DATA 621 Homework 2

Bin Lin

2017-3-15

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.

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

```
classification_output <- read.csv("C:/Users/blin261/Desktop/DATA621/classification-output-data.c
sv", header = TRUE)
head(classification_output)
```

```
##   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
##   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
```

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?

The rows represent the predicted class. On the other hand, columns represent the actual class.

```
classification_subset <- classification_output[, c("class", "scored.class",
"scored.probability")]
head(classification_subset)
```

```
##   class scored.class scored.probability
## 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
```

```
confusion_matrix <- table(classification_subset[,"scored.class"],
classification_subset[,"class"])
colnames(confusion_matrix) <- c("Actual 0", "Actual 1")
rownames(confusion_matrix) <- c("Predicted 0", "Predicted 1")
confusion_matrix
```

```
##
##               Actual 0 Actual 1
##   Predicted 0      119       30
##   Predicted 1        5       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.

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

```
get_accuracy <- function(data_frame, predicted,  actual)
{
  confusion_matrix <- table(data_frame[,predicted], data_frame[,actual])

  TN <- confusion_matrix[1,1]
  FN <- confusion_matrix[1,2]
  FP <- confusion_matrix[2,1]
  TP <- confusion_matrix[2,2]
  accuracy <- (TP + TN) /(TP + TN + FP + FN)
  return (accuracy)
}

get_accuracy(classification_subset, "scored.class", "class")
```

```
## [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.

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

```
classification_error_rate <- function(data_frame, predicted,  actual)
{
  confusion_matrix <- table(data_frame[,predicted], data_frame[,actual])

  TN <- confusion_matrix[1,1]
  FN <- confusion_matrix[1,2]
  FP <- confusion_matrix[2,1]
  TP <- confusion_matrix[2,2]
  classification_error_rate <- (FP + FN) /(TP + TN + FP + FN)
  return (classification_error_rate)
}

classification_error_rate(classification_subset, "scored.class", "class")
```

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

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

```
get_accuracy(classification_subset, "scored.class", "class") + classification_error_rate(classif
ication_subset, "scored.class", "class")
```

```
## [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.

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

```
get_precision <- function(data_frame, predicted,  actual)
{
  confusion_matrix <- table(data_frame[,predicted], data_frame[,actual])

  TN <- confusion_matrix[1,1]
  FN <- confusion_matrix[1,2]
  FP <- confusion_matrix[2,1]
  TP <- confusion_matrix[2,2]
  precision <- (TP) /(TP + FP)
  return (precision)
}

get_precision(classification_subset, "scored.class", "class")
```

```
## [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.

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

```
get_sensitivity <- function(data_frame, predicted,  actual)
{
  confusion_matrix <- table(data_frame[,predicted], data_frame[,actual])

  TN <- confusion_matrix[1,1]
  FN <- confusion_matrix[1,2]
  FP <- confusion_matrix[2,1]
  TP <- confusion_matrix[2,2]
  sensitivity <- (TP) /(TP + FN)
  return (sensitivity)
}

get_sensitivity(classification_subset, "scored.class", "class")
```

```
## [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.

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

```
get_specificity <- function(data_frame, predicted,  actual)
{
  confusion_matrix <- table(data_frame[,predicted], data_frame[,actual])

  TN <- confusion_matrix[1,1]
  FN <- confusion_matrix[1,2]
  FP <- confusion_matrix[2,1]
  TP <- confusion_matrix[2,2]
  specificity <- (TN) /(TN + FP)
  return (specificity)
}

get_specificity(classification_subset, "scored.class", "class")
```

```
## [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.

$F1 \quad Score = (2 \times Precision \times Sensitivity)/(Presicision + Sensivity)$

```
F1_score <- function(data_frame, predicted,  actual)
{
  precision <- get_precision(data_frame, predicted, actual)
  sensitivity <- get_sensitivity(data_frame, predicted, actual)
  F1_score <- (2 * precision * sensitivity) / (precision + sensitivity)
  return (F1_score)
}

F1_score(classification_subset, "scored.class", "class")
```

```
## [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.)

$Precision = \frac{TP}{TP+FP}$ $Sensitivity = \frac{TP}{TP+FN}$
$F1 \quad Score = (2 \times Precision \times Sensitivity)/(Presicision + Sensivity)$

According to the formula above, for precision and sensitivity since denominator are both larger than the numerator, both precision and sensitivity has a range of (0,1). Therefore, the product of precision and sensitivity will be smaller than either one of them. For example:

(1) 0 < a < 1

(2) 0 < b < 1

(3) 0 < ab < a

(4) 0 < ab < b

Add (3) and (4)

0 < 2ab < a + b

So that, 2ab will always be smaller than a + b . The upper bound for F1 score is 1. If 2ab approaches 0, then F1 score approaches 0. The lower bound of F1 score is 0.

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.

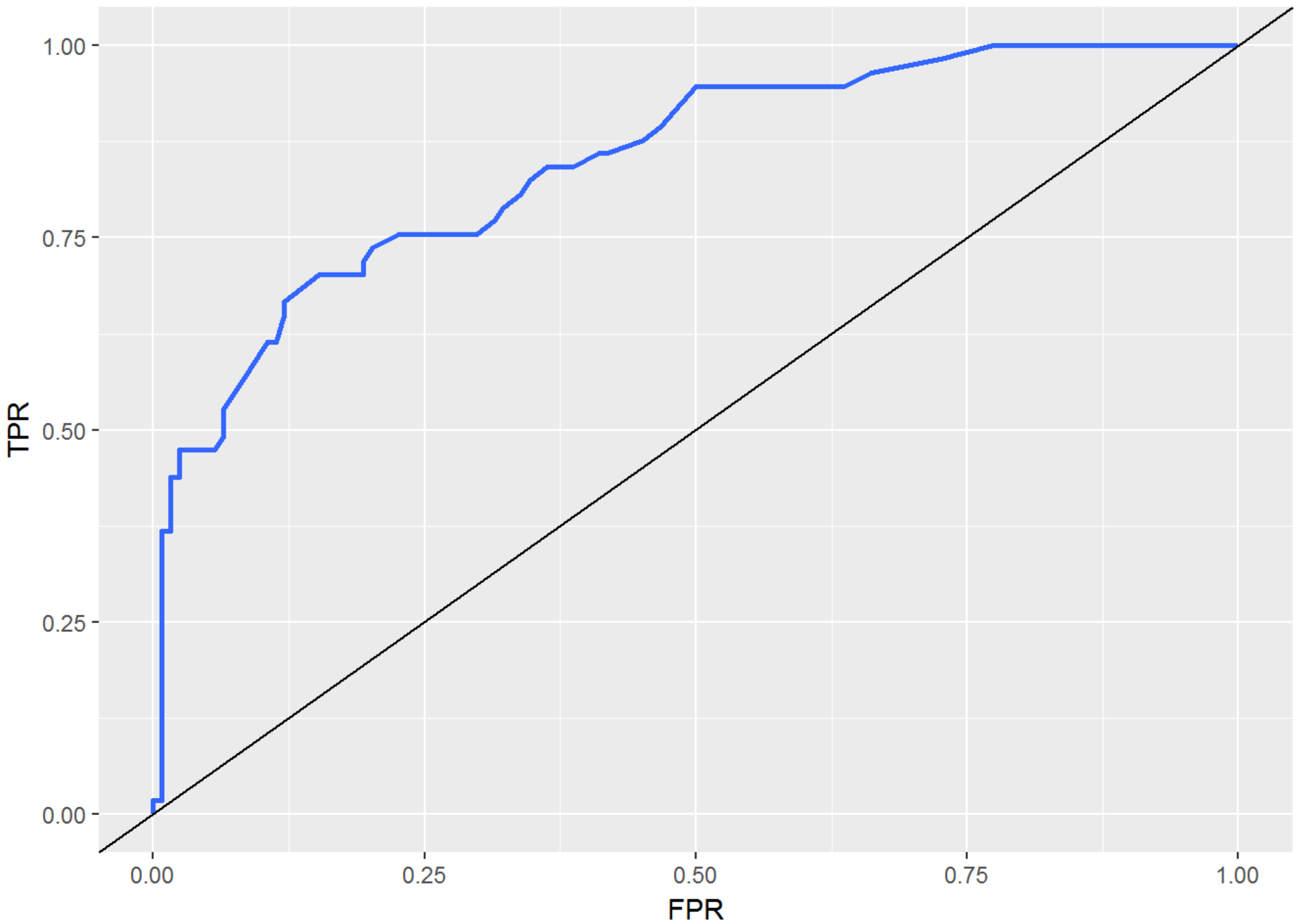
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
ROC_curve <- function(data_frame)
{
  FPR = numeric(0)
  TPR = numeric(0)

  for (i in seq(0.01, 1, by = 0.01))
    {
    predicted <- as.numeric(data_frame$scored.probability > i)
    predicted <- factor(predicted, levels = c(0, 1))
    actual <- factor(data_frame$class, levels = c(0, 1))

    confusion_matrix <- table(predicted, actual)
    TN <- confusion_matrix[1,1]
    FN <- confusion_matrix[1,2]
    FP <- confusion_matrix[2,1]
    TP <- confusion_matrix[2,2]

    specificity <- (TN) /(TN + FP)
    sensitivity <- (TP) /(TP + FN)
    FPR <- c(FPR, 1 - specificity)
    TPR <- c(TPR, sensitivity)
  }

  output <- data.frame(FPR, TPR)
  print (ggplot(output, aes(x = FPR, y = TPR)) + geom_smooth(stat="identity") + geom_abline(inte
rcept = 0, slope = 1))

  #The following code for AUC in this function was inspired by a blog. Here is the link to acces
s the blog: https://www.r-bloggers.com/calculating-auc-the-area-under-a-roc-curve/

  dFPR <- c(abs(diff(FPR)), 0)
  dTPR <- c(abs(diff(TPR)), 0)
  AUC <- (sum(TPR * dFPR) + sum(dTPR * dFPR)/2)
  paste ("Area under the curve: ", toString(AUC), collapse = "")
}

ROC_data <- classification_output[, c("class", "scored.probability")]
head(ROC_data)
```

```
##   class scored.probability
## 1     0         0.32845226
## 2     0         0.27319044
## 3     1         0.10966039
## 4     0         0.05599835
## 5     0         0.10049072
## 6     0         0.05515460
```

```
ROC_curve(ROC_data)
```




```
## [1] "Area under the curve:  0.859083191850594"
```

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

```
get_accuracy(classification_subset, "scored.class", "class")
```

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

```
classification_error_rate(classification_subset, "scored.class", "class")
```

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

```
get_precision(classification_subset, "scored.class", "class")
```

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

```
get_sensitivity(classification_subset, "scored.class", "class")
```

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

```
get_specificity(classification_subset, "scored.class", "class")
```

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

```
F1_score(classification_subset, "scored.class", "class")
```

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

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?

The results from using the caret package are identical to those from using my own function. However, the confusionMatrix function from caret package is more useful in my oppinion, because it can not only provides the confusion matrix, it also lists out all the important value in the meantime, such as accuracy, sensitivity, specificity et cetera.

```
library("caret")
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

```
confusion_m <- confusionMatrix(data = classification_subset[,"scored.class"], reference = classi
fication_subset[,"class"])
confusion_m
```

```
## 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
##
```

```
sensitivity(data = as.factor(classification_subset[,"scored.class"]), reference = as.factor(clas
sification_subset[,"class"]))
```

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

```
specificity(data = as.factor(classification_subset[,"scored.class"]), reference = as.factor(clas
sification_subset[,"class"]))
```

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

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?

```
library(pROC)
```

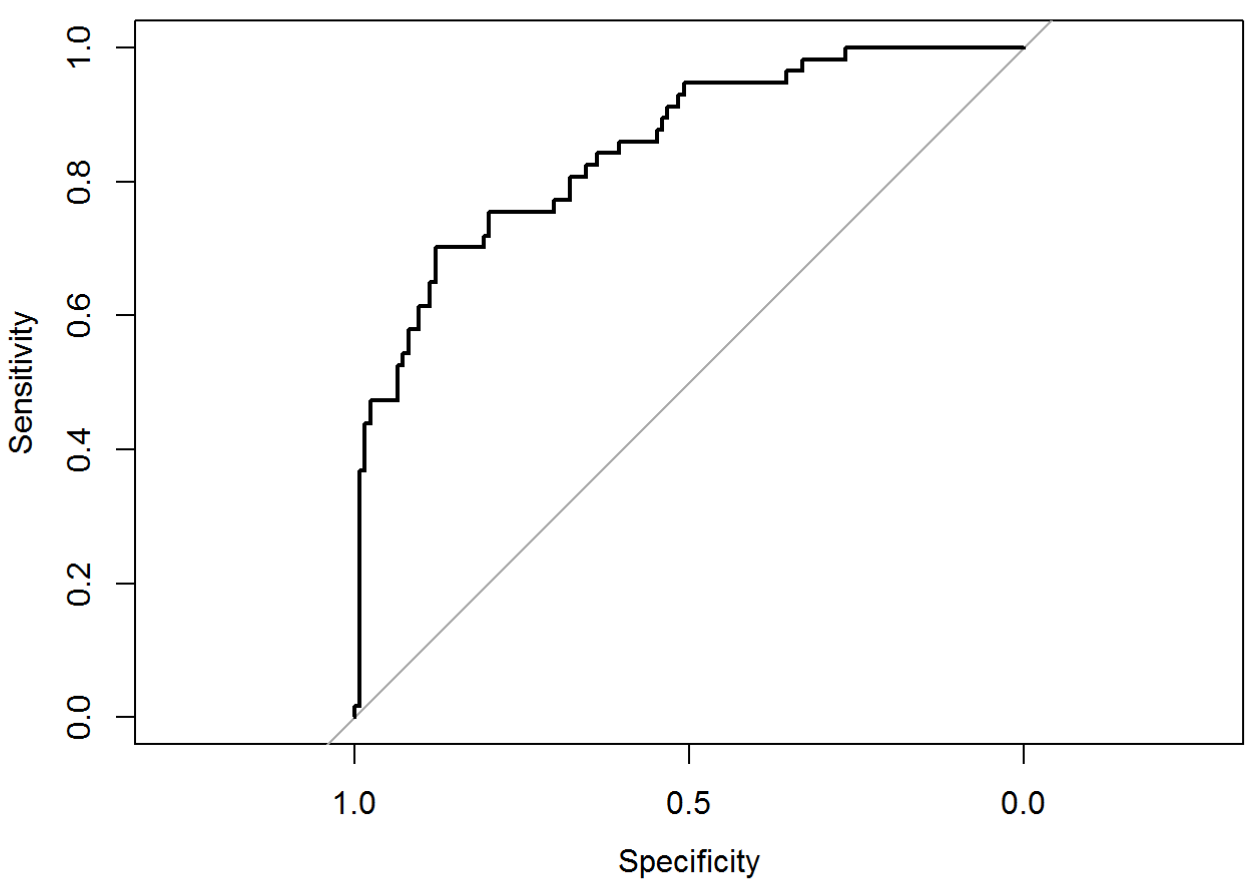
```
## Warning: package 'pROC' was built under R version 3.3.3
```

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

```
##
## Attaching package: 'pROC'
```

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

```
ROC_curve <- roc(classification_subset[,"class"], classification_subset[,"scored.probability"])
plot(ROC_curve)
```




```
auc(ROC_curve)
```

```
## Area under the curve: 0.8503
```

The ROC curve looks similar to that generated by my function. The AUC differs a little. It was 0.8590 for my function, while it was 0.8503 for pROC functions. It could be because pROC package sets up different thresholds for calculating the AUC.