Data621_weizhou_miachen

Wei Zhou / Mia Chen

3/15/2020

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.

Instructions

Complete each of the following steps as instructed:

1. Download the classification output data set (attached in Blackboard to the assignment).

```
data <- read.csv("https://raw.githubusercontent.com/miachen410/DATA621/master/HW%232/classification-out
```

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

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?

```
data <- data %>% select(class, pred_class=scored.class, pred_prob=scored.probability)
head(data)
```

```
##   class pred_class  pred_prob
## 1     0          0 0.32845226
## 2     0          0 0.27319044
## 3     1          0 0.10966039
## 4     0          0 0.05599835
## 5     0          0 0.10049072
## 6     0          0 0.05515460
```

```r
table(data[, 1:2])
```

```
##      pred_class
## class   0   1
##     0 119   5
##     1  30  27
```

The rows represent actual class, and the columns represent predicted class.

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.

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

```r
# Define accuracy function and compute the accuracy of the data set
accuracy <- function(dt) {
  df <- as.data.frame(table(dt[, 1:2]))
  TN <- df %>% filter(class==0 & pred_class==0) %>% select(Freq)
  FN <- df %>% filter(class==1 & pred_class==0) %>% select(Freq)
  FP <- df %>% filter(class==0 & pred_class==1) %>% select(Freq)
  TP <- df %>% filter(class==1 & pred_class==1) %>% select(Freq)
  return((TP+TN)/(TP+FP+TN+FN))
}
accuracy(data)
```

```
##        Freq
## 1 0.8066298
```

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.

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

```r
# Define error rate function and compute the error rate of the data set
error_rate <- function(dt) {
  df <- as.data.frame(table(dt[, 1:2]))
  TN <- df %>% filter(class==0 & pred_class==0) %>% select(Freq)
  FN <- df %>% filter(class==1 & pred_class==0) %>% select(Freq)
  FP <- df %>% filter(class==0 & pred_class==1) %>% select(Freq)
  TP <- df %>% filter(class==1 & pred_class==1) %>% select(Freq)
  return((FP+FN)/(TP+FP+TN+FN))
}
error_rate(data)
```

```
##        Freq
## 1 0.1933702
```

Verify that you get an accuracy and an error rate that sums to one.

```r
# Verify that accuracy and error rate sum to one
accuracy(data) + error_rate(data)
```

```
##   Freq
## 1    1
```

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.

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

```r
# Define precision function and compute the precision of the data set
precision <- function(dt) {
  df <- as.data.frame(table(dt[, 1:2]))
  TN <- df %>% filter(class==0 & pred_class==0) %>% select(Freq)
  FN <- df %>% filter(class==1 & pred_class==0) %>% select(Freq)
  FP <- df %>% filter(class==0 & pred_class==1) %>% select(Freq)
  TP <- df %>% filter(class==1 & pred_class==1) %>% select(Freq)
  return((TP)/(TP+FP))
}
precision(data)
```

```
##      Freq
## 1 0.84375
```

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.

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

```r
# Define sensitivity function and compute the sensitivity of the data set
sensitivity <- function(dt) {
  df <- as.data.frame(table(dt[, 1:2]))
  TN <- df %>% filter(class==0 & pred_class==0) %>% select(Freq)
  FN <- df %>% filter(class==1 & pred_class==0) %>% select(Freq)
  FP <- df %>% filter(class==0 & pred_class==1) %>% select(Freq)
  TP <- df %>% filter(class==1 & pred_class==1) %>% select(Freq)
  return((TP)/(TP+FN))
}
sensitivity(data)
```

```
##        Freq
## 1 0.4736842
```

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.

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

```r
# Define specificity function and compute the specificity of the data set
specificity <- function(dt) {
  df <- as.data.frame(table(dt[, 1:2]))
  TN <- df %>% filter(class==0 & pred_class==0) %>% select(Freq)
  FN <- df %>% filter(class==1 & pred_class==0) %>% select(Freq)
  FP <- df %>% filter(class==0 & pred_class==1) %>% select(Freq)
  TP <- df %>% filter(class==1 & pred_class==1) %>% select(Freq)
  return((TN)/(TN+FP))
}
specificity(data)
```

```
##        Freq
## 1 0.9596774
```

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.

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

```r
# Define F1 score function and compute the F1 score of the data set
f1_score <- function(dt) {
  return( (2*precision(dt)*sensitivity(dt)) / (precision(dt)+sensitivity(dt)) )
}
f1_score(data)
```

```
##        Freq
## 1 0.6067416
```

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 $0 < \quad < 1$ and $0 < \quad < 1$ then $\quad < \quad$.)

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

Both Precision and Sensitivity have a range from 0 to 1. Consider if $0 < \quad < 1$ and $0 < \quad < 1$ then $\quad <$ a.

Therefore,

(Precision x Sensitivity) < Precision

(Precision x Sensitivity) < Sensitivity

(Precision x Sensitivity + Precision x Sensitivity) < (Precision + Sensitivity)

(2 x Precision x Sensitivity) < (Precision + Sensitivity)

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.

```
df = read.csv("https://raw.githubusercontent.com/miachen410/DATA621/master/HW%232/classification-output-

ROC <- function(df)
{
  data1 = df
  thresholds <- seq(0,1,0.01)
  Y <- c()
  X <- c()
  for (threshod in thresholds) {
    data1$scored.class <- ifelse(data1$scored.probability > threshod,1,0)
    X <- append(X,1-specificity(data1))
    Y <- append(Y,sensitivity(data1))
    }
  df1 <- data.frame(X=X,Y=Y)
  df1 <- na.omit(df1)
  g <- ggplot(df1,aes(X,Y)) + geom_line() + ggtitle('Custom ROC Curve') +
    xlab('Specificity') + ylab('Sensitivity')
  height = (df1$Y[-1]+df1$Y[-length(df1$Y)])/2
  width = -diff(df1$X)
  area = round(sum(height*width),4)
  return(list(Plot =g,AUC = area))
}
```

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

```
library(knitr)
Name <- c('Accuracy','Classification Error Rate', 'Precision', 'Sensitivity','Specificity', 'F1 Score')
Value <- c(accuracy(data), error_rate(data), precision(data), sensitivity(data), specificity(data), f1_
df1 <- as.data.frame(cbind(Name, Value))
kable(df1)
```

	Name	Value
Freq	Accuracy	0.806629834254144
Freq.1	Classification Error Rate	0.193370165745856
Freq.2	Precision	0.84375
Freq.3	Sensitivity	0.473684210526316
Freq.4	Specificity	0.959677419354839
Freq.5	F1 Score	0.606741573033708

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?

```
require("caret")
```

```
## Loading required package: caret
```

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

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked _by_ '.GlobalEnv':
##
##     precision, sensitivity, specificity
```

```
df = read.csv("https://raw.githubusercontent.com/miachen410/DATA621/master/HW%232/classification-output-
d_tab <- table(df$class,df$scored.class)
confusionMatrix(d_tab, reference = df$class)
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 119   5
##   1  30  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.8232
##     P-Value [Acc > NIR] : 0.7559
##
##                   Kappa : 0.4916
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.7987
##             Specificity : 0.8438
##          Pos Pred Value : 0.9597
##          Neg Pred Value : 0.4737
##              Prevalence : 0.8232
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.6851
##       Balanced Accuracy : 0.8212
##
##        'Positive' Class : 0
##
```

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

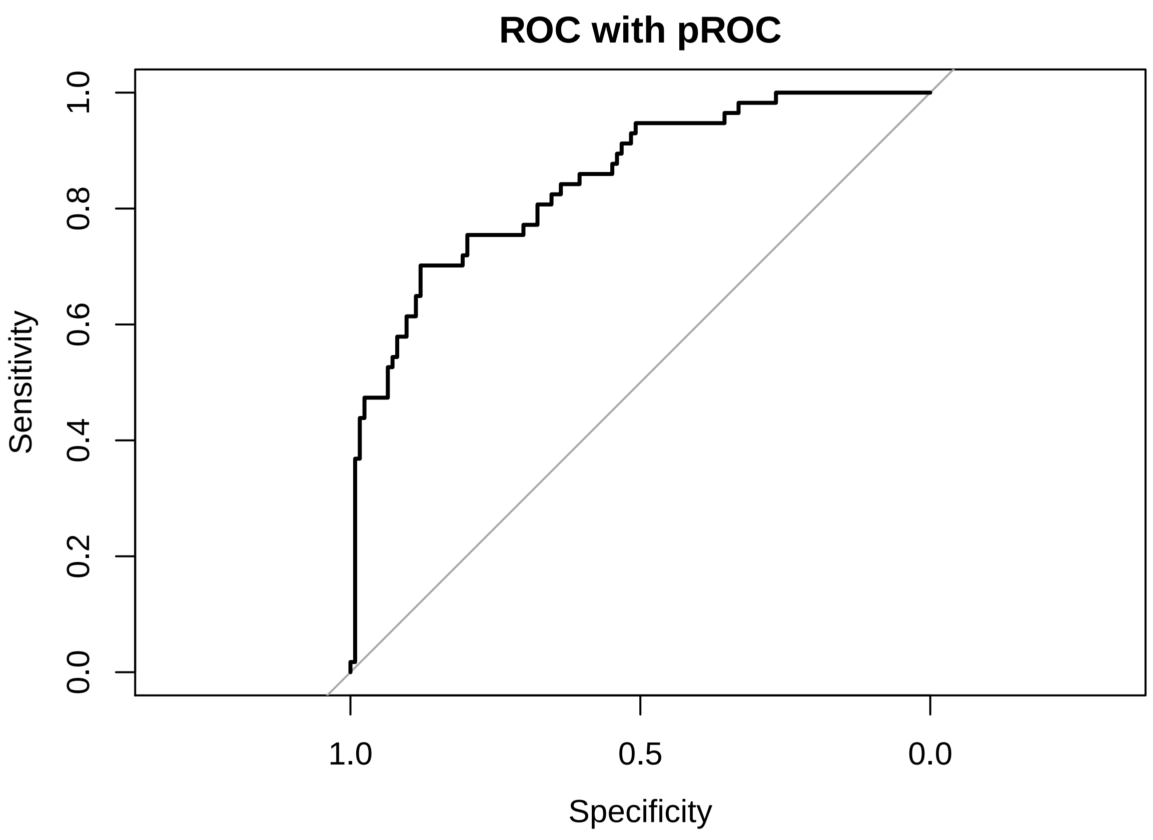
```r
require("pROC")
```

```
## Loading required package: pROC

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
d_roc <- roc(df$class,df$scored.probability)
```

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

## Setting direction: controls < cases
```

```r
plot(d_roc, main = "ROC with pROC")
```


ROC with pROC
Sensitivity
Specificity