Homework 2

Alice Ding

2023-10-05

Overview

In this homework assignment, we will work through various classification metrics. We will be asked to create functions in R to carry out the various calculations. You will also investigate some functions in packages that will let we obtain the equivalent results. Finally, we will create graphical output that also can be used to evaluate the output of classification models, such as binary logistic regression.

Exercise 1

First, we will import the data.

```
df <- read.csv('https://raw.githubusercontent.com/addsding/data621/main/homework2/classification-output-
head(df)
```

```
##   pregnant glucose diastolic skinfold insulin  bmi pedigree age class
## 1        7     124        70       33     215 25.5    0.161  37     0
## 2        2     122        76       27     200 35.9    0.483  26     0
## 3        3     107        62       13      48 22.9    0.678  23     1
## 4        1      91        64       24       0 29.2    0.192  21     0
## 5        4      83        86       19       0 29.3    0.317  34     0
## 6        1     100        74       12      46 19.5    0.149  28     0
##   scored.class scored.probability
## 1            0         0.32845226
## 2            0         0.27319044
## 3            0         0.10966039
## 4            0         0.05599835
## 5            0         0.10049072
## 6            0         0.05515460
```

Exercise 2

The data set has three key columns we will use:

- `class`: the actual class for the observation
- `scored.class`: the predicted class for the observation (based on a threshold of 0.5)
- `scored.probability`: the predicted probability of success for the observation

Now, we will use the table() function to get the raw confusion matrix for this scored dataset.

```r
data <- df[, c("class", "scored.class")]
confusion_matrix <- table(data)
confusion_matrix
```

```
##      scored.class
## class   0   1
##     0 119   5
##     1  30  27
```

Within this new confusion matrix, `class` represents the actual class (rows) while `scored.class` represents the predicted class (the columns).

We can read this as:

- 119 true negative observations (TN)
- 5 false positive observations (FP)
- 30 false negative observations (FN)
- 27 true positive observations (TP)

Exercise 3

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the accuracy of the predictions.

```r
accuracy_function <- function(x) {
  TP <- sum(x$class == 1 & x$scored.class == 1)
  TN <- sum(x$class == 0 & x$scored.class == 0)
  round((TP + TN)/nrow(x), 4)
}

accuracy_calc <- accuracy_function(data)
accuracy_calc
```

```
## [1] 0.8066
```

The accuracy is 0.8066.

Exercise 4

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the classification error rate of the predictions.

```r
classification_error_function <- function(x) {
  FP <- sum(x$class == 0 & x$scored.class == 1)
  FN <- sum(x$class == 1 & x$scored.class == 0)
  round((FP + FN)/nrow(x), 4)
}

classification_error_calc <- classification_error_function(data)
classification_error_calc
```

```
## [1] 0.1934
```

The error is 0.1934.

Next, we'll Verify that the accuracy error rate sum to one.

```
classification_error_calc + accuracy_calc
```

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

Exercise 5

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the precision of the predictions.

```
precision_function <- function(x) {
  TP <- sum(x$class == 1 & x$scored.class == 1)
  FP <- sum(x$class == 0 & x$scored.class == 1)
  round(TP/(TP + FP), 4)
}

precision_calc <- precision_function(data)
precision_calc
```

```
## [1] 0.8438
```

The precision is 0.8438.

Exercise 6

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the sensitivity of the predictions. Sensitivity is also known as recall.

```
sensitivity_function <- function(x) {
  TP <- sum(x$class == 1 & x$scored.class == 1)
  FN <- sum(x$class == 1 & x$scored.class == 0)
  round(TP/(TP + FN), 4)
}

sensitivity_calc <- sensitivity_function(data)
sensitivity_calc
```

```
## [1] 0.4737
```

The sensitivity is 0.4737.

Exercise 7

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the specificity of the predictions.

```r
specificity_function <- function(x) {
  TN <- sum(x$class == 0 & x$scored.class == 0)
  FP <- sum(x$class == 0 & x$scored.class == 1)
  round(TN/(TN + FP), 4)
}

specificity_calc <- specificity_function(data)
specificity_calc
```

```
## [1] 0.9597
```

The specificity is 0.9597.

Exercise 8

Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the F1 score of the predictions.

```r
f1_function <- function(x) {
  p <- precision_function(x)
  s <- sensitivity_function(x)
  return(round((2 * p * s)/(p + s), 4))
}

f1_calc <- f1_function(data)
f1_calc
```

```
## [1] 0.6068
```

The F1 score is 0.6068.

Exercise 9

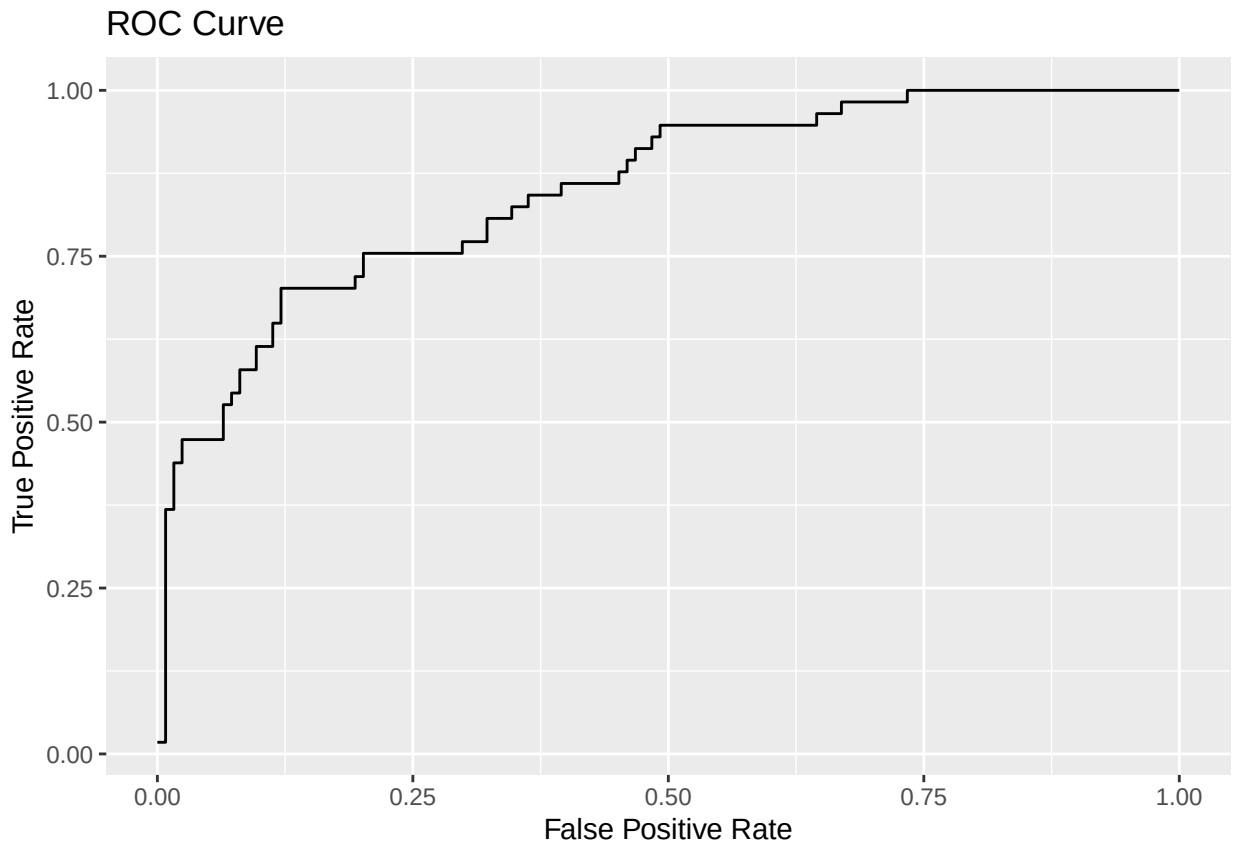
Before we move on, let's consider a question that was asked: What are the bounds on the F1 score? Show that the F1 score will always be between 0 and 1. (Hint: If $0 < a < 1$ and $0 < b < 1$ then $ab < a$.)

> So both `precision` and `sensitivity`, which are used to calculate the F1 score, are bounded between 0 and 1. Given the restriction on these two fields, the F1 score will also be between 0 and 1.

Exercise 10

Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area under the curve (AUC). Note that I recommend using a sequence of thresholds ranging from 0 to 1 at 0.01 intervals.

```r
library(ggplot2)

roc_function <- function(class, scored_prob) {
  # sort the class and scored_prob in descending order of probabilities
  order_idx <- order(scored_prob, decreasing = TRUE)
  class <- class[order_idx]
  scored_prob <- scored_prob[order_idx]

  n <- length(class)

  # initialize variables for ROC curve
  TP <- 0
  FP <- 0
  TPR <- numeric(n)
  FPR <- numeric(n)

  # initialize AUC vector
  auc_vector <- numeric(n)

  # iterate through the data
  for (i in 1:n) {
    if (class[i] == 1) {
      TP <- TP + 1
    } else {
      FP <- FP + 1
    }

    TPR[i] <- TP / sum(class == 1)
    FPR[i] <- FP / sum(class == 0)

    if (i > 1) {
      auc_vector[i] <- 0.5 * (FPR[i] - FPR[i - 1]) * (TPR[i] + TPR[i - 1])
    }
  }

  # plot the ROC curve
  plot_roc <- ggplot(data.frame(FPR = FPR, TPR = TPR)) +
    geom_line(aes(x = FPR, y = TPR)) +
    xlab("False Positive Rate") + ylab("True Positive Rate") +
    ggtitle("ROC Curve")

  # return plot and auc as a vector
  return(list(roc_plot = plot_roc, auc = auc_vector))
}

roc_data <- roc_function(df$class, df$scored.probability)
roc_data$roc_plot
```




```
auc_calc <- sum(roc_data$auc)
auc_calc
```

```
## [1] 0.8503113
```

The AUC is 0.8503.

Exercise 11

Use your created R functions and the provided classification output data set to produce all of the classification metrics discussed above.

```
metrics <- c(accuracy_calc, classification_error_calc, f1_calc, precision_calc, sensitivity_calc, speci
names(metrics) <- c("Accuracy", "Error", "F1", "Precision", "Sensitivity", "Specificity")

data.frame(metrics)
```

```
##             metrics
## Accuracy     0.8066
## Error        0.1934
## F1           0.6068
## Precision    0.8438
## Sensitivity  0.4737
## Specificity  0.9597
```

Exercise 12

Investigate the caret package. In particular, consider the functions confusionMatrix, sensitivity, and specificity. Apply the functions to the data set. How do the results compare with your own functions? 13. Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
factors <- data |>
  select(scored.class, class) |>
  mutate(scored.class = as.factor(scored.class),
         class = as.factor(class))

matrix <- confusionMatrix(factors$scored.class, factors$class, positive = "1")

caret_package <- c(matrix$overall["Accuracy"]
                   , matrix$byClass["Sensitivity"]
                   , matrix$byClass["Specificity"]
                   , matrix$byClass["Precision"]
                   , matrix$byClass["F1"]
                   )
written_function <- c(accuracy_function(data)
                      , sensitivity_function(data)
                      , specificity_function(data)
                      , precision_function(data)
                      , f1_function(data)
                      )
comparison <- cbind(caret_package, written_function)
data.frame(comparison)
```

```
##             caret_package written_function
## Accuracy        0.8066298           0.8066
## Sensitivity     0.4736842           0.4737
## Specificity     0.9596774           0.9597
## Precision       0.8437500           0.8438
## F1              0.6067416           0.6068
```

The results are very close – they only vary after 3-4 decimal points for most calculations so I would say they are comparable, it's just rounding that makes them slightly off.