131HW4

Scott Shang (8458655)

May 10, 2022

Question1

```r
library("tidyverse")
```

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## 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()
```

```r
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()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```r
library("dplyr")
library("yardstick")
library(readr)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(discrim)
```

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

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

```
library(poissonreg)
library(corrr)
library(klaR)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(knitr)
library(MASS)
library(ggplot2)
tt=read_csv('titanic.csv')
```

```
## Rows: 891 Columns: 12

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): survived, name, sex, ticket, cabin, embarked
## dbl (6): passenger_id, pclass, age, sib_sp, parch, fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tt$survived=factor(tt$survived)
tt$survived=relevel(tt$survived,"Yes")
tt$pclass=factor(tt$pclass)
head(tt)
```

```
## # 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>
```

```
set.seed(1234)
tt_split=initial_split(tt,prop=0.80,strata=survived)
train=training(tt_split)
test=testing(tt_split)
```

```
nrow(tt)
```

```
## [1] 891
```

```
nrow(train)
```

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

```
nrow(test)
```

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

```
nrow(train)+nrow(test)
```

```
## [1] 891
```

```
nrow(train)/nrow(tt)
```

```
## [1] 0.7991
```

```
nrow(test)/nrow(tt)
```

```
## [1] 0.2009
```

The training and testing data sets have the appropriate number of obs that we set above.

Question2

```r
fold=vfold_cv(train,v=10)
fold
```

```
## #  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
```

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

Question3

In question 2, we randomly split the training data sets into 10 folds of roughly equal size. k-fold cross-validation is a method that apply our learned model to different groups of data so that we can learn our model using limited observations in the training data set. If we simply fitting and testing models on the entire training set, it could be a waste of data. If we did use the entire training set, that will be bootstrap.

Question4

```r
tt_recipe=recipe(survived~pclass+sex+age+sib_sp+parch+fare,data=train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_impute_linear(age,impute_with=imp_vars(all_predictors())) %>%
  step_interact(terms=~sex_male:fare+age:fare)

log_reg=logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_wf=workflow() %>%
  add_model(log_reg) %>%
  add_recipe(tt_recipe)

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

lda_wf=workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(tt_recipe)

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

```
qda_wf=workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(tt_recipe)
```

There are 3 models, fitted with 10 folds each. In total, there will be 30 models.

Question5

```
log_fold=log_wf %>%
  fit_resamples(fold)

lda_fold=lda_wf %>%
  fit_resamples(fold)

qda_fold=qda_wf %>%
  fit_resamples(fold)
```

Question6

```
collect_metrics(log_fold)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.806    10  0.0195 Preprocessor1_Model1
## 2 roc_auc  binary     0.850    10  0.0188 Preprocessor1_Model1
```

```
collect_metrics(lda_fold)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.798    10  0.0193 Preprocessor1_Model1
## 2 roc_auc  binary     0.852    10  0.0181 Preprocessor1_Model1
```

```
collect_metrics(qda_fold)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.785    10  0.0180 Preprocessor1_Model1
## 2 roc_auc  binary     0.845    10  0.0223 Preprocessor1_Model1
```

As we can see, Logistic Regression has the best mean accuracy, and Quadratic Discriminant Analysis has the smallest standard error. Overall speaking, Logistic Regression is the best of the three, with best mean accuracy and acceptable standard error.

Question7

```
fitted=fit(log_wf,train)
acc=augment(fitted,new_data=train) %>%
  accuracy(truth=survived,estimate=.pred_class)
acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.819
```

Our model is pretty accurate!

Question8

```
tt_pf=predict(fitted,new_data=test)
tt_pf=bind_cols(tt_pf)
tt_pf=augment(fitted,new_data=test) %>%
  accuracy(truth=survived,estimate=.pred_class)
tt_pf
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.782
```

Testing accuracy is 0.78 which slightly lower than what we got across training folds. My intuition is that our model might overfitted the training folds a little bit, but overall speaking, it's acceptable.