Homework 2

Classification Metrics

Group 2

3/10/2021

Contents

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

Download the classification output data set.

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

```r
data_raw%>%head(50)
```

```
##    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
## 7         9      89        62        0       0 22.5    0.142  33     0
## 8         8     120        78        0       0 25.0    0.409  64     0
## 9         1      79        60       42      48 43.5    0.678  23     0
## 10        2     123        48       32     165 42.1    0.520  26     0
## 11        5      88        78       30       0 27.6    0.258  37     0
## 12        5     108        72       43      75 36.1    0.263  33     0
## 13       13      76        60        0       0 32.8    0.180  41     0
## 14        0     100        70       26      50 30.8    0.597  21     0
## 15        7     194        68       28       0 35.9    0.745  41     1
## 16       12      92        62        7     258 27.6    0.926  44     1
## 17        0     173        78       32     265 46.5    1.159  58     0
## 18        3     171        72       33     135 33.3    0.199  24     1
## 19        8     196        76       29     280 37.5    0.605  57     1
## 20        5      99        74       27       0 29.0    0.203  32     0
## 21        2     100        70       52      57 40.5    0.677  25     0
## 22        3     111        62        0       0 22.6    0.142  21     0
## 23        1     119        54       13      50 22.3    0.205  24     0
## 24        1     138        82        0       0 40.1    0.236  28     0
## 25        0     189       104       25       0 34.3    0.435  41     1
## 26        3     130        78       23      79 28.4    0.323  34     1
## 27        9     102        76       37       0 32.9    0.665  46     1
## 28        0     151        90       46       0 42.1    0.371  21     1
## 29        1      71        48       18      76 20.4    0.323  22     0
## 30        0     101        64       17       0 21.0    0.252  21     0
## 31        3     116        74       15     105 26.3    0.107  24     0
## 32        6     107        88        0       0 36.8    0.727  31     0
## 33        1     128        88       39     110 36.5    1.057  37     1
## 34        0     111        65        0       0 24.6    0.660  31     0
## 35        7     187        50       33     392 33.9    0.826  34     1
## 36        0     180        90       26      90 36.5    0.314  35     1
## 37        5     139        64       35     140 28.6    0.411  26     0
## 38        8     126        74       38      75 25.9    0.162  39     0
## 39        1     196        76       36     249 36.5    0.875  29     1
## 40       10      75        82        0       0 33.3    0.263  38     0
## 41        0     102        64       46      78 40.6    0.496  21     0
## 42        1      90        68        8       0 24.5    1.138  36     0
## 43        1     112        72       30     176 34.4    0.528  25     0
## 44        2     130        96        0       0 22.6    0.268  21     0
```

```
## 45        8     100        76        0        0 38.7     0.190  42      0
## 46        3      89        74       16       85 30.4     0.551  38      0
## 47        0     125        96        0        0 22.5     0.262  21      0
## 48        2      91        62        0        0 27.3     0.525  22      0
## 49        7     114        64        0        0 27.4     0.732  34      1
## 50        7     136        90        0        0 29.9     0.210  50      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
## 7             0         0.10711542
## 8             0         0.45994744
## 9             0         0.11702368
## 10            0         0.31536320
## 11            0         0.12518925
## 12            0         0.27062482
## 13            0         0.20980960
## 14            0         0.09358589
## 15            1         0.88484573
## 16            0         0.39665216
## 17            1         0.89139491
## 18            1         0.53454900
## 19            1         0.94633418
## 20            0         0.14491618
## 21            0         0.21763796
## 22            0         0.07521357
## 23            0         0.08843254
## 24            0         0.30346820
## 25            1         0.72448003
## 26            0         0.27497369
## 27            0         0.42486483
## 28            0         0.43092552
## 29            0         0.02322803
## 30            0         0.04596084
## 31            0         0.12798534
## 32            0         0.29933706
## 33            0         0.45909503
## 34            0         0.10479581
## 35            1         0.86309177
## 36            1         0.63997495
## 37            0         0.35818434
## 38            0         0.37216467
## 39            1         0.81110322
## 40            0         0.16812736
## 41            0         0.15127796
## 42            0         0.10700703
## 43            0         0.18796139
## 44            0         0.13719711
## 45            0         0.30047491
## 46            0         0.13688715
## 47            0         0.09786911
```

```
## 48             0          0.06290701
## 49             0          0.26941931
## 50             0          0.48854279
```

Task 2

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?

```r
# 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(class, scored.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()
```

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

Functions

```r
# 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.

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

```r
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.

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

```
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.

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

```r
#note: there is a built in function in package caret called sensitivity
# head(data)

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.

$$F1Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitvity}$$

```
# 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

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

```
#data1<-data.frame(rep(0,5),rep(0,5))
#data1<-data.frame(rep(0,5),rep(1,5))
#data1<-data.frame(rep(1,5),rep(0,5))
data1<-data.frame(rep(1,5),rep(1,5))
colnames(data1)<-c('class','scored.class')
data1
```

```
##   class scored.class
## 1     1            1
## 2     1            1
## 3     1            1
## 4     1            1
## 5     1            1
```

```r
precision(data1)
```

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

```r
sensitivity(data1)
```

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

```r
f1_score(data1)
```

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

Task 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.

Task 11

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

Task 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?*

```r
#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
##
```

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

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

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

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

Task 13

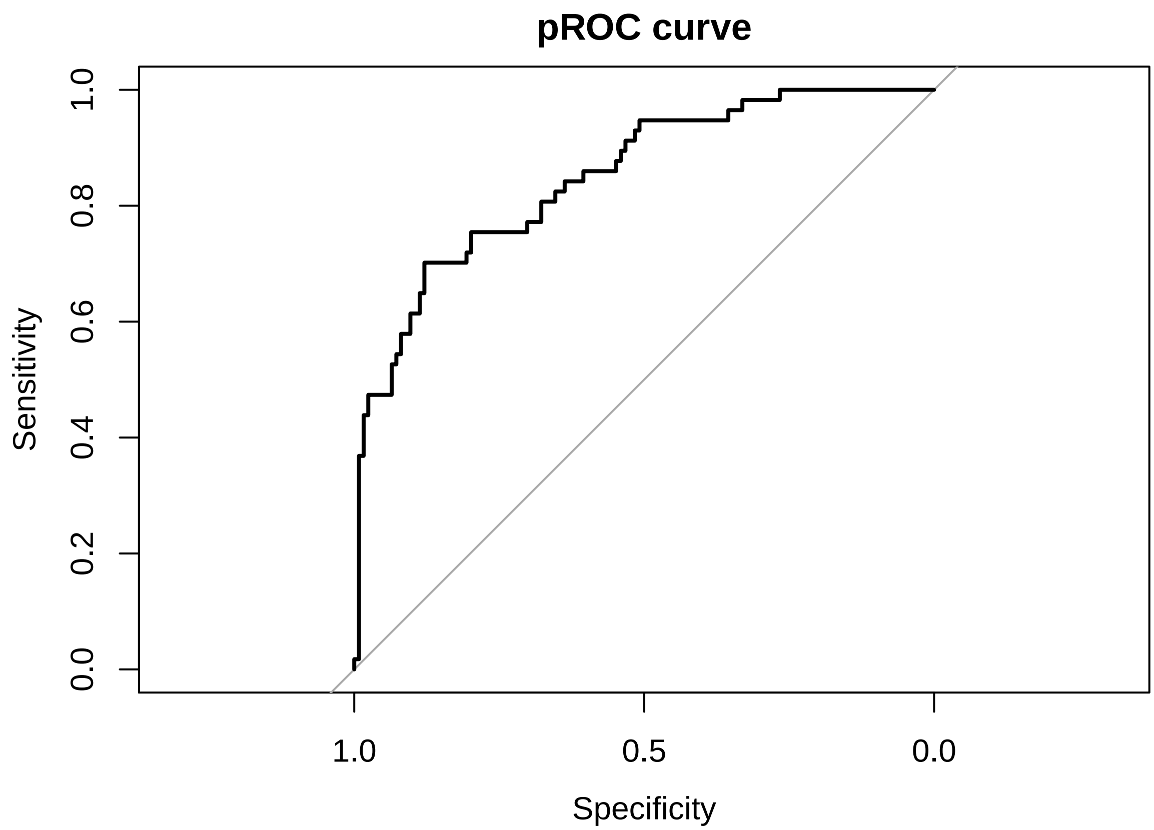
*Investigate the **pROC** package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?*

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

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

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

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


pROC curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity