Homework 3

PSTAT 131/231

Contents

Classification 1

Classification

For this assignment, we will be working with part of a Kaggle data set that was the subject of a machine learning competition and is often used for practicing ML models. The goal is classification; specifically, to predict which passengers would survive the Titanic shipwreck.

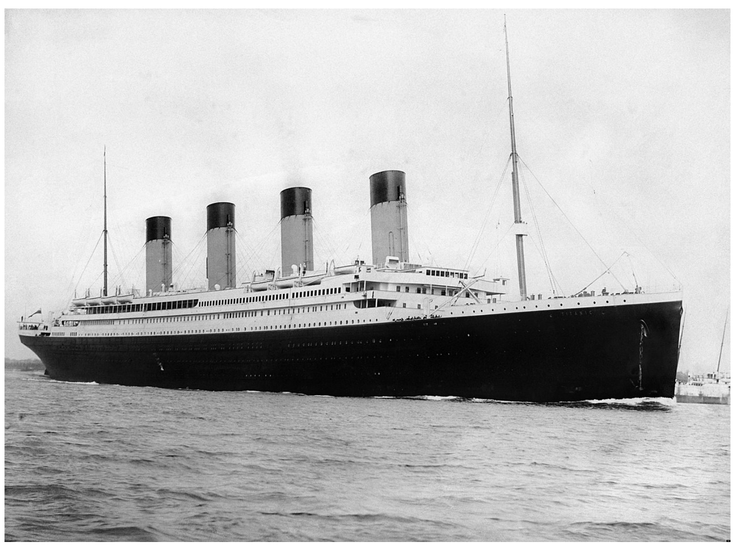


Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

Load the data from `data/titanic.csv` into *R* and familiarize yourself with the variables it contains using the codebook (`data/titanic_codebook.txt`).

```
library(dplyr)
library(tidyverse)
library(tidymodels)
library(ggplot2)
library(klaR)
library(discrim)
library(poissonreg)
library(corrr)
library(pROC)
tidymodels_prefer()
```

```r
titanic = read.csv("D:/UCSB/Spring 2022/PSTAT 131/PSTAT_131_HW/HW2/PSTAT-131/homework-3/homework-3/data/
titanic$survived = factor(titanic$survived, levels = c("Yes","No"))
titanic$pclass = factor(titanic$pclass)
titanic$sex = factor(titanic$sex)
head(titanic,6)
```

```
##   passenger_id survived pclass
## 1            1       No      3
## 2            2      Yes      1
## 3            3      Yes      3
## 4            4      Yes      1
## 5            5       No      3
## 6            6       No      3
##                                                  name    sex age sib_sp parch
## 1                             Braund, Mr. Owen Harris   male  22      1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38      1     0
## 3                              Heikkinen, Miss. Laina female  26      0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35      1     0
## 5                            Allen, Mr. William Henry   male  35      0     0
## 6                                    Moran, Mr. James   male  NA      0     0
##             ticket    fare cabin embarked
## 1        A/5 21171  7.2500  <NA>        S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

Notice that `survived` and `pclass` should be changed to factors. When changing `survived` to a factor, you may want to reorder the factor so that *"Yes"* is the first level.

Make sure you load the `tidyverse` and `tidymodels`!

Remember that you'll need to set a seed at the beginning of the document to reproduce your results.

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.

```r
set.seed(3435)
survived_split = initial_split(titanic, prop = 0.7, strata = survived)
survived_train = training(survived_split)
survived_test = testing(survived_split)

nrow(survived_train)
```

```
## [1] 623
```

```r
nrow(survived_test)
```

```
## [1] 268
```

```
sum(is.na(survived_train$age))
```

```
## [1] 128
```

```
sum(is.na(survived_train$sex))
```

```
## [1] 0
```

```
sum(is.na(survived_train$sib_sp))
```

```
## [1] 0
```

There are 611 missing values in total, which may cause serious issues.

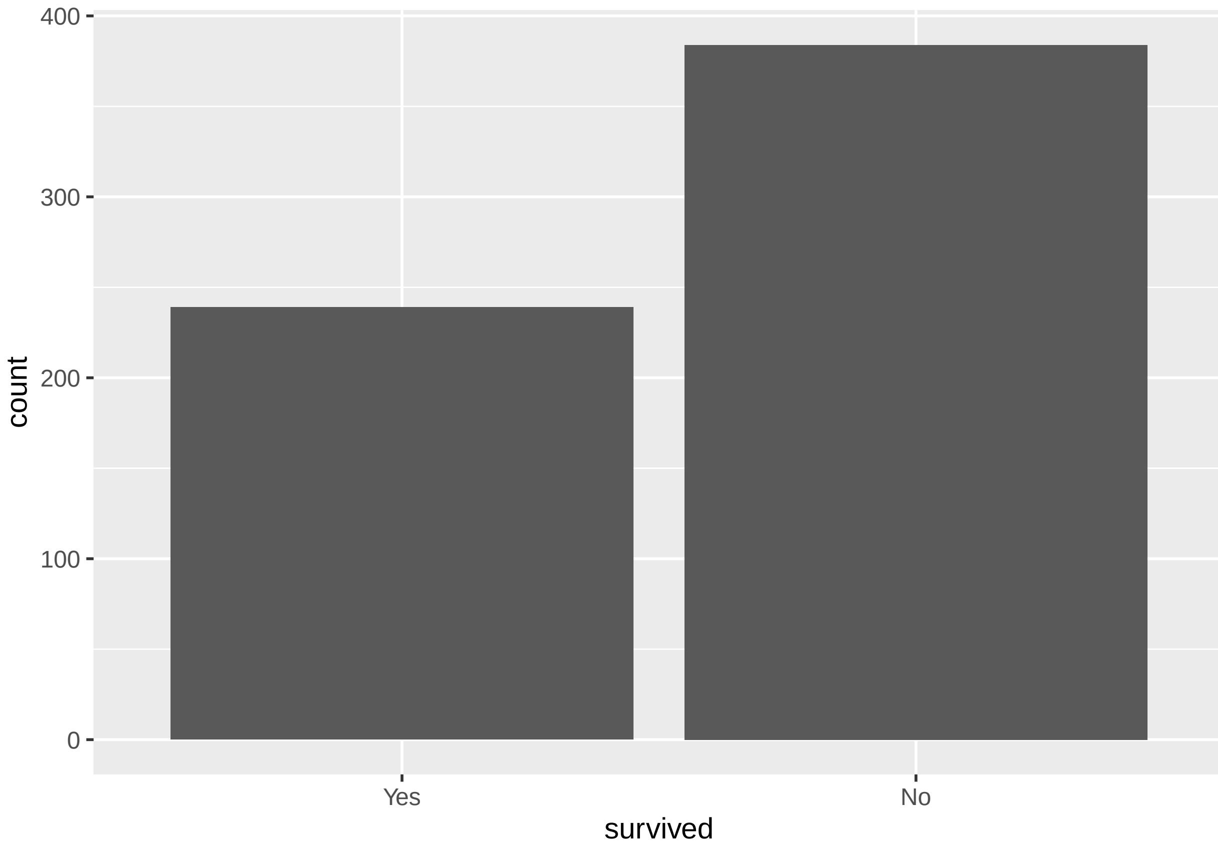
Why is it a good idea to use stratified sampling for this data?

We find that the data contains passengers divided into different social classes. We use stratified sampling to extract data from different classes so the sample can best represent the entire population being studied.

Question 2

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

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

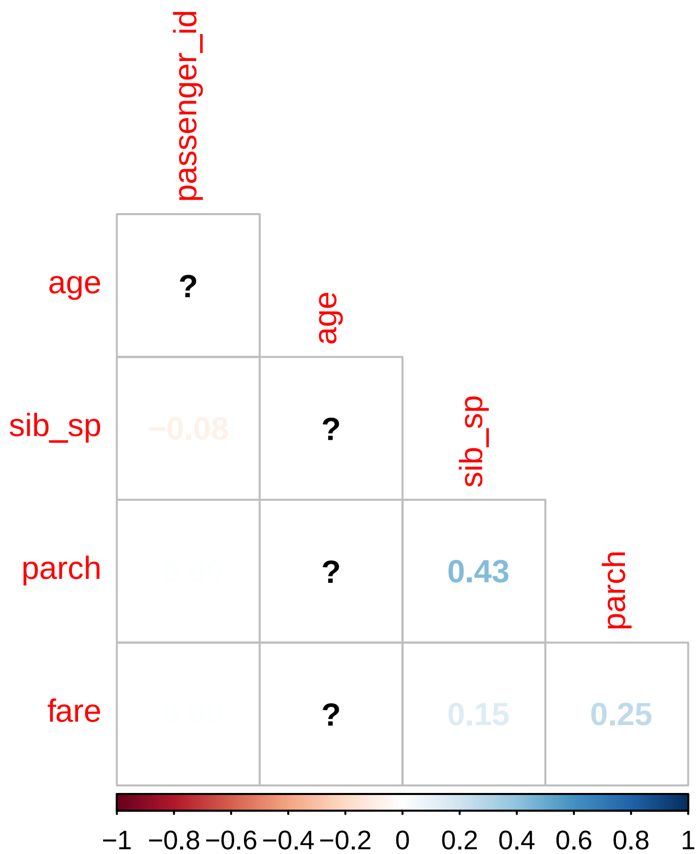


Overall, the number of people who were not survived are significant larger than the number of people who were survived.

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?

```
library(corrplot)
survived_train %>%
  dplyr::select(is.numeric ) %>%
  cor() %>%
  corrplot(type = 'lower', diag = FALSE, method = "number")
```



We find that there exists a positive correlation between sib_sp(number of siblings / spouses aboard the Titanic) and parch(number of parents / children aboard the Titanic). Also, Passenger fare shows a weak positive correlation with sib_sp and parch.

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.

```
survived_recipe = recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = survived_train) %
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ fare:starts_with("sex") + age:fare) #%>%
  #prep(survived_train)

summary(survived_recipe)
```

```
## # A tibble: 7 x 4
##   variable type    role      source
##   <chr>    <chr>   <chr>     <chr>
## 1 pclass   nominal predictor original
## 2 sex      nominal predictor original
## 3 age      numeric predictor original
## 4 sib_sp   numeric predictor original
## 5 parch    numeric predictor original
## 6 fare     numeric predictor original
## 7 survived nominal outcome   original
```

```
prepped_data <-
  survived_recipe %>% # use the recipe object
  prep() %>% # perform the recipe on training data
  juice()
glimpse(prepped_data)
```

```
## Rows: 623
## Columns: 10
## $ age            <dbl> 22.00000, 35.00000, 54.00000, 2.00000, 39.00000, 14.00~
## $ sib_sp         <int> 1, 0, 0, 3, 1, 0, 4, 1, 3, 3, 0, 1, 1, 1, 1, 0, 1, 2, ~
## $ parch          <int> 0, 0, 0, 1, 5, 0, 1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ fare           <dbl> 7.2500, 8.0500, 51.8625, 21.0750, 31.2750, 7.8542, 29.~
## $ survived       <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No~
## $ pclass_X2      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ~
## $ pclass_X3      <dbl> 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, ~
## $ sex_male       <dbl> 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ~
## $ fare_x_sex_male <dbl> 7.2500, 8.0500, 51.8625, 21.0750, 31.2750, 0.0000, 29.~
## $ fare_x_age     <dbl> 159.5000, 281.7500, 2800.5750, 42.1500, 1219.7250, 109~
```

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.

Hint: Make sure to store the results of `fit()`. You'll need them later on.

```
log_reg = logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_wkflow = workflow() %>%
  add_model(log_reg) %>%
  add_recipe(survived_recipe)

log_fit = fit(log_wkflow, survived_train)

log_fit %>% tidy()
```

```
## # A tibble: 10 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)      -4.40      0.683       -6.45  1.13e-10
```

```
##   2 age              0.0629    0.0136       4.62  3.77e- 6
##   3 sib_sp           0.437     0.132        3.30  9.52e- 4
##   4 parch            0.151     0.153        0.989 3.23e- 1
##   5 fare            -0.00116   0.0107      -0.108 9.14e- 1
##   6 pclass_X2        1.25      0.363        3.46  5.48e- 4
##   7 pclass_X3        2.44      0.382        6.39  1.62e-10
##   8 sex_male         2.15      0.303        7.09  1.32e-12
##   9 fare_x_sex_male  0.0139    0.00836      1.66  9.65e- 2
## 10 fare_x_age       -0.000360  0.000206    -1.75  8.03e- 2
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.814
```

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(survived_recipe)

lda_fit = fit(lda_wkflow, survived_train)
```

```
lda_acc <- augment(lda_fit, new_data = survived_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc
```

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

Question 7

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

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

```r
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(survived_recipe)

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

```r
qda_acc <- augment(qda_fit, new_data = survived_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
qda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.778
```

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(survived_recipe)

nb_fit <- fit(nb_wkflow, survived_train)
```

```r
nb_acc <- augment(nb_fit, new_data = survived_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
nb_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.787
```

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.

```r
pred_models = bind_cols(predict(log_fit, survived_test),
                    predict(nb_fit, survived_test),
                    predict(qda_fit, survived_test),
```

```
                    predict(lda_fit, survived_test))
colnames(pred_models) = c("log_pred","nb_pred","qda_pred", "lda_pred")
head(pred_models)
```

```
## # A tibble: 6 x 4
##   log_pred nb_pred qda_pred lda_pred
##   <fct>    <fct>   <fct>    <fct>
## 1 Yes      Yes     Yes      Yes
## 2 Yes      Yes     Yes      Yes
## 3 No       No      No       No
## 4 No       No      No       No
## 5 No       No      No       No
## 6 No       No      No       No
```

```
accuracies <- c(log_reg_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      0.814 Logistic Regression
## 2      0.793 LDA
## 3      0.787 Naive Bayes
## 4      0.778 QDA
```

Which model achieved the highest accuracy on the training data?

The Logistic Regression achieved the highest accuracy on the training data.

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?

```
head(predict(log_fit, new_data = survived_test, type = "prob"),10)
```

```
## # A tibble: 10 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1     0.933   0.0671
##  2     0.924   0.0763
##  3     0.119   0.881
```

```
## 4     0.183    0.817
## 5     0.230    0.770
## 6     0.239    0.761
## 7     0.120    0.880
## 8     0.119    0.881
## 9     0.0456   0.954
## 10    0.174    0.826
```

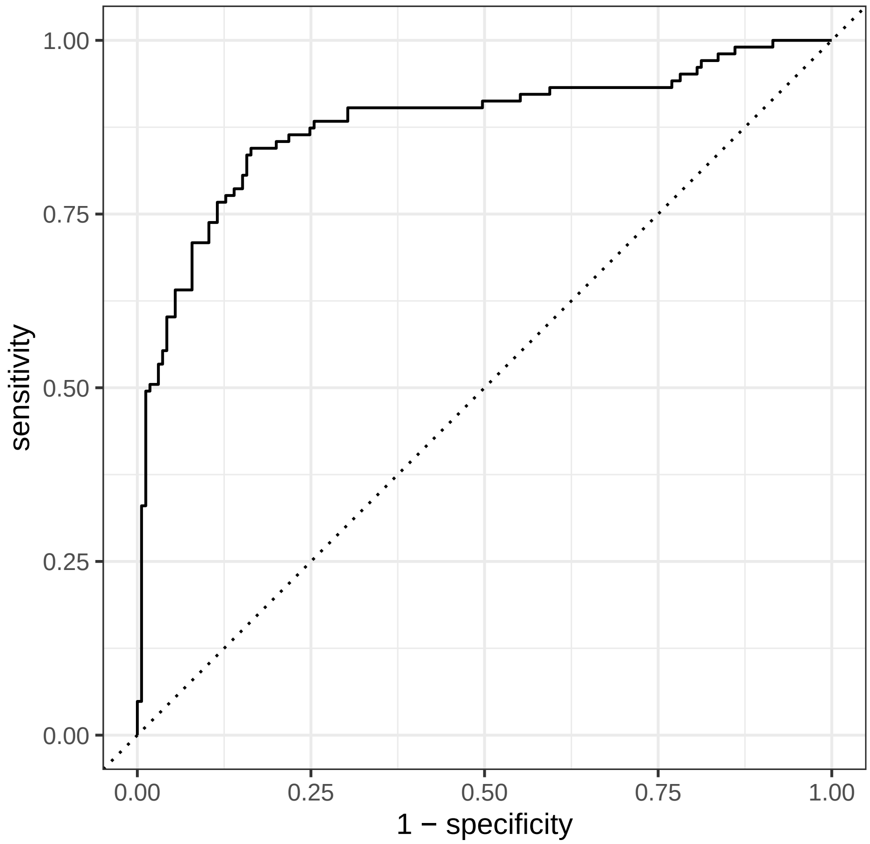
```
# confusion matrix
augment(log_fit, new_data = survived_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##             Truth
## Prediction Yes  No
##        Yes  75  17
##        No   28 148
```

```
# accuracy
log_acc_test <- augment(log_fit, new_data = survived_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc_test
```

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

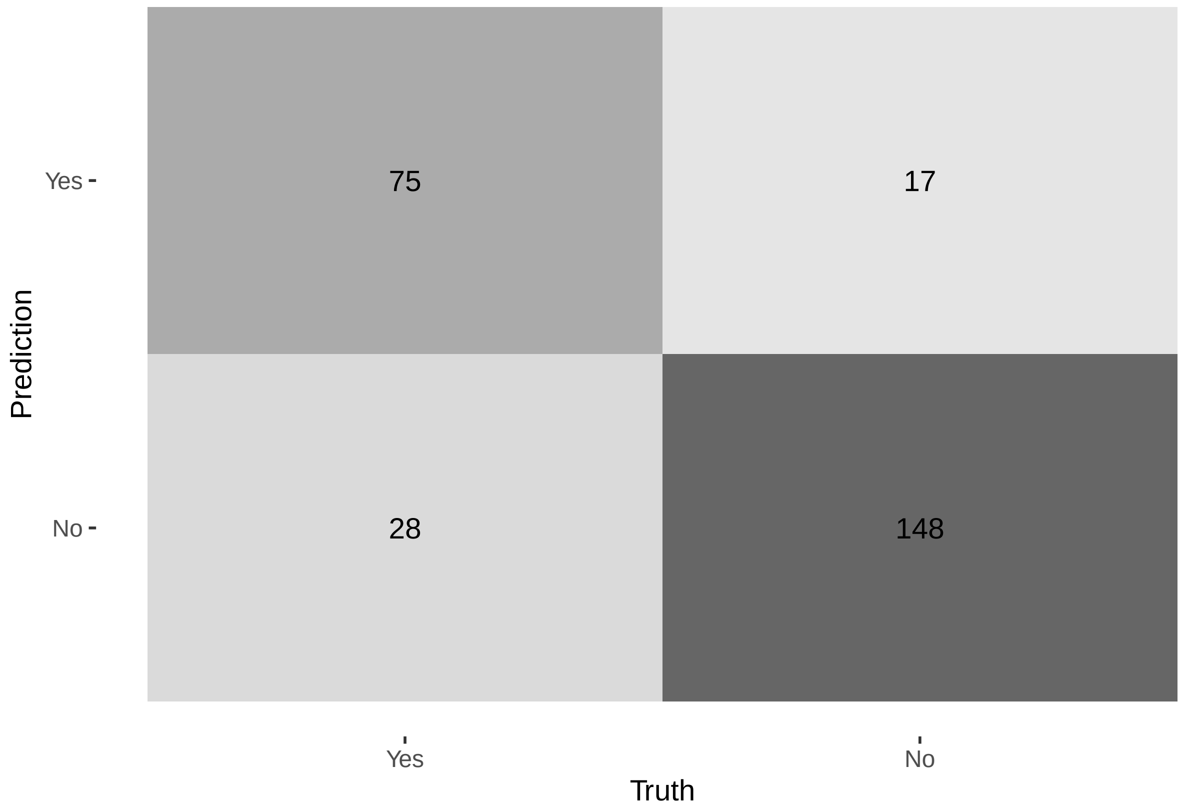
```
# ROC
augment(log_fit, new_data = survived_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```




```
augment(log_fit, new_data = survived_test) %>%
  roc(survived, .pred_Yes)
```

```
##
## Call:
## roc.data.frame(data = ., response = survived, predictor = .pred_Yes)
##
## Data: .pred_Yes in 103 controls (survived Yes) > 165 cases (survived No).
## Area under the curve: 0.88
```

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



Therefore, AUC is 0.88. The test accuracy of Logistic Regression is 0.8321 and the train accuracy is 0.8138, and they are quite similar to each other. It is surprised that test accuracy is higher. It maybe related to the splitting process where I set the ratio too high.