Homework 4

PSTAT 131/231

Contents

Resampling

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.

```
titanic <- read.csv(file =  "~/Downloads/homework-4/data/titanic.csv") %>%
  mutate(survived = factor(survived,
                           levels = c("Yes", "No")),
         pclass = factor(pclass))
head(titanic)
```

```
##   passenger_id survived pclass
## 1            1       No      3
## 2            2      Yes      1
## 3            3      Yes      3
## 4            4      Yes      1
## 5            5       No      3
## 6            6       No      3
##                                                 name    sex age sib_sp parch
## 1                             Braund, Mr. Owen Harris   male  22      1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38      1     0
## 3                              Heikkinen, Miss. Laina female  26      0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35      1     0
## 5                            Allen, Mr. William Henry   male  35      0     0
## 6                                    Moran, Mr. James   male  NA      0     0
##             ticket    fare cabin embarked
## 1        A/5 21171  7.2500  <NA>        S
## 2         PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

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

```
set.seed(343)
titanic_split <- initial_split(titanic, strata = survived, prop = 0.70)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)

titanic_recipe <- recipe(survived ~ pclass + sex + age +
                           sib_sp + parch + fare, titanic_train) %>%
  step_impute_linear(age, impute_with = imp_vars(sib_sp)) %>%
  # choice of predictors to impute with is up to you
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with("sex"):age + age:fare)
```

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(343)
titanic_split <- initial_split(titanic, strata = survived, prop = 0.70)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)


#verify the training and testing data sets
dim(titanic_train)
```

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

```
dim(titanic_test)
```

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

Question 2

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

```
set.seed(333)
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?

A k-fold cross validation is a procedure used to estimate the skill of a model in a new data. Rather than simply fitting and testing models on the entire training set, we use the k-fold cross validation to find the best value of degree that yields the "closest" fit. If we did use the entire training set, the resampling method used would be cross-validation.

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.

We would have 30 models, total, across all folds, fitting to the data. This is because for each model, which would be logistic, lda, and qda modeling, we would have 10 models each modeling method.

```
#1.

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

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

#2.
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

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


#3.
```

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

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

qda_mod <- discrim_regularized(frac_common_cov = 0, frac_identity = 0) %>%
  set_engine("klaR") %>%
  translate()
```

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

#tune_res <- tune_grid(
#  object = log_wkflow,
#  resamples = titanic_folds,
#  grid = degree_grid
#)

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

#tune_res_qda <- tune_grid(
#  object = qda_wkflow,
#  resamples = titanic_folds,
#  grid = degree_grid
#ke )


log_fit_rs <-
  log_wkflow %>%
fit_resamples(titanic_folds)

lda_fit_rs <-
  lda_wkflow %>%
fit_resamples(titanic_folds)

qda_fit_rs <-
  qda_wkflow %>%
```

```
fit_resamples(titanic_folds)
```

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.

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.)*

```
collect_metrics(log_fit_rs)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.820    10  0.0197 Preprocessor1_Model1
## 2 roc_auc  binary     0.864    10  0.0195 Preprocessor1_Model1
```

```
collect_metrics(lda_fit_rs)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.804    10  0.0238 Preprocessor1_Model1
## 2 roc_auc  binary     0.858    10  0.0205 Preprocessor1_Model1
```

```
collect_metrics(qda_fit_rs)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.790    10  0.0212 Preprocessor1_Model1
## 2 roc_auc  binary     0.833    10  0.0142 Preprocessor1_Model1
```

```
best_degree <- select_by_one_std_err(log_fit_rs, mean, metric = "accuracy")
best_degree
```

```
## # A tibble: 1 x 8
##   .metric  .estimator  mean     n std_err .config              .best .bound
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>                <dbl>  <dbl>
## 1 accuracy binary     0.820    10  0.0197 Preprocessor1_Model1 0.820  0.800
```

Note that larger mean implies greater accuracy. Since log has the largest mean and smallest standard error, we can conclude that the log model fits the best.

Question 7

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

```
final_wf_log <- workflow() %>%
  add_recipe(titanic_recipe) %>%
  add_model(log_reg)

#final_wf_qda

library(discrim)


final_fit_log <- parsnip::fit(final_wf_log, titanic_train)

final_fit_log
```

```
## == 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
##     -3.6835470       0.0190203       0.2683065       0.1917930       0.0135381
##      pclass_X2       pclass_X3        sex_male  sex_male_x_age      age_x_fare
##      1.3963566       2.3694788       1.0036425       0.0704243      -0.0004859
##
## Degrees of Freedom: 622 Total (i.e. Null);  613 Residual
## Null Deviance:       829.6
## Residual Deviance: 529   AIC: 549
```

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.

The accuracy to its average accuracy across folds is 0.8201741, and my accuracy on the testing data is 0.7835821.

```
pred_log = predict(final_fit_log, new_data = titanic_test, type = "class")
bind <- bind_cols(log = pred_log$.pred_class, survived = titanic_test$survived)

log_acc <- bind %>%
  accuracy(truth = survived, estimate = log)
log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.784
```

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.

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?