Homework 4

Eunseo Kang

Contents

Resampling

For this assignment, we will continue working with part of a Kaggle data set that was the subject of a machine learning competition and is often used for practicing ML models. The goal is classification; specifically, to predict which passengers would survive the Titanic shipwreck.

Load the data from `data/titanic.csv` into *R* and familiarize yourself with the variables it contains using the codebook (`data/titanic_codebook.txt`).

Notice that `survived` and `pclass` should be changed to factors. When changing `survived` to a factor, you may want to reorder the factor so that *"Yes"* is the first level.

Make sure you load the `tidyverse` and `tidymodels`!

Remember that you'll need to set a seed at the beginning of the document to reproduce your results.

Create a recipe for this dataset **identical** to the recipe you used in Homework 3.

```
data <- read.csv('data/titanic.csv')
data$survived <- factor(data$survived, levels=c("Yes","No"))
data$pclass <- factor(data$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(514222)

titanic_split <- initial_split(data, prop = 0.70, strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

titanic_split
```

```
## <Training/Testing/Total>
## <623/268/891>
```

```
dim(titanic_train)
```

```
## [1] 623  12
```

```
dim(titanic_test)
```

```
## [1] 268  12
```

- The training and testing data sets have the appropriate number of observations. The proportion of training data is 0.6992144, which is approximately 0.7.

```
#Creating an identical recipe with HW3
unique(titanic_train$sex) %>% sort()
```

```
## [1] "female" "male"
```

```
titanic_recipe <- recipe(survived ~ pclass+sex+age+sib_sp+parch+fare, data = titanic_train) %>%
  step_impute_linear(age) %>%
  step_dummy(sex) %>%
  step_interact(terms=~fare:starts_with("sex")+age:starts_with("sex")) %>%
  step_poly(degree = tune())
```

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 [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 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?

- It would be ideal to set aside a validation set and use it to assess the performance of our prediction model. However, since data are often scarce, this is usually not possible. In this case, k-fold cross-validation uses part of the available data to fit the model, and a different part to test it. In question 2, what we are doing is to make 10 partitions, where we fit the model using 9 parts out of them and calculate the prediction error of the fitted model when predicting the one left part of the data. We do this 10 times for each part and combine the 10 estimates of prediction error. If we use the entire training set, then we are just using validation method.

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
log_fit <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

log_wkflow <- workflow() %>%
  add_model(log_fit) %>%
  add_recipe(titanic_recipe)

#lda
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

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

#qda
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

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

- Since there are 10 folds for each model, and since there are 3 models, I will totally fit 30 (multiplication of 3 and 10) models to the data.

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.*

```
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
```

```
#logit
tune_log <- tune_grid(
  object = log_wkflow,
  resamples = titanic_folds,
  grid = degree_grid)

#lda
tune_lda <- tune_grid(
  object = lda_wkflow,
  resamples = titanic_folds,
  grid = degree_grid)

#qda
tune_qda <- tune_grid(
  object = qda_wkflow,
  resamples = titanic_folds,
  grid = degree_grid
)
save(tune_log, tune_lda, tune_qda, file = "mydata.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 four models.

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

collect_metrics(tune_log)
```

```
## # A tibble: 20 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      1 accuracy binary     0.817    10  0.0171 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.878    10  0.0177 Preprocessor01_Model1
##  3      2 accuracy binary     0.817    10  0.0171 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.878    10  0.0177 Preprocessor02_Model1
##  5      3 accuracy binary     0.817    10  0.0171 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.878    10  0.0177 Preprocessor03_Model1
##  7      4 accuracy binary     0.817    10  0.0171 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.878    10  0.0177 Preprocessor04_Model1
##  9      5 accuracy binary     0.817    10  0.0171 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.878    10  0.0177 Preprocessor05_Model1
## 11      6 accuracy binary     0.817    10  0.0171 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.878    10  0.0177 Preprocessor06_Model1
## 13      7 accuracy binary     0.817    10  0.0171 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.878    10  0.0177 Preprocessor07_Model1
## 15      8 accuracy binary     0.817    10  0.0171 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.878    10  0.0177 Preprocessor08_Model1
## 17      9 accuracy binary     0.817    10  0.0171 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.878    10  0.0177 Preprocessor09_Model1
## 19     10 accuracy binary     0.817    10  0.0171 Preprocessor10_Model1
```

```
## 20     10 roc_auc  binary     0.878    10  0.0177 Preprocessor10_Model1
```

```
collect_metrics(tune_lda)
```

```
## # A tibble: 20 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      1 accuracy binary     0.814    10  0.0162 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.881    10  0.0178 Preprocessor01_Model1
##  3      2 accuracy binary     0.814    10  0.0162 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.881    10  0.0178 Preprocessor02_Model1
##  5      3 accuracy binary     0.814    10  0.0162 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.881    10  0.0178 Preprocessor03_Model1
##  7      4 accuracy binary     0.814    10  0.0162 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.881    10  0.0178 Preprocessor04_Model1
##  9      5 accuracy binary     0.814    10  0.0162 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.881    10  0.0178 Preprocessor05_Model1
## 11      6 accuracy binary     0.814    10  0.0162 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.881    10  0.0178 Preprocessor06_Model1
## 13      7 accuracy binary     0.814    10  0.0162 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.881    10  0.0178 Preprocessor07_Model1
## 15      8 accuracy binary     0.814    10  0.0162 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.881    10  0.0178 Preprocessor08_Model1
## 17      9 accuracy binary     0.814    10  0.0162 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.881    10  0.0178 Preprocessor09_Model1
## 19     10 accuracy binary     0.814    10  0.0162 Preprocessor10_Model1
## 20     10 roc_auc  binary     0.881    10  0.0178 Preprocessor10_Model1
```

```
collect_metrics(tune_qda)
```

```
## # A tibble: 20 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      1 accuracy binary     0.779    10  0.0153 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.853    10  0.0243 Preprocessor01_Model1
##  3      2 accuracy binary     0.779    10  0.0153 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.853    10  0.0243 Preprocessor02_Model1
##  5      3 accuracy binary     0.779    10  0.0153 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.853    10  0.0243 Preprocessor03_Model1
##  7      4 accuracy binary     0.779    10  0.0153 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.853    10  0.0243 Preprocessor04_Model1
##  9      5 accuracy binary     0.779    10  0.0153 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.853    10  0.0243 Preprocessor05_Model1
## 11      6 accuracy binary     0.779    10  0.0153 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.853    10  0.0243 Preprocessor06_Model1
## 13      7 accuracy binary     0.779    10  0.0153 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.853    10  0.0243 Preprocessor07_Model1
## 15      8 accuracy binary     0.779    10  0.0153 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.853    10  0.0243 Preprocessor08_Model1
## 17      9 accuracy binary     0.779    10  0.0153 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.853    10  0.0243 Preprocessor09_Model1
## 19     10 accuracy binary     0.779    10  0.0153 Preprocessor10_Model1
## 20     10 roc_auc  binary     0.853    10  0.0243 Preprocessor10_Model1
```

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.)*

- Mean accuracy for logit model is 0.82 and standard error is 0.017. Mean accuracy for lda model is 0.81 and standard error is 0.016. Mean accuracy for qda model is 0.78 and standard error is 0.015. While there are almost no variation in standard errors, logit model shows the highest mean accuracy rate, therefore, logit model is chosen.

Question 7

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

```
best_degree <- select_by_one_std_err(tune_log, degree, metric = "accuracy")
final_wf <- finalize_workflow(log_wkflow, best_degree)
final_fit <- fit(final_wf, titanic_train)
final_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_poly()
##
## -- Model ----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##     (Intercept)          pclass2          pclass3              age
##      -3.5091249        1.4705678        2.6762152        0.0005347
##          sib_sp            parch             fare         sex_male
##       0.4912987        0.0881554        0.0002467        1.4874142
## fare_x_sex_male   sex_male_x_age
##      -0.0020048        0.0627360
##
## Degrees of Freedom: 622 Total (i.e. Null);  613 Residual
## Null Deviance:       829.6
## Residual Deviance: 499   AIC: 519
```

Question 8

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

```
log_predict <- predict(final_fit, new_data = titanic_test, type = "prob")

log_reg_acc <- augment(final_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_reg_acc
```

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

```
## 1 accuracy binary         0.769
```

Compare your model's testing accuracy to its average accuracy across folds. Describe what you see.

*The average accuracy across folds for the logit model was 0.82 while the same for the testing data set is 0.77, which is lower. It is consistent with the expectation that the accuracy for the training data would be higher since the model is fitted based on the training data set.

Required for 231 Students

Consider the following intercept-only model, with $\epsilon \sim N(0, \sigma^2)$:

$$Y = \beta + \epsilon$$

where β is the parameter that we want to estimate. Suppose that we have n observations of the response, i.e. $y_1, ..., y_n$, with uncorrelated errors.

Question 9

Derive the least-squares estimate of β.

$$\hat{\beta} = arg_{\beta} min \sum_{i=1}^{n} \epsilon_i^2$$

$$= min \sum_{i=1}^{n} (Y_i - \beta)^2$$

$$FOC) - 2 \sum (Y_i - \hat{\beta}) = 0$$

$$\therefore \sum Y_i = n\hat{\beta}$$

$$\hat{\beta} = \bar{Y}$$

Question 10

Suppose that we perform leave-one-out cross-validation (LOOCV). Recall that, in LOOCV, we divide the data into n folds. What is the covariance between $\hat{\beta}^{(1)}$, or the least-squares estimator of β that we obtain by taking the first fold as a training set, and $\hat{\beta}^{(2)}$, the least-squares estimator of β that we obtain by taking the second fold as a training set?

- The covariance is going to be zero since each estimates is going to follow the uncorrelated distribution since the each error is uncorrelated.