Homework 2

Classification Metrics

Group 2

3/10/2021

Contents

Assignment Overview . . . 1

Task 1: Downloading Data . . . 2

Task 2: Raw Confusion Matrix . . . 2

Functions . . . 3

Task 3: Accuracy Function . . . 3

Task 4: Classification Error Rate Function . . . 4

Task 5: Precision Function . . . 5

Task 6: Sensitivity Function . . . 6

Task 7: Specificity Function . . . 6

Task 8: F1 Score Function . . . 7

Task 9: F1 Score Bounds . . . 8

Task 10: ROC curve . . . 9

Task 11: Classification metrics . . . 11

Task 12: Caret Package . . . 12

Task 13: pROC Package . . . 13

Group 2 members: *Alice Friedman, Diego Correa, Jagdish Chhabria, Orli Khaimova, Richard Zheng, Stephen Haslett.*

Assignment Overview

In this homework assignment, you will work through various classification metrics. You 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 you obtain the equivalent results. Finally, you will create graphical output that also can be used to evaluate the output of classification models, such as binary logistic regression.

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

Task 1: Downloading Data

Download the classification output data set.

```
data_raw <- read.csv("https://raw.githubusercontent.com/Jagdish16/CUNY_DATA_621/main/homework_2/classifi
```

Task 2: Raw Confusion Matrix

Use the table() function to get the raw confusion matrix for this scored dataset. Make sure you understand the output. In particular, do the rows represent the actual or predicted class? The columns?

```
# For the confusion matrix, we are only interested in the class and scored.class variables,
# so we select only these variables and ignore the rest.
confusion_matrix_table <- data_raw %>%
  select(scored.class, class)

# For readability purposes, rename 'scored.class' to Predicted, and 'class' to Actual.
dplyr::rename(confusion_matrix_table, Predicted = scored.class, Actual = class) %>%
    # Convert numeric boolean values to human readable values.
    mutate(Predicted = recode(Predicted,
                               '0' = 'Negative',
                               '1' = 'Positive'),
         Actual = recode(Actual,
                        '0' = 'Negative',
                        '1' = 'Positive')) %>%
    table()
```

```
##           Actual
## Predicted  Negative Positive
##   Negative      119       30
##   Positive        5       27
```

Functions

```
# We only need the class and scored.class variables from the dataset so we
# extract them and leave everything else.
data <- data_raw %>%
  select(class, scored.class)
```

Task 3: Accuracy Function

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

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

```
accuracy <- function(df,col1,col2) {
  true = df[,col1]
  predict = df[,col2]
  # total events
  len = length(true)
  # total correct predictions
  correct = 0
  for (i in seq(len)){
    if (true[i] == predict[i]){
      correct = correct + 1
    }
  }
  # accuracy
  return (correct/len)
}
#example
accuracy(data_raw,’class’,’scored.class’)
```

```
## [1] 0.8066298
```

Task 4: Classification Error Rate Function

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.

$$Classification\ Error\ Rate = \frac{FP + FN}{TP + FP + TN + FN}$$

```r
class_error_rate <- function(df,col1,col2) {
  true = df[,col1]
  predict = df[,col2]
  # total events
  len = length(true)
  # total errors
  error = 0
  for (i in seq(len)){
    if (true[i] != predict[i]){
      error = error + 1
    }
  }
  # error rate
  return (error/len)
}
#example
class_error_rate(data_raw,'class','scored.class')
```

```
## [1] 0.1933702
```

Task 5: Precision Function

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

$$Precision = \frac{TP}{TP + FP}$$

```r
#' Precision
#'
#' Given a dataset of actual and predicted classifications,
#' returns the precision of the predictions.
#'
#' @param data A dataset of actual and predicted classifications.
#'
#' @return Precision of predictions as a numeric value rounded to 2 decimal places.
precision <- function(data) {
  # Calculate the total number of true positives in the dataset.
  true_positive <- sum(data$class == 1 & data$scored.class == 1)

  # Calculate the total number of false positives in the dataset.
  false_positive <- sum(data$class == 0 & data$scored.class == 1)

  # Perform the precision calculation and round the result to 2 decimal places.
  prediction_precision <- round(true_positive / (true_positive + false_positive), 2)

  return(prediction_precision)
}

# Call the function to provide example output.
precision(data)
```

```
## [1] 0.84
```

Task 6: Sensitivity Function

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.

$$Sensitivity = \frac{TP}{TP + FN}$$

```
sensitivity <- function(data) {

  true_positive <- sum(data$class == 1 & data$scored.class == 1)
  false_negative <- sum(data$class == 1 & data$scored.class == 0)

  sensitivity <- true_positive / (true_positive + false_negative)

  return(sensitivity)
}

sensitivity(data)
```

```
## [1] 0.4736842
```

Task 7: Specificity Function

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

$$Specificity = \frac{TN}{TN + FP}$$

```
specificity <- function(data) {

  true_negative <- sum(data$scored.class == 0 & data$class == 0)
  false_positive <- sum(data$scored.class == 1 & data$class == 0)

  specificity <- true_negative / (true_negative + false_positive)

  return(specificity)
}

specificity(data)
```

```
## [1] 0.9596774
```

Task 8: F1 Score Function

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.

$$F1\ Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitvity}$$

```r
# Should be based on the previous functions, so something like the below...
f1_score <- function(data) {
  sens <- sensitivity(data)
  prec <- precision(data)
  f1 <- 2 * sens * prec / (prec+sens)
  return(f1)
}

f1_score(data)
```

```
## [1] 0.6057692
```

Task 9: F1 Score Bounds

What are the bounds on the F1 score? Show that the F1 score will always be between 0 and 1.

The bounds of the F1 score will always be between 0 and 1. It is the harmonic mean of precision and sensitivity and it denotes how accurate a model is.

Precision is bounded between 0 and 1. Likewise, sensitivity is bounded by 0 and 1. When they are multiplied together, their product will always be smaller than the original number.

$$0 \leq precision \leq 1 \qquad\qquad 0 \leq sensitivity \leq 1$$

Since the lower bounds are both zero, the lower bound of the product would also be 0. Likewise, since the upper bounds are both 1, the upper bound of the product would also be 1.

$$0 \leq precision \cdot sensitivity \leq 1$$
$$0 \cdot sensitivity \leq precision \cdot sensitivity \leq 1 \cdot sensitivity$$
$$0 \leq precision \cdot sensitivity \leq sensitivity$$

If we multiply by a factor of 2, we get:

$$2 \cdot 0 \leq 2 \cdot precision \cdot sensitivity \leq 2 \cdot sensitivity$$
$$0 \leq 2 \cdot precision \cdot sensitivity \leq 2 \cdot sensitivity$$

If we add precision and sensitivity, we get:

$$0 + 0 \leq precision + sensitivity \leq 1 + 1$$
$$0 \leq precision + sensitivity \leq 2$$

We can combine by dividing the product by the sum to get the F1 Score.

$$\frac{0}{0} \leq \frac{2 \cdot precision \cdot sensitivity}{precision + sensitivity} \leq \frac{2 \cdot sensitivity}{2} \Rightarrow 0 < \frac{2 \cdot precision \cdot sensitivity}{precision + sensitivity} \leq sensitivity \leq 1$$

It should be noted that if both precision and sensitivity are 0, then the F1 score would be undefined, since the denominator would be 0. It would also mean that the model is not accurate as there are no true positives, which is not wanted or that the sample was too small. Henceforth, it can be said that the lower bound would be 0.

The F1 score is bounded between 0 and sensitivity, when multiplied by sensitivity whose upper bound is 1. It can also be replicated when multiplying by precision. Hence, the bounds of the F1 score are always between 0 and 1.

Task 10: ROC curve

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.

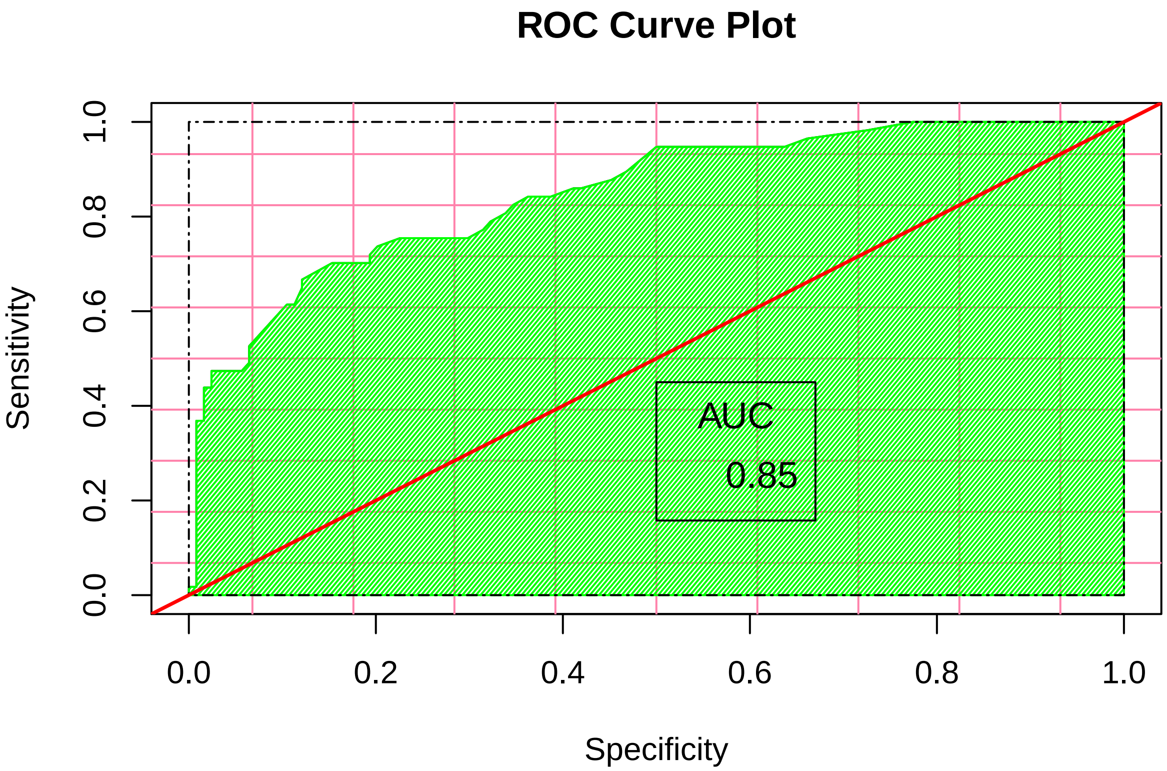
```
roc_curve_generator <- function(class, probability) {
  # Loop through the thresholds and create a dataframe for each item
  # (0 through 1 in increments of 0.01).
  for (threshold in seq(0, 1, 0.01)) {
    threshold_data <- data.frame(class = class,
                scored.class = if_else(probability >= threshold, 1, 0),
                scored.probability = probability)

    # Utilize the custom specificity() and sensitivity() functions we defined earlier
    # to calcualate the True positive Rate (TPR), and False Positive Rate (FPR).
    if (!exists(’FPR’) & !exists(’TPR’)) {
      FPR <- 1 - specificity(threshold_data)
      TPR <- sensitivity(threshold_data)
    }
    else {
      FPR <- c(FPR, 1 - specificity(threshold_data))
      TPR <- c(TPR, sensitivity(threshold_data))
    }
  }

  # Calculate the Area Under the Curve (AUC) rounded to 2 decimal places.
  roc_data <- data.frame(TPR, FPR) %>% arrange(FPR)
  AUC <- round(sum(roc_data$TPR * c(diff(roc_data$FPR), 0)) +
                sum(c(diff(roc_data$TPR)) * c(diff(roc_data$FPR), 0)) / 2, 2)

  # Generate the ROC Curve plot.
  plot(FPR, TPR, ’l’, main = ’ROC Curve Plot’, xlab = ’Specificity’, ylab = ’Sensitivity’)
  grid (10, 10, lty = 1, col = ’palevioletred1’)
  polygon(c(FPR, 1, 1), c(TPR, 0, 1), col = ’green’, density = 55, angle = 50)
  polygon(c(0, 0, 1, 1), c(0, 1, 1, 0), col = ’black’, density = 0, lty = 6)
  abline(a = 0, b = 1, col = ’red’, lwd = 1.8)
  legend(0.5, 0.45, AUC, title = ’AUC’, cex = 1.2)
}

# Call the function to generate example output.
roc_curve_generator(data_raw$class, data_raw$scored.probability)
```


ROC Curve Plot
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Specificity
AUC
0.85

Task 11: Classification metrics

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

```
acc = accuracy(data_raw,’class’,’scored.class’)
class_error = class_error_rate(data_raw,’class’,’scored.class’)
prec = precision(data)
sens = sensitivity(data)
spec = specificity(data)
f1 = f1_score(data)
paste("Accuracy is: ",acc)
```

```
## [1] "Accuracy is:  0.806629834254144"
```

```
paste("Class Error Rate is: ",class_error)
```

```
## [1] "Class Error Rate is:  0.193370165745856"
```

```
paste("Precision is: ", prec)
```

```
## [1] "Precision is:  0.84"
```

```
paste("Sensitivity is: ", sens)
```

```
## [1] "Sensitivity is:  0.473684210526316"
```

```
paste("Specificity is: ",spec)
```

```
## [1] "Specificity is:  0.959677419354839"
```

```
paste("F1 Score is: ",f1)
```

```
## [1] "F1 Score is:  0.605769230769231"
```

Task 12: Caret Package

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?*

```
#convert the variables into factors as needed for the confusionMatrix
data <- data %>%
  mutate(scored.class = as.factor(scored.class),
         class = as.factor(class))

confusionMatrix(data$scored.class, data$class, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
##
##                   Kappa : 0.4916
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.4737
##             Specificity : 0.9597
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7987
##              Prevalence : 0.3149
##          Detection Rate : 0.1492
##    Detection Prevalence : 0.1768
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 1
##
```

```
caret::sensitivity(data$scored.class, data$class, positive = "1")
```

```
## [1] 0.4736842
```

```
caret::specificity(data$scored.class, data$class, negative = "0")
```

```
## [1] 0.9596774
```

The confusion matrix is exactly the same as the one produced in task 2. Sensitivity and specificity also have the same output as created by the R functions. Precision is equal to `Pos Pred Value` in the chart.

Task 13: pROC Package

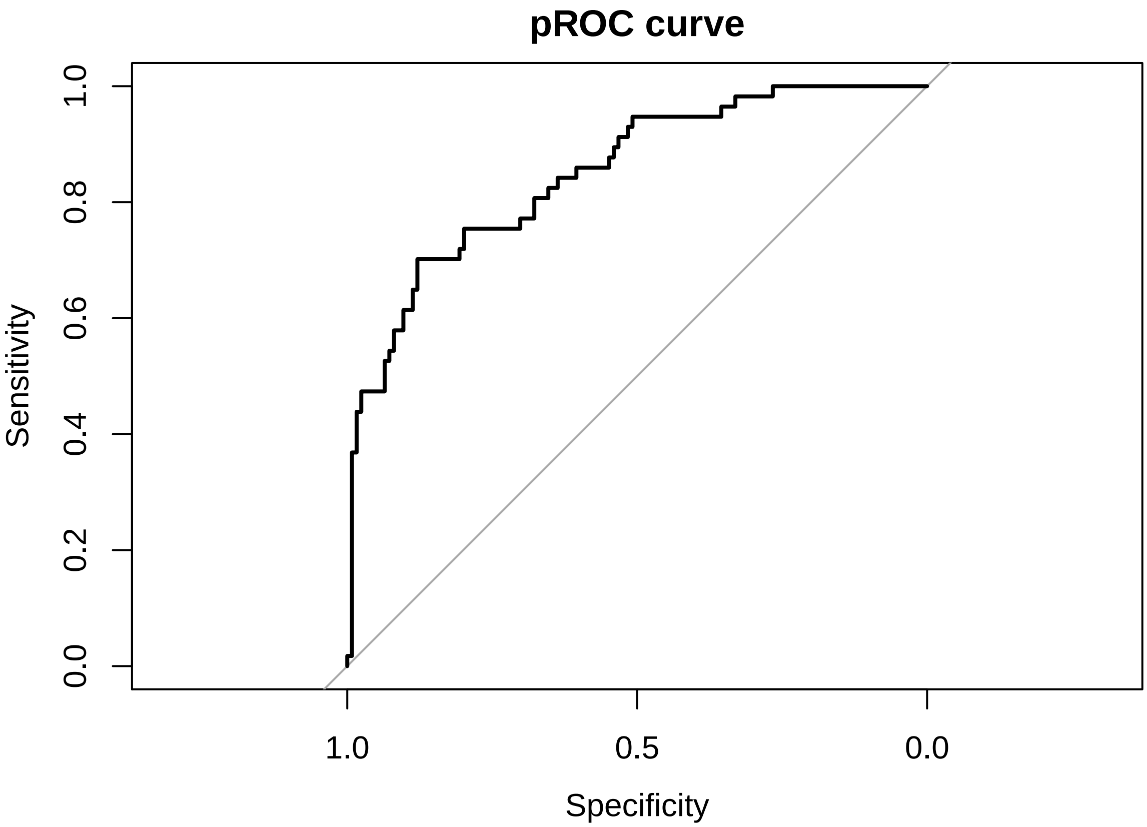
*Investigate the **pROC** package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?*

```
pROC <- roc(data_raw$class, data_raw$scored.probability)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(pROC, main = "pROC curve")
```



The ROC curve is similar to the one created in our own function in Task 10. This one is more jagged while ours has smoother lines.