Homework 4

Vardan Martirosyan

2022-11-05

First, we read in the data.

```
dataset <- read.csv("/Users/vardan/Desktop/pstat131/Homework/Homework4/data/titanic.csv")
```

Then, we load in the tidyverse and tidymodels libraries as desired. Additionally, we also load the 'ggplot2' and 'corrplot' libraries, which can help with some of the questions asked of us.

```
library(tidyverse)
library(tidymodels)
library(ggplot2)
library(corrplot)
library(MASS)
library(klaR)
library(discrim)
tidymodels_prefer()
```

We also are told that 'survived' and 'pclass' should be changed to factors, and that when changing 'survived' to a factor, to reorder the factor so that 'Yes' is the first factor. We do this as follows:

```
#Changing them to factors.
dataset$survived <- as.factor(dataset$survived)
dataset$pclass <- as.factor(dataset$pclass)

#Reordering the factor so that 'Yes' is the first factor.
dataset$survived <- relevel(dataset$survived, 'Yes')
```

Finally, we set the seed so that our code can be reproduced.

```
set.seed(69)
```

Finally, we are asked to create a recipe for this dataset that is identical to the recipe that we used in Homework 3. We do this as follows:

```
survived_recipe <-
  #Defining the predictors and response variables.
  recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = dataset) %>%
  #imputing the variable age.
  step_impute_linear(age) %>%
  #creating dummy variables for all categorical variables.
  step_dummy(all_nominal_predictors()) %>%
```

```
  #creating the interaction terms.
  step_interact( ~ starts_with("sex"):fare) %>%
  step_interact( ~ age:fare)
```

Question 1

We are asked to split the data, stratifying on the outcome variable 'survived'. We are asked to choose the proportions to split the data into. We choose 80/20 again, as this has worked out consistently in the past for us in previous homeworks and lab assignments.

```
dataset_split <- initial_split(dataset, prop = 0.8, strata = survived)

dataset_train <- training(dataset_split)
dataset_test <- testing(dataset_split)
```

We then want to verify that the training and testing data sets have an appropriate number of observations. We check their sizes as follows:

```
#First, we check the size of the original dataset.
nrow(dataset)
```

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

```
#Then, we check the size of the training and test datasets.
nrow(dataset_train)
```

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

```
nrow(dataset_test)
```

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

Looking at the training and testing datasets, it looks like they each have a good number of observations for the purposes they serve.

Question 2

We are now asked to fold the training data, using k-fold cross validation, with k = 10. We do this as follows:

```
titanic_folds <- vfold_cv(dataset_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

We are now asked to explain (in our own words) what we are doing in Question 2. In particular, we are asked what is k-fold cross validation, why we should use it (as opposed to fitting and testing the models on the entire testing dataset), and if we did use the entire training dataset, what the resampling method would be. We answer these questions as follows.

First, we describe k-fold cross validation. First, we take the dataset we are working with (in the case of Question 2, it is the titanic training dataset). Then, we partition this data set into k groups (ie, folds) of equal sizes. We note that sometimes, splitting up data into equal data sets is not possible, so we note that these groups are all of similar size, if not exactly equal. Then, we consider an arbitrary fold as the 1st fold, and set this group aside (to use as the validation set later on). We then fit the model on the other k-1 groups (so, the k-1 groups that we fit the model on can be considered as the 'training set'). Then, we compute the Mean Square Error on all of the observations that were contained in the first fold. We repeat this above process for each of the k groups that we have partitioned, which lets each fold (ie, group) of the data set that we partitioned be used as a validation set. After we calculate k MSE's, we can calculate a k-fold Cross Validation estimate of the test Mean Square error by summing up all of the mean square errors, and dividing it by the number of folds that we have.

We should use k-fold cross validation as opposed to fitting and testing the models on the entire testing dataset because k-fold cross validation lets us develop a model that is only trained using the training dataset, not the testing dataset. If we fit and test models on the entire testing dataset, then our models will be specifically tuned for the testing dataset, which means the results that they produce would not hold up on other testing datasets. K-Fold cross validation lets us determine the performance of models without having to rely on the testing dataset, which leaves the testing data set as an actual testing set of observations which we can use on our models later on.

If we used the entire training dataset, our resampling method would be called validation set approach. Having answered all of the questions, we can now answer what we did in Question 2 as follows. In question 2, we created 10 folds of the training dataset that was obtained from the Titanic dataset.

Question 4

We are now asked to set up workflows for 3 different models. The first model is a logistic regression model with the 'glm' engine. The second model is a linear discriminant analysis with the 'MASS' engine. The third model is a quadratic discriminant analysis with the 'MASS' engine. We do this as follows:

```
#First, we define the logistic regression model using 'glm'.
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
```

```
#We then define the first workflow and add the recipe.
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(survived_recipe)


#Second, we define the LDA model using 'MASS'.
lda_mod <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

#We then define the second workflow and add the recipe.
lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(survived_recipe)


#Third, we define the model for QDA.
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

#Creating the third workflow and adding the recipe.
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(survived_recipe)
```

We are asked how many models, total, across all folds, will we be fitting to the data. We note that we have 10 folds and 3 models. This means that, for each fold, we will be fitting one of the three models. Since there are 10 folds, we will be fitting 30 models, total, across all folds, on the data.

Question 5

We are asked to fit each of the models created in Question 4 to the folded data. We do this as follows:

```
fit1 <- fit_resamples(log_wkflow, titanic_folds)

fit2 <- fit_resamples(lda_wkflow, titanic_folds)

fit3 <- fit_resamples(qda_wkflow, titanic_folds)
```

Question 6

We are now asked to use collect_metrics() to print the mean and standard errors of the performance metric accuracy across all folds for each of the three models. We do this as follows:

```
#Collecting metrics on logistic regression.
collect_metrics(fit1) %>% filter(.metric == 'accuracy') %>% select(.metric, mean, std_err)
```

```
## # A tibble: 1 x 3
##   .metric   mean std_err
##   <chr>    <dbl>   <dbl>
## 1 accuracy 0.802  0.0175
```

```
#Collecting metrics on the linear discriminant analysis.
collect_metrics(fit2) %>% filter(.metric == 'accuracy') %>% select(.metric, mean, std_err)
```

```
## # A tibble: 1 x 3
##   .metric   mean std_err
##   <chr>    <dbl>   <dbl>
## 1 accuracy 0.788  0.0172
```

```
#Collecting metrics on the quadratic discriminant analysis.
collect_metrics(fit3) %>% filter(.metric == 'accuracy') %>% select(.metric, mean, std_err)
```

```
## # A tibble: 1 x 3
##   .metric   mean std_err
##   <chr>    <dbl>   <dbl>
## 1 accuracy 0.773  0.0237
```

We are then asked to decide which of the 3 fitted models has performed the best. I believe that the Logistic Regression fitted model performed the best. This is because out of the three fitted models, it has the highest accuracy, and the second lowest standard error, which implies that it performs the best (when compared to the other models).

Question 7

We are now asked to fit our chosen model to the entire training dataset (not to the folds). We do this as follows:

```
log_fit <- fit(log_wkflow, dataset_train)
```

Question 8

We are now asked to use predict(), bind_cols(), and accuracy() to assess our model's performance on the testing data. We do this as follows:

```
#Using predict() and bind_cols() to get a data frame with all of the data we need to use accuracy().
dataset_test_res <- predict(log_fit, new_data = dataset_test %>% select(-survived))
dataset_test_res <- bind_cols(dataset_test_res, dataset_test %>% select(survived))

#Using the accuracy() function to assess our model's performance on the testing data.
dataset_test_res %>% accuracy(truth = survived, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.777
```

Note that we could code this using the augment() function as well.

```
#Using the augment() function to obtain the same value.
log_reg_acc <- augment(log_fit, new_data = dataset_test) %>%
  accuracy(truth = survived, estimate = .pred_class)

log_reg_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.777
```

We are then asked to compare our model's testing accuracy to it's average accuracy across folds, and to describe what we see. We recall that our model's average accuracy across folds was approximately 80.22 percent, which is a few percent higher than our testing accuracy, which is approximately 77.65 percent. We note that 77.65 percent is a reasonable value for our testing accuracy to have, as it is still within 2 standard errors of the average accuracy across folds.