PSTAT131HW4

PSTAT 131/231

Table of Contents

## Resampling

For this assignment, we will continue working with part of a [Kaggle data set](https://www.kaggle.com/c/titanic/overview) 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](https://en.wikipedia.org/wiki/Titanic).

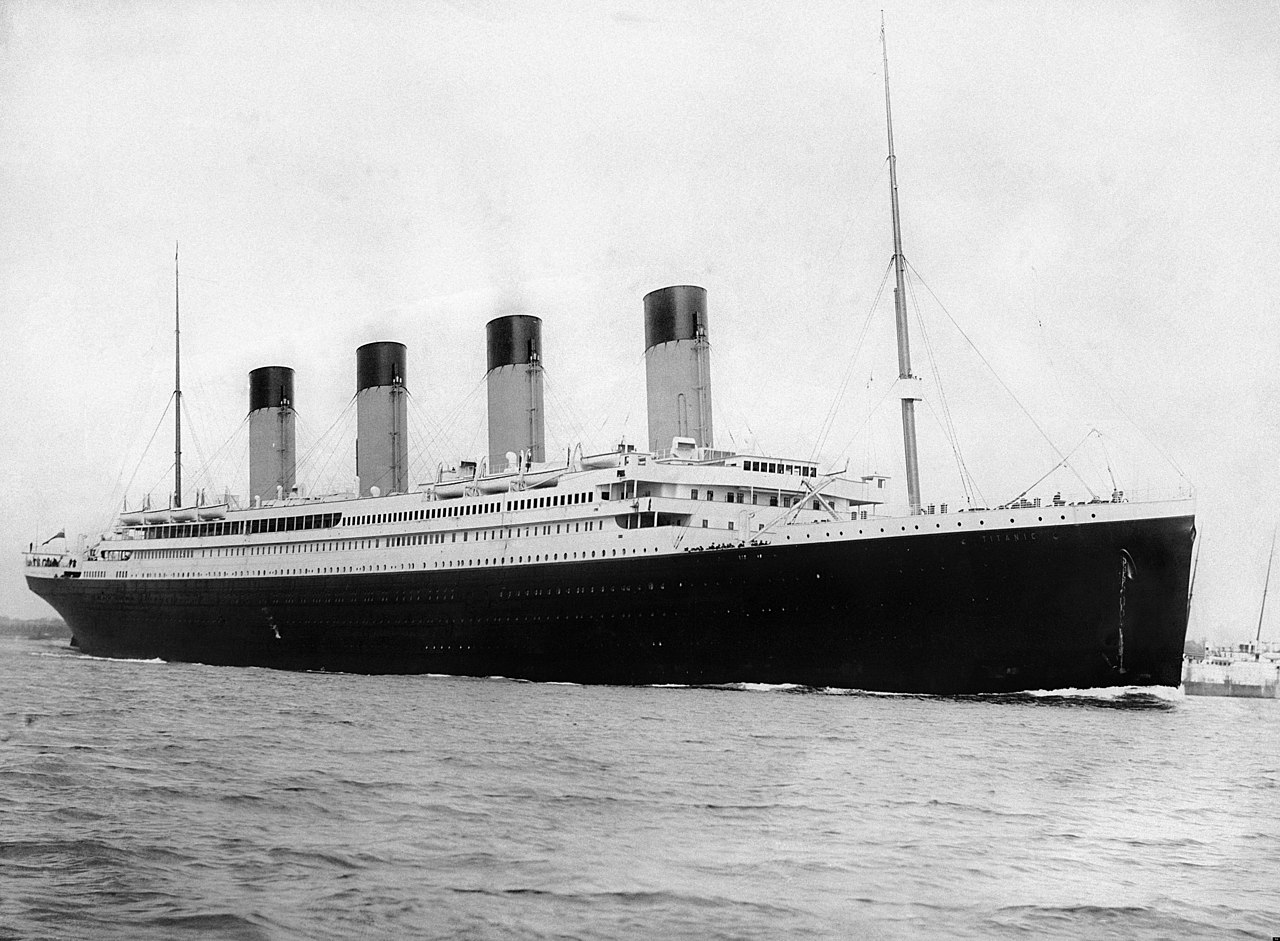


Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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

set.seed(131)  
  
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.

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 .

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*](https://www.r-bloggers.com/2017/04/load-save-and-rda-files/)*. 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 0.809 10 0.0146 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.869 10 0.0118 Preprocessor1\_Model1

The logistic regression has an accuracy of 0.8089350 and a standard error of 0.0145.

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 0.803 10 0.0129 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.868 10 0.0127 Preprocessor1\_Model1

The logistic regression has an accuracy of 0.7995136 and a standard error of 0.01826695.

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 0.783 10 0.0144 Preprocessor1\_Model1  
## 2 roc\_auc binary 0.860 10 0.0155 Preprocessor1\_Model1

The logistic regression has an accuracy of 0.7850486 and a standard error of 0.01310639.

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   
## -4.8485149 0.0638445 0.4867294 0.2053968   
## fare pclass\_X2 pclass\_X3 sex\_male   
## 0.0037451 1.3341760 2.5443384 2.7415709   
## sex\_male\_x\_fare fare\_x\_age   
## 0.0053962 -0.0003282   
##   
## Degrees of Freedom: 622 Total (i.e. Null); 613 Residual  
## Null Deviance: 829.6   
## Residual Deviance: 518.6 AIC: 538.6

### 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)) %>% accuracy(truth=survived, estimate = .pred\_class)

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

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.