Classification

Homework 3

Code

AO XU

Hide

```
library(ggplot2)
library(tidymodels)
library(ISLR)
library(ISLR2)
library(discrim)
library(poissonreg)
library(corrr)
library(klaR)
tidymodels_prefer()
```

Classification

For this assignment, we will be working with part of a Kaggle data set (https://www.kaggle.com/c/titanic/overview) that was the subject of a machine learning competition and is often used for practicing ML models. The goal is classification; specifically, to predict which passengers would survive the Titanic shipwreck (https://en.wikipedia.org/wiki/Titanic).

Fig. 1: RMS Titanic departing Southampton on April 10, 1912.

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.

Hide

```
titanic <- read.csv("titanic.csv")
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
```

Hide

```
titanic$survived<-factor(titanic$survived, levels=c('Yes','No'))
titanic$pclass<-factor(titanic$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?

Hide

```
set.seed(1888)
titanic_split<-initial_split(titanic,prop=0.80,
                             strata = survived )
titanic_train<-training(titanic_split)
titanic_test<-testing(titanic_split)
```

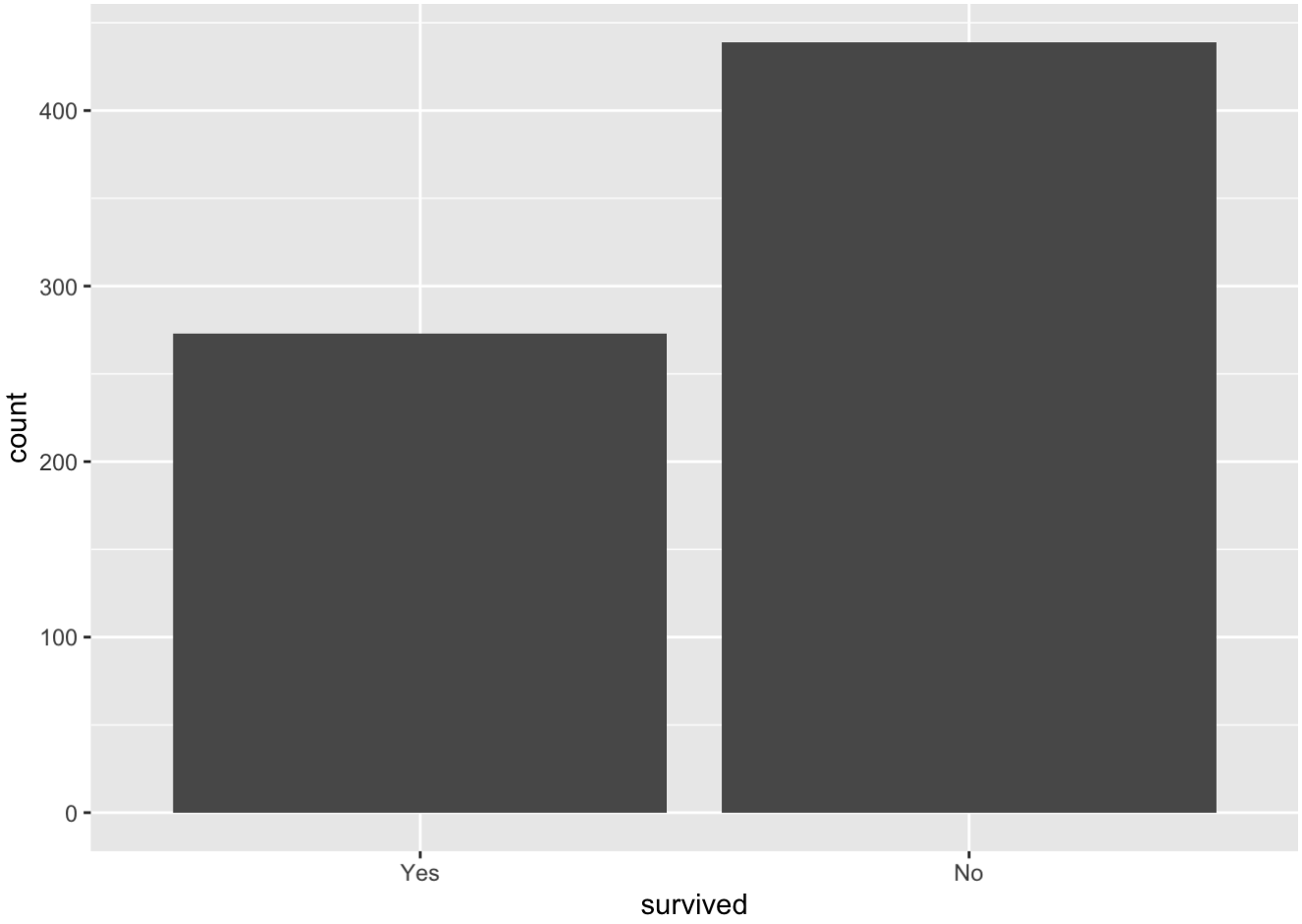
Using stratified sampling for this data can generate representations more accurately of the population.

Question 2

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

Hide

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



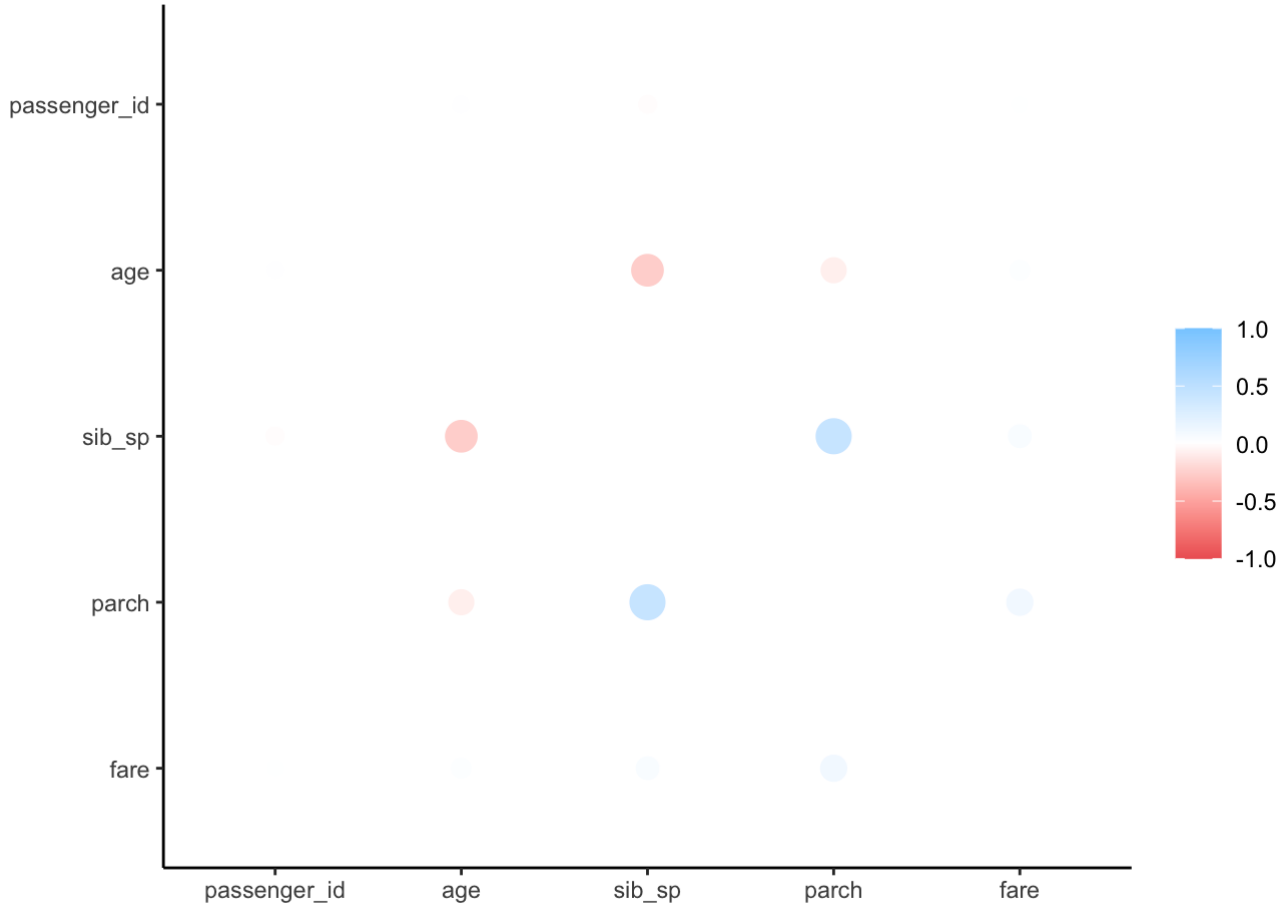
For the distribution of the outcome variable survived, the number of “no” is more than the number of “yes”.

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?

Hide

```
cor_titanic <- titanic_train %>%
  select(is.numeric) %>%
  correlate()
rplot(cor_titanic)
```



(1) sib_sp and parch (2) term and sib_sp (3) term and parch (4) passenger_od and age (5) are positively correlated.

1. term and passenger_id (2) term and age (3) passenger and parch are negatively correlated.

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.

Hide

```
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+
                  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 + age:fare
```

Hide

```
summary(titanic_recipe)
```

```
## # A tibble: 7 × 4
##   variable type    role      source
##   <chr>    <chr>   <chr>     <chr>
## 1 pclass   nominal predictor original
## 2 sex      nominal predictor original
## 3 age      numeric predictor original
## 4 sib_sp   numeric predictor original
## 5 parch    numeric predictor original
## 6 fare     numeric predictor original
## 7 survived nominal outcome   original
```

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.

Hide

```
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)
log_fit <- fit(log_wkflow, titanic_train)
```

Question 6

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

Hide

```
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.

Hide

```
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)
qda_fit <- fit(qda_wkflow, 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`.

Hide

```
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)
nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)
nb_fit <- fit(nb_wkflow, 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?

Hide

```
log_acc <- predict(log_fit, new_data = titanic_train, type = 'class') %>%
  bind_cols(titanic_train %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_acc
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.809
```

Hide

```
lda_acc <- predict(lda_fit, new_data = titanic_train, type = 'class') %>%
  bind_cols(titanic_train %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
lda_acc
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.806
```

Hide

```
qda_acc <- predict(qda_fit, new_data = titanic_train, type = 'class') %>%
  bind_cols(titanic_train %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
qda_acc
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.777
```

Hide

```
nb_acc <- predict(nb_fit, new_data = titanic_train, type = 'class') %>%
  bind_cols(titanic_train %>% select(survived)) %>%
  accuracy(truth = survived, estimate = .pred_class)
nb_acc
```

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

Logistic Regression model achieved the highest accuracy on the training data since 0.8089888 is higher than others.

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?

Hide

```
predict(log_fit, new_data = titanic_test, type = "prob")
```

```
## # A tibble: 179 × 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1     0.946   0.0542
##  2     0.900   0.0996
##  3     0.834   0.166
##  4     0.158   0.842
##  5     0.764   0.236
##  6     0.189   0.811
##  7     0.157   0.843
##  8     0.632   0.368
##  9     0.631   0.369
## 10     0.151   0.849
## # … with 169 more rows
```

Hide

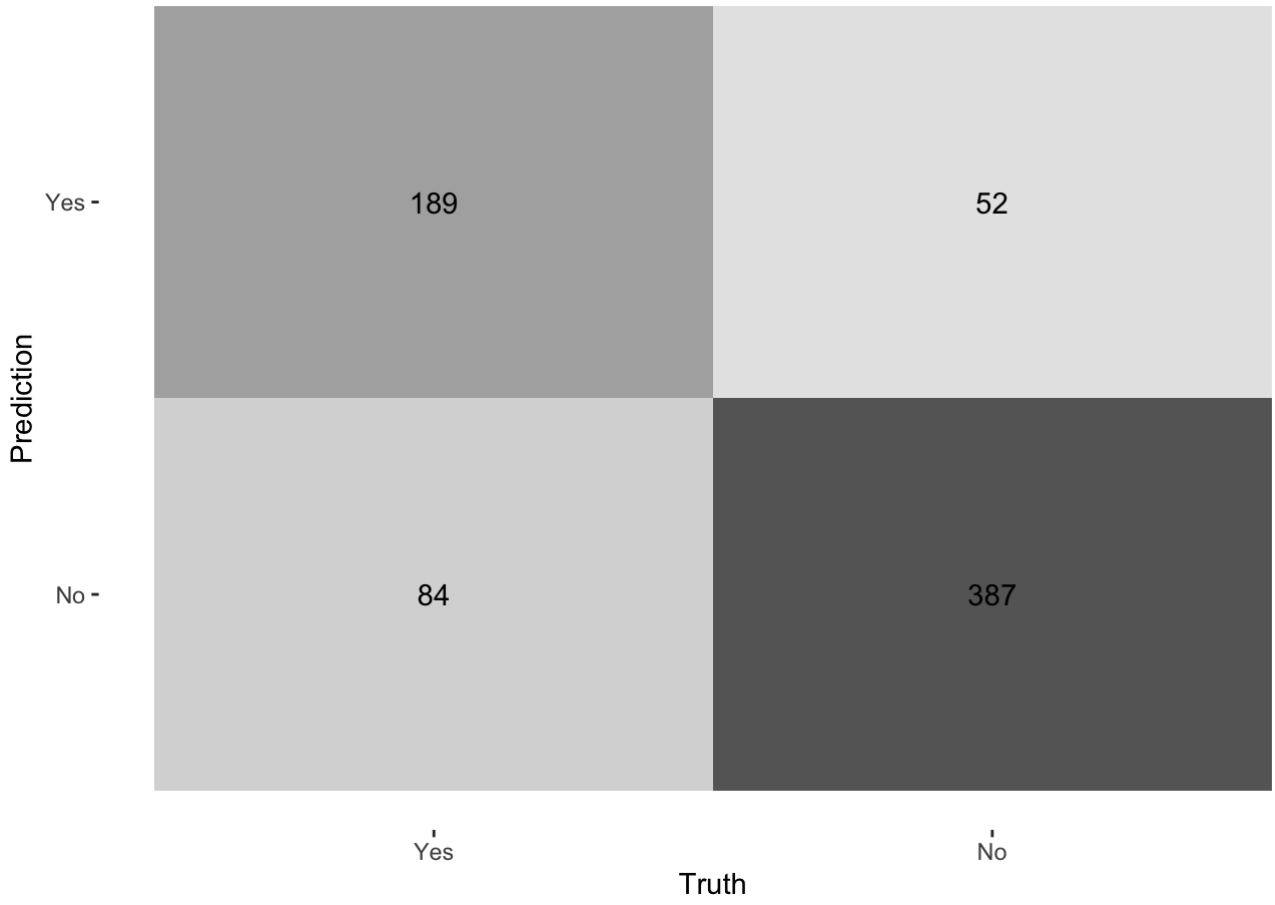
```
multi_metric <- metric_set(accuracy, sensitivity, specificity)

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         0.799
## 2 sensitivity binary         0.652
## 3 specificity binary         0.891
```

The accuracy is 0.7988827.

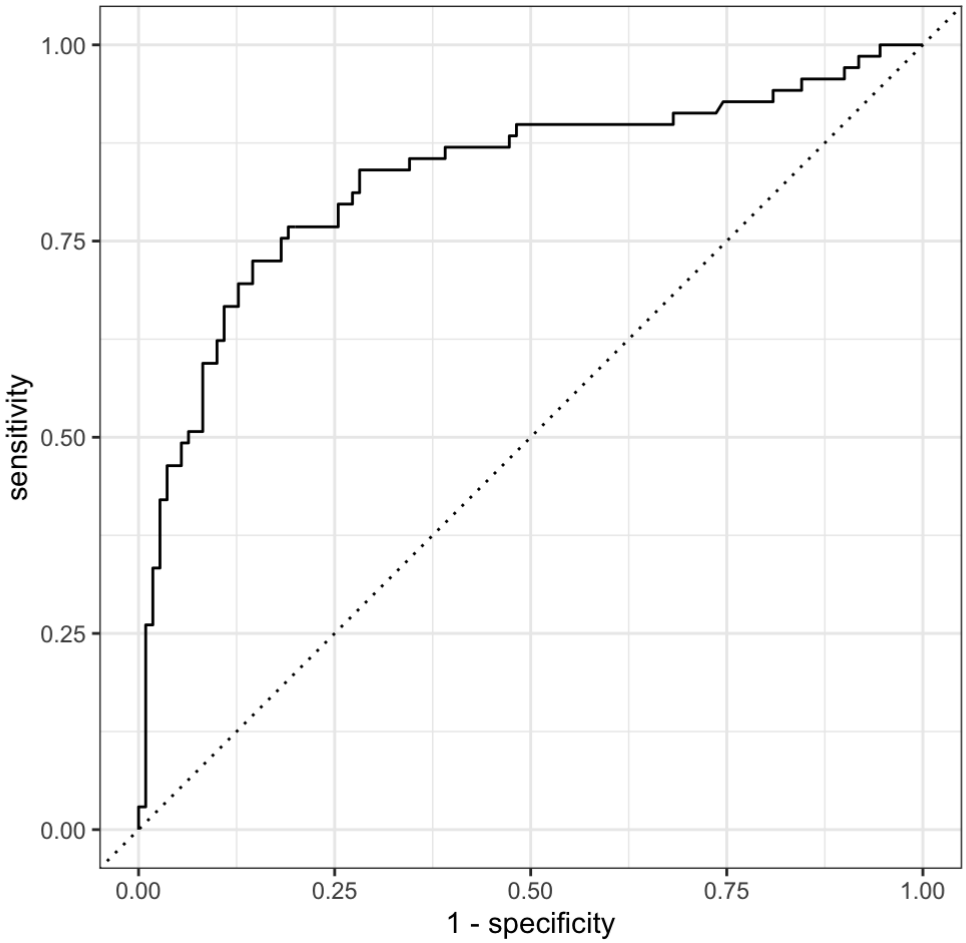
Hide

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



Hide

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



Hide

```
augment(log_fit, new_data = titanic_test) %>%
  roc_auc(survived,.pred_Yes)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.831
```

The model performs accurately. Training accuracy (0.8089888) is little higher than testing accuracy (0.7988827) The value differ might due to that training accuracy means that identical images are used both for training and testing, while test accuracy represents that the trained model identifies independent images that were not used in training.

Required for 231 Students

In a binary classification problem, let p represent the probability of class label 1, which implies that $1 - p$ represents the probability of class label 0. The *logistic function* (also called the "inverse logit") is the cumulative distribution function of the logistic distribution, which maps a real number *z* to the open interval

$(0, 1)$.

Question 11

Given that:

$$p(z) = \frac{e^z}{1 + e^z}$$

Prove that the inverse of a logistic function is indeed the *logit* function:

$$z(p) = ln\left(\frac{p}{1-p}\right)$$

Proof:

p(z) = $\frac{e^z}{1+e^z} \Longrightarrow p(z)(1 + e^z) = e^z \Longrightarrow p(z) + p(z)e^z = e^z \Longrightarrow (1-p)e^z = p \Longrightarrow e^2 = \frac{p}{1-p} \Longrightarrow$
$z = ln(\frac{p}{1-p})$

Therefore, $z(p) = ln(\frac{p}{1-p})$.

Question 12

Assume that $z = \beta_0 + \beta_1 x_1$ and $p = logistic(z)$. How do the odds of the outcome change if you increase x_1 by two? Demonstrate this.

Assume now that β_1 is negative. What value does p approach as x_1 approaches ∞? What value does p approach as x_1 approaches $-\infty$?

Solution:

Odds: $e^{\beta_0+\beta_1 x_1}$

If I increase x_1 by two, then the odds of the outcome would be $e^{\beta_0+\beta_1(x_1+2)} = e^{\beta_0+\beta_1 x_1}e^2$, meaning that it would be e^2 times of original outcome.

p = $\frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$

As x_1 approaches ∞, p = $\frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$ = 0

As x_1 approaches $-\infty$, p = $\frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}}$ = 1