Classification Metrics

David Moste, Saida Perveen, Vanita Thompson

3/21/2021

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

```
library(tidyverse)
library(knitr)
library(caret)
library(pROC)
class_data <- read_csv("https://raw.githubusercontent.com/Vthomps000/DATA621/main/Data/classification-ou
```

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?

```
class_df <- dplyr::select(class_data, scored.class, class)
table(class_df)
```

```
##              class
## scored.class   0   1
##            0 119  30
##            1   5  27
```

The rows represent the predicted class and the columns represent the actual class.

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_calc <- function(df, actual, predicted){
TP <- sum(actual == 1 & predicted == 1)
TN <- sum(actual == 0 & predicted == 0)
(TP + TN)/nrow(df)
}
```

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.

```
error_calc <- function(df, actual, predicted){
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)
  (FP + FN)/nrow(df)
}
```

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_calc <- function(df, actual, predicted){
  TP <- sum(actual == 1 & predicted == 1)
  FP <- sum(actual == 0 & predicted == 1)
  TP/(TP + FP)
}
```

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_calc <- function(df, actual, predicted){
  TP <- sum(actual == 1 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)
  TP/(TP + FN)
}
```

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.

```
spec_calc <- function(df, actual, predicted){
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  TN/(TN + FP)
}
```

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 <-function(df, actual, predicted){
  TP <- sum(actual == 1 & predicted == 1)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  prec <- TP/(TP + FP)
  sens <- TP/(TP + FN)

  (2*prec*sens)/(prec+sens)
}
```

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.

If the FP and FN are close to 0 (small), then the f1 score would be close to 1. If the FP or FN is close to 1, the f1 score will be close to 0. Therefore, the score will always be close to 0 and 1.

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).

```r
roc_func <- function(x,y){
  x <- x[order(y, decreasing=TRUE)]
  TP = cumsum(x)/sum(x)
  FP = cumsum(!x)/sum(!x)

  df <- data.frame(TP, FP)
  diffFP <- c(diff(FP), 0)
  diffTP <- c(diff(TP), 0)
  auc <- sum(TP * diffFP) +
    sum(diffTP * diffFP)/2

  return(c(df=df, auc = auc))
}
```

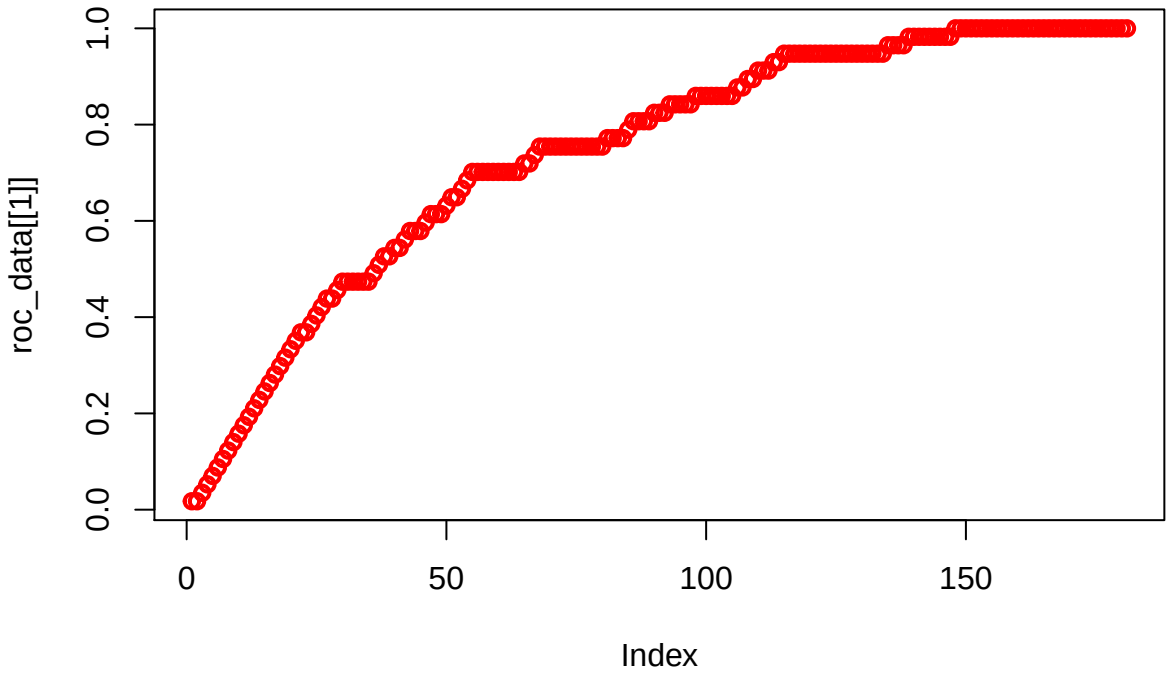
11. Use your created R functions and the provided classification output data set to produce all of the classification metrics discussed above.

```r
accuracy_var <- accuracy_calc(class_data, class_data$class, class_data$scored.class)
error_var <- error_calc(class_data, class_data$class, class_data$scored.class)
percision_var <- precision_calc(class_data, class_data$class, class_data$scored.class)
sensitivity_var <- sensitivity_calc(class_data, class_data$class, class_data$scored.class)
spec_var <- spec_calc(class_data, class_data$class, class_data$scored.class)
f1_var <- f1_score(class_data, class_data$class, class_data$scored.class)

df1 <- c(accuracy_var, error_var, f1_var, percision_var, sensitivity_var, spec_var)
names(df1) <- c("Accuracy", "Error", "F1", "Precision", "Sensitivity", "Specificity")
df1
```

```
##    Accuracy       Error          F1   Precision Sensitivity Specificity
##   0.8066298   0.1933702   0.6067416   0.8437500   0.4736842   0.9596774
```

```r
roc_data <- roc_func(class_data$class, class_data$scored.probability)
plot(roc_data[[1]],
     col="red",
     lwd=2)
```




```
roc_data$auc
```

```
## [1] 0.8503113
```

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?

```
# Convert class and scored.class into factors for use with caret
class_df$class <- as.factor(class_df$class)
class_df$scored.class <- as.factor(class_df$scored.class)

# Produce a confusion matrix with caret
caret::confusionMatrix(data = class_df$scored.class,
                       reference = class_df$class,
                       mode = 'everything')

# Calculate the sensitivity with caret
caret::sensitivity(data = class_df$scored.class,
                   reference = class_df$class,
                   positive = 1)

# Calculate the specificity with caret
caret::specificity(data = class_df$scored.class,
                   reference = class_df$class,
                   negative = 0)
```

The results of these functions is identical to the functions I created.

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?

```
# Use pROC to obtain an roc curve
rocCurve <- pROC::roc(response = class_data$class,
                      predictor = class_df$scored.probability)
auc(rocCurve)
plot(rocCurve, legacy.axes = TRUE)
```

Again, the results of these functions are the same as the functions I created.