453560e-01
## pclass_X3        1.530848e+00
## sex_male         2.104707e+00
```

```
## sex_male_x_fare -1.979945e-05
## age_x_fare      -1.244060e-04
```

## Question 7

We are then asked to repeat Question 5 (again!), but this time, specifying a quadratic discriminant analysis model for classification. We do this as follows:

```
#Defining the model for QDA.
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

#Creating the workflow and adding the recipe.
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(survived_recipe)

#Fitting the model.
qda_fit <- fit(qda_wkflow, dataset_train)

#Printing out the fit to look at the values.
qda_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_quad()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * 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  sex_male
## Yes 28.58747 0.5091575 0.4871795 46.44966 0.2527473 0.3589744 0.3223443
## No  29.94800 0.5626424 0.3302961 21.60706 0.1731207 0.6833713 0.8519362
##     sex_male_x_fare age_x_fare
## Yes        12.23347  1468.4764
## No         18.20878   662.4589
```

## Question 8

We are once again asked to repeat Question 5, but this time, to specify a naive Bayes model for classification using the "klaR" engine. We are also asked to set the 'usekernel' argument to FALSE. We do this as follows:

```
#We fit the naive Bayes model as asked of us.
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)

#We create the workflow, and add our recipe into it.
nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(survived_recipe)

#We then fit our model on the training dataset.
nb_fit <- fit(nb_wkflow, dataset_train)

#We print out nb_fit just to see what it looks like.
nb_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: naive_Bayes()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## $apriori
## grouping
##      Yes       No
## 0.383427 0.616573
##
## $tables
## $tables$age
##         [,1]     [,2]
## Yes 28.58747 14.13024
## No  29.94800 13.47444
##
## $tables$sib_sp
##          [,1]      [,2]
## Yes 0.5091575 0.7431726
## No  0.5626424 1.2732113
##
## $tables$parch
##          [,1]      [,2]
```

```
## Yes 0.4871795 0.8139526
## No  0.3302961 0.8180429
##
## $tables$fare
##         [,1]     [,2]
## Yes 46.44966 62.83323
## No  21.60706 28.73865
##
## $tables$pclass_X2
##          [,1]      [,2]
## Yes 0.2527473 0.4353854
## No  0.1731207 0.3787833
##
## $tables$pclass_X3
##          [,1]      [,2]
## Yes 0.3589744 0.4805807
## No  0.6833713 0.4656919
##
## $tables$sex_male
##          [,1]      [,2]
## Yes 0.3223443 0.4682324
## No  0.8519362 0.3555684
##
## $tables$sex_male_x_fare
##         [,1]     [,2]
## Yes 12.23347 38.24548
## No  18.20878 27.95337
##
## $tables$age_x_fare
##          [,1]     [,2]
## Yes 1468.4764 2274.679
## No   662.4589 1208.455
##
## ...
## and 1446 more lines.
```

## Question 9

We are now asked to use predict() and bind_cols() to generate predictions using each of these four models and our training data. We are then asked to use the 'accuracy' metric to assess the performance of each of the four models, and to state which model achieved the highest accuracy on our training data. We do this as follows:

```
#First, we want to write code to get the predicted classes of the passengers, where the class indicates

#We obtain the predictions of which passengers survived and didn't survive for each model.
log_prb_pred <- predict(log_fit, new_data = dataset_train, type = "class")
lda_prb_pred <- predict(lda_fit, new_data = dataset_train, type = "class")
qda_prb_pred <- predict(qda_fit, new_data = dataset_train, type = "class")
nb_prb_pred <- predict(nb_fit, new_data = dataset_train, type = "class")
```

```
#Then, we append these columns using bind_cols() to the true class that each passenger had, where the c
model_predictions <- bind_cols(dataset_train %>% select(survived), log_prb_pred, lda_prb_pred, qda_prb_

#Renaming the column names so they make more sense.
colnames(model_predictions) <- c('Truth', 'LOG', 'LDA', 'QDA', 'NB')

#Printing out the first few rows of the dataframe for us to view.
head(model_predictions)
```

```
##   Truth LOG LDA QDA NB
## 1    No  No  No  No No
## 2    No  No  No  No No
## 3    No  No  No  No No
## 4    No  No  No  No No
## 5    No  No  No  No No
## 6    No  No  No  No No
```

Thus, we have used predict() and bind_cols() to generate predictions using each of these four models and our training data. We then turn to the accuracy metric portion of the question.

```
#We then use the 'accuracy' metric to assess the performance of each of the four models.

#Getting the logistic regression accuraccy.
log_reg_acc <- augment(log_fit, new_data = dataset_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

#Getting the LDA accuracy.
lda_acc <- augment(lda_fit, new_data = dataset_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

#Getting the QDA accuracy.
qda_acc <- augment(qda_fit, new_data = dataset_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

#Getting the Naive Bayes accuracy.
nb_acc <- augment(nb_fit, new_data = dataset_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

#Putting all of the accuracies together.
accuracies <- c(log_reg_acc$.estimate, lda_acc$.estimate,
                nb_acc$.estimate, qda_acc$.estimate)

#Putting all of the model names together.
models <- c("Logistic Regression", "LDA", "Naive Bayes", "QDA")

#Combining the two lists into one tibble.
results <- tibble(accuracies = accuracies, models = models)

#Printing out the results and arranging them nicely.
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 4 x 2
```

```
##   accuracies models
##        <dbl> <chr>
## 1      0.805 Logistic Regression
## 2      0.794 LDA
## 3      0.781 QDA
## 4      0.768 Naive Bayes
```

The model that achieves the highest accuracy on the training data is Logistic Regression, with an accuracy of approximately 80.47 percent.

## Question 10

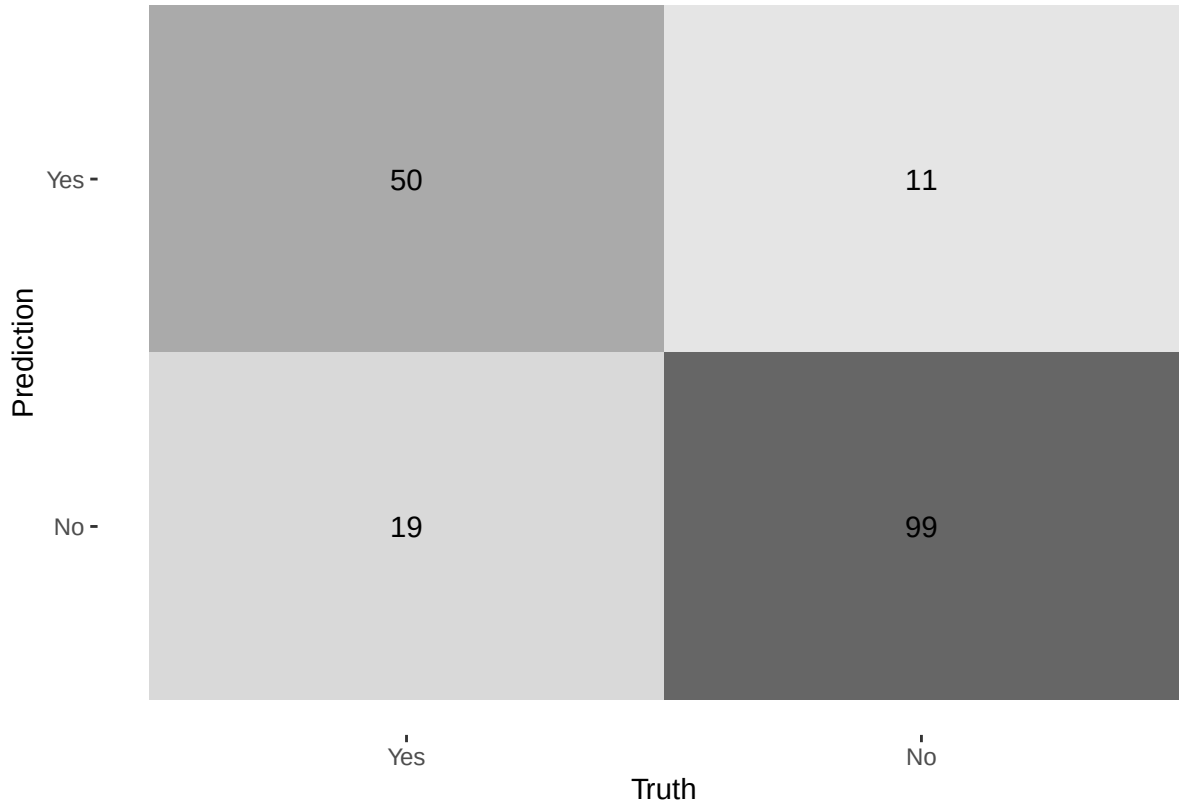
We are now asked to fit the model with the highest training accuracy to the testing data, and to report the accuracy of the model on the testing data. We do this as follows:

```
#We fit the model on the testing data now.
log_fit_test <- fit(log_wkflow, dataset_test)

#We then calculate it's accuracy.
log_reg_acc_test <- augment(log_fit_test, new_data = dataset_test) %>%
  accuracy(truth = survived, estimate = .pred_class)

#Printing out the accuracy value.
log_reg_acc_test
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.832
```

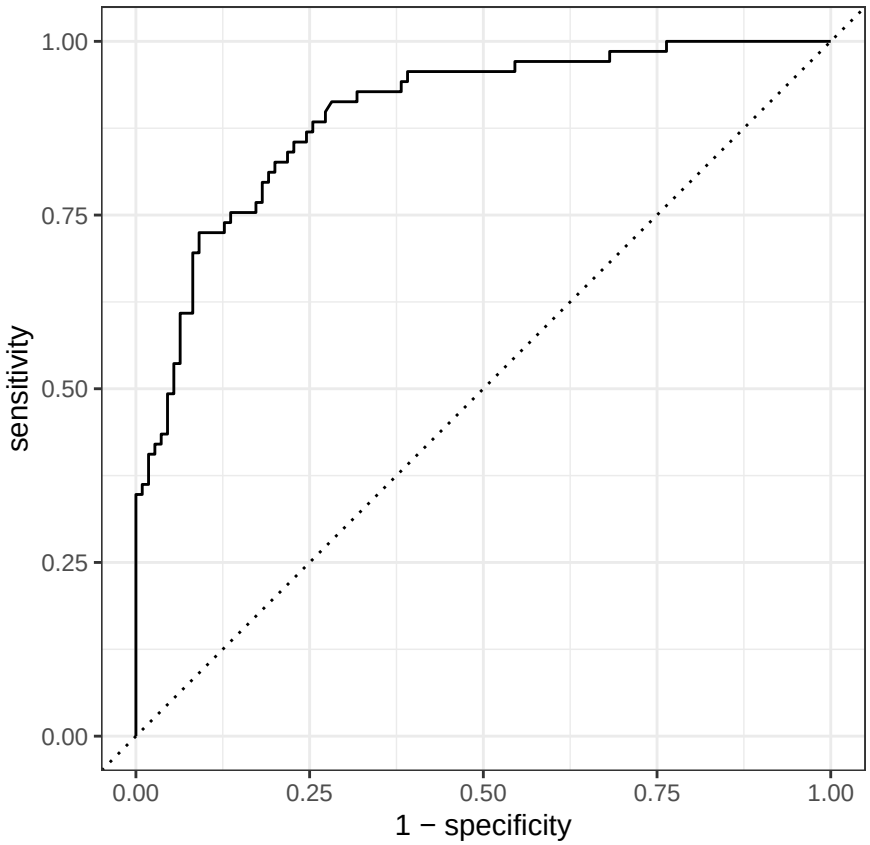
We see that, with the testing data, the accuracy of the Logistic Regression model is approximately 83.24 percent. We are now asked to, with the testing data, create a confusion matrix and visualize it. We do this as follows:

```
#Creating the confusion matrix.
augment(log_fit_test, new_data = dataset_test) %>%
  conf_mat(truth = survived, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




Looking at this visualization at face value, we can see that the model seems to be performing mostly well, as it correctly is classifying most of the observations. Finally, we are asked to plot an ROC curve, and to calculate the area under it (AUC).

```
#Here, we plot the ROC curve.
augment(log_fit_test, new_data = dataset_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```




This is the ROC graph. We can get it's area under the accuracy of the curve as follows:

```
augment(log_fit_test, new_data = dataset_test) %>%
  roc_auc(survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.895
```

Thus, we see that the area under the curve is equal to about 89.52. We are then asked how the model performed. I personally think that the model performed pretty well, as it did very well on both the training and testing datasets. In particular, it's training accuracy was approximately 80.47 percent, while it's testing accuracy was approximately 83.24 percent. I think that the values differ because the training and the testing datasets are two different datasets, and thus, they have different data within them. Thus, the model may perform better on one dataset than on the other, or vice-versa, which is why the numbers/values for the accuracy differ from one another.