HW2

Zachary Palmore

3/11/2021

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.

Supplemental Material

```
- Applied Predictive Modeling, Ch. 11 (provided as a PDF file).

- Web tutorials: http://www.saedsayad.com/model_evaluation_c.htm
```

Deliverables

Upon following the instructions below, use your created R functions and the other packages to generate the classification metrics for the provided data set. A write-up of your solutions submitted in PDF format.

Instructions

Complete each of the following steps as instructed.

```r
library(tidyverse)
library(kableExtra)
library(caret)
library(pROC)
```

Step 1

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

```r
dt <- read_csv('https://raw.githubusercontent.com/palmorezm/data621/main/HW2/classification-output-data
```

```r
head(dt)
```

```
## # A tibble: 6 x 11
##   pregnant glucose diastolic skinfold insulin   bmi pedigree   age class
##      <dbl>   <dbl>     <dbl>    <dbl>   <dbl> <dbl>    <dbl> <dbl> <dbl>
```

```
## 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
## # ... with 2 more variables: scored.class <dbl>, scored.probability <dbl>
```

Step 2

The data set has three key columns we will use:

1. `class: the actual class for the observation`
2. `scored.class: the predicted class for the observation (based on a threshold of 0.5)`
3. `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?

Rows represent the actual class observation of the 'class' field while the columns represent the 'scored.class' field which contains the predicted values in each row. This is shown in the table below.

```
dt.conf.tbl <- dt %>%
  select(class, scored.class) %>%
  table()
colnames(dt.conf.tbl) <- c("Predicted -", "Predicted +")
row.names(dt.conf.tbl) <- c("Observed -", "Observed +")
kbl(dt.conf.tbl, booktabs = T, caption = "Confusion Matrix") %>%
  kable_styling(latex_options = c("striped", "hold_position"),  full_width = F)
```

Table 1: Confusion Matrix

	Predicted -	Predicted +
Observed -	119	5
Observed +	30	27

```
# dt.conf.tbl
```

Using this table, we could conclude that there were 119 true negatives, 27 true positives, 5 false negatives, and 30 false positives based on the differences of predicted values and actual observations.

Step 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
# Calculating accuracy function
Ac <- function(x){
  tp <- sum(x$class == 1 & x$scored.class == 1)
  tn <- sum(x$class == 0 & x$scored.class == 0)
return((tp + tn)/nrow(x))
}
# Accuracy of this data set is shown below
Ac(dt)
```

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

Step 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.

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

```r
# Classification Error Rate function
errt <- function(x){
  total <- nrow(x)
  fn <- sum(x$class == 1 & x$scored.class ==0)
  fp <- sum(x$class == 0 & x$scored.class ==1)
  return((fn+fp)/total)
}
# Error rate of this data set is shown below
errt(dt)
```

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

Step 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
# Precision function
precision <- function(x){
  tp <- sum(x$class == 1 & x$scored.class == 1)
  fp <- sum(x$class == 0 & x$scored.class == 1)
  return(tp/(tp + fp))
}
# Precision of this data set is shown below
precision(dt)
```

```
## [1] 0.84375
```

Step 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.

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN}$$

```
# Sensitivity function
recall <- function(x){
  fn <- sum(x$class == 1 & x$scored.class ==0)
  tp <- sum(x$class == 1 & x$scored.class ==1)
  return(tp/(tp+fn))
}
# Sensitivity of this data set is shown below
recall(dt)
```

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

Step 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.

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

```
# Specificity function
specif <- function(x){
  tn <- sum(x$class == 0 & x$scored.class == 0)
  fp <- sum(x$class == 0 & x$scored.class == 1)
  return(tn/(tn + fp))
}
# Specificity of this data set is shown below
specif(dt)
```

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

Step 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.

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

```
# F1 Score function

F1 <- function(x) {
  tp <- sum(x$class == 1 & x$scored.class == 1)
  fp <- sum(x$class == 0 & x$scored.class == 1)
```

```
  fn <- sum(x$class == 1 & x$scored.class ==0)
  precision <- (tp/(tp + fp))
  sensitivity <- (tp/(tp+fn))
  return( (2*precision*sensitivity) /
            (precision + sensitivity))
}
# F1 Score of this data set is shown below
F1(dt)
```

```
## [1] 0.6067416
```

Step 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 < < 1 and 0 < < 1 then < .)

Consider that the F1 score is first and foremost a harmonic mean measure of the accuracy on a binary classification system. This binary system is usually given two classifications types (hence the binary part), of either positive or negative. In a numerical sense, these could be either a 0 or 1. Basing the entire calculation on this system is not likely to result in a mean outside of its inherent classification system.

Second, consider the formula. If the predictions have a low values of precision and sensitivity, then the results would be closer to 0. Alternatively, if the quantities of errors are large, then the score will be closer to 1. Both concepts can be demonstrated mathematically. Consider the following inequalities;

$$0 \geq P \geq 1$$
$$0 \geq S \geq 1$$
$$PS \leq S \ or \ P$$
$$Thus, \ 0 \leq PS \leq S \ or \ P \leq 1$$

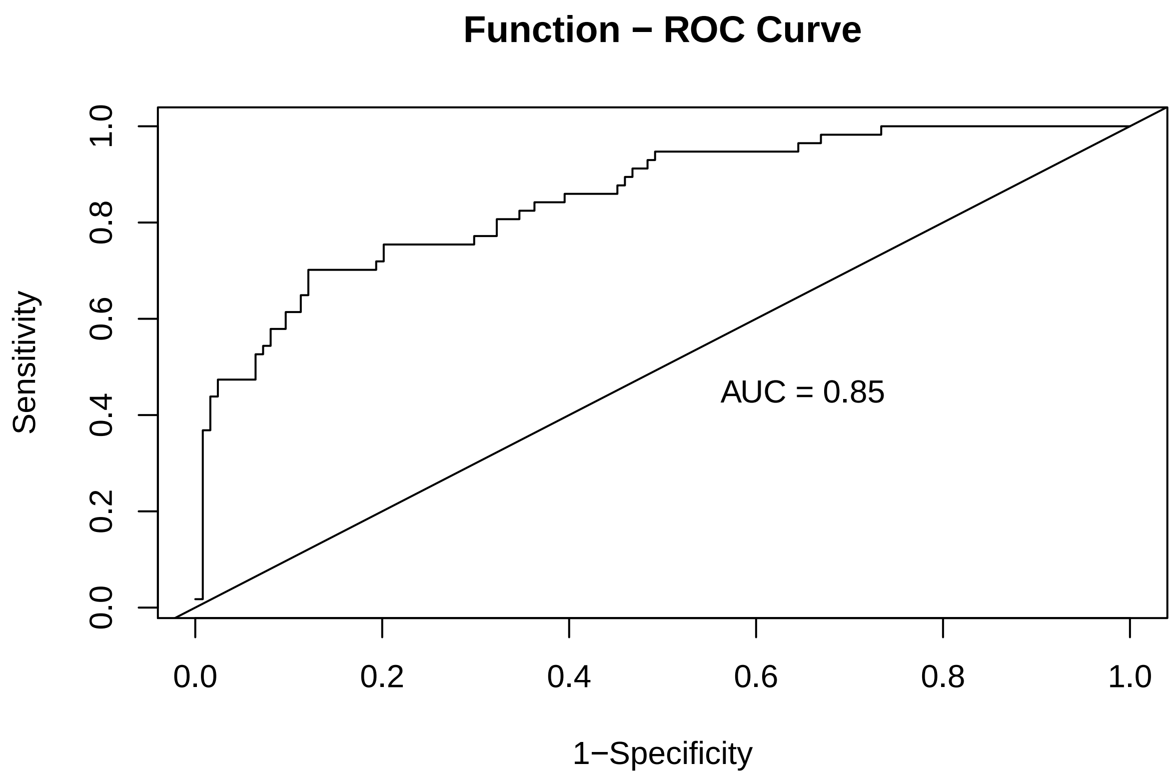
The range of precision (P) is such that it must be between 0 and 1. The same applies to sensitivity (S). When we compute these in a function, our denominator will always be the sum of those values while the numerator is a multiple of the two. Thus, the denominator will always be between 0 and 2, while the numerator stays between 0 and 1. Dividing those two ranges will also always results in a value between 0 and 1.

Step 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.

```
ROC <- function (x, y) {
  x <- x[order(y, decreasing=T)]
  d <- data.frame(TPR=cumsum(x)/sum(x),
            FPR=cumsum(!x)/sum(!x),
            x)
  auc <- ((sum(d$TPR * c(diff(d$FPR),0)) +
                  sum(c(diff(d$TPR),0) * c(diff(d$FPR),0))/2))
  plot(d$FPR, d$TPR,
```

```r
      type = 'l',
      xlim=c(0.0,1.0),
      main = "Function - ROC Curve",
      xlab = "1-Specificity",
      ylab = "Sensitivity") +
  abline(coef = c(0,1)) +
  text(0.65, 0.45, labels = print(paste("AUC =", round(auc, 3))))
  return(head(d, 10))
}
ROC(dt$class, dt$scored.probability)
```




```
## [1] "AUC = 0.85"

##           TPR         FPR x
## 1  0.01754386 0.000000000 1
## 2  0.01754386 0.008064516 0
## 3  0.03508772 0.008064516 1
## 4  0.05263158 0.008064516 1
## 5  0.07017544 0.008064516 1
## 6  0.08771930 0.008064516 1
## 7  0.10526316 0.008064516 1
## 8  0.12280702 0.008064516 1
## 9  0.14035088 0.008064516 1
## 10 0.15789474 0.008064516 1
```

Step 11

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

```
conf.tbl <- c( round(Ac(dt),3),
         round(errt(dt),3),
         round(precision(dt),3),
         round(recall(dt),3),
         round(specif(dt),3),
         round(F1(dt),3))
names(conf.tbl) <- c("Accuracy", "Classification Error", "Precision",
                "Sensitivity", "Specificity", "F1 Score")
conf.tbl <-as.data.frame(conf.tbl)
names(conf.tbl)[1]<-'Scores'
kbl(conf.tbl)%>%
   kable_styling(latex_options = c("striped", "hold_position"), full_width = F)
```

	Scores
Accuracy	0.807
Classification Error	0.193
Precision	0.844
Sensitivity	0.474
Specificity	0.960
F1 Score	0.607

Step 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?

Running the confusionMatrix function to compare accuracy, sensitivity, and specificity to those scores above.

```
conf.mat.caret <- confusionMatrix(data = as.factor(dt$scored.class),
                                  reference = as.factor(dt$class), positive = '0')
caret.sens <- conf.mat.caret$byClass["Sensitivity"]
caret.accu <- conf.mat.caret$overall["Accuracy"]
caret.spec <- conf.mat.caret$byClass["Specificity"]
caret.stats <- as.data.frame(cbind(
  round(caret.accu, 3),
  round(caret.sens, 3),
  round(caret.spec, 3)
  ))
colnames(caret.stats) <- c("Accuracy", "Sensitivity", "Specificity")
row.names(caret.stats) <- "Scores"
kbl(caret.stats) %>%
  kable_styling(latex_options = c("striped", "hold_position"), full_width = F)
```

	Accuracy	Sensitivity	Specificity
Scores	0.807	0.96	0.474

Alternatively, to compare all stats, we might want to compare the full scope of statistics.

```r
conf.mat.caret
```

```
## 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 : 0.9597
##             Specificity : 0.4737
##          Pos Pred Value : 0.7987
##          Neg Pred Value : 0.8438
##              Prevalence : 0.6851
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.8232
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
##
```

At more precise decimals the values are slightly different. However, when rounded to the hundredths place there appears to be no major differences between the caret package functions and those created above.

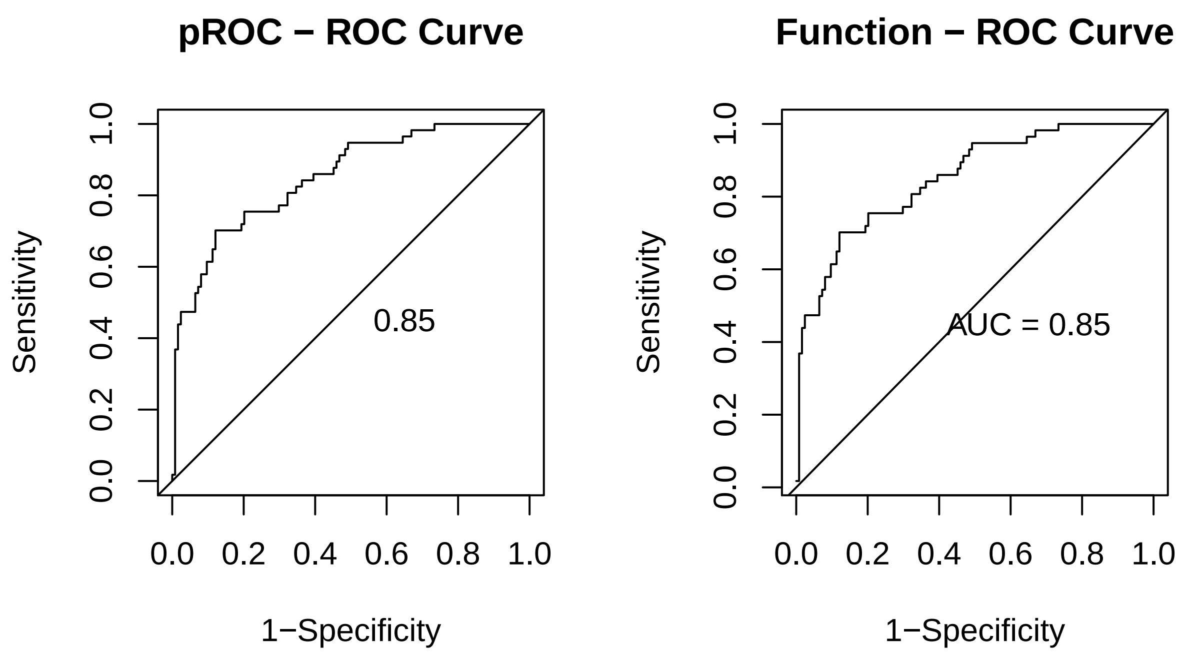
Step 13

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
rocdf <- pROC::roc(dt$class, dt$scored.probability)
rocauc <- auc(dt$class, dt$scored.probability)
par(mfrow = c(1, 2), pty = 's')
plot(1-rocdf$specificities, rocdf$sensitivities, type='s',
     main = "pROC - ROC Curve",
     xlab = "1-Specificity",
     ylab = "Sensitivity") +
  abline(c(0,1))+
  text(0.65, 0.45, labels = round(rocauc, 3))
```

```
## integer(0)
```

```r
ROC(dt$class, dt$scored.probability)
```




```
## [1] "AUC = 0.85"
```

```
##           TPR         FPR x
## 1  0.01754386 0.000000000 1
## 2  0.01754386 0.008064516 0
## 3  0.03508772 0.008064516 1
## 4  0.05263158 0.008064516 1
## 5  0.07017544 0.008064516 1
## 6  0.08771930 0.008064516 1
## 7  0.10526316 0.008064516 1
## 8  0.12280702 0.008064516 1
## 9  0.14035088 0.008064516 1
## 10 0.15789474 0.008064516 1
```

Placed side-by-side, they are nearly identical at this scale. Our function was intentionally created this way to mimic the results of the pROC package. However, if we look closer at the exact values of each statistic (shown in the results from the caret package and each function above), we will notice that in general they begin to differ slightly beyond the hundreths decimal place. This is expected since we rounded some of the results prior to calculation.