Homework 3

PSTAT 131/231

Contents


```
library(tidyverse)
library(tidymodels)
library(ISLR)
library(ISLR2)
library(discrim)
library(poissonreg)
library(corrr)
library(klaR)

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

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?

```
set.seed(1234)
titanic_split <- initial_split(titanic, prop = 0.80, strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
nrow(titanic)
```

```
## [1] 891
```

```
nrow(titanic_train)  # about 80% of the data
```

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

```
nrow(titanic_test)  # about to be 20% of the data
```

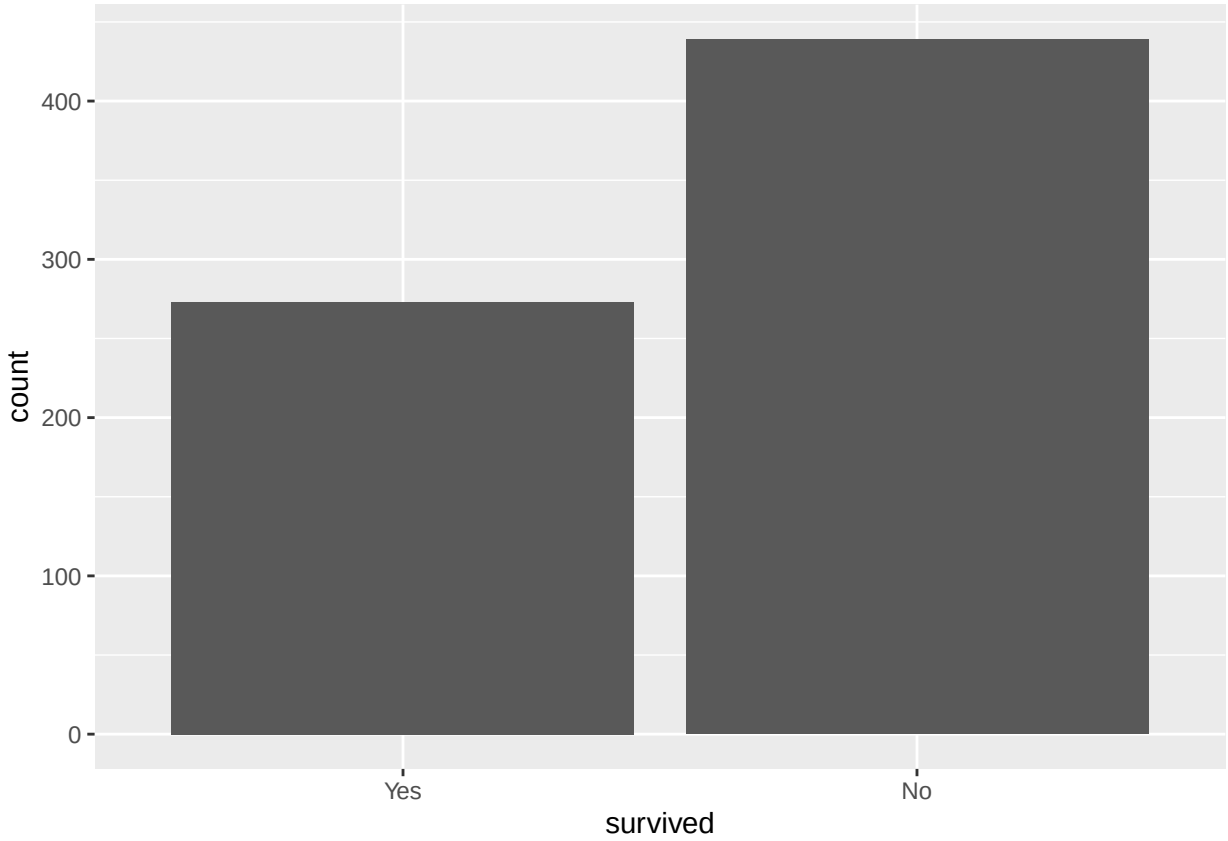
```
## [1] 179
```

There seems to be quite a few NAs in the Cabin and Age variables (just a handful in Embarked as well). This may cause problems for our model and we need to impute values for them if used in the model (or drop them).
It is a good idea to use stratified sampling for this data because one result (did not survive) outnumbers the other (survived) by quite a bit. We want a balanced distribution of these responses so that our model gets enough data on each response type. Thus, if we do not utilize stratified sampling, it may lead to low quality training and testing data.

Question 2

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

```
ggplot(data = titanic_train, aes(x = survived)) + geom_bar()
```



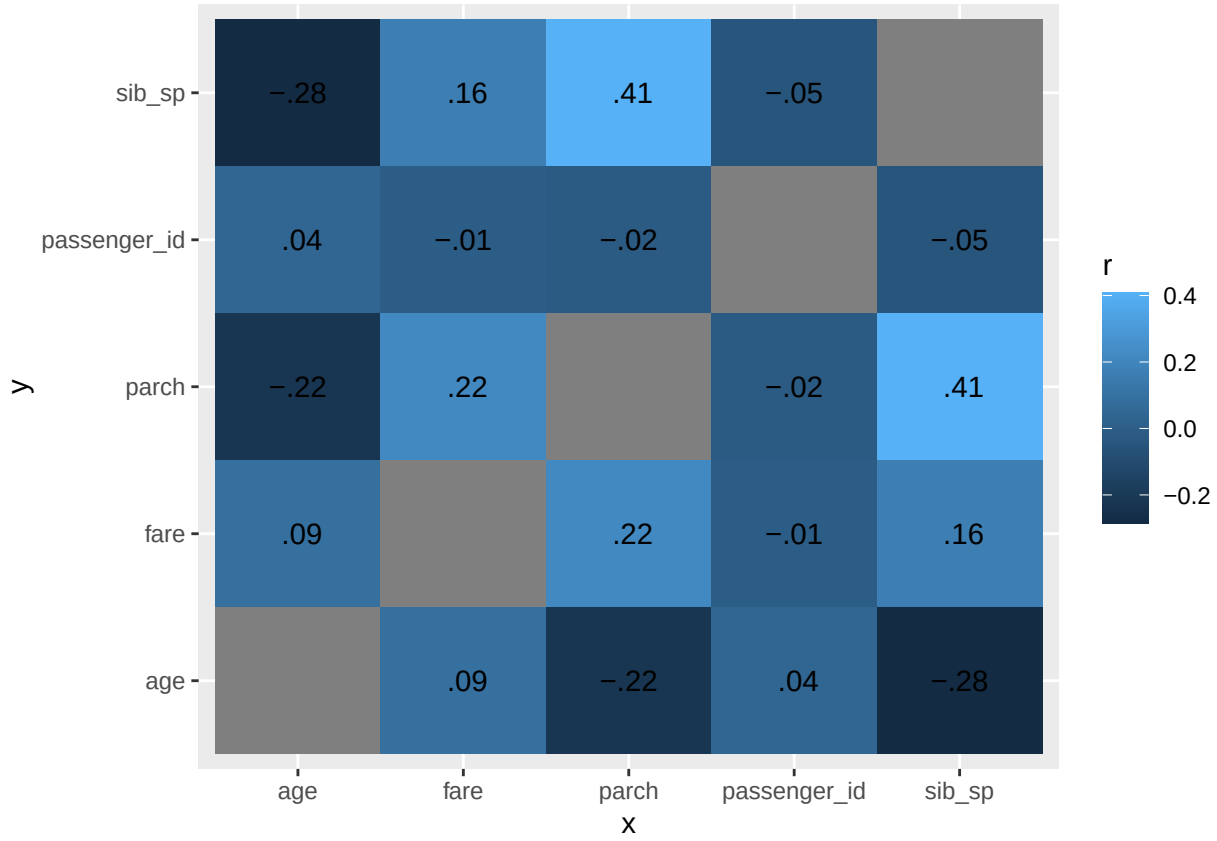
It seems that there are more people who did not survive than survived, with about 200 people more people in the "did not survive" category. Quite an imbalanced distribution.

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 <- titanic_train %>%
  dplyr::select(age, sib_sp, parch, fare, passenger_id) %>%
  correlate()

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



I see that fare & parch, fare & sib_sp, age & fare, and parch & sib_sp are all *positively* correlated with each other. Additionally, I see that age & parch and age & sib_sp are all *negatively* correlated with each other. There are other correlations, but they are very close to zero and aren't too significant.

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, impute_with = imp_vars(all_predictors())) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~starts_with("sex"):fare) %>%
  step_interact(terms = ~ 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.

Hint: Make sure to store the results of `fit()`. You'll need them later on.

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

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

log_titanic_fit <- fit(log_titanic_workflow, titanic_train)
```

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

lda_titanic_fit <- fit(lda_titanic_workflow, 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_mode("classification") %>%
  set_engine("MASS")

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

qda_titanic_fit <- fit(qda_titanic_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_titanic_workflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)

nb_titanic_fit <- fit(nb_titanic_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?

```
# logistic regression
log_titanic_res <- predict(log_titanic_fit, new_data = titanic_train %>%
                             dplyr::select(-survived))
log_titanic_res <- bind_cols(log_titanic_res, new_data = titanic_train %>%
                             dplyr::select(survived))
log_titanic_acc <- accuracy(log_titanic_res, truth = survived, estimate = .pred_class)

# linear discriminant analysis
lda_titanic_res <- predict(lda_titanic_fit, new_data = titanic_train %>%
                             dplyr::select(-survived))
lda_titanic_res <- bind_cols(lda_titanic_res, new_data = titanic_train %>%
                             dplyr::select(survived))
lda_titanic_acc <- accuracy(lda_titanic_res, truth = survived, estimate = .pred_class)

# quadratic discriminant analysis
qda_titanic_res <- predict(qda_titanic_fit, new_data = titanic_train %>%
                             dplyr::select(-survived))
qda_titanic_res <- bind_cols(qda_titanic_res, new_data = titanic_train %>%
                             dplyr::select(survived))
qda_titanic_acc <- accuracy(qda_titanic_res, truth = survived, estimate = .pred_class)

# naive bayes
nb_titanic_res <- predict(nb_titanic_fit, new_data = titanic_train %>%
                             dplyr::select(-survived))
nb_titanic_res <- bind_cols(nb_titanic_res, new_data = titanic_train %>%
                             dplyr::select(survived))
nb_titanic_acc <- accuracy(nb_titanic_res, truth = survived, estimate = .pred_class)
```

```
accuracies <- c(log_titanic_acc$.estimate, lda_titanic_acc$.estimate,
                nb_titanic_acc$.estimate, qda_titanic_acc$.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      0.819 Logistic Regression
## 2      0.803 LDA
## 3      0.789 QDA
## 4      0.784 Naive Bayes
```

The model which achieved the highest accuracy on the training data was logistic regression.

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?

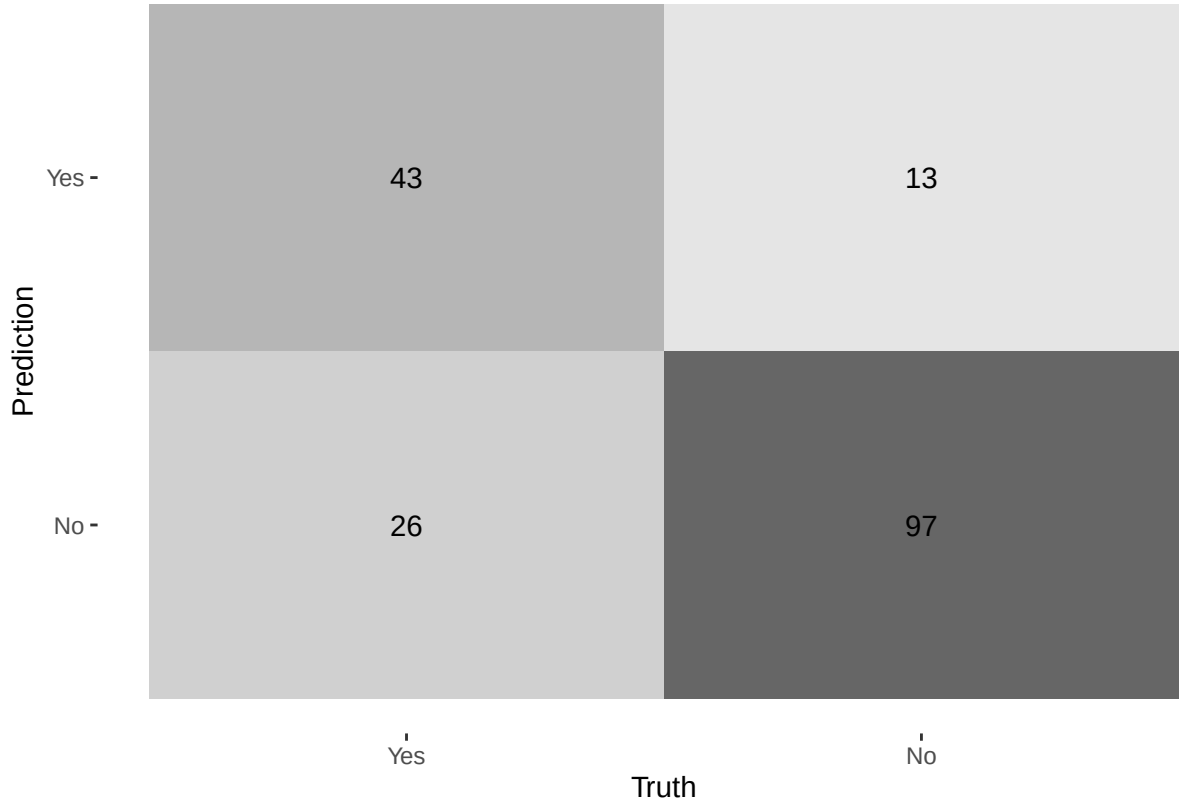
```
# accuracy on testing data
log_titanic_test_res <- predict(log_titanic_fit, new_data = titanic_test %>%
                           dplyr::select(-survived))
log_titanic_test_res <- bind_cols(log_titanic_test_res, new_data = titanic_test %>%
                           dplyr::select(survived))
log_titanic_test_acc <- accuracy(log_titanic_test_res, truth = survived, estimate = .pred_class)
log_titanic_test_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.782
```

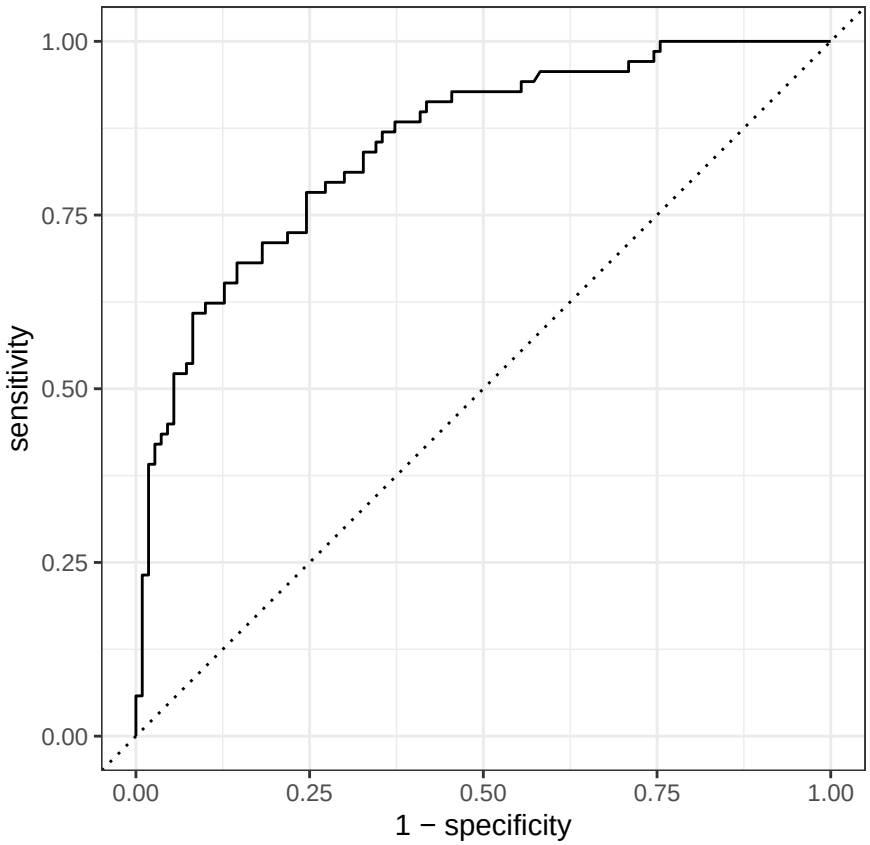
```
# creating the confusion matrix
augment(log_titanic_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##           Truth
## Prediction Yes No
##        Yes  43 13
##        No   26 97
```

```
augment(log_titanic_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
# creating the ROC curve
augment(log_titanic_fit, new_data = titanic_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```




```
augment(log_titanic_fit, new_data = titanic_test) %>%  # calculating AUC
  roc_auc(survived, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.850
```

The accuracy of the logistic regression model on the testing data is 78.21%. The AUC is about 85%. I would say that the model performed quite well since it was able to accurately predict 78.21% of classes on data it has never seen before. Additionally, it is better than the baseline of randomly guessing classes, which is about 50% accuracy. The accuracy of the logistic regression model on the training data is 81.88%. The accuracies are different because, generally, models do not perform as well on testing data compared to training data. This is because the values in the testing data may not have been seen by the model before, so it makes sense that the testing accuracy is lower. Additionally, the model is almost always more suited to the training data as opposed to the testing data since the model uses training data to train.