Homework 4

Jules Merigot (8488256)

November 09, 2022

PSTAT 131/231 Statistical Machine Learning - Fall 2022

Resampling

Before we get started, let's first load the titanic data and change the `survived` and `pclass` variables to factors.

```
# loading the data
titanic_data <- read.csv(file = "C:/Users/jules/OneDrive/Desktop/homework-4/data/titanic.csv")
head(titanic_data)

titanic_data$survived <- factor(titanic_data$survived, labels = c("Yes", "No"))
titanic_data$pclass <- factor(titanic_data$pclass)

str(titanic_data)
```

Question 1

Let's set a seed, and randomly split the data to create a training and testing set. We'll choose appropriate proportions and stratify on the outcome variable, `survived`.

```
# setting the seed
set.seed(8488)

titanic_split <- initial_split(titanic_data, prop=0.70, strata=survived)

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

For splitting the data, I chose a proportion of 0.70 because it allows for more training data, while retaining enough data to be tested since there is a limited amount of observations. The training data has 623 observations while the testing data has 268 observations.

Next, let's make our recipe, while accounting for missing values and creating the proper interactions.

```
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare,
                         data=titanic_train) %>%
  step_impute_linear(age, impute_with = imp_vars(all_predictors())) %>%
  step_dummy(all_nominal_predictors()) %>%
```

```
  step_interact(terms = ~ sex_male:fare + age:fare)

titanic_recipe %>% prep() %>% juice()
```

```
## # A tibble: 623 x 10
##      age sib_sp parch  fare survived pclass_X2 pclass_X3 sex_m~1 sex_m~2 fare_~3
##    <dbl>  <int> <int> <dbl> <fct>        <dbl>     <dbl>   <dbl>   <dbl>   <dbl>
##  1  38        1     0 71.3  No               0         0       0     0     2709.
##  2  35        1     0 53.1  No               0         0       0     0     1858.
##  3  27        0     2 11.1  No               0         1       0     0      301.
##  4  14        1     0 30.1  No               1         0       0     0      421.
##  5  32.6      0     0 13    No               1         0       1    13      424.
##  6  34        0     0 13    No               1         0       1    13      442
##  7  28        0     0 35.5  No               0         0       1    35.5    994
##  8  25.7      0     0  7.88 No               0         1       0     0      202.
##  9  25.7      0     0  7.75 No               0         1       0     0      199.
## 10  14        1     0 11.2  No               0         1       0     0      157.
## # ... with 613 more rows, and abbreviated variable names 1: sex_male,
## #   2: sex_male_x_fare, 3: fare_x_age
```

Question 2

Now, we fold the training data using k-fold cross-validation, with k=10 (represented by v in the ISLR package terminology).

```
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 [560/63]> Fold01
##  2 <split [560/63]> Fold02
##  3 <split [560/63]> Fold03
##  4 <split [561/62]> Fold04
##  5 <split [561/62]> Fold05
##  6 <split [561/62]> Fold06
##  7 <split [561/62]> Fold07
##  8 <split [561/62]> Fold08
##  9 <split [561/62]> Fold09
## 10 <split [561/62]> Fold10
```

Question 3

In Question 2 above, we are randomly splitting our titanic training dataset into 10 groups or folds, as denoted by the parameter k, of approximately equal size. Essentially, k-fold cross validation is a statistical method or resampling procedure used to evaluate the skill of machine learning models on a limited dataset. When fitting the model, the model is trained on (k-1) folds and tested on the one left out fold. This process is then repeated k times until the model is trained and tested on all the folds. This method allows for users to fit and test models on various groups of data, as designated by the k=10 folds, which is easy to implement

and thus results in skill estimates that generally have a lower bias than other methods. If instead we used a resampling method on the entire training set, it would be considered the Validation Set Approach, which purely uses a training dataset to test and fit the model, and then a testing dataset to fit the final model.

Question 4

We will now set up workflows for 3 models:

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)
```

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)
```

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)
```

With 10 folds and with the 3 models created above, we will be fitting a total of 30 models to the data.

Question 5

We will now fit each of the created models to the folded data.

```
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

We will now use `collect_metrics()` to print the mean and standard errors of the performance metric accuracy across all folds for each of the three models.

```
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     0.815    10  0.0178 Preprocessor1_Model1
## 2 roc_auc  binary     0.855    10  0.0187 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     0.807    10  0.0126 Preprocessor1_Model1
## 2 roc_auc  binary     0.857    10  0.0194 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     0.799    10  0.0125 Preprocessor1_Model1
## 2 roc_auc  binary     0.851    10  0.0165 Preprocessor1_Model1
```

The fitted model that performed the best is the linear discriminant analysis model. While this model did not have the highest mean accuracy, 0.8072 compared to the 0.8152 mean accuracy of the logistic regression model, it is considered more accurate relative to the standard error accuracy. The standard error of the linear discriminant analysis model is 0.01263, while the standard error of the logistic regression model is higher at 0.01782. Since its standard error is much lower than that of the logistic regression model, even though the linear discriminant analysis model has a slightly lower mean accuracy, it is still considered the more accurate fitted model in this case.

Question 7

Now that we've chosen the linear discriminant analysis model as the most accurate, let's fit the model to the entire training dataset.

```
final_log_fit <- fit(log_wkflow, titanic_train)
final_log_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
##        5.082673       -0.070660       -0.468572        0.003331
##            fare        pclass_X2        pclass_X3        sex_male
##       -0.004168       -1.267485       -2.787656       -2.520565
## sex_male_x_fare      fare_x_age
##       -0.012563        0.000418
##
## Degrees of Freedom: 622 Total (i.e. Null);  613 Residual
## Null Deviance:       830
## Residual Deviance: 519   AIC: 539
```

Question 8

Finally, with our fitted model, we will use `predict()`, `bind_cols()`, and `accuracy()` to assess our model's performance on the testing data!

```
test_model_acc <- predict(final_log_fit, titanic_test) %>%
  bind_cols(titanic_test$survived) %>%
  accuracy(truth = titanic_test$survived, estimate = .pred_class)
```

```
## New names:
## * `` -> `...2`
```

```
test_model_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.787
```

```
Test_model_acc <- predict(final_log_fit, titanic_test, type="class") %>%
  bind_cols(titanic_test %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
Test_model_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.787
```

As we can see above, our model's testing accuracy is slightly lower than our average accuracy across the folds. Our mean accuracy across all folds for the linear discriminant analysis is 0.8072, while our model's testing accuracy is 0.7873. This is normal and expected since the model was optimized for the training data and the folds in its earlier stages, therefore it will tend to have a lower accuracy when applied to the testing data.

Overall, this model is rather accurate with an accuracy nearing 80%. The linear discriminant analysis model is definitely the better choice in this case.