Pstat131 Hw3

Noe Arambula ID: 3561131

2022-04-13

Contents

Loading libraries

```
library(discrim)
library(ggplot2)
library(tidyverse)
library(tidymodels)
library(corrplot)
library(ggthemes)
tidymodels_prefer()
library(readr)
library(corrr)
library(klaR)
library(MASS)
```

Read in csv data file and set seed

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

```
## Rows: 891 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): survived, name, sex, ticket, cabin, embarked
## dbl (6): passenger_id, pclass, age, sib_sp, parch, fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
set.seed(777)
```

Changing survived and pclass to factors

```
titanic$survived =  factor(titanic$survived, levels = c("Yes", "No")) # Note can use parse_factor() in
titanic$pclass =  factor(titanic$pclass)

class(titanic$survived)
```

```
## [1] "factor"
```

```
class(titanic$pclass)
```

```
## [1] "factor"
```

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.

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

head(titanic_train)
```

```
## # A tibble: 6 x 12
##   passenger_id survived pclass name  sex     age sib_sp parch ticket  fare cabin
##          <dbl> <fct>    <fct>  <chr> <chr> <dbl>  <dbl> <dbl> <chr>  <dbl> <chr>
## 1            5 No       3      Alle~ male     35      0     0 373450  8.05 <NA>
## 2            7 No       1      McCa~ male     54      0     0 17463  51.9  E46
## 3            8 No       3      Pals~ male      2      3     1 349909 21.1  <NA>
## 4           13 No       3      Saun~ male     20      0     0 A/5. ~  8.05 <NA>
## 5           14 No       3      Ande~ male     39      1     5 347082 31.3  <NA>
## 6           15 No       3      Vest~ fema~    14      0     0 350406  7.85 <NA>
## # ... with 1 more variable: embarked <chr>
```

There are some missing values for age which could pose an issue. There are also missing values for cabin however this would mean they did not have a cabin could change cabin to true or false

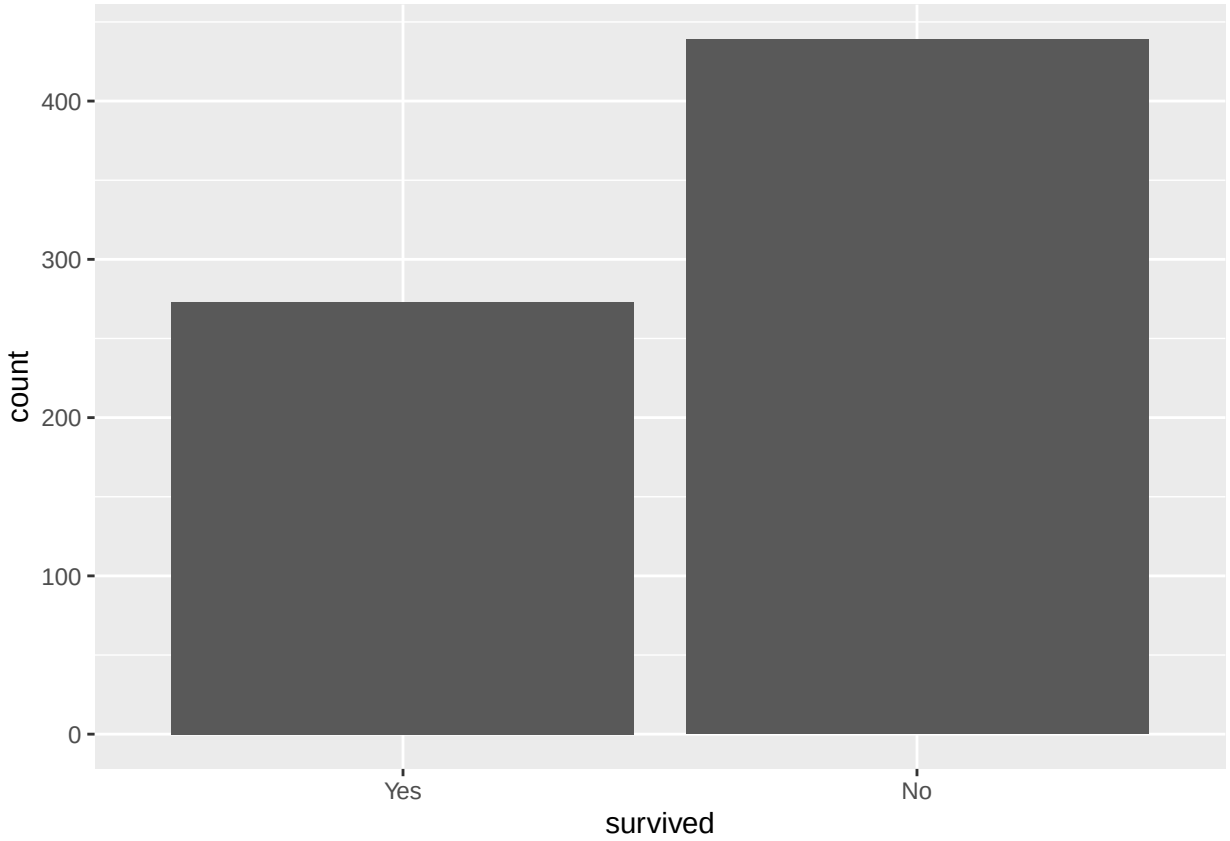
Why is it a good idea to use stratified sampling for this data?.

It is a good idea to use stratified sampling for this data because survived has 2 different levels and we want a proportionate amount of yes and no's for the data otherwise it could be skewed towards did not survive or not have any survived observations(yes value for survived)

Question 2

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

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



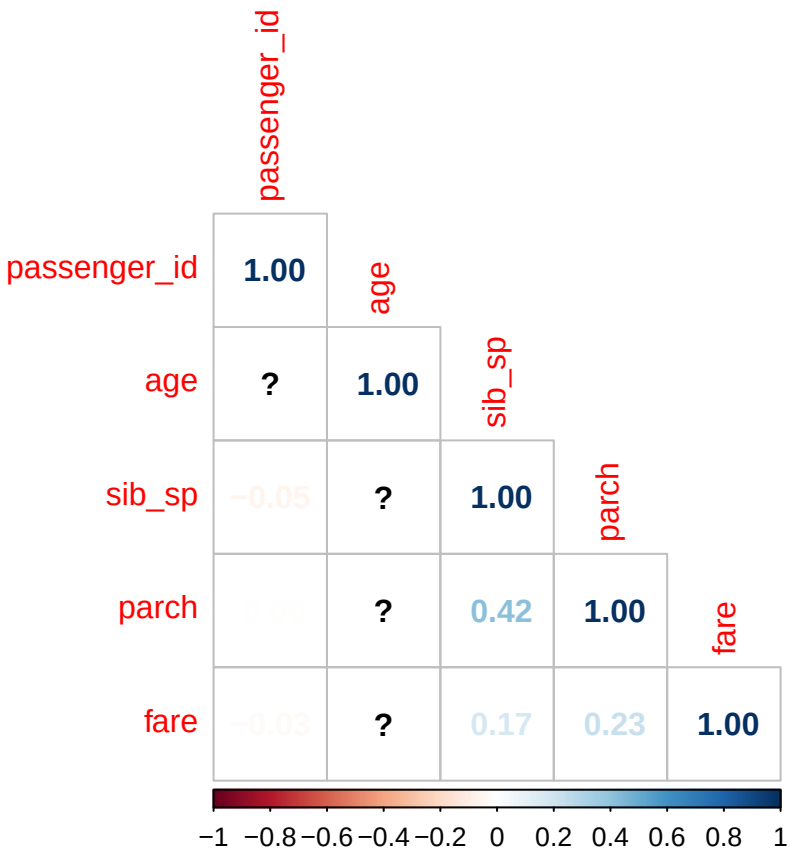
There are almost double the amount of people that did not survive then did survive i.e. the amount of No(did not survive) value is almost twice as big as Yes value

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?

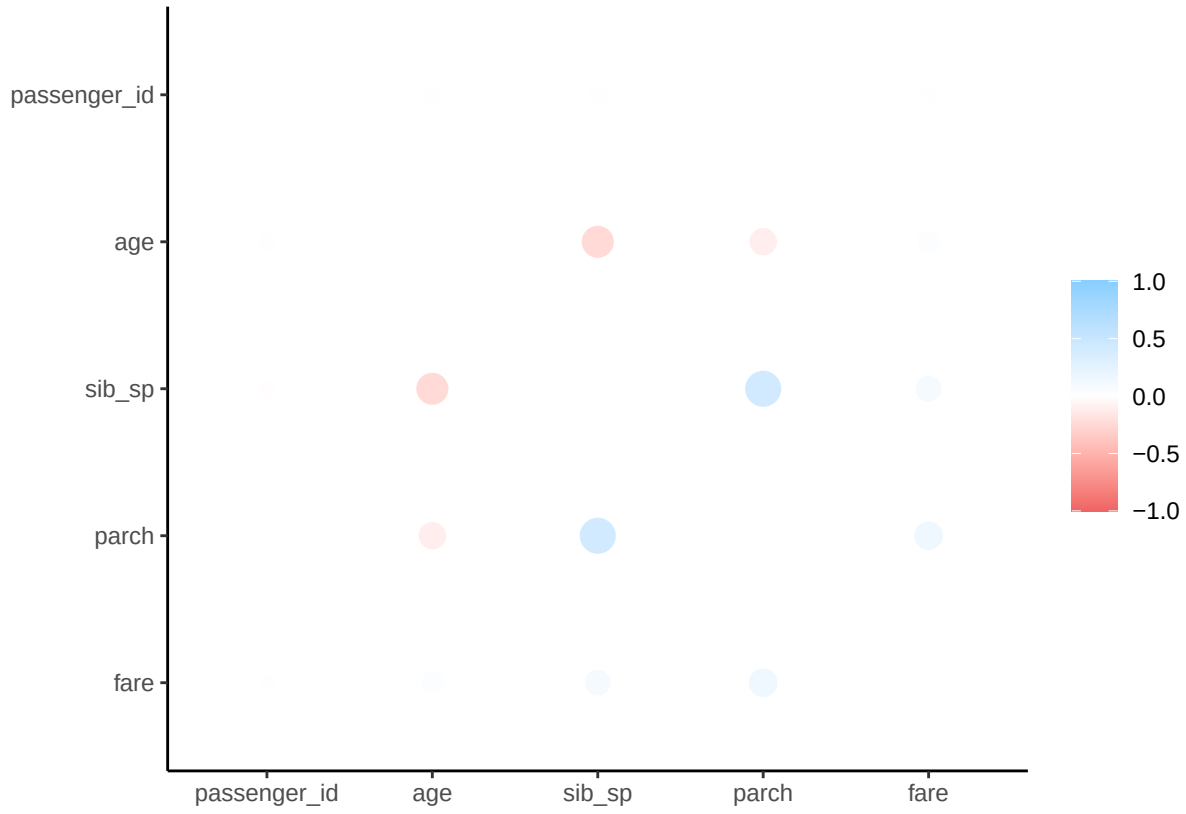
Correlation Matrix

```
titanic_train %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(type = 'lower',
           method = 'number')
```



Visualization of Correlation Matrix

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



Age and sib_sp are negatively correlated and sib_sp and parch are positively 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.

```
titanic_recipe <- recipe(survived ~ pclass + sex + age + sib_sp + parch + fare, data = titanic_train) %
  step_impute_linear(age) %>%  # imputes age to fix missing values
  step_dummy(all_nominal_predictors()) %>%   # creates dummy variables
  step_interact(terms = sex_male ~ fare) %>%    # Note: had to use sex_male since sex was made into a d
  step_interact(terms = age ~ fare)  # Interactions created

titanic_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          6
##
## Operations:
##
## Linear regression imputation for age
## Dummy variables from all_nominal_predictors()
## Interactions with sex_male, 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.

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

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

 # creating workflow
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)

 # using fit to apply workflow to training data
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.

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

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

 # using fit to apply workflow to training data
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.

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

 # creating workflow
qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(titanic_recipe)

 # using fit to apply workflow to training data
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`.

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

 # creating workflow
nb_wkflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(titanic_recipe)

 # using fit to apply workflow to training data
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.

Logistic Regression Model Predictions + Accuracy

```
titanic_log_pred <- predict(log_fit, new_data = titanic_train %>% select(-survived))
```

```
titanic_log_pred = bind_cols(titanic_log_pred, titanic_train %>% select(survived))

titanic_log_pred %>%
  head
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 Yes         No
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.810
```

Linear Discriminant Analysis Model Predictions + Accuracy

```
titanic_lda_pred <- predict(lda_fit, new_data = titanic_train %>% select(-survived))

titanic_lda_pred = bind_cols(titanic_lda_pred, titanic_train %>% select(survived))

titanic_lda_pred %>%
  head()
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 Yes         No
```

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

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

Quadratic Discriminant Analysis Model Predictions + Accuracy

```
titanic_qda_pred <- predict(qda_fit, new_data = titanic_train %>% select(-survived))

titanic_qda_pred = bind_cols(titanic_qda_pred, titanic_train %>% select(survived))

titanic_qda_pred %>%
  head()
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 Yes         No
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.803
```

Naive Bayes Model Predictions + Accuracy

```
titanic_nb_pred <- predict(nb_fit, new_data = titanic_train %>% select(-survived))

titanic_nb_pred = bind_cols(titanic_nb_pred, titanic_train %>% select(survived))

titanic_nb_pred %>%
  head()
```

```
## # A tibble: 6 x 2
##   .pred_class survived
##   <fct>       <fct>
## 1 No          No
## 2 No          No
## 3 No          No
## 4 No          No
## 5 No          No
## 6 Yes         No
```

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

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.798
```

Which model achieved the highest accuracy on the training data?

The model that achieved the highest accuracy on the training data was the logistic Regression model with an accuracy of 0.81 or 81%

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).

Fitting and Accuracy

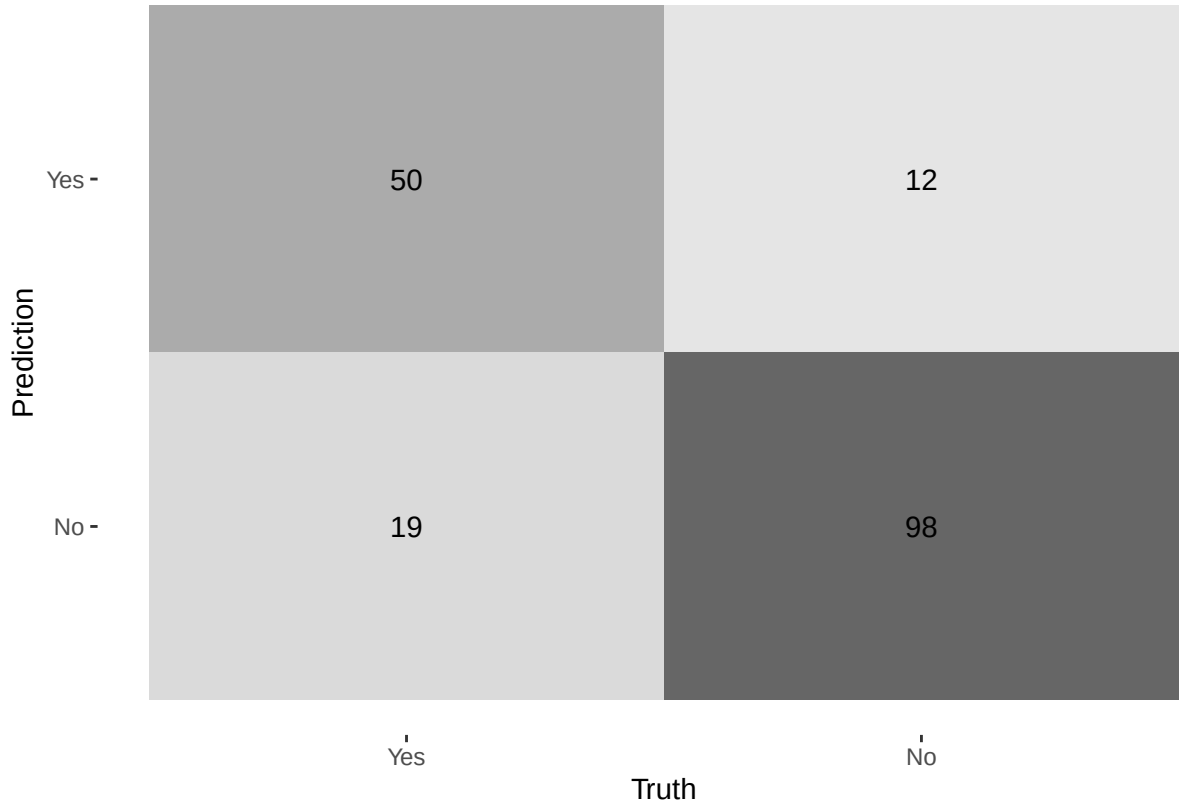
```
# Fitting logistic model (model w/ highest accuracy) to testing data
log_testing_fit <- fit(log_wkflow, titanic_test)

  # Accuracy on testing data
log_testing_acc <- augment(log_testing_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
log_testing_acc
```

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

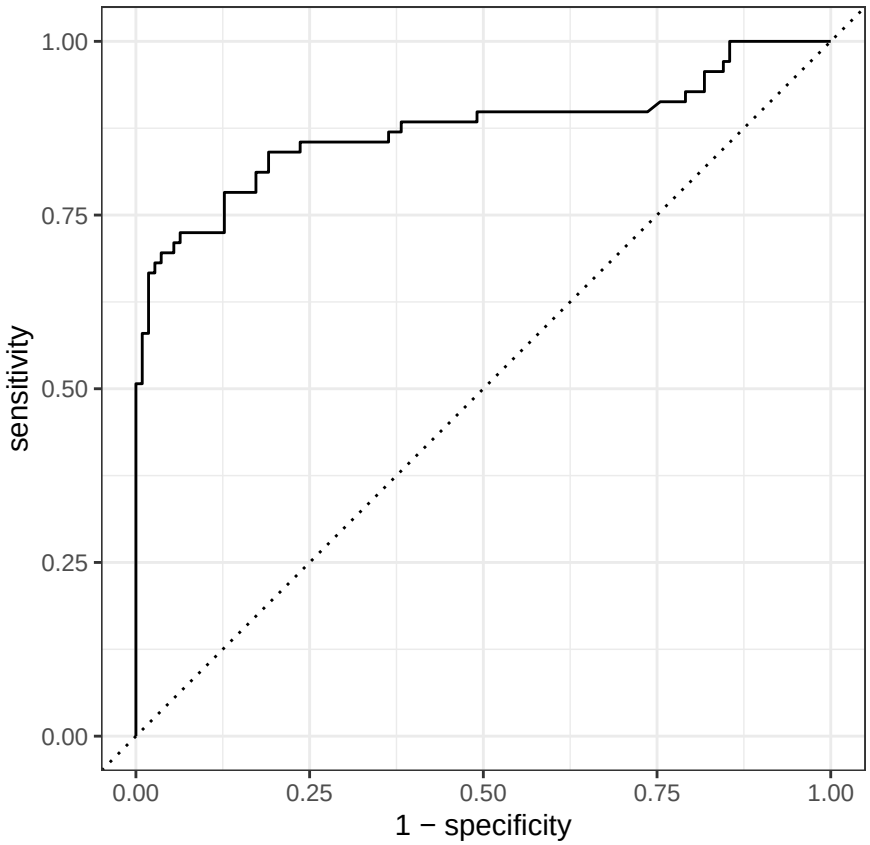
Confusion Matrix and Visualization

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



ROC Curve and AUC Calculation

```
# Plot ROC curve
augment(log_testing_fit, new_data = titanic_test) %>%
  roc_curve(survived, .pred_Yes) %>%
  autoplot()
```




```
# Calculate AUC
augment(log_testing_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         0.873
```

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

The model performed fairly well at predicting those who did survive as the ROC curve is closer to the top left of the graph indicating a good performance if it was close to the diagnol dotted line the predictions would be no better than random guessing.

In addition, we have an AUC of 87.3% which is pretty good as well as the higher it is the better, AUC tells how accurate the models predictions are.

When we take a look at the confusion matrix above we can see that we have an overwhelming amount of true positives/negatives compared to a few false positives/negatives which is very good

```
log_testing_acc
```

```
## # A tibble: 1 x 3
```

```
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.827
```

```
log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.810
```

The testing accuracy was a little higher at 0.827 or 82.7% compared to the training accuracy of 81%

The reason this might have been higher is that the training data might have just had points that were a little better suited for our model since we stratified the testing set it makes sense that the accuracy would be similar to both sets some variation.