SPerveen_Homework2

Sadia Perveen

3/18/2021

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

```
cod <-
read.csv("https://raw.githubusercontent.com/sperveen/DATA621/main/classificat
ion-output-data.csv")
head(cod, 10)
```

```
##    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
##    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
```

```
confusionMatrix <- data.frame(table("scored.class" = cod$scored.class,
"class" = cod$class))
confusionMatrix
```

```
##   scored.class class Freq
## 1            0     0  119
## 2            1     0    5
## 3            0     1   30
## 4            1     1   27
```

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. $TP + TN$ $Accuracy = TP+FP+TN+FN$

```
accuracyPredictions <- function(confusionMatrix){
  tp <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==1,]$Freq
  tn <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==0,]$Freq
  fp <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==1,]$Freq
  fn <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==0,]$Freq
  total <- tp + tn + fp + fn
  return((tp+tn)/total)
}
accuracyPredictionsValue <- accuracyPredictions(confusionMatrix)
accuracyPredictionsValue

## [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. FP $+ FN$ $Classification\ Error\ Rate = TP + FP + TN + FN$ Verify that you get an accuracy and an error rate that sums to one.

```
errorRate <- function(confusionMatrix){
  tp <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==1,]$Freq
  tn <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==0,]$Freq
  fp <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==1,]$Freq
  fn <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==0,]$Freq
  total <- tp + tn + fp + fn
  return((fp+fn)/total)
}

errorRateValue <- errorRate(confusionMatrix)
errorRateValue

## [1] 0.1933702
```

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. $TP\ Precision = TP + FP$

```
percision <- function(confusionMatrix){
  tp <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==1,]$Freq
  fp <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==1,]$Freq
  return(tp/(tp+fp))
}

percisionValue <- percision(confusionMatrix)
percisionValue

## [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. $TP\ Sensitivity = TP + FN$

```
sensitivity <- function(confusionMatrix){
  tp <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==1,]$Freq
  fn <- confusionMatrix[confusionMatrix$class==1 &
confusionMatrix$scored.class==0,]$Freq
  return(tp/(tp+fn))
}

sensitivityValue <- sensitivity(confusionMatrix)
sensitivityValue

## [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. $TN\ Specificity = TN + FP$

```
specificity <- function(confusionMatrix){
  tn <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==0,]$Freq
  fp <- confusionMatrix[confusionMatrix$class==0 &
confusionMatrix$scored.class==1,]$Freq
  return(tn/(tn+fp))
}

specificityValue <- specificity(confusionMatrix)
specificityValue

## [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. $2 \times Precision \times Sensitivity\ Precision + Sensitivity$

```
f1Score <- function(confusionMatrix){
  percisionValue <- percision(confusionMatrix)
  sensitivityValue <- sensitivity(confusionMatrix)
  return((2 * percisionValue * sensitivityValue)/(percisionValue +
sensitivityValue))
}
f1ScoreValue <- f1Score(confusionMatrix)
f1ScoreValue

## [1] 0.6067416
```

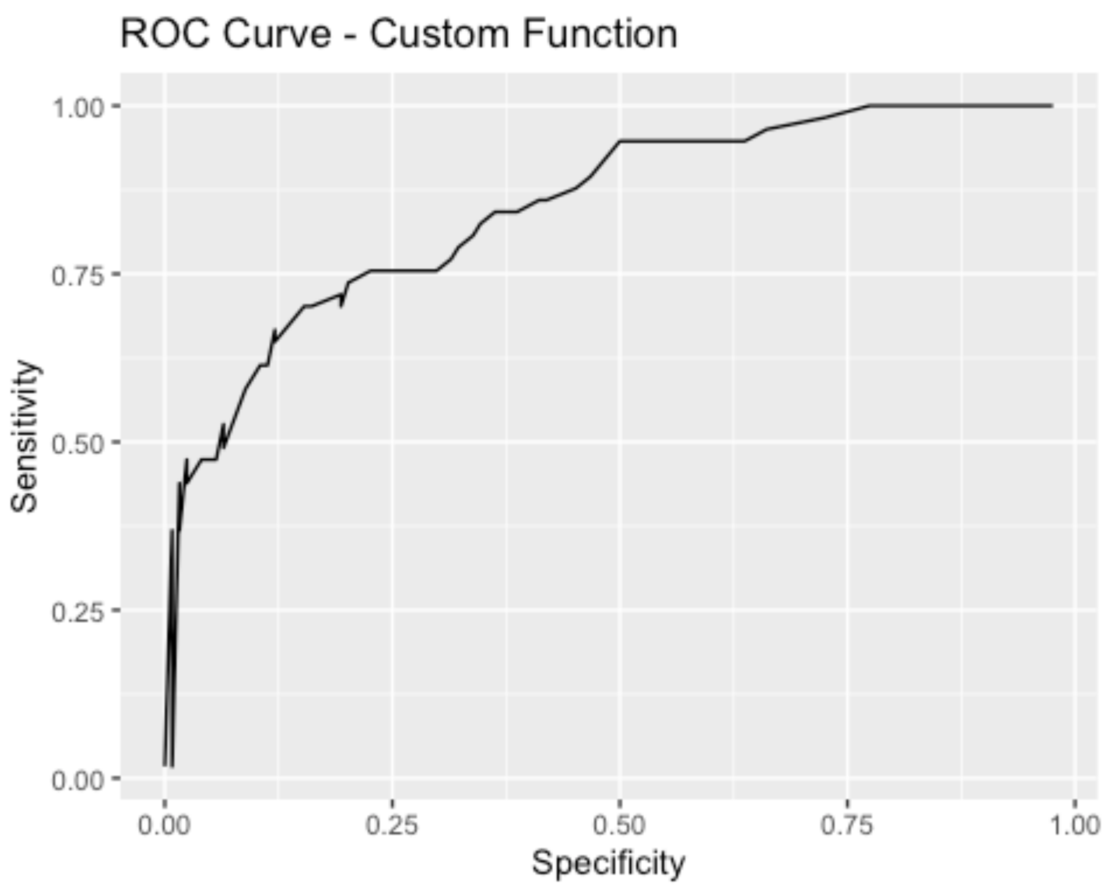
9. Before we move on, let's consider a question that was asked: What are the bounds on the F1 score? Show that the F1 score will always be between 0 and 1. (Hint: If $0 < a < 1$ and $0 < b < 1$ then $ab < a$.)

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.

```
rocPlot <- function(cod){
  cod2 <- cod
  thresholds <- seq(0,1,0.01)
  sensitivityPoints <- c()
  specificityPoints <- c()
  for (t in thresholds) {
    cod2$scored.class <- ifelse(cod2$scored.probability > t,1,0)
    specificityPoints <- append(specificityPoints,1-
specificity(data.frame(table("scored.class" = cod2$scored.class, "class" =
cod2$class))))
    sensitivityPoints <-
append(sensitivityPoints,sensitivity(data.frame(table("scored.class" =
cod2$scored.class, "class" = cod2$class))))
  }
  codDf <- data.frame(X=specificityPoints,Y=sensitivityPoints)
  codDf <- na.omit(codDf)
  g <- ggplot(codDf,aes(X,Y)) + geom_line() + ggtitle('ROC Curve - Custom
Function') +
    xlab('Specificity') + ylab('Sensitivity')
  height = (codDf$Y[-1]+codDf$Y[-length(codDf$Y)])/2
  width = -diff(codDf$X)
  area = sum(height*width)
  return(list(Plot =g,AUC = area))
}
```

```r
rocPlot(cod)
```

```
## $Plot
```




```
##
## $AUC
## [1] 0.8247029
```

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

```r
functionName <- c('Accuracy','Classification Error Rate', 'Precision',
'Sensitivity','Specificity', 'F1 Score')
functionValue <- c(accuracyPredictions(confusionMatrix),
errorRate(confusionMatrix), percision(confusionMatrix),
sensitivity(confusionMatrix), specificity(confusionMatrix),
f1Score(confusionMatrix))
functionOutputs <- as.data.frame(cbind(functionName, functionValue))
functionOutputs
```

```
##                functionName     functionValue
## 1                  Accuracy 0.806629834254144
## 2 Classification Error Rate 0.193370165745856
## 3                 Precision           0.84375
```

```
## 4                Sensitivity 0.473684210526316
## 5                Specificity 0.959677419354839
## 6                   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':
##
##     sensitivity, specificity

caret::confusionMatrix(table(cod$class,cod$scored.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

plot(pROC::roc(cod$class, cod$scored.probability), main="ROC Curve - pROC")

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

## Setting direction: controls < cases
```