Homework-3

Caleb Mazariegos

2022-04-18

```
# setting survived and pclass as factors, reordering survived so that "Yes" is the first level
titanic_codebook$survived <- as.factor(titanic_codebook$survived)
titanic_codebook$survived <- factor(titanic_codebook$survived, levels = c("Yes", "No"))

titanic_codebook$pclass <- as.factor(titanic_codebook$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. Take a look at the training data and note any potential issues, such as missing data. Why is it a good idea to use stratified sampling for this data?

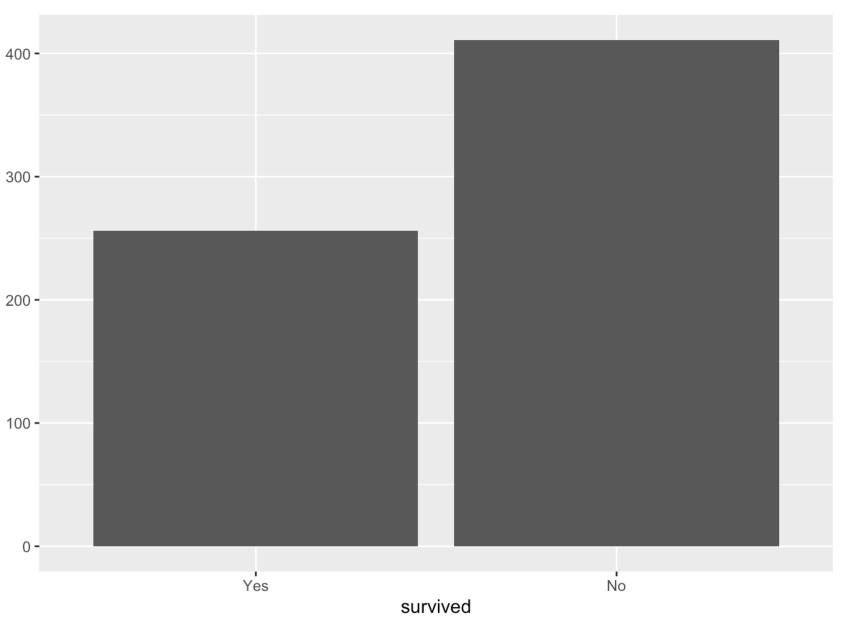
```
# Setting the seed
set.seed(3435)

titanic_split <- initial_split(titanic_codebook, prop = 0.75, strata = survived)

titanic_train <- training(titanic_split)

titanic_test <- testing(titanic_split)
```

There are some potential issues regarding missing data, some ages and cabins are missing. It is a good idea to use stratified sampling for this data because we want to group the passengers of the titanic into 2 groups, based on their survival.

Question 2

Using the training data set, explore/describe the distribution of the outcome variable survived.

```
titanic_train %>%
  ggplot(aes(x = survived)) + geom_bar()
```



The distribution between surviving and dying does not look equal. There are more people that did not survive than those that did survive.

Question 3

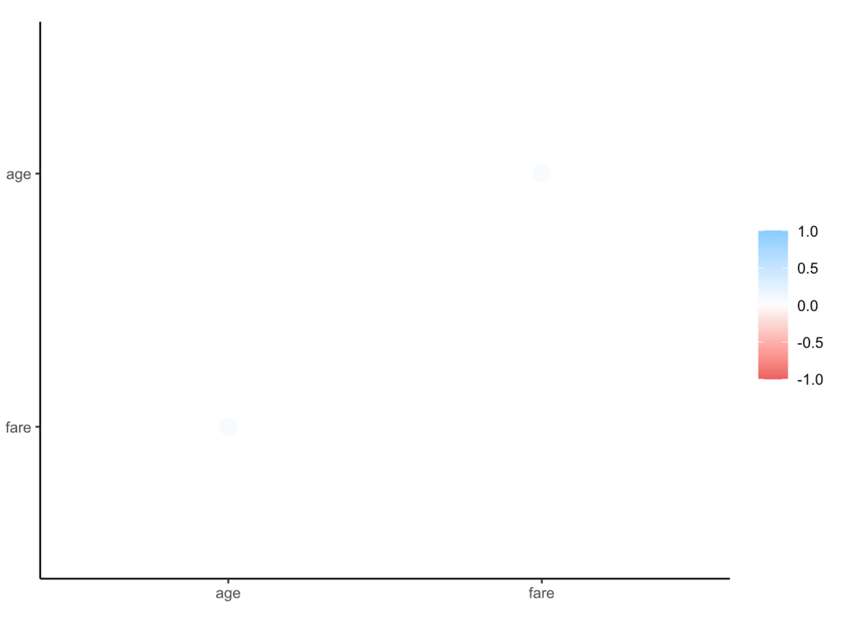
Using the training data set, create a correlation matrix of all continuous variables. Create a visualization of the matrix, and describe any patterns you see. Are any predictors correlated with each other? Which ones, and in which direction?

```
cor_titanic_train <- titanic_train %>%
  select(age, fare) %>%
   correlate()
```

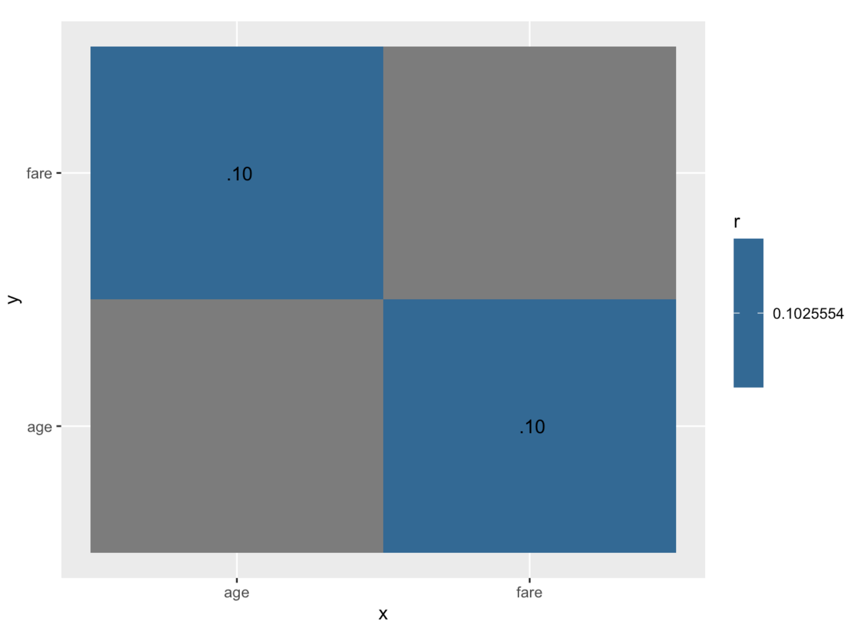
```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
rplot(cor_titanic_train)
```

```
## Don't know how to automatically pick scale for object of type noquote. Defaulting to continuous.
```




```
cor_titanic_train %>%
  stretch() %>%
  ggplot(aes(x, y, fill = r)) +
  geom_tile() +
  geom_text(aes(label = as.character(fashion(r))))
```



The only continuous variables are age and fare. These 2 predictor variables have a weak positive correlation of 0.10.

Question 4

Using the training data, create a recipe predicting the outcome variable survived. Include the following predictors: ticket class, sex, age, number of siblings or spouses aboard, number of parents or children aboard, and passenger fare.

Recall that there were missing values for age. To deal with this, add an imputation step using step_impute_linear(). Next, use step_dummy() to dummy encode categorical predictors. Finally, include interactions between:

- Sex and passenger fare, and

- Age and passenger fare.

You'll need to investigate the tidymodels documentation to find the appropriate step functions to use.

```
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train) %>%

step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ starts_with("sex"):fare) %>%
  step_interact(terms = ~ age:fare)
titanic_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for age
## Dummy variables from all_nominal_predictors()
## Interactions with starts_with("sex"):fare
## Interactions with age:fare
```

Question 5

Specify a logistic regression model for classification using the "glm" engine. Then create a workflow. Add your model and the appropriate recipe. Finally, use fit() to apply your workflow to the training data.

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

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

log_fit <- fit(log_workflow, titanic_train)

log_fit %>%
  tidy()
```

```
## # A tibble: 10 × 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -4.02      0.644       -6.25  4.14e-10
##  2 age              0.0531    0.0126       4.21  2.54e- 5
##  3 sib_sp           0.462     0.132        3.50  4.70e- 4
##  4 parch            0.183     0.147        1.25  2.13e- 1
##  5 fare            -0.00729   0.0106      -0.689 4.91e- 1
##  6 pclass_X2        1.10      0.351        3.14  1.67e- 3
##  7 pclass_X3        2.20      0.364        6.05  1.48e- 9
##  8 sex_male         2.36      0.301        7.85  4.29e-15
##  9 sex_male_x_fare  0.0133    0.00836      1.60  1.10e- 1
## 10 age_x_fare      -0.000237  0.000190    -1.25  2.12e- 1
```

Question 6

Repeat Question 5, but this time specify a linear discriminant analysis model for classification using the "MASS" engine.

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


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

lda_fit <- fit(lda_wkflow, titanic_train)
```

Question 7

Repeat Question 5, but this time specify a quadratic discriminant analysis model for classification using the "MASS" engine.

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

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

qda_fit <- fit(qda_workflow, titanic_train)
```

Question 8

Repeat Question 5, but this time specify a naive Bayes model for classification using the "klaR" engine. Set the usekernel argument to FALSE.

```
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)

nb_workflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)

nb_fit <- fit(nb_workflow, titanic_train)
```

Question 9

Now you've fit four different models to your training data.

Use predict() and bind_cols() to generate predictions using each of these 4 models and your training data. Then use the accuracy metric to assess the performance of each of the four models.

Which model achieved the highest accuracy on the training data?

```
log_predict <- bind_cols(predict(log_fit, new_data = titanic_train), titanic_train %>% dplyr :: select(survived))
augment(log_fit, new_data = titanic_train) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 178  45
##        No   78 366
```

```
log_acc <- augment(log_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc #0.8155922
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.816
```

```
lda_predict <- bind_cols(predict(lda_fit, new_data = titanic_train), titanic_train %>% dplyr :: select(survived))
augment(lda_fit, new_data = titanic_train) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 177  54
##        No   79 357
```

```
lda_acc <- augment(lda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc # 0.8
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.801
```

```
qda_preedict <- bind_cols(predict(qda_fit, new_data = titanic_train), titanic_train %>% dplyr :: select(survived)
)
augment(qda_fit, new_data = titanic_train) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes 131  26
##        No  125 385
```

```
qda_acc <- augment(qda_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
qda_acc #0.773
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.774
```

```
nb_predict <- suppressWarnings(bind_cols(predict(nb_fit, new_data = titanic_train), titanic_train %>% dplyr :: se
lect(survived)))
suppressWarnings(augment(nb_fit, new_data = titanic_train) %>%
  conf_mat(truth = survived, estimate = .pred_class))
```

```
##           Truth
## Prediction Yes  No
##        Yes 131  28
##        No  125 383
```

```
nb_acc <- suppressWarnings(augment(nb_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class))
nb_acc #0.7706147
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.771
```

Logistic regression is the model that had the highest accuracy on the training data.

Question 10

Fit the model with the highest training accuracy to the testing data. Report the accuracy of the model on the testing data.

Again using the testing data, create a confusion matrix and visualize it. Plot an ROC curve and calculate the area under it (AUC).

How did the model perform? Compare its training and testing accuracies. If the values differ, why do you think this is so?

```
# fitting the model to testing data
predict(log_fit, new_data = titanic_test, type = "prob")
```

```
## # A tibble: 224 × 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.926    0.0737
##  2    0.114    0.886
##  3    0.167    0.833
##  4    0.224    0.776
##  5    0.257    0.743
##  6    0.114    0.886
##  7    0.0550   0.945
##  8    0.160    0.840
##  9    0.537    0.463
## 10    0.906    0.0942
## # … with 214 more rows
```

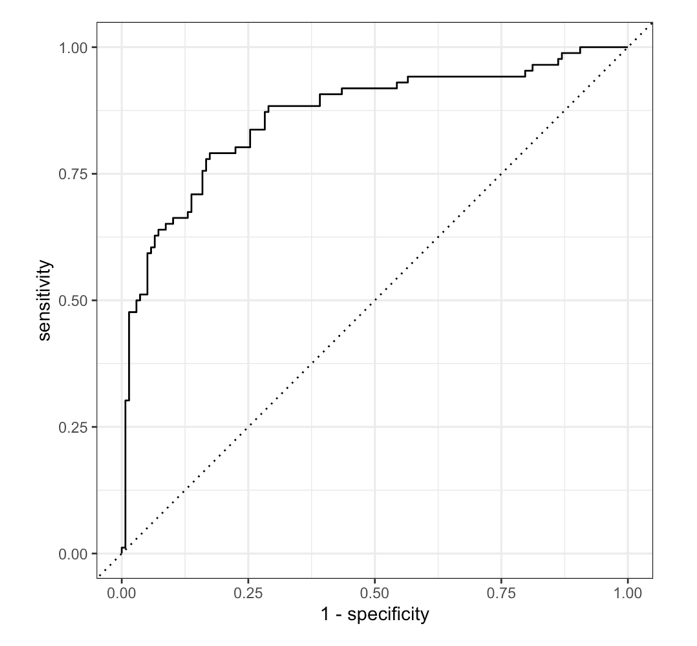
```
# viewing confusion matrix on testing data
augment(log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes  No
##        Yes  57  18
##        No   29 120
```

```
# looking at testing accuracy
multi_metric <- metric_set(accuracy, sensitivity, specificity)

augment(log_fit, new_data = titanic_test) %>%
  multi_metric(truth = survived, estimate = .pred_class)
```

```
## # A tibble: 3 × 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.790
## 2 sensitivity binary         0.663
## 3 specificity binary         0.870
```

```
# ROC Curve
augment(log_fit, new_data = titanic_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```



The model performed pretty well. There is a 79% accuracy.