PSTAT 131 Homework 4

Matthew Zhang

Spring 2022

```r
library(tidyverse)
library(tidymodels)
library(corrr)
library(discrim)
library(ggplot2)
library(poissonreg)
library(klaR)

#Load data
tidymodels_prefer()
titanic <- read.csv("data/titanic.csv")

titanic$survived <-factor(titanic$survived, levels=c("Yes", "No"))
titanic$pclass <-factor(titanic$pclass)
```

Resampling

For this assignment, we will continue 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.

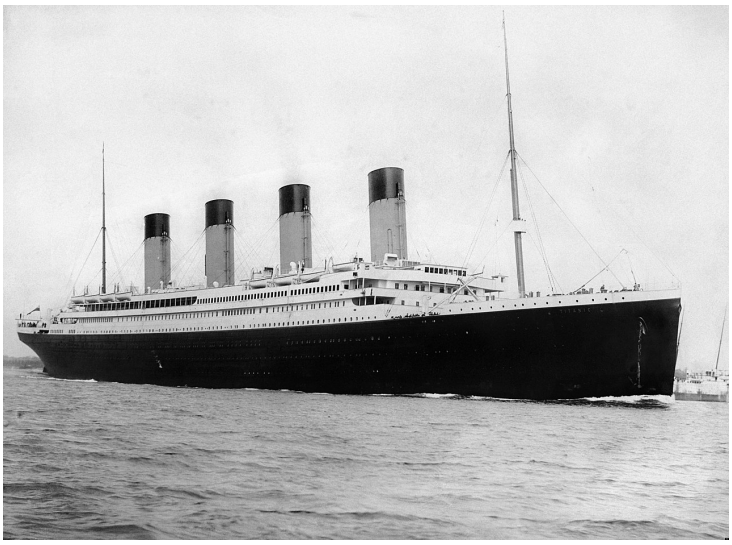

Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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`!

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

Create a recipe for this dataset **identical** to the recipe you used in Homework 3.

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.

```
set.seed(123)

titanic_split <- initial_split(titanic, prop = 0.80,
                               strata = survived)

titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

titanic_split
```

```
## <Analysis/Assess/Total>
## <712/179/891>
```

```
dim(titanic_train)
```

```
## [1] 712  12
```

```
dim(titanic_test)
```

```
## [1] 179  12
```

```
#Create recipe
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) %>%
                         step_interact(terms = ~ 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
## Interactions with age:fare
```

Question 2

Fold the **training** data. Use k-fold cross-validation, with $k = 10$.

```
titanic_folds <- vfold_cv(titanic_train, v = 10)
titanic_folds
```

```
## #  10-fold cross-validation
```

```
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [640/72]> Fold01
##  2 <split [640/72]> Fold02
##  3 <split [641/71]> Fold03
##  4 <split [641/71]> Fold04
##  5 <split [641/71]> Fold05
##  6 <split [641/71]> Fold06
##  7 <split [641/71]> Fold07
##  8 <split [641/71]> Fold08
##  9 <split [641/71]> Fold09
## 10 <split [641/71]> Fold10
```

Question 3

In your own words, explain what we are doing in Question 2. What is k-fold cross-validation? Why should we use it, rather than simply fitting and testing models on the entire training set? If we **did** use the entire training set, what resampling method would that be?

In Question 2, we are performing k-folds cross validation which is a form of cross validation that takes multiple subsets of the training data to fit the model on. This is effective because it allows all observations to be input into the model which reduces bias. In essence, there are multiple iterations of validation where taking a certain fold assesses the model while the remaining are used to fit the model and thus, re-sampling.

Question 4

Set up workflows for 3 models:

1. A logistic regression with the `glm` engine;
2. A linear discriminant analysis with the `MASS` engine;
3. A quadratic discriminant analysis with the `MASS` engine.

How many models, total, across all folds, will you be fitting to the data? To answer, think about how many folds there are, and how many models you'll fit to each fold.

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

#LDA
lda_model <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")
lda_wf <- workflow() %>%
  add_model(lda_model) %>%
  add_recipe(titanic_recipe)

#QDA
qda_model <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")
qda_wf <- workflow() %>%
```

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

Since there are 3 primary models, I will be fitting 30 models in total as a result of k=10 for k-folds cross validation.

Question 5

Fit each of the models created in Question 4 to the folded data.

IMPORTANT: *Some models may take a while to run – anywhere from 3 to 10 minutes. You should NOT re-run these models each time you knit. Instead, run them once, using an R script, and store your results; look into the use of loading and saving. You should still include the code to run them when you knit, but set* `eval = FALSE` *in the code chunks.*

```
#Logistic
log_fit <-
  log_wf %>%
  fit_resamples(titanic_folds)

#LDA
lda_fit <-
  lda_wf %>%
  fit_resamples(titanic_folds)

#QDA
qda_fit <-
  qda_wf %>%
  fit_resamples(titanic_folds)
```

Question 6

Use `collect_metrics()` to print the mean and standard errors of the performance metric *accuracy* across all folds for each of the four models.

Decide which of the 3 fitted models has performed the best. Explain why. *(Note: You should consider both the mean accuracy and its standard error.)*

```
#Logistic
collect_metrics(log_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.806    10  0.0126 Preprocessor1_Model1
## 2 roc_auc  binary     0.842    10  0.0120 Preprocessor1_Model1
```

```
#LDA
collect_metrics(lda_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.796    10  0.0123 Preprocessor1_Model1
## 2 roc_auc  binary     0.843    10  0.0105 Preprocessor1_Model1
```

```
#QDA
collect_metrics(qda_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.760    10  0.0152 Preprocessor1_Model1
## 2 roc_auc  binary     0.840    10  0.0112 Preprocessor1_Model1
```

The best model is the Logistic model because it has the highest mean accuracy and a similar standard error to the LDA model which means it is overall evaluated to have a better performance. Additionally, the QDA model has a lower mean accuracy and higher standard error which eliminates it as a high performing model compared to the other two.

Question 7

Now that you've chosen a model, fit your chosen model to the entire training dataset (not to the folds).

```
log_fit <- fit(log_wf, titanic_train)
log_fit %>% tidy()
```

```
## # A tibble: 10 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -4.71      0.623       -7.55  4.26e-14
##  2 age              0.0698    0.0131       5.32  1.02e- 7
##  3 sib_sp           0.298     0.133        2.24  2.49e- 2
##  4 parch            0.116     0.134        0.868 3.85e- 1
##  5 fare             0.0150    0.00963      1.55  1.20e- 1
##  6 pclass_X2        1.18      0.336        3.51  4.50e- 4
##  7 pclass_X3        2.51      0.349        7.20  6.19e-13
##  8 sex_male         2.38      0.279        8.54  1.33e-17
##  9 sex_male_x_fare  0.00786   0.00656      1.20  2.31e- 1
## 10 age_x_fare      -0.000711  0.000260    -2.73  6.32e- 3
```

Question 8

Finally, with your fitted model, use `predict()`, `bind_cols()`, and `accuracy()` to assess your model's performance on the testing data!

Compare your model's testing accuracy to its average accuracy across folds. Describe what you see.

```
predict(log_fit, new_data = titanic_test) %>% bind_cols(titanic_test %>% dplyr::select(survived)) %>%

accuracy(survived, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.810
```

The model's testing accuracy is approximately the same compared to the average accuracy across folds where the average is around .806 versus .810. We can conclude that the model predicts whether or not a passenger survived from the titanic very well and that there may be a low variance.