HW4

Ben Hertzberg

2022-11-03

```
titanic <- read.csv('./data/titanic.csv')
library(tidymodels)
library(tidyverse)
library(ggplot2)
library(discrim)
library(poissonreg)
library(corrr)
library(klaR)
tidymodels_prefer()
```

```
titanic$survived <- as.factor(titanic$survived)
titanic$pclass <- as.factor(titanic$pclass)
```

Question 1

```
set.seed(619)

ttnc_split <- initial_split(titanic, prop = 0.80, strata = survived)
ttnc_train <- training(ttnc_split)
ttnc_test <- testing(ttnc_split)
```

```
dim(ttnc_train)
```

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

```
dim(ttnc_test)
```

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

712 training observations, 179 testing observations, which is about 80% training and 20% testing.

```
ttnc_rec <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = ttnc_train) %>%

  step_impute_linear() %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_interact(terms = ~starts_with("sex"):fare) %>%
  step_interact(terms = ~age:fare)
```

Question 2

```
ttnc_folds <- vfold_cv(ttnc_train, v = 10)
ttnc_folds
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [640/72]> Fold01
##  2 <split [640/72]> Fold02
##  3 <split [641/71]> 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 [641/71]> Fold10
```

Question 3

In Q2, we randomly assigned observations from the training data into 10 roughly equal subsets to be used in k-fold cross-validation. K-fold CV offers a way to check how well our model is doing before applying it on the testing data. This is done by training a model on every fold except one, and then effectively testing on that last fold. K total models are trained, each one using a different fold to test on and training on all the others. This offers the benefit of checking how a model works on data it was not trained on before actually applying it to the test data. If we fit and train models on the entire training set right away, we have to use the testing data to do this check, and then we can't make adjustments to the model without risking data leakage. If we resampled from the entire training set, we would be using bootstrapping.

Question 4

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

log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(ttnc_rec)

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

lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(ttnc_rec)

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

```
  set_engine('MASS') %>%
  set_mode('classification')

qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(ttnc_rec)
```

I am fitting 3 models to 10 folds, which leads to a total of 30 models.

Question 5

```
log_fit_k <- fit_resamples(log_wkflow, ttnc_folds)
lda_fit_k <- fit_resamples(lda_wkflow, ttnc_folds)
qda_fit_k <- fit_resamples(qda_wkflow, ttnc_folds)
```

```
write_rds(log_fit_k, file = 'SavedModels/log_res.rds')
write_rds(lda_fit_k, file = 'SavedModels/lda_res.rds')
write_rds(qda_fit_k, file = 'SavedModels/qda_res.rds')
```

```
log_res <- read_rds(file = 'SavedModels/log_res.rds')
lda_res <- read_rds(file = 'SavedModels/lda_res.rds')
qda_res <- read_rds(file = 'SavedModels/qda_res.rds')
```

Question 6

```
collect_metrics(log_res)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.800    10  0.0206 Preprocessor1_Model1
## 2 roc_auc  binary     0.857    10  0.0187 Preprocessor1_Model1
```

```
collect_metrics(lda_res)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.793    10  0.0206 Preprocessor1_Model1
## 2 roc_auc  binary     0.857    10  0.0190 Preprocessor1_Model1
```

```
collect_metrics(qda_res)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.759    10  0.0174 Preprocessor1_Model1
## 2 roc_auc  binary     0.847    10  0.0169 Preprocessor1_Model1
```

QDA performed the worst. LDA and Logistic Regression performed very similarly, but logistic regression has a slightly higher mean accuracy. The standard error of the two is almost identical, but the LDA's is barely lower. The difference in mean accuracy was larger than the difference in standard error, so I believe the logistic regression model performed the best.

Question 7

```
log_fit_final <- fit(log_wkflow, ttnc_train)
```

Question 8

```
predict(log_fit_final, new_data = ttnc_test, type = 'prob')
```

```
## # A tibble: 179 x 2
##    .pred_No .pred_Yes
##       <dbl>     <dbl>
##  1   0.0725     0.927
##  2   0.851      0.149
##  3   0.421      0.579
##  4   0.511      0.489
##  5   0.236      0.764
##  6   0.369      0.631
##  7  NA         NA
##  8  NA         NA
##  9   0.851      0.149
## 10   0.0990     0.901
## # ... with 169 more rows
```

```
log_acc <- augment(log_fit_final, new_data = ttnc_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.755
```

The LDA model test accuracy is 75.52%, lower than the average across folds which was 79.97%. The accuracy of a model on new data often is slightly lower, so these values are reasonable.