DATA621 Homework 2

Javern Wilson, Joseph Simone, Paul Perez, Jack Russo

3/6/2020

Contents

Overview In this homework assignment, we will work through various classification metrics. Functions are in R to carry out the various calculations. We will also investigate some functions in packages that will let us obtain the equivalent results. Finally, we will create graphical output that also can be used to evaluate the output of classification models.

Libraries

```
class_output <- read.csv("classification-output-data.csv", header = T)
head(class_output)
```

Data Import

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

```
df <- read.csv(paste0("https://raw.githubusercontent.com/josephsimone/Data621/master/project2/1/classifi
```

```
confusion_matix <- table("Predictions" = class_output$scored.class, "Actual" = class_output$class)
confusion_matix
```

Table() Function

```
##            Actual
## Predictions   0   1
##           0 119  30
##           1   5  27
```

The rows represent predictions while the columns represent the actual observations.

ACCURACY Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the accuracy of the predictions

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

Accuracy can be defined as the fraction of predicitons our model got right. Also known as the error rate, the accuracy rate makes no distinction about the type of error being made.

```
cl_accuracy <- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
  FP <- cm[2,1]
  FN <- cm[1,2]

  return((TP + TN)/(TP + FP + TN + FN))
}
```

CLASSIFICATION ERROR RATE 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.

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

The Classification Error Rate calculates the number of incorrect predictions out of the total number of predictions in the dataset.

```
cl_cer <- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
  FP <- cm[2,1]
  FN <- cm[1,2]

  return((FP + FN)/(TP + FP + TN + FN))
}
```

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

```
(cl_accuracy(class_output)+ cl_cer(class_output)) == 1
```

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

PRECISION Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the precision of the predictions.

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

This is the positive value or the fraction of the positive predictions that are actually positive.

```
cl_precision <- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
  FP <- cm[2,1]
  FN <- cm[1,2]

  return(TP/(TP + FP))
}
```

SENSITIVITY 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.

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

The sensitivity is sometimes considered the true positive rate since it measures the accuracy in the event population.

```
cl_sensitivity <- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
  FP <- cm[2,1]
  FN <- cm[1,2]

  return((TP)/(TP + FN))
}
```

SPECIFICITY Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the specificity of the predictions.

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

This is the true negatitive rate or the proportion of negatives that are correctly identified.

```
cl_specificity<- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
```

```
  FP <- cm[2,1]
  FN <- cm[1,2]

  return((TN)/(TN + FP))
}
```

F1 SCORE OF PREDICTIONS 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

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

The F1 Score of Predictions measures the test's accuracy, on a scale of 0 to 1 where a value of 1 is the most accurate and the value of 0 is the least accurate.

```
cl_f1score <- function(df){
  cm <- table("Predictions" = df$scored.class, "Actual" = df$class)

  TP <- cm[2,2]
  TN <- cm[1,1]
  FP <- cm[2,1]
  FN <- cm[1,2]

  f1score <- (2 * cl_precision(df) * cl_sensitivity(df)) / (cl_precision(df) + cl_sensitivity(df))
  return(f1score)
}
```

F1 SCORE BOUNDS Before Ze moYe 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 $0 < a < 1$ then $ab < a$.)

```
f1_score_function <- function(cl_precision, cl_sensitivity){
  f1_score <- (2*cl_precision*cl_sensitivity)/(cl_precision+cl_sensitivity)
  return (f1_score)
}
```

```
(f1_score_function(0, .5))
```

```
## [1] 0
```

```
(f1_score_function(1, 1))
```

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

```
p <- runif(100, min = 0, max = 1)
s <- runif(100, min = 0, max = 1)
f <- (2*p*s)/(p+s)
summary(f)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01915 0.25298 0.49870 0.46308 0.66196 0.92541
```

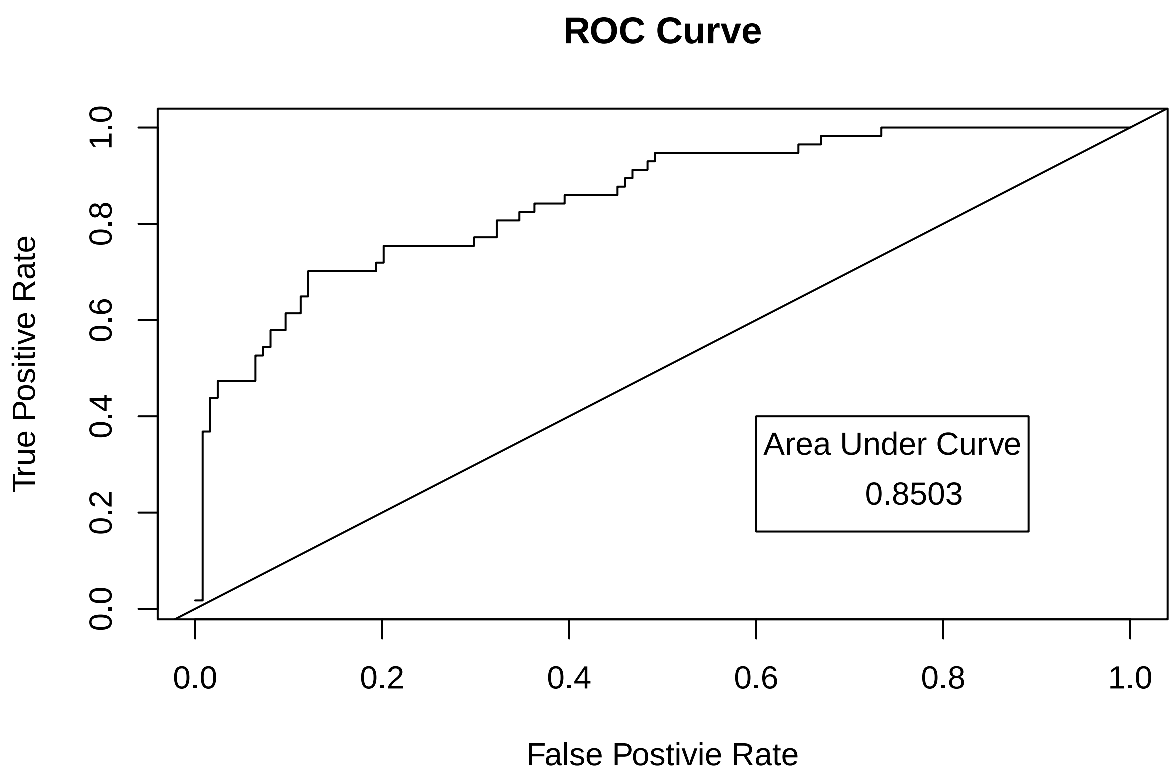
ROC CURVE Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area under the curve (AUC).

```
ROC <- function(x, y){
  x <- x[order(y, decreasing = TRUE)]
 t_p_r <- cumsum(x) / sum(x)
 f_p_r <- cumsum(!x) / sum(!x)
  xy <- data.frame(t_p_r,f_p_r, x)

 f_p_r_df <- c(diff(xy$f_p_r), 0)
 t_p_r_df <- c(diff(xy$t_p_r), 0)
  A_U_C <- round(sum(xy$t_p_r *f_p_r_df) + sum(t_p_r_df *f_p_r_df)/2, 4)

  plot(xy$f_p_r, xy$t_p_r, type = "l",
       main = "ROC Curve",
       xlab = "False Postivie Rate",
       ylab = "True Positive Rate")
  abline(a = 0, b = 1)
  legend(.6, .4, A_U_C, title = "Area Under Curve")
}
```

```
ROC(df$class,df$scored.probability)
```



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

```
N <- c('Accuracy','Classification Error Rate', 'Precision', 'Sensitivity','Specificity', 'F1 Score')
V <- round(c(cl_accuracy(df), cl_cer(df), cl_precision (df), cl_sensitivity(df), cl_specificity(df), cl_
df_1 <- as.data.frame(cbind(N, V))
kable(df_1)
```

N	V
Accuracy	0.8066
Classification Error Rate	0.1934
Precision	0.8438
Sensitivity	0.4737
Specificity	0.9597
F1 Score	0.6067

CARET 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?

```
confusionMatrix(data = factor(class_output$scored.class), reference = factor(class_output$class), posit
```

```
## 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
##
```

```
# Caret - sensitivity
sensitivity(data = factor(class_output$scored.class), reference = factor(class_output$class), positive =
```

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

```r
# Created - sensitivity
cl_sensitivity(df=df)
```

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

The homebrew function matches the result of the caret sensitivity function.

```r
# Caret - specificity
specificity(data = factor(class_output$scored.class), reference = factor(class_output$class), negative =
```

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

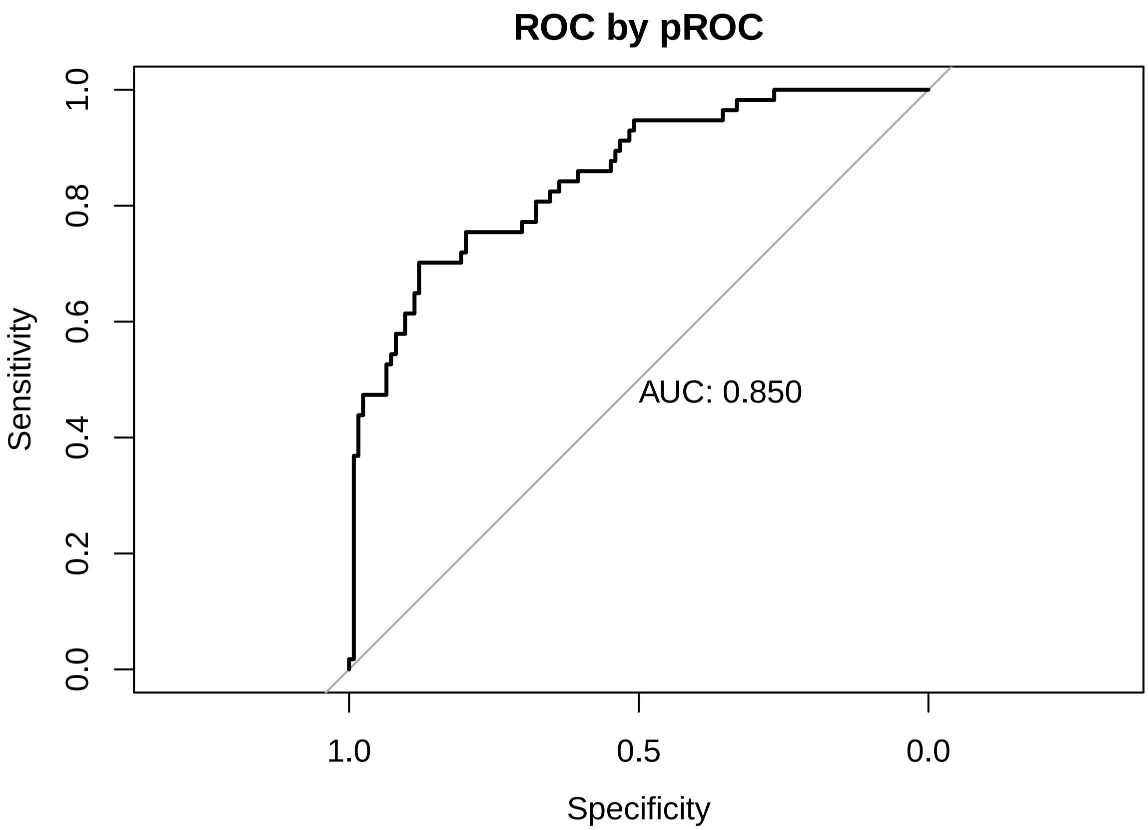
```r
# Created - specificity
cl_specificity(df=df)
```

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

The function matches the result of the caret sensitivity function.

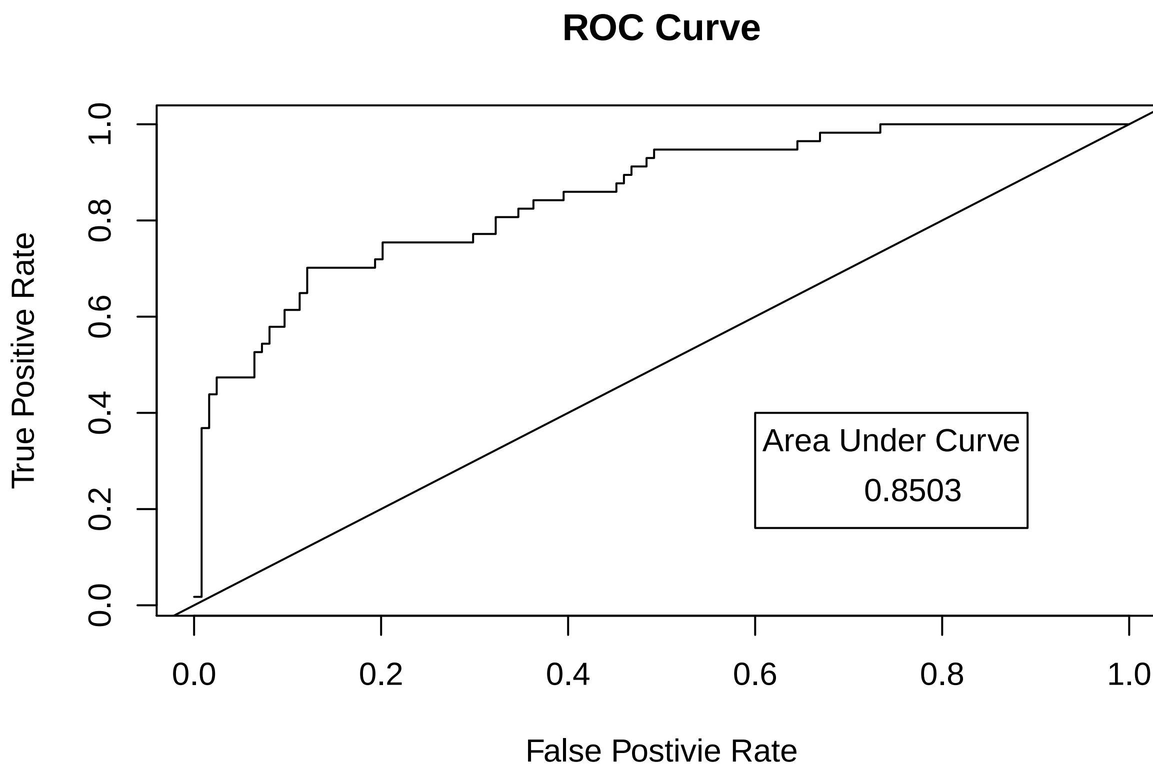
pROC Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```r
plot(roc(df$class, df$scored.probability), print.auc = TRUE , main = "ROC by pROC")
```



pROC

```
ROC(df$class,df$scored.probability)
```



R Function Created

While the two graphs, yield the same result. There are slight differences. The `pROC` package places values on the X-label in a range of 1.5<->-0.5. The function we wrote for this assingment, places values 0<-> 1 on the X-label. In addition, the function we wrote for this assignment extends the findings for the Area Underneath the Curve and extra decimal value.