Homework 3

PSTAT 131/231

Contents

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.

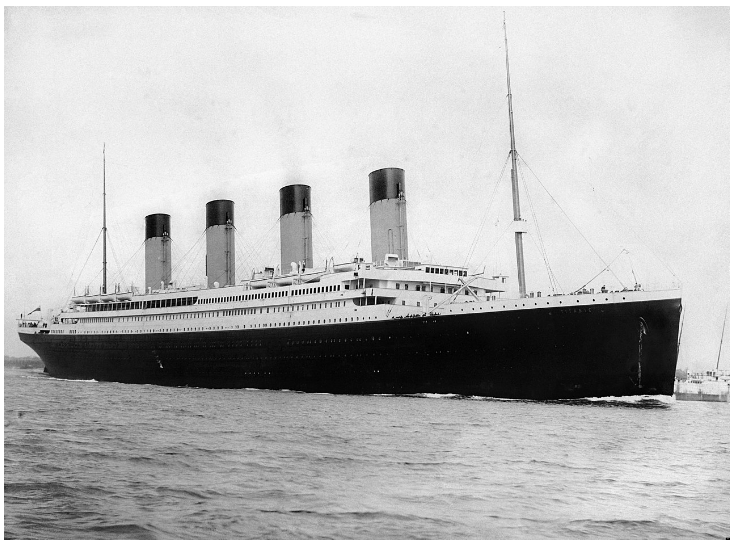


Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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

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`!

```r
titanic_data <- read.csv(file = "titanic.csv")
titanic_data$survived = as.factor(titanic_data$survived)
titanic_data$pclass = as.factor(titanic_data$pclass)
```

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.

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

```
set.seed(596)

titanic_split <- initial_split(titanic_data, prop = 0.8, strata = survived )
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

# check for the missing values
sum(is.na(titanic_train))
```

```
## [1] 695
```

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

```
## [1] 146
```

```
sum(is.na(titanic_train$cabin))
```

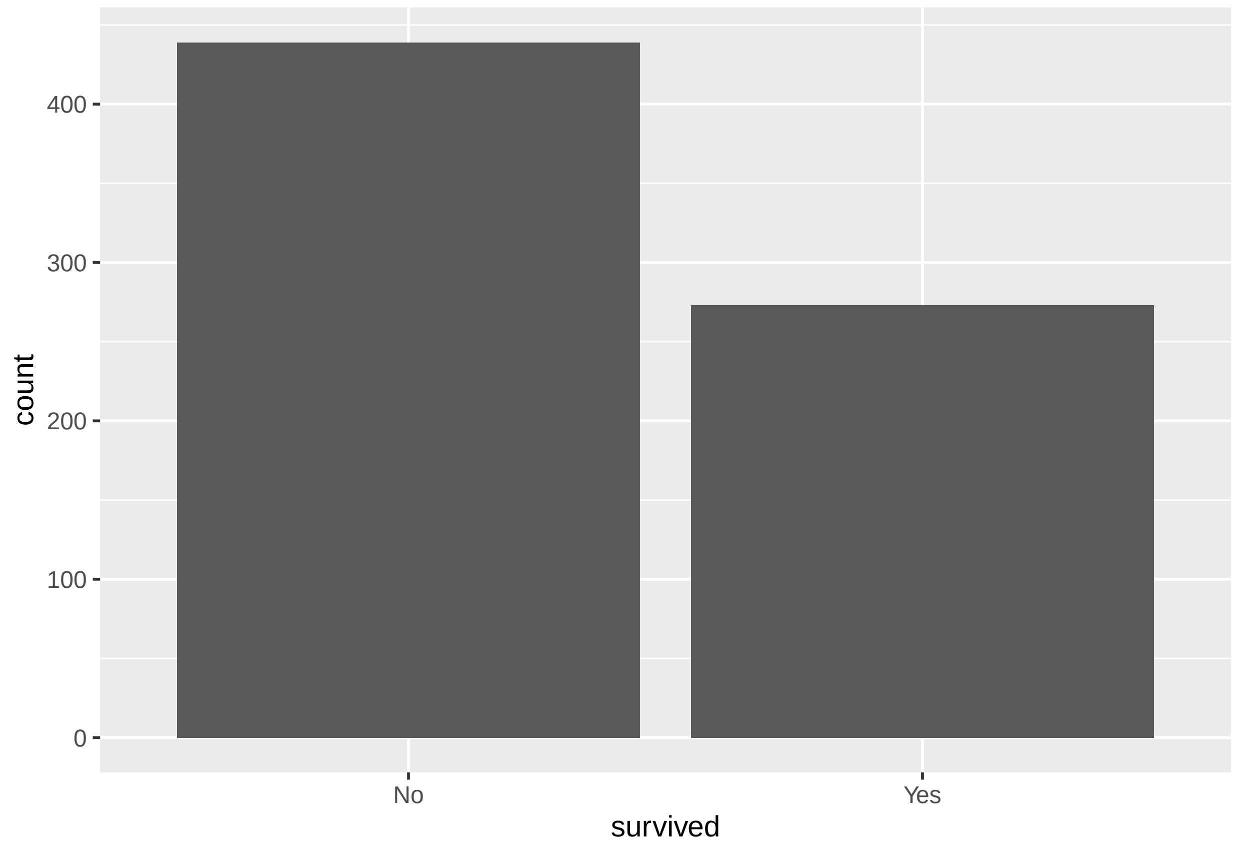
```
## [1] 548
```

So, after splitting the data into train and test. Now train data has 667 rows and testing data has 224 rows. The train data holds the 75% of the whole dataset. The spliting of the dataset could help us try several models in training set and find the best fit value to testing set. For missing value I used is.na to check for the missing value of the training set and after digging inside we could see the missing values are about age and cabin.

Question 2

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

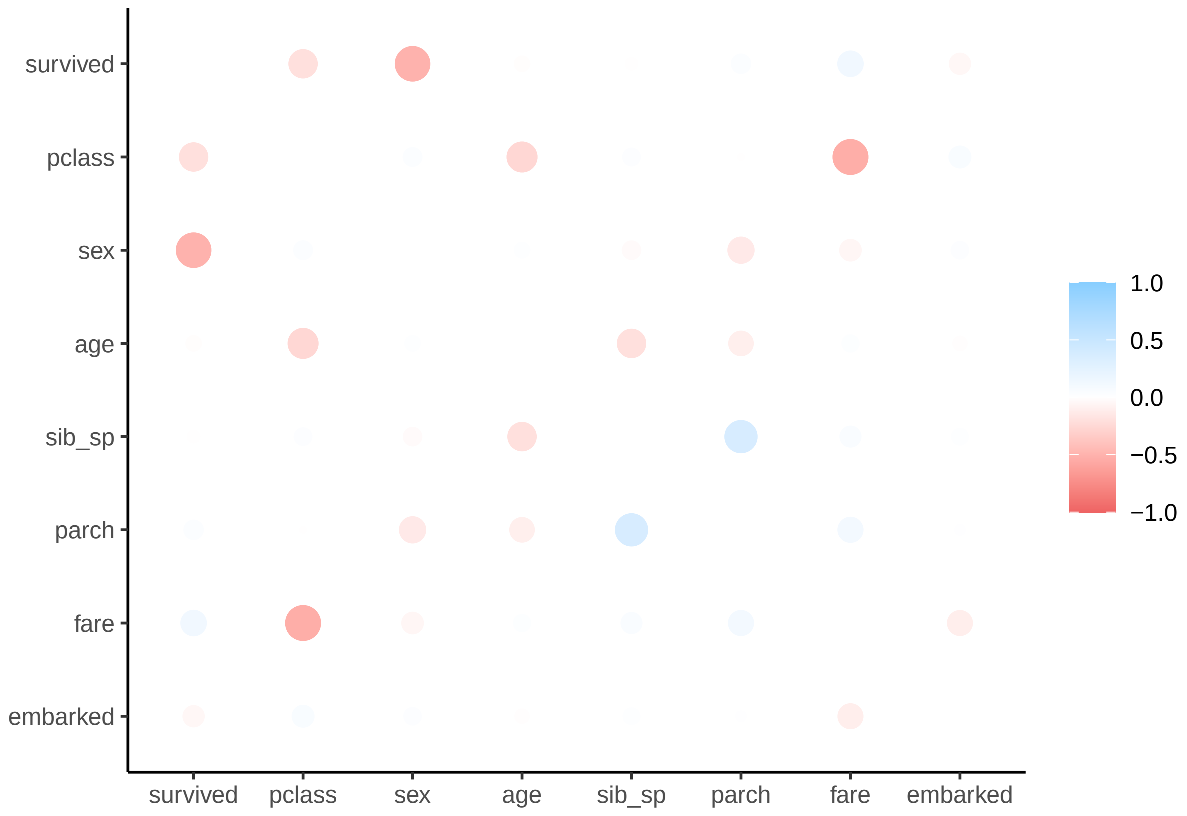
```
titanic_train %>%
  ggplot(aes( x = survived)) +
  geom_bar()
```



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?

```r
cor_titanic_train <- titanic_train %>%
  dplyr::select(-passenger_id, -name, -cabin, -ticket) %>%
  mutate(sex = fct_recode(sex,"0" = "male","1" = "female")) %>%
  mutate(embarked = fct_recode(embarked,"1" = "C","2" = "Q","3"="S"))  %>%
  mutate(sex = as.integer(sex),
         pclass = as.integer(pclass),
         survived = as.integer(survived),
         embarked = as.integer(embarked)) %>%
  correlate(use = "pairwise.complete.obs", method = "pearson")
rplot(cor_titanic_train)
```



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_interact(terms = ~ starts_with("sex"):fare +
                  age:fare)

titanic_recipe
```

```
## Recipe
##
```

```
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for age
## Dummy variables from all_nominal_predictors()
## Interactions with starts_with("sex"):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.

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

```
titanic_log <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

titanic_wkflow <- workflow() %>%
  add_model(titanic_log) %>%
  add_recipe(titanic_recipe)

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

Question 6

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

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

lda_wkflow <- workflow() %>%
  add_model(titanic_lda) %>%
  add_recipe(titanic_recipe)
class(titanic_train)
```

```
## [1] "data.frame"
```

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

Question 7

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

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

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

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

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`.

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

bayes_wkflow <- workflow() %>%
  add_model(titanic_bay) %>%
  add_recipe(titanic_recipe)

bayes_fit <- fit(bayes_wkflow, titanic_train)
```

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?

```
log_predic <- predict(log_fit, titanic_train)
lda_predict <- predict(lda_fit, titanic_train)
qda_predict <- predict(qda_fit, titanic_train)
bayes_predict <- predict(bayes_fit, titanic_train)

predict_bound <- bind_cols(log_predic, lda_predict,qda_predict, bayes_predict, titanic_train$survived)
colnames(predict_bound) = c("Log Predict", "LDA Predict", "QDA Predict",
                               "Bayes Predict", "Survived")

predict_bound
```

```
## # A tibble: 712 x 5
##    `Log Predict` `LDA Predict` `QDA Predict` `Bayes Predict` Survived
##    <fct>         <fct>         <fct>         <fct>           <fct>
##  1 No            No            No            No              No
##  2 No            No            No            No              No
##  3 No            No            No            No              No
##   4 Yes           Yes           No            No              No
##   5 No            No            No            No              No
##   6 No            Yes           No            No              No
##   7 No            No            No            No              No
##   8 No            No            No            No              No
##   9 No            No            No            Yes             No
##  10 No            No            No            No              No
## # ... with 702 more rows
```

```
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)
bayes_acc <- augment(bayes_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.795
```

```
lda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.782
```

```
qda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.771
```

```
bayes_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.763
```

We could say log has the highest prediction.

Question 10

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

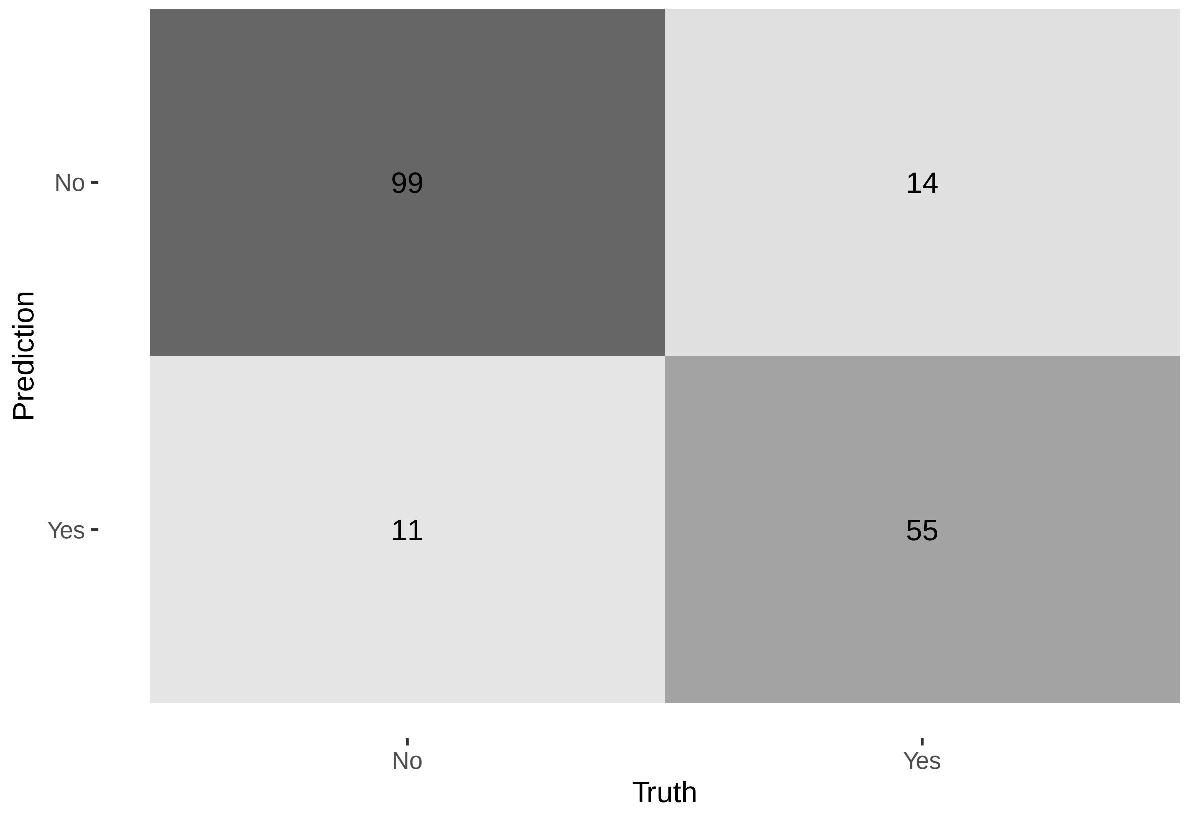
Again using the **testing** data, create a confusion matrix and visualize it. Plot an ROC curve and calculate the area under it (AUC).

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

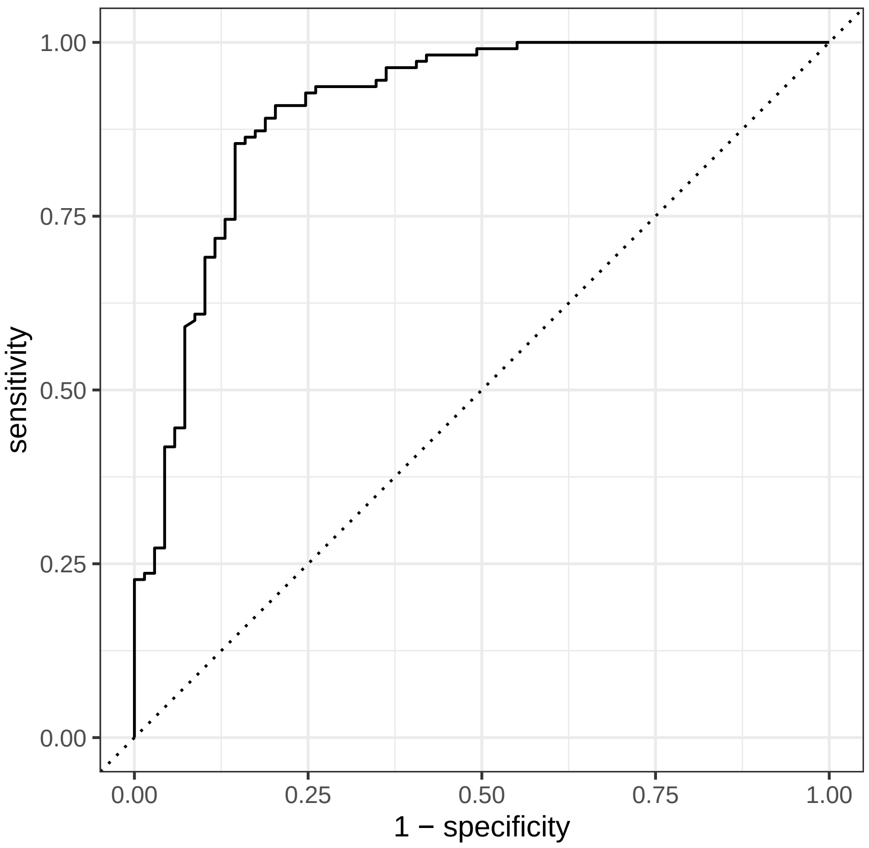
```
predict(log_fit, titanic_test)
```

```
## # A tibble: 179 x 1
##    .pred_class
##    <fct>
##  1 Yes
##  2 No
##  3 No
##  4 No
##  5 Yes
##  6 Yes
##  7 No
##  8 No
##  9 Yes
## 10 No
## # ... with 169 more rows
```

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



```
augment(log_fit, titanic_test) %>%
  roc_curve(survived, .pred_No) %>%
  autoplot()
```



Required for 231 Students

In a binary classification problem, let p represent the probability of class label 1, which implies that $1-p$ represents the probability of class label 0. The *logistic function* (also called the "inverse logit") is the cumulative distribution function of the logistic distribution, which maps a real number z to the open interval $(0,1)$.

Question 11

Given that:

$$p(z) = \frac{e^z}{1+e^z}$$

Prove that the inverse of a logistic function is indeed the *logit* function:

$$z(p) = ln\left(\frac{p}{1-p}\right)$$

Question 12

Assume that $z = \beta_0 + \beta_1 x_1$ and $p = logistic(z)$. How do the odds of the outcome change if you increase x_1 by two? Demonstrate this.

Assume now that β_1 is negative. What value does p approach as x_1 approaches ∞? What value does p approach as x_1 approaches $-\infty$?