Untitled

Yiting Zhang

2022-05-02

```
# load the libraries
library(tidymodels)
library(ggplot2)
library(discrim)
library(corrr)
library(klaR)
library(caret)
library(ggplot2)
library(tidyverse)
library(corrplot)
library(ggthemes)
library(cli)
library(recipes)
library(pROC)
library(yardstick)
library(MASS)
library(poissonreg)
library(naivebayes)
tidymodels_prefer()

# load and factor data.
titanic <- read.csv(file = 'titanic.csv')
titanic$survived <- factor(titanic$survived, levels = c("Yes","No"))
titanic$pclass <- factor(titanic$pclass)
```

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.

```
set.seed(100)
titanic_split <- initial_split(titanic, prop = 0.80,
                               strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)



data<-dim(titanic)[1]
train<-dim(titanic_train)[1]
```

```
test<-dim(titanic_test)[1]
train/data
```

```
## [1] 0.7991021
```

```
test/data
```

```
## [1] 0.2008979
```

```
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 + age:fare)
```

we can verify the number of observations was split correctly.

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?

One of the resampling method is k-fold cross-validation. The training data is partitioned at random into sets of equal size which are called folds. For each iteration of resampling in 10-fold cross validation, one fold is kept as an assessment set to evaluate the model, and the remaining 9 folds are utilized as an analysis set to fit the model. The averages of each iteration make up the final resampling estimate of model performance. Because the model is developed based on the training data set, just fitting and testing models on the training set will result in very good performance. And the validation set approach would be used if we used the whole training set for resampling.

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.

```
# logistic regression with glm engine
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

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

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

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

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

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

There will be 30 models I am fitting, there are 10 folds for each engine and there are 3 different engine.

Question 5

Fit each of the models created in Question 4 to the folded data.

```
# logistic regression
log_res <- log_wkflow %>%
  fit_resamples(resamples = titanic_folds)
```

```
# linear discriminant
lda_res <- lda_wkflow %>%
  fit_resamples(resamples = titanic_folds)
```

```
# quadratic discriminant
qda_res <- qda_wkflow %>%
  fit_resamples(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.)*

```
log_acc <- collect_metrics(log_res)
log_acc
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.803    10  0.0186 Preprocessor1_Model1
## 2 roc_auc  binary     0.848    10  0.0169 Preprocessor1_Model1
```

```
#95% confidence interval
log_acc$mean[1] - 1.96*sqrt(log_acc$std_err[1]/10)
```

```
## [1] 0.7187804
```

```
log_acc$mean[1] + 1.96*sqrt(log_acc$std_err[1]/10)
```

```
## [1] 0.8876359
```

```
lda_acc <- collect_metrics(lda_res)
lda_acc
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.796    10  0.0188 Preprocessor1_Model1
## 2 roc_auc  binary     0.846    10  0.0181 Preprocessor1_Model1
```

```
#95% confidence interval
lda_acc$mean[1] - 1.96*sqrt(lda_acc$std_err[1]/10)
```

```
## [1] 0.7111858
```

```
lda_acc$mean[1] + 1.96*sqrt(lda_acc$std_err[1]/10)
```

```
## [1] 0.8812242
```

```
qda_acc <- collect_metrics(qda_res)
qda_acc
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.770    10  0.0116 Preprocessor1_Model1
## 2 roc_auc  binary     0.852    10  0.0168 Preprocessor1_Model1
```

```
#95% confidence interval
qda_acc$mean[1] - 1.96*sqrt(qda_acc$std_err[1]/10)
```

```
## [1] 0.7029349
```

```
qda_acc$mean[1] + 1.96*sqrt(qda_acc$std_err[1]/10)
```

```
## [1] 0.8364626
```

```
mean_accuracy <- c(log_acc$mean[1],lda_acc$mean[1], qda_acc$mean[1])
Standard_error <- c(log_acc$std_err[1],lda_acc$std_err[1], qda_acc$std_err[1])
models <- c("Logistic Regression", "LDA", "QDA")
results <- tibble(accuracies = mean_accuracy, Standard_error = Standard_error, models = models)
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 3 x 3
##   accuracies Standard_error models
##        <dbl>          <dbl> <chr>
## 1      0.803         0.0186 Logistic Regression
## 2      0.796         0.0188 LDA
## 3      0.770         0.0116 QDA
```

logistic model has the best performance in this example it has highest mean accuracy. The logistic regression also have the highest lower bound and upper bound.

Question 7

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

```
log_fit <- fit(log_wkflow, titanic_train)
log_fit %>% tidy()
```

```
## # A tibble: 10 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -4.69      0.664       -7.06  1.67e-12
##  2 age              0.0677    0.0128       5.28  1.27e- 7
##  3 sib_sp           0.493     0.123        4.00  6.38e- 5
##  4 parch            0.110     0.142        0.769 4.42e- 1
```

```
##  5 fare            -0.00381   0.0109      -0.349 7.27e- 1
##  6 pclass_X2        1.40      0.360        3.89  1.00e- 4
##  7 pclass_X3        2.61      0.373        6.99  2.73e-12
##  8 sex_male         2.24      0.291        7.71  1.26e-14
##  9 sex_male_x_fare  0.0129    0.00838      1.53  1.25e- 1
## 10 fare_x_age      -0.000279  0.000191    -1.46  1.46e- 1
```

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.

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

```
## # A tibble: 179 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1     0.112    0.888
##  2     0.767    0.233
##  3     0.467    0.533
##  4     0.234    0.766
##  5     0.428    0.572
##  6     0.222    0.778
##  7     0.755    0.245
##  8     0.253    0.747
##  9     0.108    0.892
## 10     0.616    0.384
## # ... with 169 more rows
```

```
log_reg_acc <- augment(log_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
x <- collect_metrics(log_res)[1,][1:3]
names(x)[3] <- '.estimate'
bind_rows(log_reg_acc,x)%>%
add_column(models=c('testing','folds'), .before = ".metric")
```

```
## # A tibble: 2 x 4
##   models  .metric  .estimator .estimate
##   <chr>   <chr>    <chr>          <dbl>
## 1 testing accuracy binary         0.860
## 2 folds   accuracy binary         0.803
```

What I see is that the model's testing accuracy is a little bit higher than the average accuracy across folds.