Pstat 131 Hw 4

Noe Arambula

2022-04-28

Contents

output: pdf_document: toc: true html_document: toc: true toc_float: true code_folding: show —

Loading Libraries

```
library(tidymodels)
library(ISLR)
library(ISLR2)
library(tidyverse)
library(discrim)

tidymodels_prefer()
```

Read in CSV Data File and Set Seed

```
titanic <- 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.
```

```
set.seed(777)
```

Changing Survived and pclass to Factors

```
titanic$survived =  factor(titanic$survived, levels = c("Yes", "No"))
# Note can use parse_factor() in order to give a warning when there is a value not in the set

titanic$pclass =  factor(titanic$pclass)

class(titanic$survived)
```

```
## [1] "factor"
```

```
class(titanic$pclass)
```

```
## [1] "factor"
```

```
titanic
```

```
## # A tibble: 891 x 12
##    passenger_id survived pclass name       sex     age sib_sp parch ticket  fare
##             <dbl> <fct>    <fct>  <chr>      <chr> <dbl>  <dbl> <dbl> <chr>  <dbl>
##   1             1 No       3      Braund, M~ male     22      1     0 A/5 2~  7.25
##   2             2 Yes      1      Cumings, ~ fema~    38      1     0 PC 17~ 71.3
##   3             3 Yes      3      Heikkinen~ fema~    26      0     0 STON/~  7.92
##   4             4 Yes      1      Futrelle,~ fema~    35      1     0 113803 53.1
##   5             5 No       3      Allen, Mr~ male     35      0     0 373450  8.05
##   6             6 No       3      Moran, Mr~ male     NA      0     0 330877  8.46
##   7             7 No       1      McCarthy,~ male     54      0     0 17463  51.9
##   8             8 No       3      Palsson, ~ male      2      3     1 349909 21.1
##   9             9 Yes      3      Johnson, ~ fema~    27      0     2 347742 11.1
## 10            10 Yes      2      Nasser, M~ fema~    14      1     0 237736 30.1
## # ... with 881 more rows, and 2 more variables: cabin <chr>, embarked <chr>
```

Q.1 Splitting Data Create A Recipe Like in HW 3

```r
titanic_split <- titanic %>%
  initial_split(strata = survived, prop = 0.8)

titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

dim(titanic_train)
```

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

```r
dim(titanic_test)
```

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

Each data set has approximately the right number of observations, the training data has 712 obs. which is about 80% of the full data set, which contains 891 observations.

```r
# Titanic recipe like in HW 3
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train) %
  step_impute_linear(age, impute_with = imp_vars(sib_sp)) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with("sex"):age + age:fare)
```

Question 2

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

```r
# Creating tuned recipe

titanic_tuned_rec <- titanic_recipe %>%
  step_poly(pclass, sex, age, sib_sp, parch, fare, degree = tune())  # polynomial regression
```

```r
#linear regression model
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

titanic_tuned_wf <- workflow() %>%
  add_recipe(titanic_tuned_rec) %>%
  add_model(lm_spec)
```

```r
# rsample object of the cross-validation resamples

titanic_folds <- vfold_cv(titanic_train, v = 10) # Here ISLR uses v instead of k but they are interchan
titanic_folds # creates the k-Fold data set
```

```
## #  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
```

```r
# tibble with hyperparameter values we are exploring
degree_grid <- grid_regular(degree(range = c(1, 10)), levels = 10)
degree_grid
```

```
## # A tibble: 10 x 1
##    degree
##     <dbl>
##  1      1
##  2      2
##  3      3
##  4      4
##  5      5
##  6      6
##  7      7
##  8      8
##  9      9
## 10     10
```

```r
# Do k-fold cross validation; tune_grid() fits the models within each fold for each value specified in

tune_res <- tune_grid(
  object = titanic_tuned_wf,
  resamples = titanic_folds,
  grid = degree_grid
)
```

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 re-sampling method would that be?

Answer: K-fold cross-validation is a re-sampling method in order to help evaluate machine learning models by splitting the data into further smaller samples/subsets that we can perform analysis on separately in order to test our models on various amounts of subsets of the data. So , in order to do this we need to create a workflow object with one parameter set for tuning, in part 2 this was the first step I created the recipe and workflow. Next we need to actually split the data into the folds/multiple smaller training sets which we did using vfold_cv. Lastly, we had to create a tibble with all the values we want to test, so in this case the polynomial degrees we wanted to test.

We should use k-fold cross-validation because it is useful when we smaller amount of observations since we have multiple subsets we can test on. It helps reduce selection bias as we test on many subsets with different observations. It is also useful because it helps us see how our model would perform on new data like the testing set before having to use the actual testing set thus we can see trends such as bias, variance, over fitting, etc and we can try and correct it before using it on the actual testing set. It allows us to fit multiple models on multiple subsets to see how they do.

If we used the entire training set that would be a validation approach.

Question 4

Set up workflows for 3 models:

A logistic regression with the glm engine;

A linear discriminant analysis with the MASS engine;

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.

Answer: I will be fitting 300 models across all folds to the data since there are we are fitting 3 models to 10 folds each, so we have 3 engines so 10x3 = 30

Logistic Regression Workflow

```r
# We will use the recipe created in Question 2 to create workflows

# Logistic regression
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

titanic_log_wf <- workflow() %>%  # log workflow
```

```
  add_recipe(titanic_recipe) %>%
  add_model(log_reg)
```

LDA Workflow

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

titanic_lda_wf <- workflow() %>%  # lda workflow
  add_model(lda_mod) %>%
  add_recipe(titanic_recipe)
```

QDA Workflow

```
# Quadratic discriminant analysis
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

titanic_qda_wf <- workflow() %>%  # qda workflow
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)
```

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.

```
# logistic regression fit
fit_log <- tune_grid(
  object = titanic_log_wf,
  resamples = titanic_folds,
  grid = degree_grid,
  control = control_resamples(verbose = TRUE))

# fit_resamples fits computes performance metrics across our specified resamples
# the control option prints out progress which helps with models that take a long time to fit
```

```
# linear discriminant analysis fit
fit_lda <- fit_resamples(
  titanic_lda_wf,
  titanic_folds,
  control = control_resamples(verbose = TRUE))
```

```
# Quadratic discriminant analysis fit
fit_qda <- fit_resamples(
  titanic_qda_wf,
  titanic_folds,
  control = control_resamples(verbose = TRUE))


save(fit_log, fit_lda, fit_qda, file = "fittedmodels.rda")
```

I saved all the fitted models in an R script that I previously ran to save time and will now just load them in

```
load(file = "fittedmodels.rda")
```

Question 6

Use `collect_metrics()` to print the mean and standard errors of the performance metric *accuracy* across all folds for each of the three 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.)*

Logistic Regression

```
# Logistic regression
collect_metrics(fit_log)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.806    10  0.0152 Preprocessor1_Model1
## 2 roc_auc  binary     0.852    10  0.0167 Preprocessor1_Model1
```

Linear Discriminant Analysis

```
# Linear discriminant analysis
collect_metrics(fit_lda,summarize = F)
```

```
## # A tibble: 20 x 5
##    id     .metric  .estimator .estimate .config
##    <chr>  <chr>    <chr>          <dbl> <chr>
##  1 Fold01 accuracy binary         0.819 Preprocessor1_Model1
##  2 Fold01 roc_auc  binary         0.889 Preprocessor1_Model1
##  3 Fold02 accuracy binary         0.708 Preprocessor1_Model1
##  4 Fold02 roc_auc  binary         0.737 Preprocessor1_Model1
##  5 Fold03 accuracy binary         0.831 Preprocessor1_Model1
##  6 Fold03 roc_auc  binary         0.857 Preprocessor1_Model1
##  7 Fold04 accuracy binary         0.845 Preprocessor1_Model1
##  8 Fold04 roc_auc  binary         0.906 Preprocessor1_Model1
##  9 Fold05 accuracy binary         0.873 Preprocessor1_Model1
## 10 Fold05 roc_auc  binary         0.934 Preprocessor1_Model1
## 11 Fold06 accuracy binary         0.803 Preprocessor1_Model1
## 12 Fold06 roc_auc  binary         0.811 Preprocessor1_Model1
## 13 Fold07 accuracy binary         0.789 Preprocessor1_Model1
## 14 Fold07 roc_auc  binary         0.853 Preprocessor1_Model1
## 15 Fold08 accuracy binary         0.761 Preprocessor1_Model1
## 16 Fold08 roc_auc  binary         0.914 Preprocessor1_Model1
## 17 Fold09 accuracy binary         0.676 Preprocessor1_Model1
## 18 Fold09 roc_auc  binary         0.770 Preprocessor1_Model1
## 19 Fold10 accuracy binary         0.789 Preprocessor1_Model1
## 20 Fold10 roc_auc  binary         0.849 Preprocessor1_Model1
```

```
# note adding the summarize = F option allows us to see the estimate for each fold
collect_metrics(fit_lda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.789    10  0.0192 Preprocessor1_Model1
## 2 roc_auc  binary     0.852    10  0.0201 Preprocessor1_Model1
```

Quadratic Discriminant Analysis

```
# Quadratic discriminant analysis
collect_metrics(fit_qda)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.784    10  0.0159 Preprocessor1_Model1
## 2 roc_auc  binary     0.828    10  0.0194 Preprocessor1_Model1
```

Best Model

Now, after this analysis I have chosen the logistic regression model to be the final model. It performed the best because it had the lowest standard error out of all the models and highest mean for accuracy.

Second best model was qda

Question 7

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

```r
# Fitting logistic model to entire training data set
final_fit <- fit(titanic_log_wf, titanic_train)

final_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##    (Intercept)            age         sib_sp          parch           fare
##     -4.0364890      0.0212189      0.3017504      0.1546627      0.0151621
##      pclass_X2      pclass_X3       sex_male  sex_male_x_age     age_x_fare
##      1.6165940      2.7304424      1.0653427      0.0624705     -0.0004132
##
## Degrees of Freedom: 711 Total (i.e. Null);  702 Residual
## Null Deviance:       948
## Residual Deviance: 628.2     AIC: 648.2
```

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.

```r
predict(final_fit, new_data = titanic_test, type = "class") %>%
  bind_cols(titanic_test %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.816
```

The model's testing accuracy was 0.8156425 while the average accuracy for the folds was 0.8061228 so it did a bit better on the testing set then on the folds