Homework 4

PSTAT 131/231

Contents

Resampling

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.

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.

Answer 1

```r
set.seed(3435)
library(dplyr)
library(tidymodels)
library(discrim)
library(poissonreg)
library(tune)
tidymodels_prefer()
titanic <- read.csv("titanic.csv", header=TRUE)

titanic$pclass <- as.factor(titanic$pclass)
titanic$survived <- as.factor(titanic$survived)
titanic$sex <- as.factor(titanic$sex)

titanic <- titanic %>%  mutate(survived=relevel(survived,ref="Yes"))
#titanic$pclass <- as.numeric(titanic$pclass)
#titanic$survived <- as.numeric(titanic$survived)
titanic$sex <- as.numeric(titanic$sex)
```

```r
titanic_split <- initial_split(titanic, prop = 0.80,
                                strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
titanic_split
```

```
## <Analysis/Assess/Total>
## <712/179/891>
```

```r
dim(titanic_train)
```

```
## [1] 712  12
```

```r
dim(titanic_test)
```

```
## [1] 179  12
```

Question 2

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

Answer 2

```r
Auto_folds <- vfold_cv(titanic_train, v = 10)
Auto_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
```

```r
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
```

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?

Answer 3

We are dividing training data into 10-folds, roughly equal size. We hold out one set to fit the model on all the sets. If we are using k folds then We leave out part k, fit the model to the other K ??? 1 parts (combined), and then obtain predictions for the left-out kth part. This method ensure that every observation from the original dataset has the chance of appearing in training and test set.

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.

Answer 4

3 models with 10 folds across each model selected so total will be 30.

```
##Tuned data recipe
poly_tuned_rec <-  recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train)%
  step_poly(fare, degree = tune())

##Logisitic regression

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


poly_log_wkflow <- workflow() %>%
  add_model(log_mod) %>%
  add_recipe(poly_tuned_rec)

##linear discriminant analysis
lda_mod <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

poly_lda_wkflow <- workflow() %>%
  add_model(lda_mod) %>%
  add_recipe(poly_tuned_rec)


##quadratic discriminant analysis
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

poly_qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(poly_tuned_rec)
```

```
Auto_folds <- vfold_cv(titanic_train, v = 10)
Auto_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
```

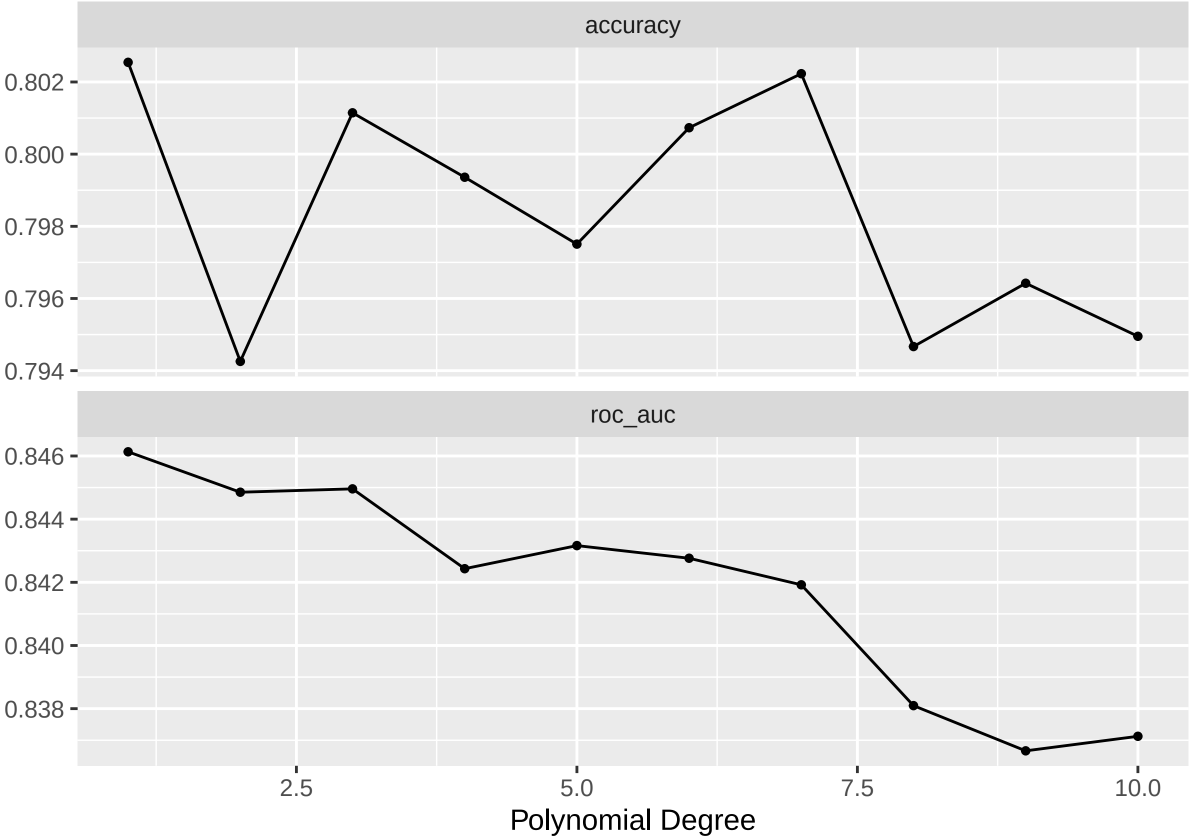
```
degree_grid <- grid_regular(degree(range = c(1, 10)), levels = 10)
degree_grid
```

```
## # 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
```

Question 5

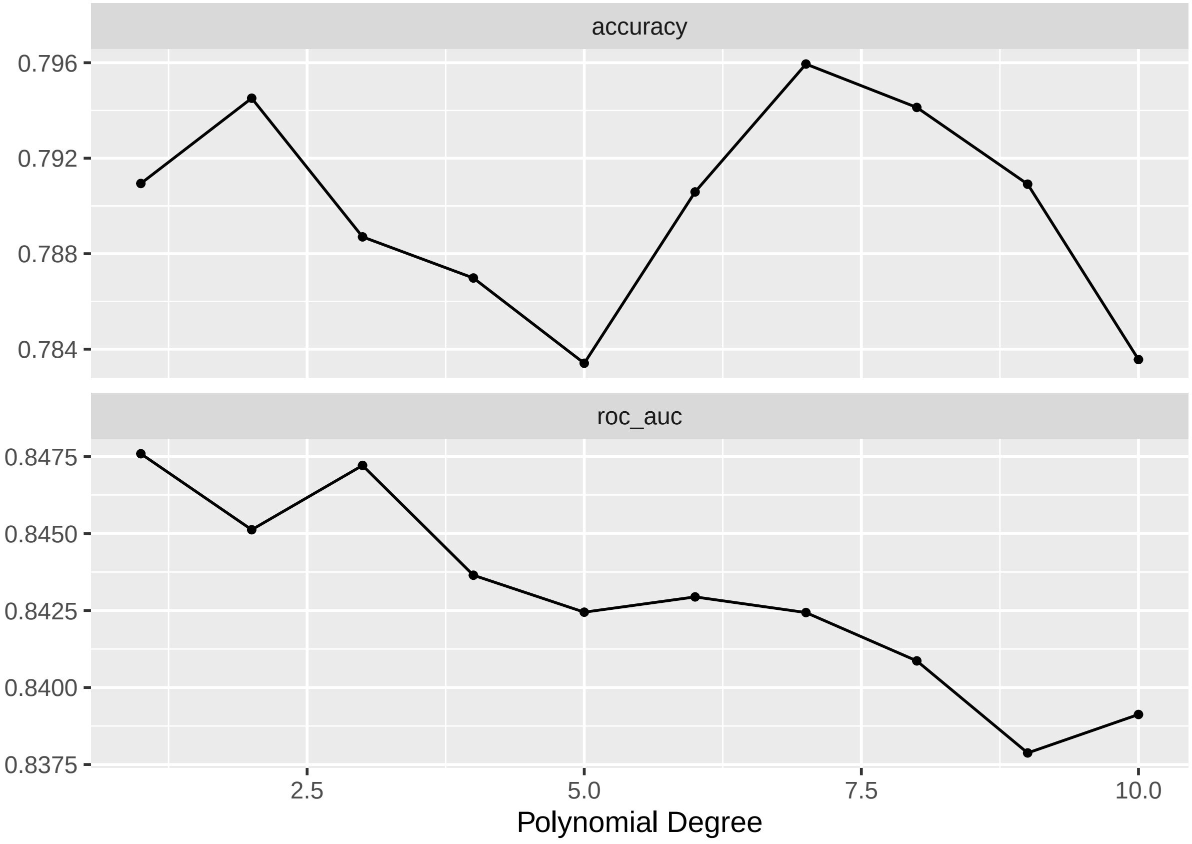
Fit each of the models created in Question 4 to the folded data. ###Answer 5

```
tune_res_log <- tune_grid(
  object = poly_log_wkflow,
  resamples = Auto_folds,
  grid = degree_grid,
  control = control_grid(verbose = TRUE)

)

autoplot(tune_res_log)
```

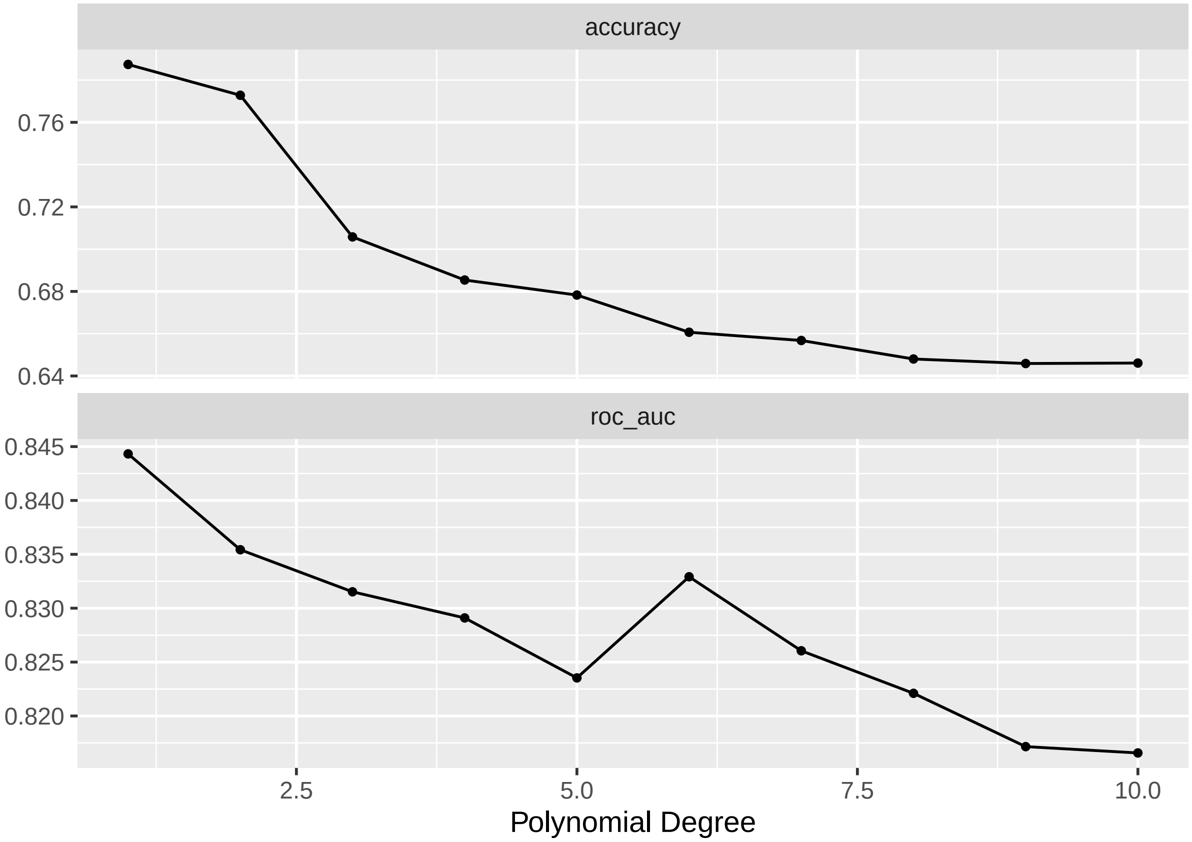


```
tune_res_lda<- tune_grid(
  object = poly_lda_wkflow,
  resamples = Auto_folds,
  grid = degree_grid,
  control = control_grid(verbose = TRUE)

)
```

```
autoplot(tune_res_lda)
```



```
tune_res_qda<- tune_grid(
  object = poly_qda_wkflow,
  resamples = Auto_folds,
  grid = degree_grid,
  control = control_grid(verbose = TRUE)

)


autoplot(tune_res_qda)
```



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(tune_res_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     0.803    10  0.0115 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.846    10  0.0141 Preprocessor01_Model1
##  3      2 accuracy binary     0.794    10  0.0132 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.845    10  0.0138 Preprocessor02_Model1
##  5      3 accuracy binary     0.801    10  0.0124 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.845    10  0.0134 Preprocessor03_Model1
##  7      4 accuracy binary     0.799    10  0.0127 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.842    10  0.0129 Preprocessor04_Model1
##  9      5 accuracy binary     0.798    10  0.0132 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.843    10  0.0127 Preprocessor05_Model1
## 11      6 accuracy binary     0.801    10  0.0134 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.843    10  0.0129 Preprocessor06_Model1
## 13      7 accuracy binary     0.802    10  0.0126 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.842    10  0.0120 Preprocessor07_Model1
## 15      8 accuracy binary     0.795    10  0.0129 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.838    10  0.0122 Preprocessor08_Model1
## 17      9 accuracy binary     0.796    10  0.0127 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.837    10  0.0115 Preprocessor09_Model1
## 19     10 accuracy binary     0.795    10  0.0152 Preprocessor10_Model1
## 20     10 roc_auc  binary     0.837    10  0.0113 Preprocessor10_Model1
```

```
show_best(tune_res_log, metric = "accuracy" , "roc_auc")
```

```
## # A tibble: 10 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      1 accuracy binary     0.803    10  0.0115 Preprocessor01_Model1
##  2      7 accuracy binary     0.802    10  0.0126 Preprocessor07_Model1
##  3      3 accuracy binary     0.801    10  0.0124 Preprocessor03_Model1
##  4      6 accuracy binary     0.801    10  0.0134 Preprocessor06_Model1
##  5      4 accuracy binary     0.799    10  0.0127 Preprocessor04_Model1
##  6      5 accuracy binary     0.798    10  0.0132 Preprocessor05_Model1
##  7      9 accuracy binary     0.796    10  0.0127 Preprocessor09_Model1
##  8     10 accuracy binary     0.795    10  0.0152 Preprocessor10_Model1
##  9      8 accuracy binary     0.795    10  0.0129 Preprocessor08_Model1
## 10      2 accuracy binary     0.794    10  0.0132 Preprocessor02_Model1
```

```
best_degree_log <-select_by_one_std_err(tune_res_log, degree, metric = "accuracy")
best_degree_log
```

```
## # A tibble: 1 x 9
##   degree .metric  .estimator  mean     n std_err .config             .best .bound
##    <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>               <dbl>  <dbl>
## 1      1 accuracy binary     0.803    10  0.0115 Preprocessor01_Mo~ 0.803  0.791
```

```
collect_metrics(tune_res_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     0.791    10  0.0156 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.848    10  0.0149 Preprocessor01_Model1
##  3      2 accuracy binary     0.795    10  0.0160 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.845    10  0.0147 Preprocessor02_Model1
##  5      3 accuracy binary     0.789    10  0.0145 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.847    10  0.0151 Preprocessor03_Model1
##  7      4 accuracy binary     0.787    10  0.0143 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.844    10  0.0142 Preprocessor04_Model1
##  9      5 accuracy binary     0.783    10  0.0143 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.842    10  0.0138 Preprocessor05_Model1
## 11      6 accuracy binary     0.791    10  0.0158 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.843    10  0.0141 Preprocessor06_Model1
## 13      7 accuracy binary     0.796    10  0.0152 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.842    10  0.0136 Preprocessor07_Model1
## 15      8 accuracy binary     0.794    10  0.0157 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.841    10  0.0127 Preprocessor08_Model1
## 17      9 accuracy binary     0.791    10  0.0169 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.838    10  0.0140 Preprocessor09_Model1
## 19     10 accuracy binary     0.784    10  0.0154 Preprocessor10_Model1
## 20     10 roc_auc  binary     0.839    10  0.0136 Preprocessor10_Model1
```

```
show_best(tune_res_lda, metric = "accuracy" , "roc_auc")
```

```
## # A tibble: 10 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      7 accuracy binary     0.796    10  0.0152 Preprocessor07_Model1
##  2      2 accuracy binary     0.795    10  0.0160 Preprocessor02_Model1
##  3      8 accuracy binary     0.794    10  0.0157 Preprocessor08_Model1
##  4      1 accuracy binary     0.791    10  0.0156 Preprocessor01_Model1
##  5      9 accuracy binary     0.791    10  0.0169 Preprocessor09_Model1
##  6      6 accuracy binary     0.791    10  0.0158 Preprocessor06_Model1
##  7      3 accuracy binary     0.789    10  0.0145 Preprocessor03_Model1
##  8      4 accuracy binary     0.787    10  0.0143 Preprocessor04_Model1
##  9     10 accuracy binary     0.784    10  0.0154 Preprocessor10_Model1
## 10      5 accuracy binary     0.783    10  0.0143 Preprocessor05_Model1
```

```
best_degree_lda<-select_by_one_std_err(tune_res_lda, degree, metric = "accuracy")
best_degree_lda
```

```
## # A tibble: 1 x 9
##   degree .metric  .estimator  mean     n std_err .config             .best .bound
##    <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>               <dbl>  <dbl>
## 1      1 accuracy binary     0.791    10  0.0156 Preprocessor01_Mo~ 0.796  0.781
```

```
collect_metrics(tune_res_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     0.787    10  0.0140 Preprocessor01_Model1
##  2      1 roc_auc  binary     0.844    10  0.0131 Preprocessor01_Model1
##  3      2 accuracy binary     0.773    10  0.0125 Preprocessor02_Model1
##  4      2 roc_auc  binary     0.835    10  0.0137 Preprocessor02_Model1
##  5      3 accuracy binary     0.706    10  0.0141 Preprocessor03_Model1
##  6      3 roc_auc  binary     0.832    10  0.0152 Preprocessor03_Model1
##  7      4 accuracy binary     0.685    10  0.0134 Preprocessor04_Model1
##  8      4 roc_auc  binary     0.829    10  0.0162 Preprocessor04_Model1
##  9      5 accuracy binary     0.678    10  0.0126 Preprocessor05_Model1
## 10      5 roc_auc  binary     0.824    10  0.0166 Preprocessor05_Model1
## 11      6 accuracy binary     0.661    10  0.0133 Preprocessor06_Model1
## 12      6 roc_auc  binary     0.833    10  0.0161 Preprocessor06_Model1
## 13      7 accuracy binary     0.657    10  0.0134 Preprocessor07_Model1
## 14      7 roc_auc  binary     0.826    10  0.0160 Preprocessor07_Model1
## 15      8 accuracy binary     0.648    10  0.0133 Preprocessor08_Model1
## 16      8 roc_auc  binary     0.822    10  0.0170 Preprocessor08_Model1
## 17      9 accuracy binary     0.646    10  0.0124 Preprocessor09_Model1
## 18      9 roc_auc  binary     0.817    10  0.0188 Preprocessor09_Model1
## 19     10 accuracy binary     0.646    10  0.0124 Preprocessor10_Model1
## 20     10 roc_auc  binary     0.817    10  0.0208 Preprocessor10_Model1
```

```
show_best(tune_res_qda, metric = "accuracy" , "roc_auc")
```

```
## # A tibble: 10 x 7
##    degree .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1      1 accuracy binary     0.787    10  0.0140 Preprocessor01_Model1
##  2      2 accuracy binary     0.773    10  0.0125 Preprocessor02_Model1
##  3      3 accuracy binary     0.706    10  0.0141 Preprocessor03_Model1
##  4      4 accuracy binary     0.685    10  0.0134 Preprocessor04_Model1
##  5      5 accuracy binary     0.678    10  0.0126 Preprocessor05_Model1
##  6      6 accuracy binary     0.661    10  0.0133 Preprocessor06_Model1
##  7      7 accuracy binary     0.657    10  0.0134 Preprocessor07_Model1
##  8      8 accuracy binary     0.648    10  0.0133 Preprocessor08_Model1
##  9     10 accuracy binary     0.646    10  0.0124 Preprocessor10_Model1
## 10      9 accuracy binary     0.646    10  0.0124 Preprocessor09_Model1
```

```
best_degree_qda<-select_by_one_std_err(tune_res_qda, degree, metric = "accuracy")
best_degree_qda
```

```
## # A tibble: 1 x 9
##   degree .metric  .estimator  mean     n std_err .config             .best .bound
##    <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>               <dbl>  <dbl>
## 1      1 accuracy binary     0.787    10  0.0140 Preprocessor01_Mo~  0.787  0.773
```

```
accuracies <- c(best_degree_log$.best, best_degree_lda$.best,
                best_degree_qda$.best)
models <- c("Logistic Regression", "LDA", "QDA")
results <- tibble(accuracies = accuracies, models = models)
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 3 x 2
##   accuracies models
##        <dbl> <chr>
## 1      0.803 Logistic Regression
## 2      0.796 LDA
## 3      0.787 QDA
```

Question 7

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

Answer 7

```r
final_wf <- finalize_workflow(poly_log_wkflow, best_degree_log)
final_wf
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_poly()
##
## -- Model -----------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```r
final_fit <- fit(final_wf, titanic_train)
final_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_poly()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
## (Intercept)      pclass2      pclass3          sex          age       sib_sp
##    -7.01104      1.37257      2.52578      2.62503      0.04695      0.38027
##       parch  fare_poly_1
##     0.19478     -3.91486
##
## Degrees of Freedom: 563 Total (i.e. Null);  556 Residual
##   (148 observations deleted due to missingness)
## Null Deviance:       760.3
## Residual Deviance: 500.4     AIC: 516.4
```

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.

Answer 8

Final fitted model accuracy is very close to the average accuracy across folds.

```
final_accu <- predict(final_fit, new_data = titanic_test, type = "class") %>%
  bind_cols(titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
final_accu
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.793
```