PSTAT131_HW3

Tao Wang

2022-04-13

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.

```
# load packages
library('tidyverse')
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
library('tidymodels')
```

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

## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      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

## -- 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()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library('corrr')
library('ISLR') # For the Smarket data set
library('ISLR2') # For the Bikeshare data set
```

```
##
## Attaching package: 'ISLR2'

## The following objects are masked from 'package:ISLR':
##
##     Auto, Credit
```

```
library('discrim')
```

```
##
## Attaching package: 'discrim'

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

```
library('poissonreg')
library('klaR') # for naive bayes
```

```
## 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
```

```
# set a seed
set.seed(1234)

# read the csv
titanic<- read.csv("data/titanic.csv")

# change variable `survived` and `pclass` to factors
titanic$survived<-as.factor(titanic$survived)
titanic$pclass<- as.factor(titanic$pclass)

# change `survived` to a factor,  *"Yes"* is the first level.
titanic$survived<-factor(titanic$survived, ordered = TRUE, levels = c("Yes", 'No'))
# check the levels of variable `survived`
levels(titanic$survived)
```

```
## [1] "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?

Answer:

- 1. Split the data, stratifying on the outcome variable, `survived.`

```r
# 1. Split the data, stratifying on the outcome variable, `survived.`
titanic_split <- initial_split(titanic, prop = 0.80,
                               strata = survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
```

- 2. Verify that the training and testing data sets have the appropriate number of observations.

```r
# 2. Verify that the training and testing data sets
# have the appropriate number of observations.
nrow(titanic) # the answer is 891
```

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

```r
nrow(titanic_train) # the answer is 712
```

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

```r
nrow(titanic_test) # the answer is 179
```

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

Since $891 * 0.80 = 712.8$ and $891 * 0.20 = 178.2$, then we can see that the results we get from our runs are similar to those of our theoretical calculations. So, we can conclude that the training and testing data sets have the appropriate number of observations.

- 3. Take a look at the training data and note any potential issues, such as missing data.

```r
# take a look at the first few rows of the training data
head(titanic_train)
```

```
##    passenger_id survived pclass
## 1             1       No      3
## 6             6       No      3
## 7             7       No      1
## 13           13       No      3
## 19           19       No      3
## 21           21       No      2
##                                                    name    sex age sib_sp
## 1                               Braund, Mr. Owen Harris   male  22      1
## 6                                      Moran, Mr. James   male  NA      0
## 7                               McCarthy, Mr. Timothy J   male  54      0
## 13                       Saundercock, Mr. William Henry   male  20      0
## 19 Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele) female  31      1
## 21                                 Fynney, Mr. Joseph J   male  35      0
##    parch    ticket    fare cabin embarked
## 1      0 A/5 21171  7.2500  <NA>        S
## 6      0    330877  8.4583  <NA>        Q
## 7      0     17463 51.8625   E46        S
## 13     0 A/5. 2151  8.0500  <NA>        S
## 19     0    345763 18.0000  <NA>        S
## 21     0    239865 26.0000  <NA>        S
```

```r
# sum of missing data
sum(is.na(titanic_train)) # 696
```

```
## [1] 690
```

```r
sum(is.na(titanic_train$passenger_id)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$survived)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$pclass)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$name)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$sex)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$age)) # 139
```

```
## [1] 136
```

```r
sum(is.na(titanic_train$sib_sp)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$parch)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$ticket)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$fare)) # 0
```

```
## [1] 0
```

```r
sum(is.na(titanic_train$cabin)) # 556
```

```
## [1] 552
```

```r
sum(is.na(titanic_train$embarked)) # 1
```

```
## [1] 2
```

Based on what we got here, we can see that there are some potential issues. There are 696 observations contain missing value. To be more specific, for variable age, there are 139 missing values; for variable cabin, there are 556 missing value; for variable embarked there are 1 missing value. In addition, we can see that the ticket has different format. These potential issues may cause problem.

- 4. Why is it a good idea to use stratified sampling for this data? (We haven't learned it yet.)

```r
table(titanic$survived)
```

```
##
## Yes  No
## 342 549
```

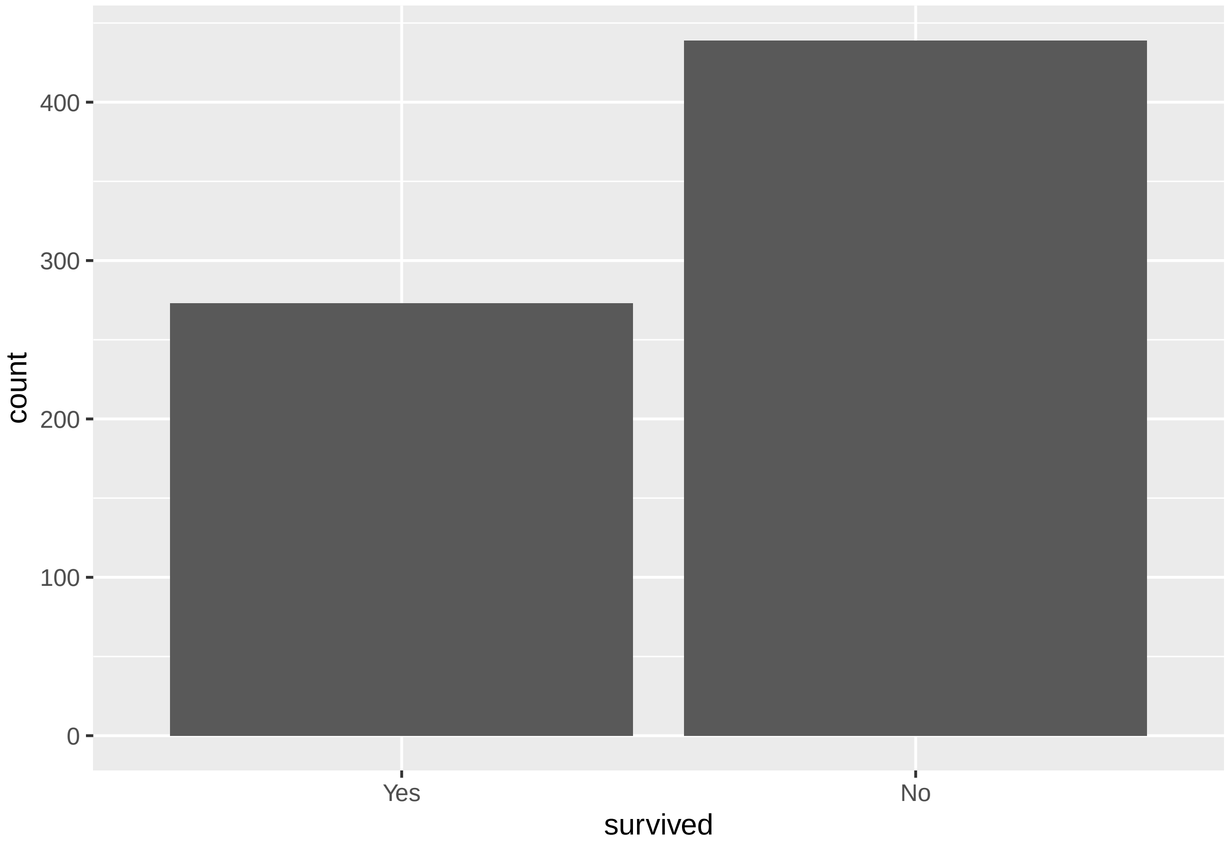
It is a good idea to use stratified sampling for this data because our goal is to predict which passengers would survive the Titanic shipwreck, and the response variable `survived` is categorical.

Variable `survived` has two levels: one is 'Yes', and the second one is 'No'. Each level has enough observations. In this case, stratification may produce a smaller error of estimation than would be produced by a simple random sample of the same size. By the professor, stratified sampling can also make sure that the distributions of different levels of the response variable are the same. This is why we want to use stratified sampling here.

Question 2

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

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



```
table(titanic_train$survived)
```

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

According to the graph, we know that there are more people didn't survive than did survive in the **training** data set. There are 273 people survived, and 439 people didn't survive.

Question 3

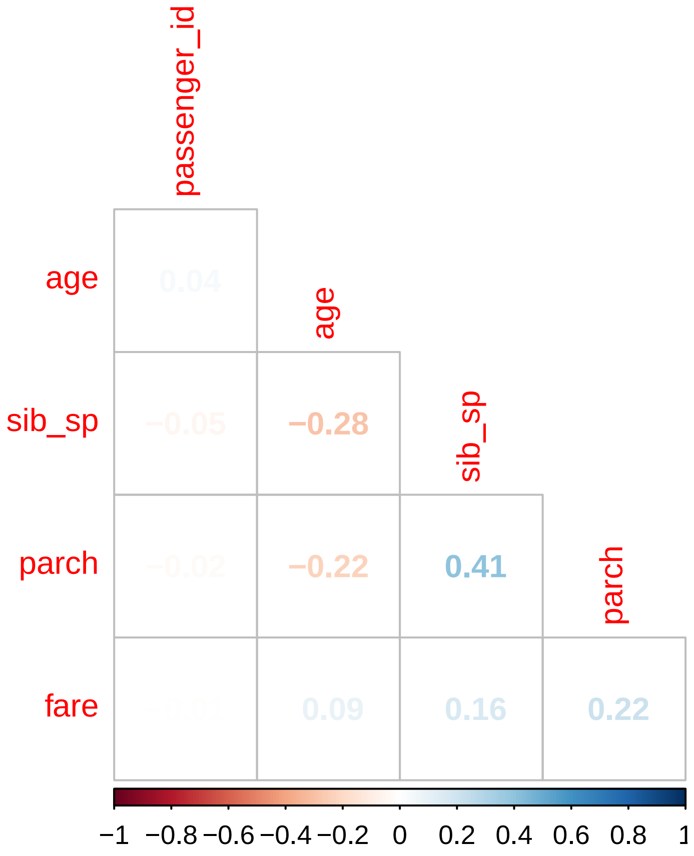
Using the **training** data set, create a correlation matrix of all continuous variables. Create a visualization of the matrix, and describe any patterns you see. Are any predictors correlated with each other? Which ones, and in which direction?

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
# create a correlation matrix of all continuous variables.
# Create a visualization of the matrix
titanic_train %>%
  dplyr::select(is.numeric) %>%
  cor(method = "pearson", use = "pairwise.complete.obs" ) %>%
  corrplot(type = 'lower', diag = FALSE,
           method = 'number')
```

```
## Warning: Predicate functions must be wrapped in `where()`.
##
##   # Bad
##   data %>% select(is.numeric)
##
##   # Good
##   data %>% select(where(is.numeric))
##
## i Please update your code.
## This message is displayed once per session.
```



* Are any predictors correlated with each other? Which ones, and in which direction?

According the matrix we got, we can know that:

- age are negatively correlated with sib_sp
- age are negatively correlated with parch
- age are positively correlated with fare
- sib_sp are positively correlated with parch
- sib_sp are positively correlated with fare *parch are positively correlated with fare

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.

```
# Using the **training** data, create a recipe predicting the outcome variable `survived`.
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)%>%
step_interact(terms = ~ age:fare)

titanic_recipe # check the titanic_train_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
## 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.

```
# Specify a **logistic regression** model for classification using the `"glm"` engine.
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

# Then create a workflow. Add your model and the appropriate recipe.
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(titanic_recipe)
```

```r
#  Finally, use `fit()` to apply my workflow to the **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.

```r
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.

```r
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`.

```r
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?

```
# Use `predict()` and `bind_cols()` to generate predictions using
# each of these 4 models and your **training** data.
# according to the professor and TA, we use let type = 'class' here.
log_predict <- predict(log_fit, new_data = titanic_train, type = "class")

lda_predict<-predict(lda_fit, new_data =  titanic_train, type = "class")

qda_predict<-predict(qda_fit, new_data = titanic_train, type = "class")

nb_predict<-predict(nb_fit, new_data = titanic_train, type = "class")

titanic_train_predictions <- bind_cols(log_predict,lda_predict,qda_predict,nb_predict)
```

```
## New names:
## * `.pred_class` -> `.pred_class...1`
## * `.pred_class` -> `.pred_class...2`
## * `.pred_class` -> `.pred_class...3`
## * `.pred_class` -> `.pred_class...4`
```

```
titanic_train_predictions %>% head()
```

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

```
# Then use the *accuracy* metric to assess the performance of each of the four models.
log_reg_acc <- augment(log_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)

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

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

nb_acc <- augment(nb_fit, new_data = titanic_train) %>%
  accuracy(truth = survived, estimate = .pred_class)
accuracies <- c(log_reg_acc$.estimate, lda_acc$.estimate,
                nb_acc$.estimate, qda_acc$.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
```

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?

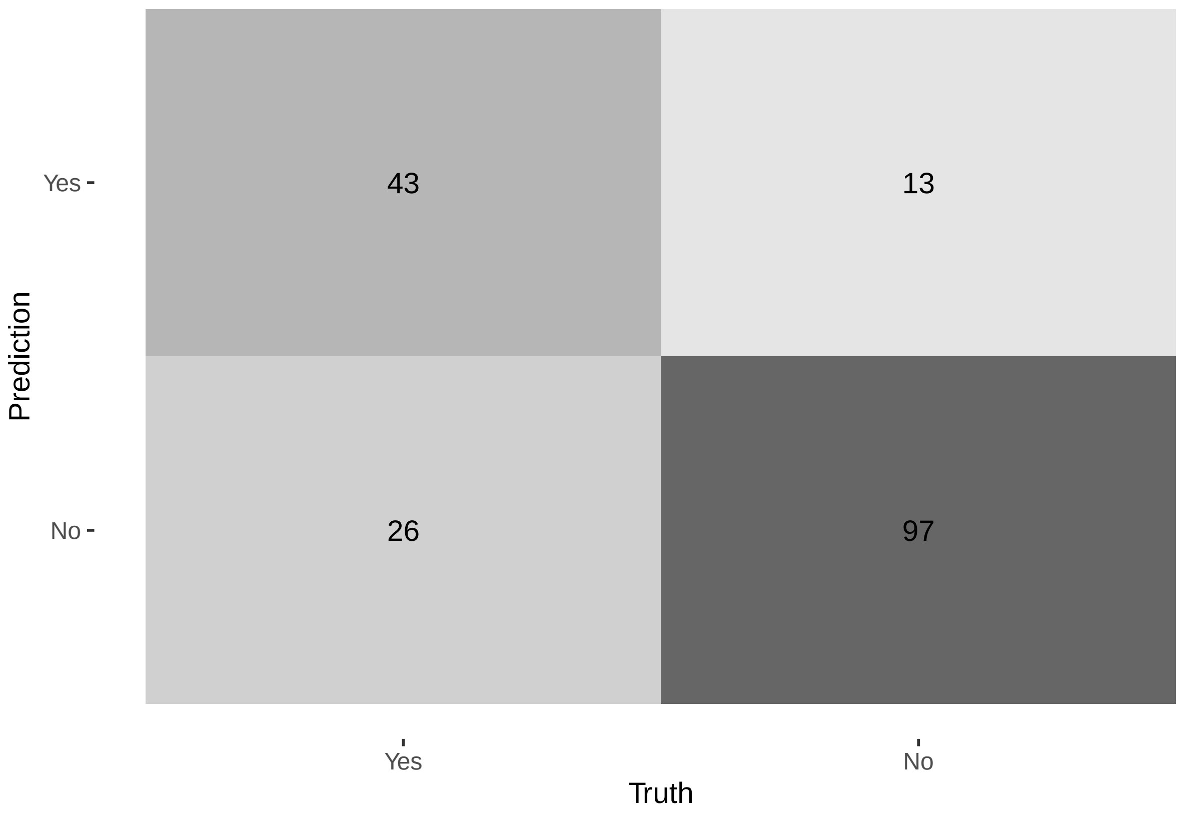
```
# Fit the model with the highest training accuracy to the **testing** data.
predict(log_fit, new_data = titanic_test, type = 'class')
```

```
## # A tibble: 179 x 1
##    .pred_class
##    <fct>
##  1 Yes
##  2 No
##  3 No
##  4 Yes
##  5 No
##  6 Yes
##  7 No
##  8 No
##  9 Yes
## 10 Yes
## # ... with 169 more rows
```

```
# Report the accuracy of the model on the **testing** data.
new_acc <- augment(log_fit, new_data = titanic_test) %>%
  accuracy(truth = survived, estimate = .pred_class)
new_acc
```

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

```
# Again using the **testing** data, create a confusion matrix and visualize it.
augment(log_fit, new_data = titanic_test) %>%
  conf_mat(truth = survived, estimate = .pred_class)%>%
  autoplot(type = "heatmap")
```

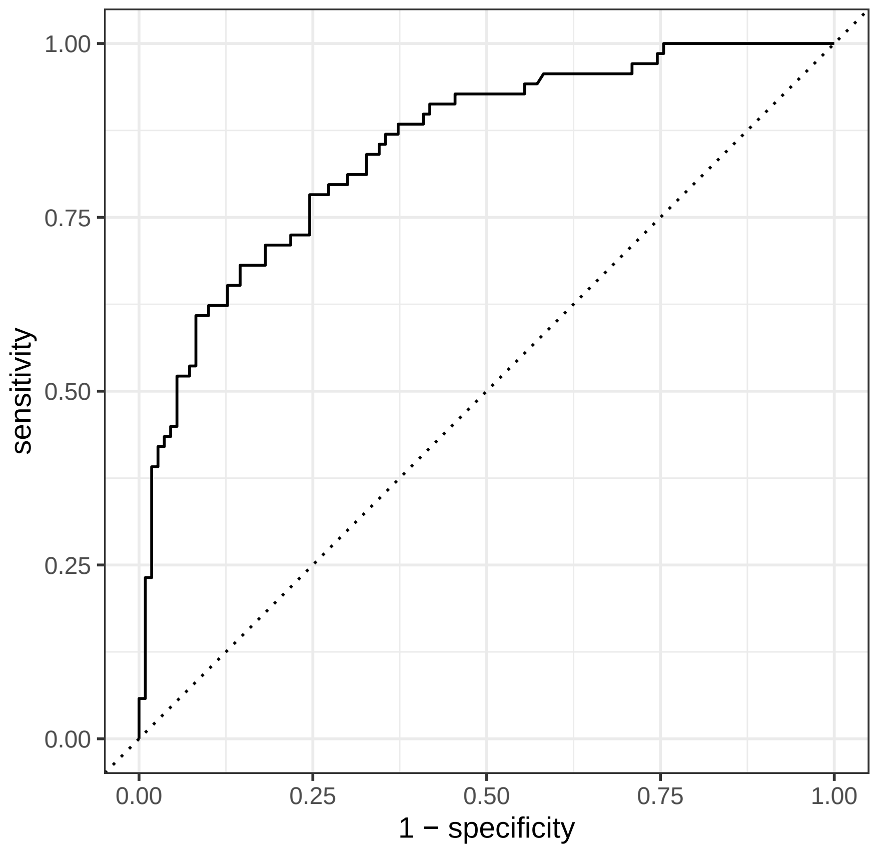


```r
# In order to help us to understand ROC better
# we  we add two other metrics, sensitivity and specificity, out of curiosit
multi_metric <- metric_set(accuracy, sensitivity, specificity)

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

```
## # A tibble: 3 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.782
## 2 sensitivity binary         0.623
## 3 specificity binary         0.882
```

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



```
# Calculate the area under it (AUC).
augment(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         0.850
```

- How did the model perform?
- Well, the AUC we got is about 0.8502 (close to 1). By what we learned about AUC, we can conclude that our model perform good. According to the professor, an AUC above 0.85 means high classification accuracy.
- Compare its training and testing accuracies.

The training accuracy is 0.818820. The testing accuracy is 0.7821229. We can that The testing accuracy is smaller than the training accuracy. They are different.

- If the values differ, why do you think this is so?
- According to the lecture, we know that the testing accuracy is smaller than the training accuracy because we use 80% of data in the training dataset and 20% in the testing dataset. When we use more data, our model will be more accurate. I think it will be a very important reason why our testing accuracy is smaller than our training accuracy.