hw3-hope-131

Evan Hope

4/19/2022

In this assignment I will use data from the passengers aboard the Titanic to help make predictions on the number of people that would and would not survive the tragedy given certain pieces of information about the passengers. Additionally, I will use 4 different kinds of models to see which one performs the best. The point of this assignment is to show my introductory skills of the 4 different kinds of models used to make my prediction, the use of correlation as well as confusion matrices, and the use of ROC/AUC plots to determine the accuracy of my models .

Downloading appropriate packages...

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.2     v rsample      1.1.1
## v dials        1.1.0     v tune         1.0.1
## v infer        1.0.4     v workflows    1.1.2
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.3     v yardstick    1.1.0
## v recipes      1.0.3
## -- 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()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggthemes)
library(ISLR2)
library(discrim)
```

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

```
library(poissonreg)
library(corrr)
library(klaR)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:ISLR2':
##
##     Boston
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
tidymodels_prefer()
```

Reading in the data and setting the seed...

```
set.seed(8888)

titanic_data <- read.csv("C:/Users/Ordai/OneDrive/Desktop/School/Stats/PSTAT 131/hw3-131-Hope/titanic.cs
```

We must now change "survived" and "plclass" to factors while also reordering the levels of the survived.

```
# Changing survived variable to a factor and reordering the levels.
titanic_data$survived <- factor(titanic_data$survived, levels = c("Yes", "No"))

# Changing pclass variable to a factor
titanic_data$pclass <- factor(titanic_data$pclass)

# Checking to make sure the above code worked.
is.factor(titanic_data$survived)
```

```
## [1] TRUE
```

```
is.factor(titanic_data$pclass)
```

```
## [1] TRUE
```

Question 1.) splitting the data into training and testing sets...

```
titanic_data_split <- initial_split(titanic_data, prop = 0.70, strata = survived)

titanic_train <- training(titanic_data_split)
titanic_test <- testing(titanic_data_split)
```

Unlike my first assignment, I use stratified sampling.

It is good to use stratified sampling here because we are dealing with qualitative data that can produce different average values.

Question 2.) Distribution of 'survived' variable.

```
titanic_train$survived
```

```
##   [1] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
##  [19] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
##  [37] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
##  [55] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
##  [73] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
##  [91] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [109] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [127] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [145] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [163] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [181] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [199] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [217] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [235] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [253] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [271] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [289] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [307] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [325] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [343] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [361] No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No
## [379] No  No  No  No  No  No  Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [397] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [415] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [433] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [451] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [469] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [487] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [505] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [523] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [541] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [559] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [577] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
```

```
## [595] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## [613] Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes
## Levels: Yes No
```

After observing the training set, we see that the number of survivors is less than the number of non-survivors. Given that there are only two possible outcomes, the survived variable has a binomial distribution with some probability 'p' that the passenger survives.

Question 3.) Correlation Matrix

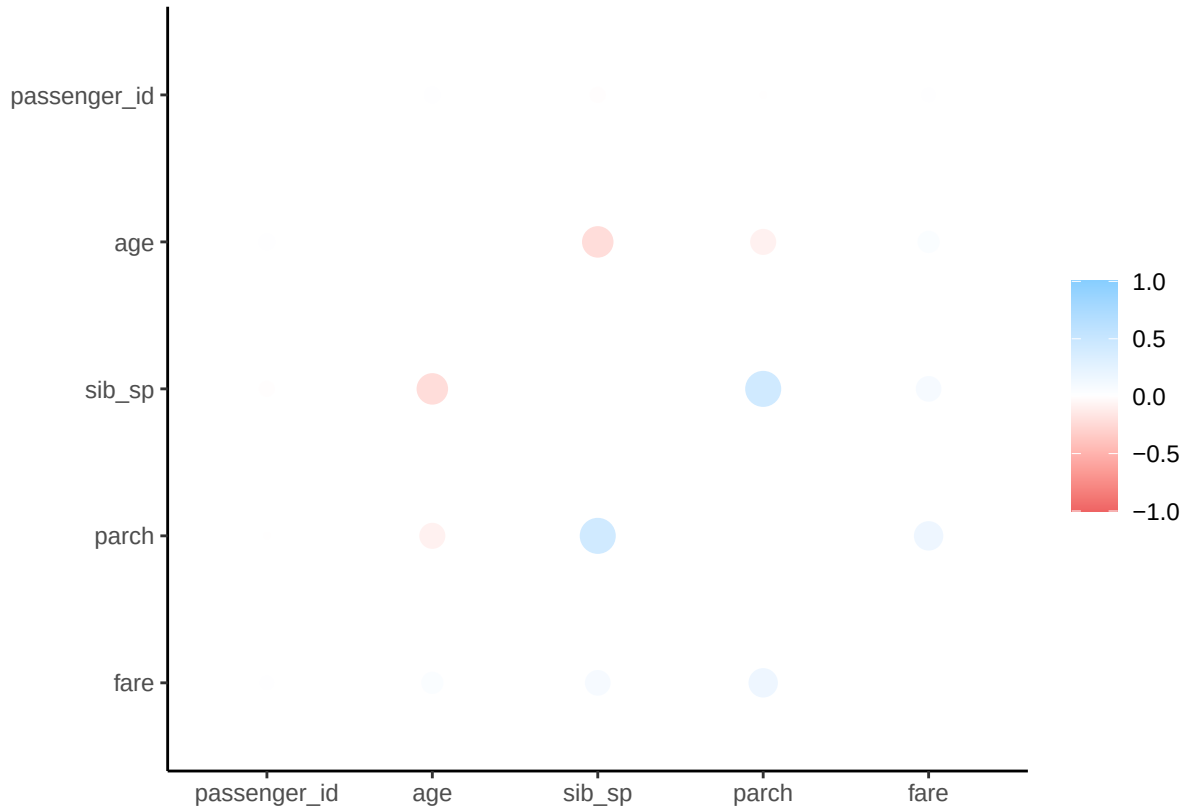
Plotting our correlation matrix for the CONTINUOUS variables only...

```
corrlate_titanic <- titanic_train %>%
  select(-survived, -pclass, -name, -sex, -ticket, -embarked, -cabin) %>%
  correlate()
```

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

```
rplot(corrlate_titanic)
```

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



By the looks of it there are some correlations between a few of the variables. One of the more noticeable

correlations is the relationship between age and the number of siblings/spouses aboard. This makes sense because as the number of siblings/spouse increases, the ages should typically be younger as this indicates a child of a family. If the number of siblings/spouse is low, this could mean that the passenger is only with their spouse indicating that they may be older in age.

Another distinct correlation is the # of siblings/spouse and # of parents/children aboard in the blue direction. In other words, if the number of siblings/spouses of a given passenger is low, then we should also expect to see a low amount of parents/children for that same passenger.

Question 4.) Recipe building.

Below I will create a recipe that imputes missing data, assign dummy variables to all nominal predictors, normalizes all of the numeric variables, and create interactions between the male sex and fare as well as the age and fare.

```
titanic_survival <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train)
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric()) %>%
  step_interact(terms = ~ sex_male:fare) %>%
  step_interact(terms = ~ age:fare)
```

I will now begin creating workflows and fitting four different types of models/analysis starting with....

Question 5.) Logistic Regression

Setting up the engine

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

Setting up the workflow...

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


#Applying the workflow...

titanic_log_fit <- fit(log_wkflow, titanic_train)
```

Lets view the results.

```
titanic_log_fit %>%
  tidy()
```

```
## # A tibble: 10 x 5
##    term            estimate std.error statistic  p.value
##    <chr>              <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)       0.661      0.111     5.97  2.35e- 9
##  2 age               0.746      0.138     5.40  6.79e- 8
##  3 sib_sp            0.394      0.141     2.80  5.04e- 3
```

```
##  4 parch             0.0316     0.113     0.280 7.80e- 1
##  5 fare             -0.0384     0.160    -0.240 8.11e- 1
##  6 pclass_X2         0.565      0.146     3.86  1.15e- 4
##  7 pclass_X3         1.26       0.183     6.86  6.86e-12
##  8 sex_male          1.22       0.114    10.7   8.33e-27
##  9 sex_male_x_fare   0.139      0.152     0.914 3.61e- 1
## 10 age_x_fare       -0.263      0.129    -2.03  4.23e- 2
```

Question 6. 7. and 8.) Repeat 5....

Linear Discriminant Analysis

```
# Engine
linear_disc_analysis <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

#Setting up the workflow...

linear_disc_wkflow <- workflow() %>%
  add_model(linear_disc_analysis) %>%
  add_recipe(titanic_survival)


#Applying the workflow...

titanic_linear_disc_fit <- fit(linear_disc_wkflow, titanic_train)
```

Quadratic Discriminant Analysis

```
# Engine
quad_disc_analysis <- discrim_quad() %>%
  set_engine("MASS") %>%
  set_mode("classification")

#Setting up the workflow...

quad_disc_wkflow <- workflow() %>%
  add_model(quad_disc_analysis) %>%
  add_recipe(titanic_survival)


#Applying the workflow...

titanic_quad_disc_fit <- fit(quad_disc_wkflow, titanic_train)
```

Naive Bayes Model

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

```
#Setting up the workflow...

naive_bayes_wkflow <- workflow() %>%
  add_model(naive_bayes_model) %>%
  add_recipe(titanic_survival)


#Applying the workflow...

titanic_naive_bayes_fit <- fit(naive_bayes_wkflow, titanic_train)
```

Question 9.)

Calculating the accuracy for all 4 models...

```
log_reg_acc <- augment(titanic_log_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

disc_linear_acc <- augment(titanic_linear_disc_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

quad_disc_acc <- augment(titanic_quad_disc_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

naive_bayes_acc <- augment(titanic_naive_bayes_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 26

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 52

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 72

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 87

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 133

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 152
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 158

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 184

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 203

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 225

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 232

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 236

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 241

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 272

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 280

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 290

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 293

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 344

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 353

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 366

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 392

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 406

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 413
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 441

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 448

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 455

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 457

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 467

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 481

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 497

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 558

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 562

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 575

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 580

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 582

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 591

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 613

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 9

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 22

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 26
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 52

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 72

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 87

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 133

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 152

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 158

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 184

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 203

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 225

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 232

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 236

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 241

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 272

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 280

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 290

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 293

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 344
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 353

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 366

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 392

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 406

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 413

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 441

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 448

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 455

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 457

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 467

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 481

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 497

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 558

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 562

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 575

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 580

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 582
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 591
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 613
```

And now comparing...

```
accuracies <- c(log_reg_acc$.estimate, disc_linear_acc$.estimate,
                quad_disc_acc$.estimate, naive_bayes_acc$.estimate)
models <- c("Logistic Regression", "LDA", "QDA", "Naive Bayes")
results <- tibble(accuracy = accuracies, model = models)
results %>%
  arrange(-accuracies)
```

```
## # A tibble: 4 x 2
##   accuracy model
##      <dbl> <chr>
## 1    0.801 Logistic Regression
## 2    0.782 QDA
## 3    0.775 LDA
## 4    0.772 Naive Bayes
```

Based off this display of accuracy scores, logistic regression appears to have the highest accuracy! This may be due to the fact that our response variable only has two possible outcomes. Survived or Not Survived, yes or no, 0 or 1, etc.

Question 10.) Since it scored the highest accuracy I will use the logistic regression model for my predictions using the test data set. Below are my calculated predictions of the probability of each passenger surviving.

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

```
## # A tibble: 268 x 2
##    .pred_Yes .pred_No
##        <dbl>    <dbl>
##  1    0.919    0.0808
##  2    0.630    0.370
##  3    0.919    0.0815
##  4    0.306    0.694
##  5    0.0369   0.963
##  6    0.786    0.214
##  7    0.475    0.525
##  8    0.104    0.896
##  9    0.473    0.527
## 10    0.211    0.789
## # ... with 258 more rows
```

Putting it into a confusion matrix...

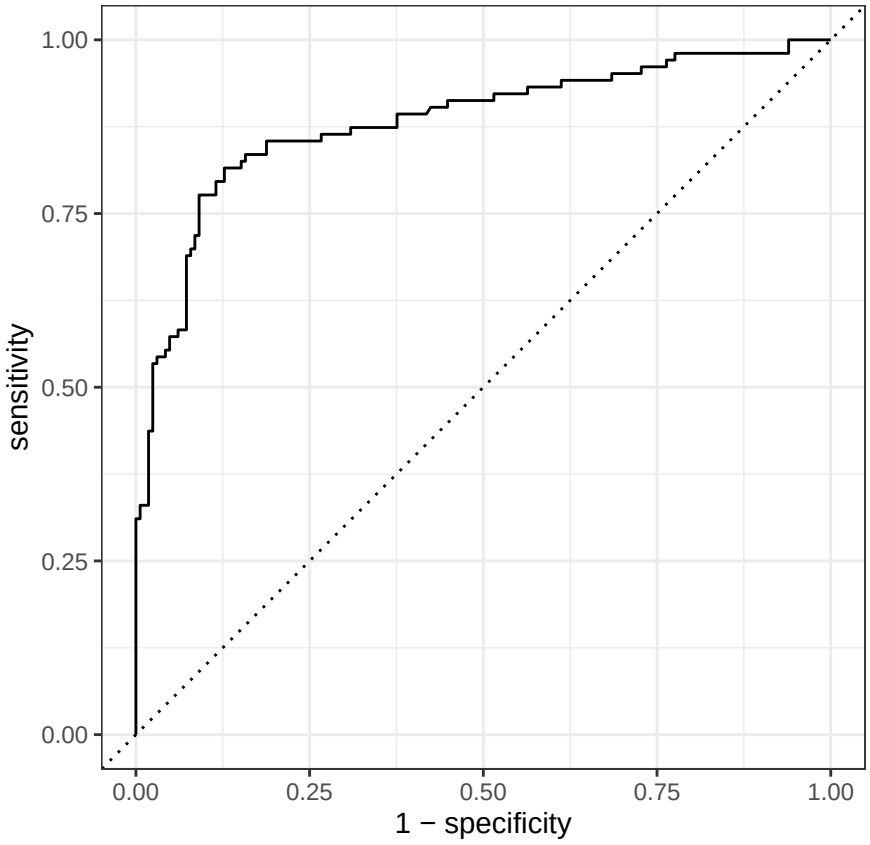
```
augment(titanic_log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)
```

```
##            Truth
## Prediction Yes  No
##        Yes  82  21
##        No   21 144
```

Now we will look at the ROC curve and find the area under the curve.

First: ROC plot

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



Now for the AUC:

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

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.882
```

As we can see, our testing accuracy (0.882) was higher than our training accuracy (0.801) ! The reason I think this happens is because there are missing values in the data that could be distorting it a bit. Also, I believe there is some repetitive data in the set or passengers with very similar data.