DATA621 HW2

Team 1: Michael ODonnell

October 8, 2020

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

data added to my github, pulled into rmd from github below:

```
# Loading the data
git_dir <- 'https://raw.github.com/odonnell31/DATA621-HW2/master/data'
class_data = read.csv(paste(git_dir, "/classification-output-data.csv", sep=""))
class_data_subset <- names(class_data) %in% c("class", "scored.class", "scored.probability")
```

Printing a quick view of the data

```
# Loading the data
head(class_data, 3)
```

```
##   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
##   scored.class scored.probability
## 1            0          0.3284523
## 2            0          0.2731904
## 3            0          0.1096604
```

Loading the required libraries for the rest of the assignment..

```
# load required packages
library(ggplot2)
library(caret)
```

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

```
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

2. The data set has three key columns we will use, print a confusion matrix for the scored dataset

```
# Loading the data
table(class_data$class, class_data$scored.class)
```

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

Function for accuracy and output below

```
# write a function to calculate accuracy
accuracy <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])

  acc <- round((TruePositive+TrueNegative)/
                 (TruePositive+TrueNegative+FalsePositive+FalseNegative), 3)

  return(acc)
}

accuracy(class_data)
```

```
## [1] 0.807
```

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.

```
# write a function to calculate classificition error
classification_error <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])
```

```
  class_error <- round((FalsePositive+FalseNegative)/
                         (TruePositive+TrueNegative+FalsePositive+FalseNegative), 3)

  return(class_error)
}

classification_error(class_data)
```

```
## [1] 0.193
```

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.

```
# write a function to calculate precision
precision <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])

  prec <- round((TruePositive)/(TruePositive+FalsePositive), 3)

  return(prec)
}

precision(class_data)
```

```
## [1] 0.844
```

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.

```
# write a function to calculate sensitivity
sensitivity <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])

  sens <- round((TruePositive)/(TruePositive+FalseNegative), 3)

  return(sens)
}
```

```
sensitivity(class_data)
```

```
## [1] 0.474
```

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.

```
# write a function to calculate specificity
specificity <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])

  specs <- round((TrueNegative)/(FalsePositive+TrueNegative), 3)

  return(specs)
}

specificity(class_data)
```

```
## [1] 0.96
```

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.

Function to create f1 score below, leveraging above precision and sensitivity functions.

```
# write a function to calculate F1 Score
f_one_score <- function(df) {
  TruePositive <- nrow(df[df$class == 1 & df$scored.class == 1,])
  TrueNegative <- nrow(df[df$class == 0 & df$scored.class == 0,])
  FalsePositive <- nrow(df[df$class == 0 & df$scored.class == 1,])
  FalseNegative <- nrow(df[df$class == 1 & df$scored.class == 0,])

  f_one <- round((2*precision(df)*sensitivity(df))/
                   (precision(df)+sensitivity(df)), 3)

  return(f_one)
}

f_one_score(class_data)
```

```
## [1] 0.607
```

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.

This is a neat question. To answer, I used runif to generate random numbers for precision and sensitivity. Another way to do this is to use the sequence function for number between 0 and 1.

```
# prove the bounds of F1 score are between 0 and 1
precsision_example <- runif(10, min = 0, max = 1)
sensitivity_example <- runif(10, min = 0, max = 1)
f_one_score_example <- (2 * precsision_example * sensitivity_example)/
  (precsision_example + sensitivity_example)

summary(f_one_score_example)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.007068 0.074329 0.236894 0.244295 0.323995 0.756510
```

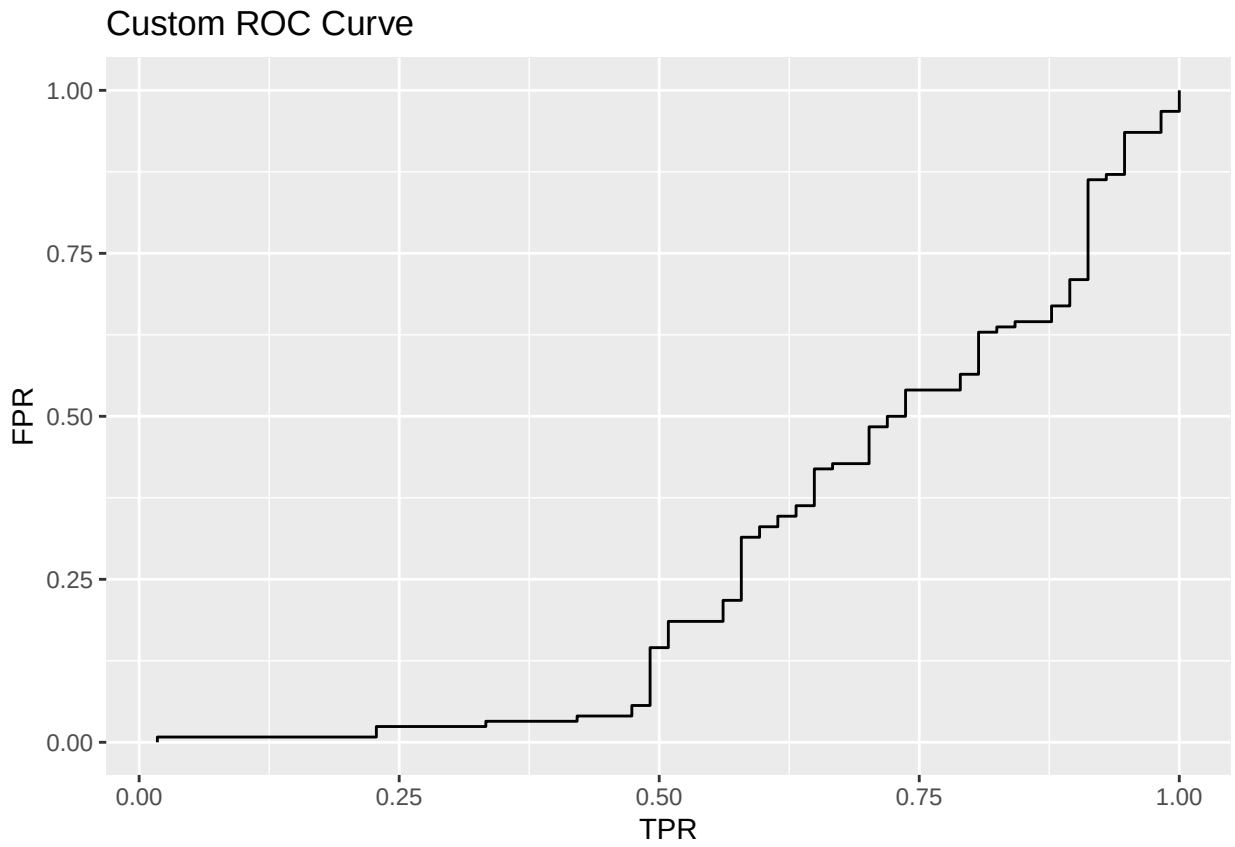
10. Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated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

niave function to create ROC Curve below

```
# prove the bounds of F1 score are between 0 and 1
roc_curve <- function(labels, scores) {
  labels <- labels[order(scores, decreasing=TRUE)]
  df <- data.frame(TPR=cumsum(labels)/sum(labels),
                   FPR=cumsum(!labels)/sum(!labels), labels)

  ggplot(df,aes(TPR,FPR)) +
    geom_line() +
    ggtitle('Custom ROC Curve')
}

roc_curve(class_data$class, class_data$scored.class)
```



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

Creating a function to print all scoring metrics:

```
# write a function to output all metrics from previous functions
all_metrics <- function(df) {
  accuracy_metric <- accuracy(df)
  precision_metric <- precision(df)
  sensitivity_metric <- sensitivity(df)
  specificity_metric <- specificity(df)
  f1_score <- f_one_score(df)

  output_df <- data.frame(accuracy_metric, precision_metric,
                          sensitivity_metric, specificity_metric,
                          f1_score)

  return(output_df)
}

all_metrics(class_data)
```

```
##   accuracy_metric precision_metric sensitivity_metric specificity_metric
## 1           0.807            0.844              0.474               0.96
##   f1_score
## 1    0.607
```

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 are nearly the same, but the caret package is very convenient.

```
# check out the caret package
confusionMatrix(table(class_data$class, class_data$scored.class),
                reference = class_data$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?

The results hold fairly similar to my own function. But, the pROC package is far more sophistocated and robust than my function. Will use pROC going forward!

```
# check out the caret package
roc1 <- roc(class_data$class,
            class_data$scored.class, percent=TRUE,
            # arguments for plot
            plot=TRUE, auc.polygon=TRUE, max.auc.polygon=TRUE, grid=TRUE,
            print.auc=TRUE, show.thres=TRUE)
```

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

```
## Setting direction: controls < cases
```