Homework 4

PSTAT 131/231

Contents

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.

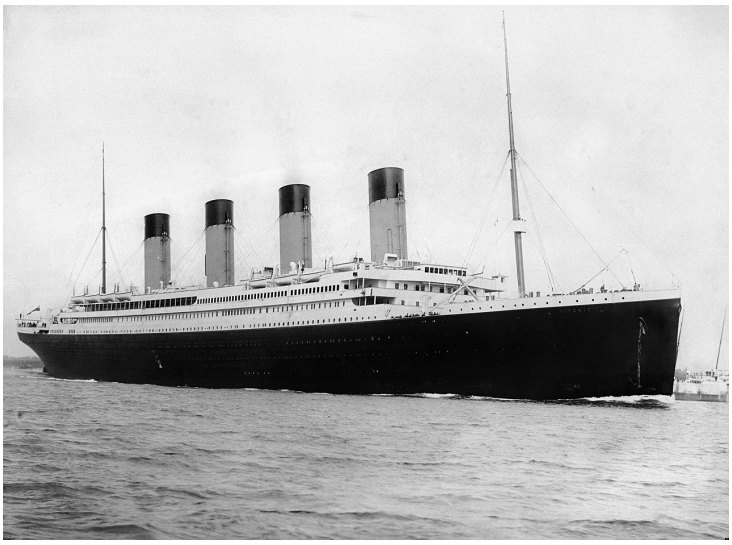

Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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.

```
set.seed(123)
library(tidyverse)
```

```
library(tidymodels)
library(discrim)
titanic <- read.csv('data/titanic.csv')
titanic$survived <-factor(titanic$survived, levels=c("Yes", "No"))
titanic$pclass <-factor(titanic$pclass)
```

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.

```
titanic_split <- initial_split(titanic, prop = 0.7,
                               strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
dim(titanic_train)
```

```
## [1] 623  12
```

```
dim(titanic_test)
```

```
## [1] 268  12
```

```
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)
```

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 [560/63]> Fold01
##  2 <split [560/63]> Fold02
##  3 <split [560/63]> Fold03
##  4 <split [561/62]> Fold04
##  5 <split [561/62]> Fold05
##  6 <split [561/62]> Fold06
##  7 <split [561/62]> Fold07
##  8 <split [561/62]> Fold08
##  9 <split [561/62]> Fold09
## 10 <split [561/62]> 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?

k-fold cross-validation divides the observations into k groups of about equal size, and uses the first group as a validation set and fits the method using the remaining groups. We should use it because it may be less computationally-expensive than other procedures, can be useful if we have limited data, and is generally less biased since all the data is used for both training and testing, rather than splitting the data into training and testing. If we did use the entire training set, that would be the validation set approach.

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 regression model/workflow using glm engine
log_reg <- logistic_reg() %>% set_engine("glm") %>% set_mode("classification")
log_workflow <- workflow() %>% add_model(log_reg) %>% add_recipe(titanic_recipe)


# linear discrimant analysis model + workflow using MASS engine
lda_model <- discrim_linear()  %>% set_mode("classification")%>% set_engine("MASS")
# workflow
lda_workflow <- workflow() %>% add_model(lda_model) %>% add_recipe(titanic_recipe)

# quadratic discriminant analysis model + workflow using MASS engine
qda_model <- discrim_quad()  %>% set_mode("classification")%>% set_engine("MASS")
# workflow
qda_workflow <- workflow() %>% add_model(qda_model) %>% add_recipe(titanic_recipe)
```

We will be fitting 30 models to the data, because for each of the 10 folds, we will be fitting the 3 models to the 9 other folds combined as training data. Hence we fit 3 models for each fold, which ends up as 30 models fitted. (Unless we do not treat the combined folds as one dataset, which would result in 27 fitted models for each fold for a total of 270 fitted models.)

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.*

```
keep_pred <- control_resamples(save_pred = TRUE, save_workflow = TRUE)

log_res <- log_workflow %>% fit_resamples(
            resamples = titanic_folds, control = keep_pred)

lda_res <- lda_workflow %>% fit_resamples(
  resamples = titanic_folds, control = keep_pred)

qda_res <- qda_workflow %>% fit_resamples(
  resamples = titanic_folds, control = keep_pred)
```

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.)*

```
# mean and standard error for accuracy
collect_metrics(log_res)[1, c("mean", "std_err")]
```

```
## # A tibble: 1 x 2
##    mean std_err
##   <dbl>   <dbl>
## 1 0.823  0.0169
```

```
collect_metrics(lda_res)[1, c("mean", "std_err")]
```

```
## # A tibble: 1 x 2
##    mean std_err
##   <dbl>   <dbl>
## 1 0.802  0.0119
```

```
collect_metrics(qda_res)[1, c("mean", "std_err")]
```

```
## # A tibble: 1 x 2
##    mean std_err
##   <dbl>   <dbl>
## 1 0.770  0.0209
```

Looking at the mean accuracy and the standard error, it appears that the logistic regression model performed the best. It had the highest mean for the accuracy metric, which means that it correctly predicted the survival of a passenger of around 0.82 of the dataset, which was the highest of the 3 models. In addition, since its standard error was only around 0.017, that signifies that the model did not perform well by chance. Since the gap between the mean of the log model and the means of the other models is larger than the standard error, it suggests that the difference in mean is a result of performance gap rather than chance.

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_workflow, titanic_train)
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.776
```

The testing accuracy is a decent bit lower than the average accuracy across folds at 0.7761194 versus 0.8233231. This could be because the estimate produced by the validation set approach (our usual training vs testing data technique) can vary a lot depending on which data points are used in the training and testing sets, whereas using the k-fold cross validation does not have very high variance nor bias.