PSTAT131HW04

Yifei Zhang

2022-04-28

Contents


```
library(ggplot2)
library(tidyverse)
library(tidymodels)
library(corrplot)
library(ggthemes)
library(discrim)
library(poissonreg)
library(corrr)
library(klaR)
library(ISLR)
library(ISLR2)
library(purrr)
tidymodels_prefer()
titanic <- read.csv("titanic.csv")
```

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.

We get 712 and 179 rows for the corresponding training and testing sets, which are approximately 80 and 20 percent of 891. The number of observations are appropriate.

```r
set.seed(1010)

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

titanic_train <- training(titanic_split)

titanic_test <- testing(titanic_split)

count(titanic_train, "passenger_id") # see how many rows
```

```
##   "passenger_id"   n
## 1   passenger_id 712
```

```r
count(titanic_test, "passenger_id") # see how many rows
```

```
##   "passenger_id"   n
## 1   passenger_id 179
```

```r
# or use dim to see the dimension and we can see we have appropriate numbers of rows for each set

dim(titanic_train)
```

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

```r
dim(titanic_test)
```

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

Question 2

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

```r
titanic_folds <- vfold_cv(titanic_train, v = 10, strata = 'survived')

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 = ~ sex_male : fare) %>%
  step_interact(terms = ~ age : fare)

titanic_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
```

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?

A K fold cross-validation is a form of cross-validation that uses the technique of spiting the data into k subsets/ folds, and we use use subsets as test sets, and training sets, more specifically (K-1) folds as the training set and 1 fold as the validation set, basically doing what we normally do but in smaller scale and more times. We use it instead of the fitting and testing models on the entire training set method is because this is more efficient, and it address the problem with the holdout method, by using the K fold cross-validation we are not making our result depend on our selection of training and testing sets. If we did use the entire training set, the resampling method would be the Validation Set Approach method.

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.

I will be fitting 30 models in total across all folds, 3 for each fold, and for 10 folds that's 30.

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


lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")
lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)

qda_mod <- discrim_quad() %>%
```

```
  set_mode("classification") %>%
  set_engine("MASS")
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)
```

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_wkflow %>%
  fit_resamples(titanic_folds)
lda_fit <- lda_wkflow %>%
  fit_resamples(titanic_folds)
qda_fit <- qda_wkflow %>%
  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.)* The model that has performed the best is the lda model because although it has a lightly lower mean accuracy compared to the log model, its standard error is considerably smaller than the log model which is good. We make less mitakes with about the same accuracy.

```
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     0.794    10  0.0146 Preprocessor1_Model1
## 2 roc_auc  binary     0.838    10  0.0116 Preprocessor1_Model1
```

```
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     0.784    10  0.0124 Preprocessor1_Model1
## 2 roc_auc  binary     0.840    10  0.0116 Preprocessor1_Model1
```

```
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     0.770    10  0.0137 Preprocessor1_Model1
## 2 roc_auc  binary     0.841    10  0.0144 Preprocessor1_Model1
```

Question 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_wkflow, titanic_train)
lda_fit_pred <- predict(lda_fit, new_data = titanic_train, type = "prob")
lda_acc <- augment(lda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc <- augment(lda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.794
```

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.

The accuracy on the testing data is slightly higher than the average training data, and this is probably because the approach we take is using a mechanistic model that has lower variance.

```
lda_test_fit <- fit(lda_wkflow, titanic_test)
titanic_testing_pred <-
  predict(lda_test_fit, titanic_test) %>%
  bind_cols(predict(lda_test_fit, titanic_test, type = "prob")) %>%
  bind_cols(titanic_test %>% select(survived))

titanic_testing_pred %>%
  accuracy(truth = survived, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.838
```

```
lda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.794
```

Code that did not workout/use

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

titanic_wf <- workflow() %>% add_recipe(titanic_recipe) %>% add_model(log_reg)

titanic_fit_rs <- titanic_wf %>% fit_resamples(titanic_folds)

titanic_fit_rs collect_metrics(titanic_fit_rs)

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

titanic_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(terms = ~ sex_male : fare) %>% step_interact(terms = ~ age : fare) %>% step_poly(pclass, sex, age, sib_sp, parch, fare, degree = tune())

titanic_tuned_wf <- workflow() %>% add_recipe(titanic_tuned_rec) %>% add_model(lm_spec)

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

titanic_rec <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train) %>% step_impute_linear(age) %>% step_dummy(all_nominal_predictors()) %>% step_interact(terms = ~ sex_male : fare) %>% step_interact(terms = ~ age : fare) %>% step_poly(pclass, sex, age, sib_sp, parch, fare, degree = 2)

titanic_wf <- workflow() %>% add_recipe(titanic_rec) %>% add_model(lm_spec)

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

tune_res <- tune_grid(object = titanic_tuned_wf, resamples = titanic_folds, grid = degree_grid)

autoplot(tune_res)

titanic_tuned_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 = ~ sex_male : fare) %>% step_interact(terms = ~ age : fare) %>% step_poly(pclass, sex, age, sib_sp, parch, fare, degree = tune())

titanic_tuned_recipe

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

titanic_tuned_wf <- workflow() %>% add_recipe(titanic_tuned_recipe) %>% add_model(lm_spec)

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

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

tune_res <- tune_grid(object = titanic_tuned_wf, resamples = titanic_folds, grid = degree_grid)

kfolds <- map_df(titanic_folds$splits, pluck, 'titanic_train') titanic_folds

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

tune_res <- tune_grid(object = titanic_tuned_wf, resamples = titanic_folds, grid = degree_grid)

collect_metrics(tune_res)

show_best(tune_res, metric = "rmse")

select_by_one_std_err(tune_res, degree, metric = "rmse")

best_degree <- select_by_one_std_err(tune_res, degree, metric = "rmse")

final_wf <- finalize_workflow(titanic_tuned_wf, best_degree)

final_fit <- fit(titanic_tuned_wf, titanic_train) titanic_fit <- lm_spec %>% fit(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train)

titanic_testing_pred %>%
roc_auc(truth = survived, .pred_PS)

titanic_testing_pred <- predict(titanic_fit, titanic_test) %>% bind_cols(predict(titanic_fit, titanicl_test, type = “prob”)) %>% bind_cols(titanic_test %>% select(survived))

titanic_testing_pred %>%
accuracy(truth = survived, .pred_survived)

collect_metrics(tune_res) show_best(tune_res, metric = “rmse”) select_by_one_std_err(tune_res, degree, metric = “rmse”) best_degree <- select_by_one_std_err(tune_res, degree, metric = “rmse”) final_wf <- finalize_workflow(titanic_tuned_wf, best_degree) final_wf final_fit <- fit(final_wf, titanic_train) final_fit augment(final_fit, new_data = titanic_test) %>% rmse(truth = survived, estimate = .pred)

titanic $ survived <-ifelse(titanic $ survived == “Yes”, 1, 0) titanic $ pclass <- as.numeric(titanic $ pclass) titanic $ sex <- ifelse(titanic$sex == “male”, 1, 0) titanic $ sib_sp <- as.numeric(titanic $ sib_sp) titanic $ parch <- as.numeric(titanic $ parch)