HW3

Seongu Lee

4/19/2022

```r
#install.packages("discrim")
#install.packages("poissonreg")
#install.packages("corrr")
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(tidymodels)
```

```
## Warning: package 'tidymodels' was built under R version 4.0.5

## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --

## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.1      v tune         0.2.0
## v infer        1.0.0      v workflows    0.2.6
## v modeldata    0.1.1      v workflowsets 0.2.1
## v parsnip      0.2.1      v yardstick    0.0.9
## v recipes      0.2.0

## Warning: package 'broom' was built under R version 4.0.5

## Warning: package 'dials' was built under R version 4.0.5

## Warning: package 'scales' was built under R version 4.0.5

## Warning: package 'infer' was built under R version 4.0.5

## Warning: package 'modeldata' was built under R version 4.0.5

## Warning: package 'parsnip' was built under R version 4.0.5

## Warning: package 'recipes' was built under R version 4.0.5

## Warning: package 'rsample' was built under R version 4.0.5

## Warning: package 'tune' was built under R version 4.0.5

## Warning: package 'workflows' was built under R version 4.0.5

## Warning: package 'workflowsets' was built under R version 4.0.5

## Warning: package 'yardstick' was built under R version 4.0.5

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```r
library(discrim)
```

```
## Warning: package 'discrim' was built under R version 4.0.5

##
## Attaching package: 'discrim'

## The following object is masked from 'package:dials':
##
##     smoothness
```

```r
library(poissonreg)
```

```
## Warning: package 'poissonreg' was built under R version 4.0.5
```

```r
library(corrr)
```

```
## Warning: package 'corrr' was built under R version 4.0.5
```

```r
library(klaR)
```

```
## Warning: package 'klaR' was built under R version 4.0.5

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
titanic <- read.csv("C:/Users/sungu/OneDrive/Desktop/homework-3/homework-3/data/titanic.csv")
titanic$survived = factor(titanic$survived,levels = c("Yes","No"))
titanic$pclass = factor(titanic$pclass)
```

Question1

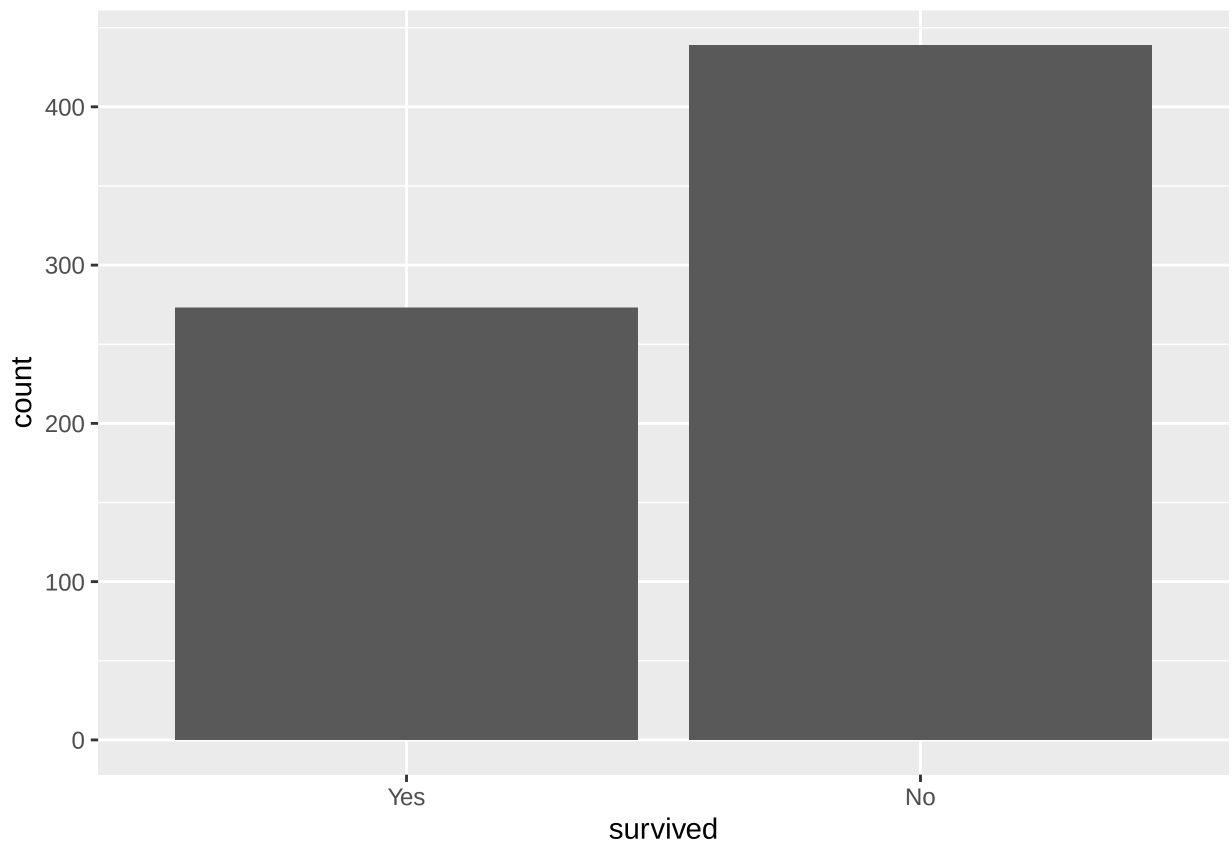
```r
set.seed(731)

split <- initial_split(titanic, prop = 0.80,strata = survived)
train <- training(split)
test<- testing(split)
table(is.na(train))
```

```
##
## FALSE  TRUE
##  7852   692
```

So, the train set will have 712 rows and test set will have 179 rows about 80% of the total data set. Also, I can see some NA values from the training set on the cabin and age columns. Stratified sampling helps to find the best distribution with survived column.

Question2

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




```r
table(train$survived)
```

```
##
## Yes  No
## 273 439
```

There are less survived people than non survived people. There are 273 survived people and 439 non survived people.

Question3

```r
num <- unlist(lapply(train, is.numeric))
num
```

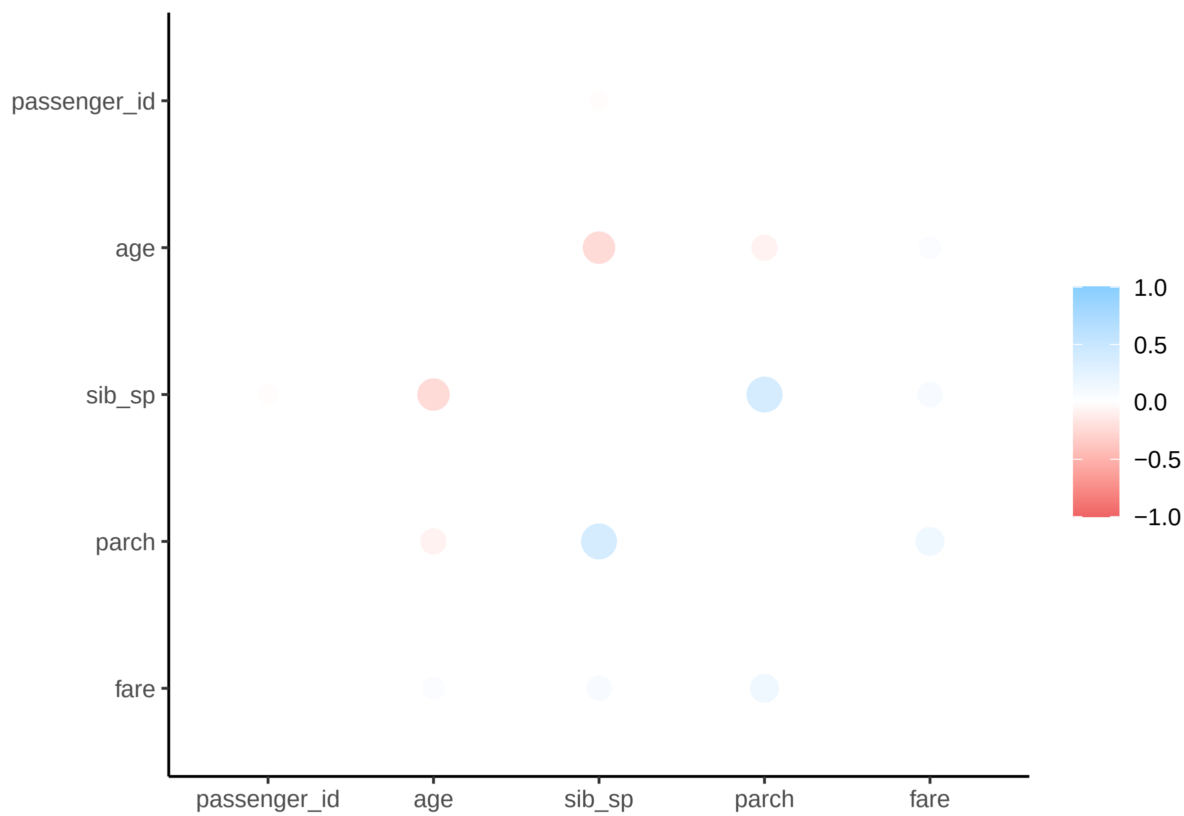
```
## passenger_id     survived       pclass         name          sex          age
##         TRUE        FALSE        FALSE        FALSE        FALSE         TRUE
##       sib_sp        parch       ticket         fare        cabin     embarked
##         TRUE         TRUE        FALSE         TRUE        FALSE        FALSE
```

```r
cor_train <- train %>%
  dplyr::select(-c(survived,pclass,name,sex,ticket,cabin,embarked)) %>%
  correlate()
```

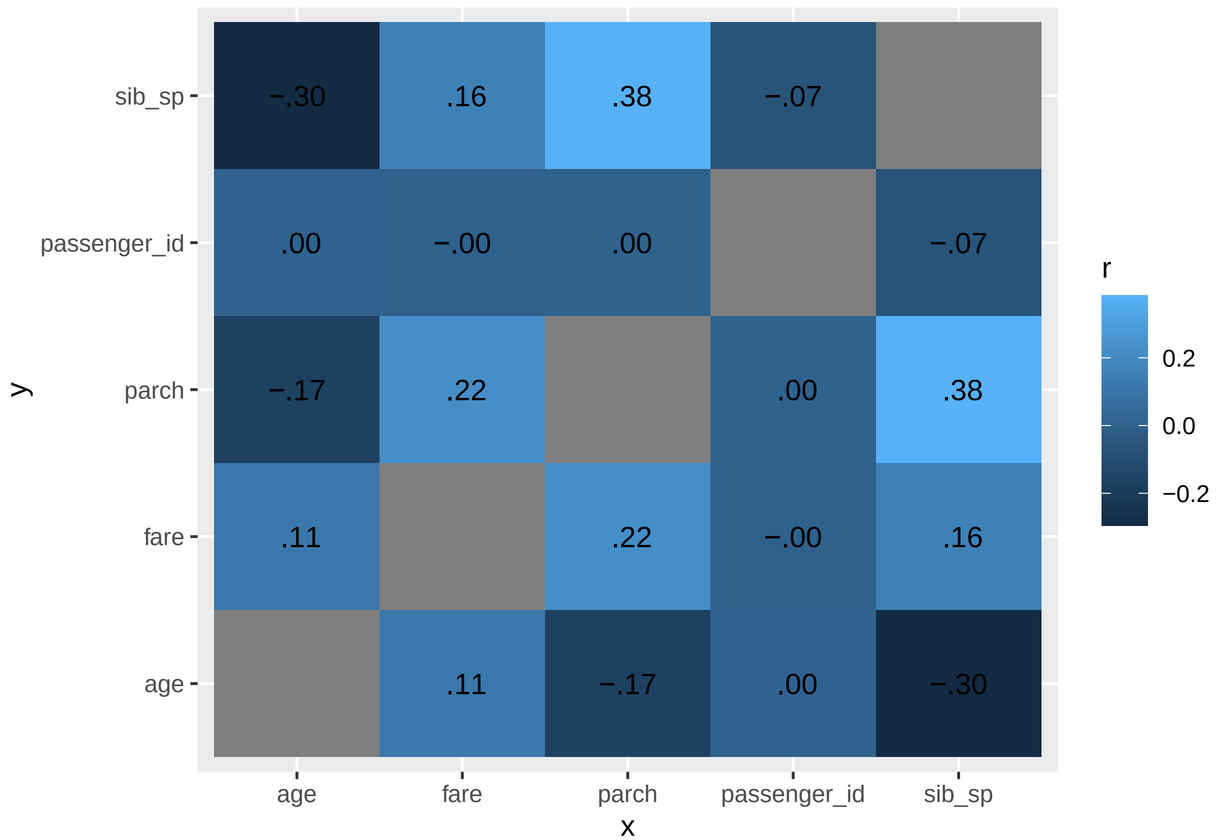
```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```r
rplot(cor_train)
```

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



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



The visualization matrix is symmetric. sib_sp are negatively correlated with passenger_id, parch are negatively correlated with age and positively correlated with fare and sib_sp. fare are positively correlated with age and sib_sp.

Question4

```r
reciped <- recipe(survived ~ pclass+sex+age+sib_sp+parch+fare, data = train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with("sex"):fare) %>%
  step_interact(~ age:fare)
reciped
```

```
## 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
```

Question5

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

```r
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(reciped)

log_fit <- fit(log_wkflow, train)
```

```r
log_fit %>%
  tidy()
```

```
## # A tibble: 10 x 5
##    term             estimate std.error statistic  p.value
##    <chr>               <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     -4.36      0.613       -7.11  1.13e-12
##  2 age              0.0573    0.0120       4.79  1.67e- 6
##  3 sib_sp           0.353     0.121        2.91  3.56e- 3
##  4 parch            0.0737    0.127        0.581 5.61e- 1
##  5 fare             0.00751   0.00857      0.876 3.81e- 1
##  6 pclass_X2        1.09      0.348        3.15  1.66e- 3
##  7 pclass_X3        2.37      0.359        6.61  3.80e-11
##  8 sex_male         2.41      0.273        8.82  1.15e-18
##  9 sex_male_x_fare  0.00935   0.00639      1.46  1.43e- 1
## 10 age_x_fare      -0.000453  0.000207    -2.19  2.83e- 2
```

Question6

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

```r
lin_wkflow <- workflow() %>%
  add_model(lin_reg) %>%
  add_recipe(reciped)
```

```
lin_fit <- fit(lin_wkflow, train)
```

Question7

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

```
qd_wkflow <- workflow() %>%
  add_model(qd_reg) %>%
  add_recipe(reciped)

qd_fit <- fit(qd_wkflow, train)
```

Question8

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

nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(reciped)

nb_fit <- fit(nb_wkflow, train)
```

Question9

Log Reg.

```
log<- bind_cols(predict(log_fit, new_data = train), train%>%dplyr::select(survived))
log_acc <- log %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc
```

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

Same value with using only predict

```
logpred<- predict(log_fit, new_data = train, type = "prob")

loga_acc <- augment(log_fit, new_data = train) %>%
  accuracy(truth = survived, estimate = .pred_class)
loga_acc
```

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

LDA.

```
lin<- bind_cols(predict(lin_fit, new_data = train), train%>%dplyr::select(survived))
lin_acc <- lin %>%
  accuracy(truth = survived, estimate = .pred_class)
lin_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.802
```

QDA.

```
qd<- bind_cols(predict(qd_fit, new_data = train), train%>%dplyr::select(survived))
qd_acc <- qd %>%
  accuracy(truth = survived, estimate = .pred_class)
qd_acc
```

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

Naive Bayes.

```
nb<- bind_cols(predict(nb_fit, new_data = train), train%>%dplyr::select(survived))
nb_acc <- nb %>%
  accuracy(truth = survived, estimate = .pred_class)
nb_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.768
```

Comparing Model Performance

```
accuracies <- c(log_acc$.estimate, lin_acc$.estimate,
                nb_acc$.estimate, qd_acc$.estimate)
models <- c("Logistic Regression", "LDA", "Naive Bayes", "QDA")
results <- tibble(accuracies = accuracies, models = models)
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 4 x 2
##   accuracies models
##        <dbl> <chr>
## 1      0.802 LDA
## 2      0.801 Logistic Regression
## 3      0.784 QDA
## 4      0.768 Naive Bayes
```

Logtistic Regression achieved the highest accuracy.

question10

```
head(predict(log_fit, new_data = test, type = "prob"))
```

```
## # A tibble: 6 x 2
##   .pred_Yes .pred_No
##       <dbl>    <dbl>
## 1     0.631    0.369
## 2     0.110    0.890
## 3     0.776    0.224
## 4     0.495    0.505
## 5     0.237    0.763
## 6     0.755    0.245
```

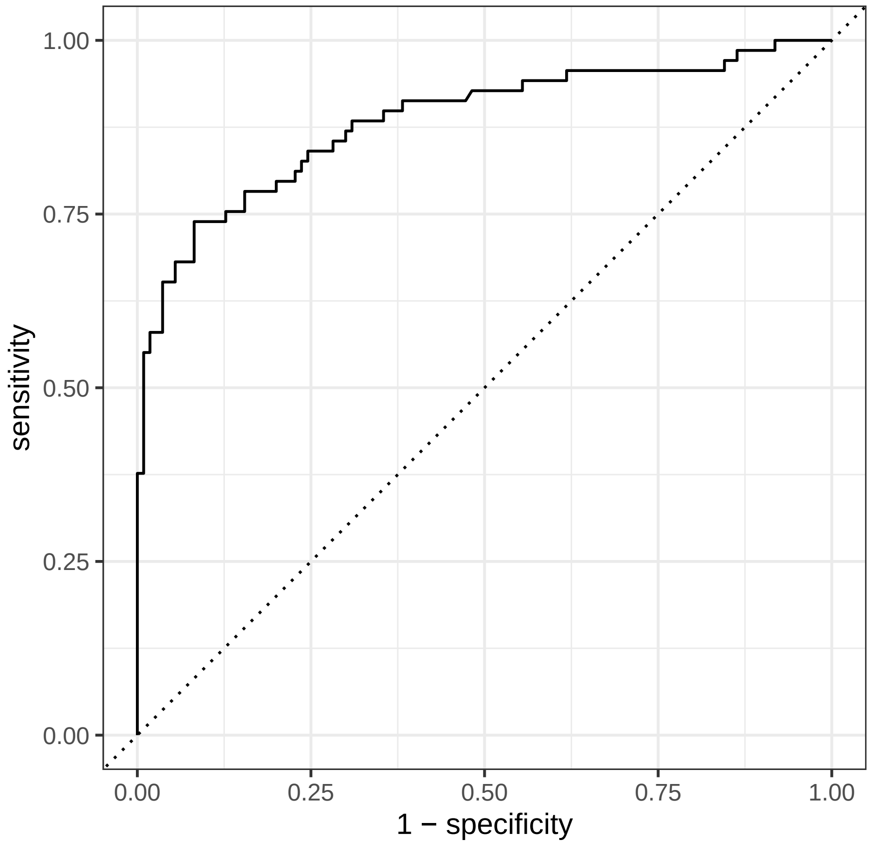
```
augment(log_fit, new_data = test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes No
##        Yes  51 13
##        No   18 97
```

```
multi_metric <- metric_set(accuracy, sensitivity, specificity)

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

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.827
## 2 sensitivity binary         0.739
## 3 specificity binary         0.882
```

```
augment(log_fit, new_data = test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```




```
augment(log_fit, new_data = test) %>%
  accuracy(truth = survived, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.827
```

This model performed well. Testing accuracy is higher than training accuracy. Testing was 0.8268 and training was 0.8005. Because training data set and testing data set are independent, the values differ.