PSTAT 131 HW 4

Raymond Lee

2022-04-24

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.2.0 --

## v broom        0.8.0     v rsample      0.1.1
## v dials        0.1.1     v tune         0.2.0
## v infer        1.0.0     v workflows    0.2.6
## v modeldata    0.1.1     v workflowsets 0.2.1
## v parsnip      0.2.1     v yardstick    0.0.9
## v recipes      0.2.0

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(discrim)
```

```
##
## Attaching package: 'discrim'
```

```
## The following object is masked from 'package:dials':
##
##     smoothness
```

```
library(klaR)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
tidymodels_prefer()

titanic = read.csv("titanic.csv")
titanic$survived = factor(titanic$survived, levels = c("Yes", "No"))
titanic$pclass = factor(titanic$pclass)

set.seed(1114)
```

1.

```
titanic_split = initial_split(titanic, prop = .70, strata = survived)
titanic_train = training(titanic_split)
titanic_test = testing(titanic_split)

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(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare) %>%
  step_nzv(all_predictors())
```

2.

```
library(tune)

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
```

3. K-fold cross-validation is when we split the training set into folds to use in assessing model performance and selecting the best model. Less data are used to figure out the best model, allowing us to set aside enough data for the testing set. This would be especially useful when we have limited data available. Using the the entire training set would be the validation set approach.

4.

```
log_reg = logistic_reg() %>%
  set_engine("glm")

log_workflow = workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)
```

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

lda_workflow = workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)
```

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

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

There are 10 folds and 3 models. Therefore, 30 models will be fitted across all folds

5.

```
log_fit_folds = log_workflow %>%
  fit_resamples(titanic_folds)
```

```
lda_fit_folds = lda_workflow %>%
  fit_resamples(titanic_folds)
```

```
qda_fit_folds = qda_workflow %>%
  fit_resamples(titanic_folds)
```

6.

```
collect_metrics(log_fit_folds)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.801    10  0.0152 Preprocessor1_Model1
## 2 roc_auc  binary     0.848    10  0.0206 Preprocessor1_Model1
```

```
collect_metrics(lda_fit_folds)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.780    10  0.0171 Preprocessor1_Model1
## 2 roc_auc  binary     0.847    10  0.0218 Preprocessor1_Model1
```

```
collect_metrics(qda_fit_folds)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.771    10  0.0155 Preprocessor1_Model1
## 2 roc_auc  binary     0.823    10  0.0227 Preprocessor1_Model1
```

The logistic regression model performed the best because it has the highest mean accuracy and lowest standard error for the accuracy.

7.

```
log_fit = fit(log_workflow, titanic_train)
```

8.

```
log_predict = predict(log_fit, new_data = titanic_test, type = "prob")
log_predict
```

```
## # A tibble: 268 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.0876    0.912
##  2    0.0778    0.922
##  3    0.312     0.688
##  4    0.737     0.263
##  5    0.810     0.190
##  6    0.149     0.851
##  7    0.0451    0.955
##  8    0.637     0.363
##  9    0.236     0.764
## 10    0.638     0.362
## # ... with 258 more rows
```

```
log_test_acc = augment(log_fit, new_data=titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_test_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.817
```

The testing accuracy is slightly higher than the average accuracy across folds, but they are quite close. The average accuracy across fields appears to be a good indicator of the testing accuracy.