DATA621_HW2

Critical group 2: Avery Davidowitz, Josh Iden, Mathew Katz, Tyler Brown, Gabriel Santos, John Ledesma

2023-03-11

```
library(tidyr);
library(dplyr);
library(kableExtra);
library(ggplot2)
library(caret)
library(pROC)
```

Introduction HW2

We have a dataset called "classification-output-data.csv", which has a set of independent variables and a class, along with a predictive classification model with scored probability and a scored class based on scored probability. We have to use the following 3 key columns to derive some key metrics from the ranking model.

- 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

Finally, a graph will be created and we can evaluate the models.

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

Loading the data

```
git_dir <- 'https://raw.github.com/GabrielSantos33/DATA621_HW2/main/'
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")

head(class_data, 6)
```

```
##   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?

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

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

We get the column of actual class and the column of the predicted class for the observation.

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.

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

Function for accuracy and output below:

```
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
```

The accuracy is: 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.

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

```
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
```

The Classification error date is 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.

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

```
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
```

The precision es 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.

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

```
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
```

The sensitivity is 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.

$$Sensitivity = \frac{TP}{TN + FP}$$

```
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
```

The specificity is 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.

$$F1score = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

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

```
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
```

The F1 score is 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. (Hint: If $0 < a < 1$ and $0 < b < 1$ then $ab < a$.)

I used runif() function to generate random numbers for precision and sensitivity.

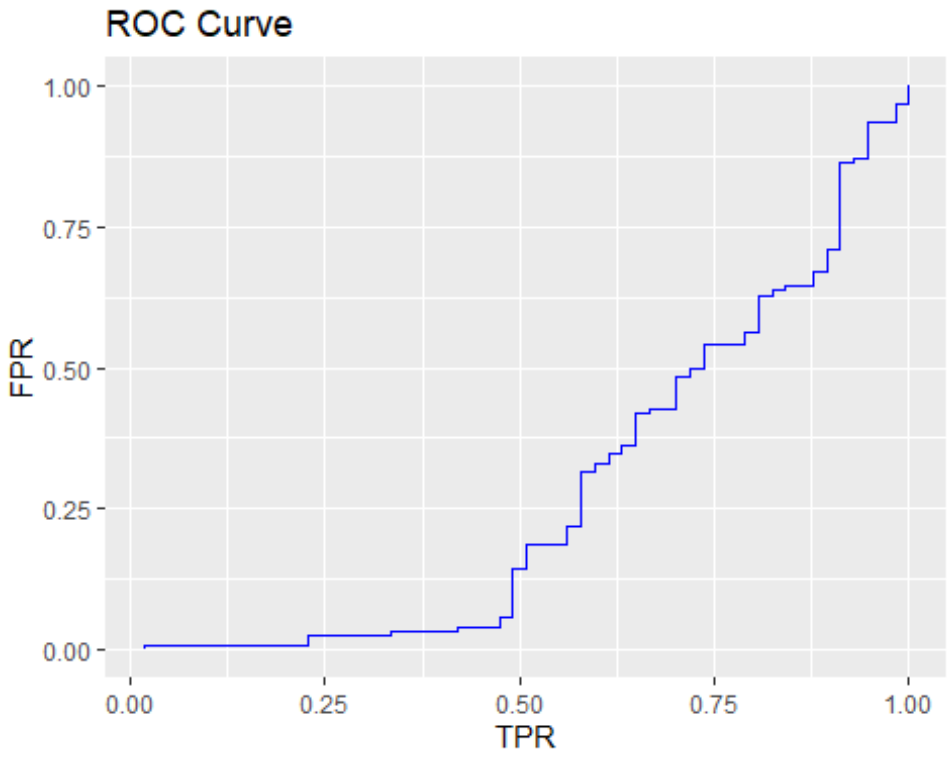
```
precsision_example <- runif(10, min = 0, max = 1)
sensitivity_example <- runif(10, min = 0, max = 1)
f_one_score_example <- (2 * precsision_example * sensitivity_example)/
  (precsision_example + sensitivity_example)
summary(f_one_score_example)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1002  0.2688  0.4193  0.4667  0.7276  0.8279
```

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_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(col="blue") +
    ggtitle('ROC Curve')
}
roc_curve(class_data$class, class_data$scored.class)
```



ROC (Receiver Operating Characteristic) Analysis is a useful way to assess the accuracy of model predictions by plotting the sensitivity versus specificity of a classification test.

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

```
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?

```
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
##
```

The results are very similar, it is useful to use the Caret Package.

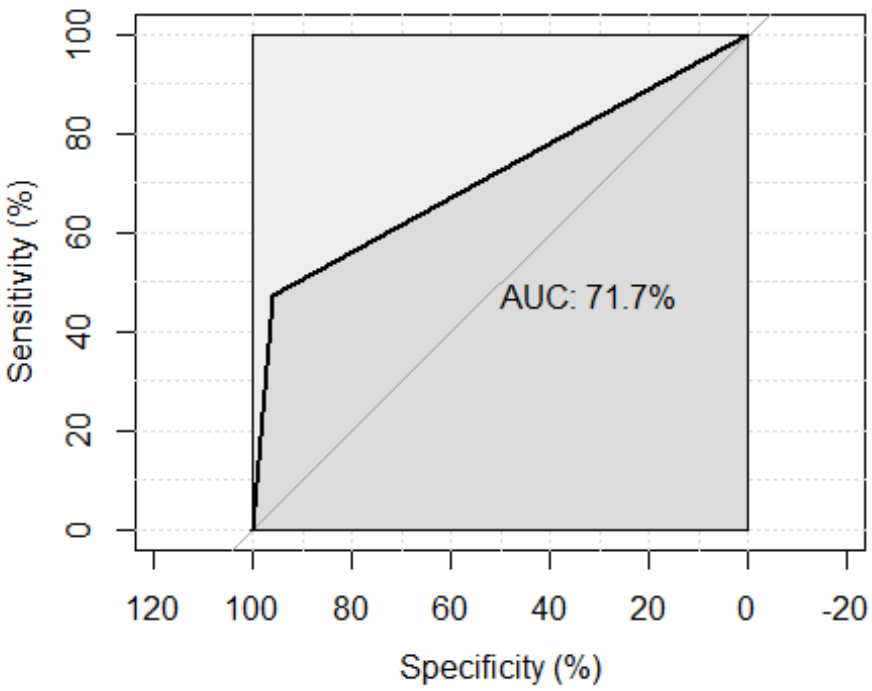
13.Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```
roc1 <- roc(class_data$class,
            class_data$scored.class, percent=TRUE,

            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
```



The results between the pROC package and the function are similar. The pROC package is very complete and gives better results, being better to use. Overall, the pROC package is a powerful tool for generating ROC curves and calculating AUC values in R, and is worth exploring further for anyone working with classification problems.