PSTAT131HW4

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.

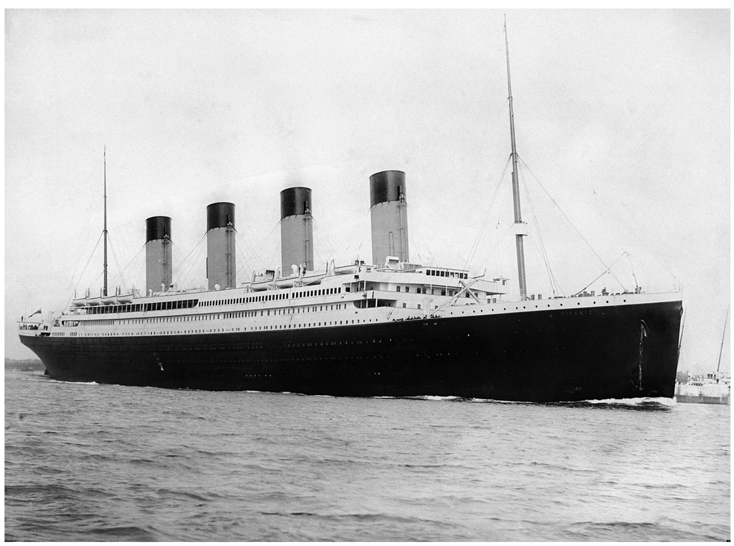

Figure 1: Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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.

Loading Packages

```r
set.seed(123)

library(tidymodels)
library(tidyverse)
library(dplyr)
library(corrr)
library(klaR)
library(MASS)
library(discrim)
library(poissonreg)

tidymodels_prefer()

titanic <- read_csv("data/titanic.csv")
titanic$survived <- factor(titanic$survived)
titanic$survived <- relevel(titanic$survived, "Yes")
titanic$pclass <- factor(titanic$pclass)
head(titanic)
```

```
## # A tibble: 6 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>
## 6            6 No       3      Mora~ male     NA      0     0 330877  8.46 <NA>
## # ... with 1 more variable: embarked <chr>
```

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.

```r
titanic_split <- initial_split(titanic, prop = 0.7, strata = survived)


# The training data sets can be materialized using the training() functions
titanic_train <- training(titanic_split)
# The testing data sets can be materialized using the testing() functions
titanic_test <- testing(titanic_split)

# use dim() to verify that the correct number of observations are now in each data set
dim(titanic_train)
```

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

```
dim(titanic_test)
```

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

Question 2

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

```
poly_tuned_rec <-
  recipe(survived ~ pclass+sex+age+sib_sp+parch+fare, data = titanic_train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with("sex"):fare + age:fare)

poly_tuned_rec
```

```
## 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
```

```
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

poly_tuned_wf <- workflow() %>%
  add_recipe(poly_tuned_rec) %>%
  add_model(lm_spec)

Auto_folds <- vfold_cv(titanic_train, v = 10)
Auto_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?

In order to solve the shortcomings of simple cross-validation, I think we need use k-fold cross-validation. Divide the dataset into different k segments,select one segment as the validation set in each cycle of k times, and use all the remaining segments as the training set. It may be less computationally-expensive than other procedures, can be useful if we have limited data. Randomly initialize the weights to train the model.

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.

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

log_workflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(poly_tuned_rec)

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

lda_workflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(poly_tuned_rec)

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

qda_workflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(poly_tuned_rec)
```

There are a total of 30 models, 10 for each engine, corresponding to 10-fold. For each of the 10 folds, we will be fitting the 3 models to the 9 other folds combined as training data. Ends up as 30 models fitted.

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.*

```
log_fit <- log_workflow %>%
  fit_resamples(resamples = Auto_folds)

lda_fit <- lda_workflow %>%
  fit_resamples(resamples = Auto_folds)

qda_fit <- qda_workflow %>%
  fit_resamples(resamples = Auto_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.)*

```
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     0.817    10  0.0141 Preprocessor1_Model1
## 2 roc_auc  binary     0.850    10  0.0141 Preprocessor1_Model1
```

The logistic regression has an accuracy of 0.824 and a standard error of 0.0157.

```
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     0.798    10  0.0111 Preprocessor1_Model1
## 2 roc_auc  binary     0.851    10  0.0141 Preprocessor1_Model1
```

The logistic regression has an accuracy of 0.804 and a standard error of 0.0192.

```
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     0.774    10  0.0104 Preprocessor1_Model1
## 2 roc_auc  binary     0.839    10  0.0157 Preprocessor1_Model1
```

The logistic regression has an accuracy of 0.766 and a standard error of 0.0209.

So, the logistic regression model performed the best because it had the highest accuracy and smallest standard error.

Question 7

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

```
final_fit <- fit(log_workflow, titanic_train)

final_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##     (Intercept)              age           sib_sp            parch
##       -4.479635         0.065303         0.591529         0.131192
##            fare        pclass_X2        pclass_X3         sex_male
##       -0.007895         0.932520         2.216490         2.530135
## sex_male_x_fare       fare_x_age
##        0.014305        -0.000260
##
## Degrees of Freedom: 622 Total (i.e. Null);  613 Residual
## Null Deviance:       829.6
## Residual Deviance: 522.8     AIC: 542.8
```

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(final_fit, new_data = titanic_test) %>% bind_cols(titanic_test%>% dplyr::select(survived)) %>% a
```

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

we can be seen that the logistic regression model is better than the test set. Using the k-fold cross validation does not have very high variance nor bias.