PSTAT131_HW4

Tao Wang

2022-05-02

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.

```
# prepare
library(klaR) # for naive bayes
```

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

```
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()
## x dplyr::select() masks MASS::select()
```

```
library(tidymodels)
```

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

```
## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      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 dplyr::select()   masks MASS::select()
## x yardstick::spec() masks readr::spec()
```

```
## x recipes::step()   masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(discrim)
```

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

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

```
library(poissonreg)
library(corrr)
tidymodels_prefer()

set.seed(1234) # can be any number
```

```
# Load the data from `data/titanic.csv` into *R*
# change `survived` and `pclass`  to factors.
# reorder the factor survived` so that *"Yes"* is the first level
titanic <- read.csv(file = "data/titanic.csv") %>%
  mutate(survived = factor(survived,
                           levels = c("Yes", "No")),
         pclass = factor(pclass))
head(titanic)
```

```
##   passenger_id survived pclass
## 1            1       No      3
## 2            2      Yes      1
## 3            3      Yes      3
## 4            4      Yes      1
## 5            5       No      3
## 6            6       No      3
##                                                 name    sex age sib_sp parch
## 1                             Braund, Mr. Owen Harris   male  22      1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38      1     0
## 3                              Heikkinen, Miss. Laina female  26      0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35      1     0
## 5                            Allen, Mr. William Henry   male  35      0     0
## 6                                    Moran, Mr. James   male  NA      0     0
##             ticket    fare cabin embarked
## 1        A/5 21171  7.2500  <NA>        S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

```
# check whether *"Yes"* is the first level
levels(titanic$survived)
```

```
## [1] "Yes" "No"
```

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.

```
# Split the data, stratifying on the outcome variable, `survived.`
titanic_split <- initial_split(titanic, prop = 0.80,
                               strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

# Verify that the training and testing data sets have the appropriate number of observations.
dim(titanic_train) # verify the training data sets
```

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

```
dim(titanic_test) # Verify the testing data sets
```

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

- Each dataset has approximately the right number of observations;
- For the training data set, 712 is almost exactly 80% of the full data set, which contains 891 observations. So, the training data set has approximately the right number of observations.
- For the testing data set, 179 is almost exactly 20% of the full data set, which contains 891 observations. So, the testing data set has approximately the right number of observations.

Question 2

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

```
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 [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 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?

Answer

- 1. In Qurstion2, we are randomly dividing the training data into 10 groups (or folds) of (roughly) equal sizes.

- 2. According to the textbook, k-fold cross-validation is an approach that involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, MSE_1, MSE_2,...,MSE_k. The k-fold CV estimate is computed by averaging these value. (Page 203)
- 3. According to the TA, if we simply fit and test models on the entire training set, it may cause overfitting. To avoid causing overfitting, we should use k-fold cross-validation.
- 4. Bootstrap

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.

```
# Create a recipe for this dataset **identical** to the recipe you used in Homework 3.
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 = ~ starts_with('sex'):fare)%>%
step_interact(terms = ~ age:fare)
```

```
# 1. A logistic regression with the `glm` engine;
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)
```

```
# 2. A linear discriminant analysis with the `MASS` engine;
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)
```

```
# 3. A quadratic discriminant analysis with the `MASS` engine.
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

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

- How many models, total, across all folds, will you be fitting to the data?
- Answer: Since we have 3 different types of models for all 10 folds, then we will fit 30 models in total to the data.

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.*

- how can we fit our model to the floded data ?
- do we need to use keep_pred <- control_resamples(save_pred = TRUE, save_workflow = TRUE)

```
log_fit <- fit_resamples(log_wkflow, titanic_folds)
```

```
lda_fit <- fit_resamples(lda_wkflow, titanic_folds)
```

```
qda_fit <- fit_resamples(qda_wkflow, 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 3 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     0.806    10  0.0195 Preprocessor1_Model1
## 2 roc_auc  binary     0.850    10  0.0188 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     0.798    10  0.0193 Preprocessor1_Model1
## 2 roc_auc  binary     0.852    10  0.0181 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     0.785    10  0.0180 Preprocessor1_Model1
## 2 roc_auc  binary     0.845    10  0.0223 Preprocessor1_Model1
```

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.)***

- By comparing these three models, we can find that the logistic regression have the highest mean accuracy and lowest standard error. So, we can conclude that the logistic regression has performed the best.

Question 7

Now that you've chosen a model, fit your chosen model to the entire training dataset (not to the folds).

```
lm_fit <- fit(log_wkflow, titanic_train)
```

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.

```
# 1. use `predict()`, `bind_cols()`, and `accuracy()` to assess your model's performance
# on the testing data!
log_acc <- predict(lm_fit, new_data = titanic_test, type = "class") %>%
  bind_cols(titanic_test %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc # it tells us the model's testing accuracy
```

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

```
# Recall, the model's average accuracy across folds
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     0.806    10  0.0195 Preprocessor1_Model1
## 2 roc_auc  binary     0.850    10  0.0188 Preprocessor1_Model1
```

- My model's testing accuracy is 0.782
- My model's average accuracy across folds is 0.806
- We can see that the model's testing accuracy is less than the model's average accuracy across folds.