PSTAT 131 - Homework Assignment 4

Akshat Ataliwala (7924145)

April 29, 2022

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.

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.

```
library(tidymodels)
library(tidyverse)
library(ggplot2)
library(corrr)
library(klaR)
library(MASS)
library(discrim)
library(poissonreg)
tidymodels_prefer()
data <- read_csv("data/titanic.csv")
data %>% head(5)
```

```
## # A tibble: 5 x 12
##   passenger_id survived pclass name  sex     age sib_sp parch ticket  fare cabin
##          <dbl> <chr>     <dbl> <chr> <chr> <dbl>  <dbl> <dbl> <chr>  <dbl> <chr>
## 1            1 No            3 Brau~ male     22      1     0 A/5 2~  7.25 <NA>
## 2            2 Yes           1 Cumi~ fema~    38      1     0 PC 17~ 71.3  C85
## 3            3 Yes           3 Heik~ fema~    26      0     0 STON/~  7.92 <NA>
## 4            4 Yes           1 Futr~ fema~    35      1     0 113803 53.1  C123
## 5            5 No            3 Alle~ male     35      0     0 373450  8.05 <NA>
## # ... with 1 more variable: embarked <chr>
```

```
data$survived <- as.factor(data$survived)
data$survived <- relevel(data$survived, "Yes")
data$pclass <- as.factor(data$pclass)
data %>% head(5)
```

```
## # A tibble: 5 x 12
##   passenger_id survived pclass name  sex     age sib_sp parch ticket  fare cabin
##          <dbl> <fct>    <fct>  <chr> <chr> <dbl>  <dbl> <dbl> <chr>  <dbl> <chr>
## 1            1 No       3      Brau~ male     22      1     0 A/5 2~  7.25 <NA>
## 2            2 Yes      1      Cumi~ fema~    38      1     0 PC 17~ 71.3  C85
## 3            3 Yes      3      Heik~ fema~    26      0     0 STON/~  7.92 <NA>
## 4            4 Yes      1      Futr~ fema~    35      1     0 113803 53.1  C123
## 5            5 No       3      Alle~ male     35      0     0 373450  8.05 <NA>
## # ... with 1 more variable: embarked <chr>
```

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(3478)

data_split <- initial_split(data,
                            prop = 0.8,
                            strata = survived)

train <- training(data_split)
test <- testing(data_split)
```

```
dim(data)
```

```
## [1] 891  12
```

```
0.8 * nrow(data)
```

```
## [1] 712.8
```

```
dim(train)
```

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

```
dim(test)
```

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

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

```
folds <- vfold_cv(data = train,
                  v = 10,
                  strata = survived)
folds
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [640/72]> Fold01
##  2 <split [640/72]> Fold02
##  3 <split [640/72]> 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 [642/70]> Fold10
```

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 is a technique used to avoid over fitting, and is used during the model training process. After we split our total data into training and testing, we want to evaluate the performance of our trained model, but showing it the test set would essentially be cheating the model. Instead, we further subdivide the training set into k folds, where each fold is essentially a training set and a validation set that is a subset of the training set. Each fold is different but from the same training set, and therefore we can use the validation set in each fold like a test set, and evaluate different hyperparameters and tune our model before we show the model the final test set. This will improve generalize-ability because we get to evaluate our model before it has seen the test set, because if we tune the model based on testing performance we can over fit to the test set. If we use the entire training set to resample, we would be bootstrapping.

4. Set up workflows for 3 models:

A logistic regression with the glm engine; A linear discriminant analysis with the MASS engine; 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.

```
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare,
                         data = train) %>%
  step_impute_linear(age, impute_with = imp_vars(all_predictors())) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ sex_male:fare) %>%
  step_interact(terms = ~ age:fare)
```

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

log_wflow <- workflow() %>%
  add_recipe(titanic_recipe) %>%
  add_model(log_reg)
```

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

lda_wflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)
```

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

qda_wflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)
```

In total, we are fitting 30 models, 10 (1 sub-model / fold) for each of the 3 different models (logistic regression, lda, qda).

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.

```
degree_grid <- grid_regular(degree(range = c(1, 10)), levels = 10)
degree_grid
```

```
## # A tibble: 10 x 1
##    degree
##     <dbl>
##  1      1
##  2      2
##  3      3
##  4      4
##  5      5
##  6      6
##  7      7
##  8      8
##  9      9
## 10     10
```

```
tune_res_log_reg <- tune_grid(
  object = log_wflow,
  resamples = folds,
  grid = degree_grid)

tune_res_lda <- tune_grid(
  object = lda_wflow,
  resamples = folds,
```

```
  grid = degree_grid)

tune_res_qda <- tune_grid(
  object = qda_wflow,
  resamples = folds,
  grid = degree_grid)
```

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.)

```
# Logistics Regression Metrics
collect_metrics(tune_res_log_reg)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.798    10  0.0114 Preprocessor1_Model1
## 2 roc_auc  binary     0.850    10  0.0152 Preprocessor1_Model1
```

```
# Linear Discriminant Analysis Metrics
collect_metrics(tune_res_lda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.794    10  0.0111 Preprocessor1_Model1
## 2 roc_auc  binary     0.851    10  0.0154 Preprocessor1_Model1
```

```
# Quadratic Discriminant Analysis Metrics
collect_metrics(tune_res_qda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.777    10  0.0184 Preprocessor1_Model1
## 2 roc_auc  binary     0.834    10  0.0208 Preprocessor1_Model1
```

Looking at the models' performance after k-fold cross validation, its clear the qda is worse than lda and logistic regression across the board. The difference between logistic regression and lda is more subtle, as logisitic regression has a higher accuracy by 0.0042. However, the lda model has a lower standard error by 0.0029 and a higher roc_auc score by 0.006, so I will be choosing the lda model since it might perform slightly better with new data given the smaller error and larger ROC.

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

```
lda_fit <- fit(lda_wflow, train)
```

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.

```
lda_predict <- predict(lda_fit, new_data = test)
predictions <- bind_cols(test %>% select(survived), lda_predict)
predictions
```

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

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.799
```

The lda model actually performed slightly better on the test set (0.7989) than it did on average with k-fold CV (0.7937). This is pretty good and means our model is generalizable in some sense, but it could probably be even better if we tuned various hyperparameters when cross validating and selecting a model because the accuracies are so similar (essentially meaning that the current cross validation didn't improve accuracy that much).