PSTAT131-HW4

Ephets Head

4/25/2022

Load the dataset titanic into R, and turn variables pclass and survived into factors.

```
#load data set
titanic <-read_csv("data2/titanic.csv", show_col_types=FALSE)
titanic
```

```
## # A tibble: 891 x 12
##    passenger_id survived pclass name       sex     age sib_sp parch ticket  fare
##           <dbl> <chr>     <dbl> <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>
```

```
#factor pclass and survived
titanic$pclass <- factor(titanic$pclass)
titanic$survived <- factor(titanic$survived)

#re-assign the base level of the factored variable "survived" to be "Yes"
titanic$survived <- relevel(titanic$survived, "Yes")
```

Question 1: Split the data, stratifying by the outcome variable survived.

```
set.seed(0427)

titan_split <- initial_split(titanic, strata = survived, prop=0.7)
titan_train <- training(titan_split)
titan_test <- testing(titan_split)

#verify that each set has an appropriate number of observations
dim(titan_split)
```

```
##   analysis assessment          n          p
##        623        268        891         12
```

The training set has 623 observations, while the testing set has 268.

Create an identical recipe to the one in HW 3.

```r
titan_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data=titan_train) %>%
  step_impute_linear() %>%
  step_dummy(sex) %>%
  step_interact(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare)
```

Question 2: For the training data, use k-fold cross validation, with $k = 10$. Use the same basic recipe as in HW 3.

```r
#first we will create a tuned recipe
titanic_tuned_recipe <-
  recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data=titan_train) %>%
  step_impute_linear() %>%
  step_dummy(sex) %>%
  step_interact(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare) %>%
  step_poly(degree=tune())

#next we will create a k-fold dataset
titan_folds <- vfold_cv(titan_train, v=10)

#before we create a workflow, we must specify a basic regression model
logr_model<-
  logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

titan_kfold_wf <- workflow() %>%
  add_recipe(titanic_tuned_recipe) %>%
  add_model(logr_model)

titan_degree_grid <- grid_regular(degree(range=c(1,10)),levels=10)

#finally we fold the training data using our workflow and k-fold dataset
titanic_res <- tune_grid(
  titan_kfold_wf,
  resamples = titan_folds,
  grid=10
)
```

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?

In k-fold cross validation, we divide our data into 'k' folds - in this case, 10 - where each fold is a set of similar size. For each re-sampling, one fold is set aside as the testing data while the rest are used for training the model. The accuracy of the trained model is then calculated using the "testing" (or validation) fold data. Averaging these cross-validation accuracies gives us a good estimate of the accuracy of our model.

Using k-fold cross-validation allows us to test more models given the same limited data, as each data point is still used only once as testing data. If we were to use the entire training set to fit our model, this is simply called cross-validation.

Question 4: Set up workflows for 3 models:

1. *A logistic regression with the `glm` engine:*

```r
#Part 1: specify and store a logistic regression model using the glm engine
logress_model<-
  logistic_reg() %>%
  set_engine("glm")

#Part 2: create a workflow and add our model and recipe
logr_flow <-
  workflow() %>%
  add_model(logress_model) %>%
  add_recipe(titanic_tuned_recipe)
```

2. *A linear discriminant analysis with the `MASS` engine:*

```r
#Part 1: specify and store a linear discriminant model using the mass engine
ld_model <-
  discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

#Part 2: create a workflow and add our model and recipe
ld_flow <-
  workflow() %>%
  add_model(ld_model) %>%
  add_recipe(titanic_tuned_recipe)
```

3. *A quadratic discriminant analysis with the `MASS` engine.*

```r
#Part 1: specify and store a quadratic discriminant model using the mass engine
qd_model <-
  discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

#Part 2: create a workflow and add our model and recipe
qd_flow <-
  workflow() %>%
  add_model(qd_model) %>%
  add_recipe(titanic_tuned_recipe)
```

How many models total, across all folds, will you be fitting to the data?

There are 10 folds, and we will be fitting the three models we have specified above to each fold. This means we will have a total of 30 fitted models.

Question 5: Fit each of the models created in question 4 to the folded data.

```r
#first we will create a workflow set
wf_set <- workflow_set(
  preproc = list(tuned = titanic_tuned_recipe),
  models = list(Logistic_Reg = logress_model, Linear_Discr = ld_model, Quadr_Discr = qd_model)
)
#now we will fit the three workflows to the folded data
```

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 models has performed the best. Explain why.

```r
metrics <- collect_metrics(fitted_folded)
cols_metrics <- metrics[c(1,5,7,9)]
acc_metrics <- cols_metrics %>%
  filter(.metric == "accuracy")

acc_metrics
```

```
## # A tibble: 9 x 4
##   wflow_id           .metric   mean std_err
##   <chr>              <chr>    <dbl>   <dbl>
## 1 tuned_Logistic_Reg accuracy 0.809  0.0136
## 2 tuned_Logistic_Reg accuracy 0.809  0.0136
## 3 tuned_Logistic_Reg accuracy 0.809  0.0136
## 4 tuned_Linear_Discr accuracy 0.803  0.0129
## 5 tuned_Linear_Discr accuracy 0.803  0.0129
## 6 tuned_Linear_Discr accuracy 0.803  0.0129
## 7 tuned_Quadr_Discr  accuracy 0.789  0.0171
## 8 tuned_Quadr_Discr  accuracy 0.789  0.0171
## 9 tuned_Quadr_Discr  accuracy 0.789  0.0171
```

The Quadratic Discriminant Analysis model has the largest standard error and the smallest mean accuracy, so it clearly performs the worst. As for which model performs the best, we will select the model with the highest mean accuracy (which would be the logistic regression model), despite the Linear Discriminant Analysis model having a slightly lower standard error.

Question 7: Now that you've chosen a model, fit your chosen model to the entire training dataset (not the folds).

```r
logr_flow_untuned <- workflow() %>%
  add_model(logress_model) %>%
  add_recipe(titan_recipe)

titanic_best_model <- fit(logr_flow_untuned, data= titan_train)
```

Question 8: Finally, with your fitted model, use `predict()`, `bind_cols()`, and `accuracy()` to assess your model's performance on the testing data!

```r
#first we create a vector of the model's fitted values on the testing data
testing_predictions <- predict(titanic_best_model, new_data=titan_test)

#next we create a vector of the testing data's actual observed outcomes
testing_observ <- titan_test$survived

#then we combine the predicted and actual values in a tibble
testing_results <- bind_cols(.pred_class = testing_predictions, outcome = testing_observ)

accuracy(data=testing_results, truth=titan_test$survived,estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.786
```

Therefore, our accuracy on the testing data is about 0.786, which is (as expected) significantly lower than how it performed on the training data. The model still performs fairly well.