Data 621 Assignment 2

Mark Gonsalves, Joshua Hummell, Claire Meyer, Chinedu Onyeka, Rathish Parayil Sasidharan

3/16/2022

Assignment Two

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).

```
class_output <- read_csv("https://raw.githubusercontent.com/jhumms/Data-621/main/Assignment-2/classifica

class_output <- class_output %>%
    select(class, scored.class, scored.probability)
```

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

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

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

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

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

```
accuracy <- function(df) {
    # create the column labels
    cols = c("TN", "FN", "FP", "TP")
    # Create a matrix with Table
    confusion_matrix <- table(Actual = df$class, Predicted = df$scored.class)
    # Add in the labels
    confusion_matrix <- data.frame(confusion_matrix, index = cols)
    # get the accuracy and return
    accuracy_value <- (confusion_matrix$Freq[4] + confusion_matrix$Freq[1])/sum(confusion_matrix$Freq)
    return(accuracy_value)
}

accuracy(class_output)
```

```
## [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.

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

```
classErrorRate <- function(df) {
    total <- nrow(class_output)
    FN <- sum(class_output$class == 1 & class_output$scored.class == 0)
    FP <- sum(class_output$class == 0 & class_output$scored.class == 1)
    ((FN + FP)/total)
}

classErrorRate(df)
```

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

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.

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

```
precision <- function(df) {
    total <- nrow(class_output)
    TP <- sum(class_output$class == 1 & class_output$scored.class == 1)
    FP <- sum(class_output$class == 0 & class_output$scored.class == 1)
    (TP/(TP + FP))
}

precision(df)
```

```
## [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.

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

```r
sensitivity <- function(df) {
    total <- nrow(class_output)
    TP <- sum(class_output$class == 1 & class_output$scored.class == 1)
    FN <- sum(class_output$class == 1 & class_output$scored.class == 0)
    (TP/(TP + FN))
}

sensitivity(df)
```

```
## [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.

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

```r
specificity <- function(df) {
    TN <- sum(df$class == 0 & df$scored.class == 0)
    FP <- sum(df$class == 0 & df$scored.class == 1)
    round(TN/(TN + FP), 4)
}
specificity(class_output)
```

```
## [1] 0.9597
```

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.

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

```r
f1_score <- function(x) {
    (2 * precision(x) * sensitivity(x))/(precision(x) + sensitivity(x))
}
f1_score(class_output)
```

```
## [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 < a < 1 then ab < a.)

Since both Precision and Sensitivity are bounded between 0 and 1. Therefore, F1 score will also be bounded between 0 and 1

We already know that both Precision and Sensitivity are between 0 and 1 hence $Precision * Sensitivity < Precision, Sensitivity$

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

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

$< \frac{Precision}{Precision+Sensitivity} + \frac{Sensitivity}{Precision+Sensitivity}$

$< \frac{Precision+Sensitivity}{Precision+Sensitivity} = 1$

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). Note that I recommend using a sequence of thresholds ranging from 0 to 1 at 0.01 intervals.

Answer: ROC (Receiver Operator Characteristic) curve essentially helps us to visualize how well the classifier is performing and works well for binary classification models. It shows a plot of True Positive Rate (TPR) also known as Sensitivity vs False Positive Rate (FPR) also known as (1 - Specificity).
$TPR = Sensitivity = \frac{TP}{TP+FN}$
$TNR = Specificity = \frac{TN}{TN+FP}$
$FPR = 1 - Specificity = \frac{FP}{TN+FP}$
The ROC curve is a probabilistic curve of TPR vs FPR at various threshold values and it's a measure of the ability of the machine learning classifier to distinguish between classes (to properly classify). In general, the higher the AUC (Area Under the ROC Curve), the better the performance of the model in distinguishing between positive and negative classes.

The function below generates an ROC curve:

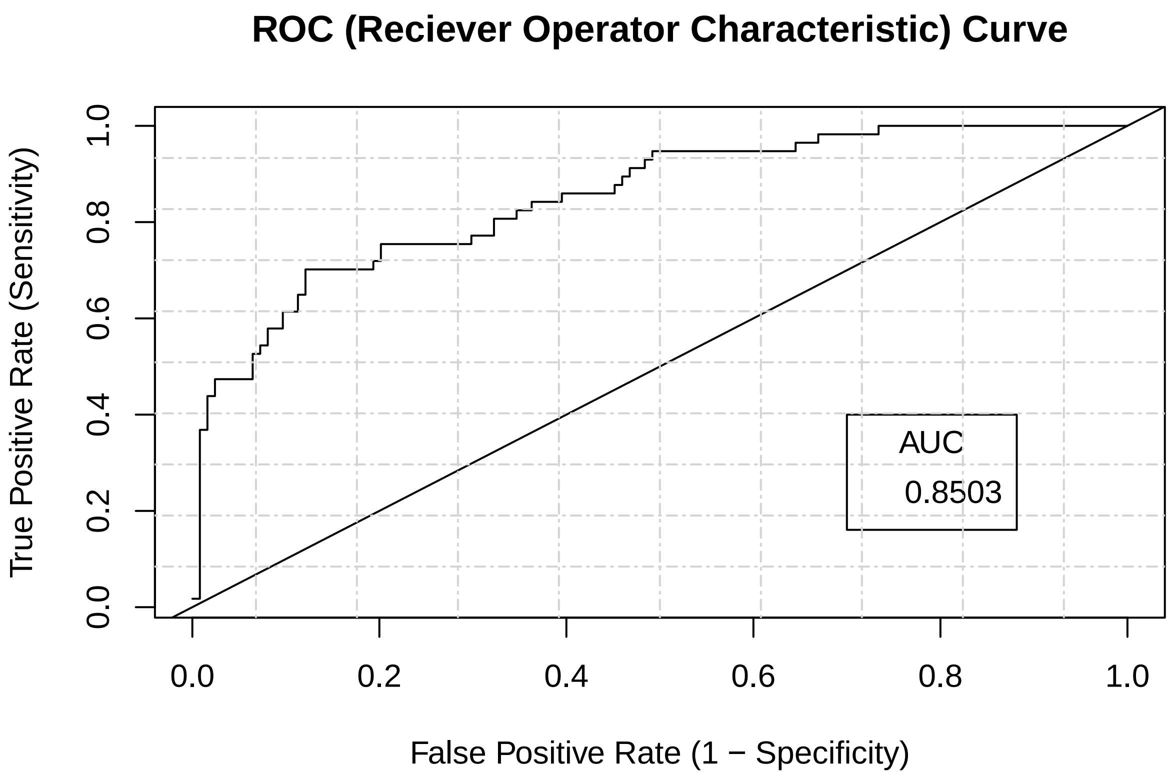
```r
generate_roc <- function(class, scored_prob) {
    # This generates ROC curve from true classification column (class) and probability column
    # scored.probability)
    class <- class[order(scored_prob, decreasing = TRUE)]
    TPR = cumsum(class)/sum(class)
    FPR = cumsum(!class)/sum(!class)
    tpr_fpr_df <- data.frame(TPR, FPR, class)

    tpr_df <- c(diff(tpr_fpr_df$TPR), 0)
    fpr_df <- c(diff(tpr_fpr_df$FPR), 0)
    AUC <- round(sum(tpr_fpr_df$TPR * fpr_df) + sum((tpr_df * fpr_df)/2), 4)

    # plot the ROC curve
    plot(tpr_fpr_df$FPR, tpr_fpr_df$TPR, type = "l", main = "ROC (Reciever Operator Characteristic) Cur
        xlab = "False Positive Rate (1 - Specificity)", ylab = "True Positive Rate (Sensitivity)")
    abline(a = 0, b = 1)
    legend(0.7, 0.4, AUC, title = "AUC")
    grid(10, 10, lty = 6, col = "lightgrey")

}

# call the generate ROC function
generate_roc(class_output$class, class_output$scored.probability)
```



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

```
created_r_functions <- data.frame(accuracy(class_output), classErrorRate(class_output), precision(class_
    sensitivity(class_output), specificity(class_output), f1_score(class_output))

created_r_functions_rownames <- c("Accuracy", "Classification Error Rate", "Precision", "Sensitivity",
    "Specificity", "F1 Score")
created_r_functions_headers <- "Created_R_functions_values"
created_r_functions <- t(created_r_functions)
rownames(created_r_functions) <- created_r_functions_rownames
colnames(created_r_functions) <- created_r_functions_headers
created_r_functions <- created_r_functions %>%
    kbl() %>%
    kable_styling()

# display the classification metrics by using the created R functions
created_r_functions
```

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?

We see near identical results to the homegrown functions above, just rounded to the 4th decimal place, e.g. a sensitivity of 0.4736842 vs. 0.4737 and a specificity of 0.9597000 vs. 0.9597. Accuracy is a match, as well as the confusion matrix itself.

	Created_R_functions_values
Accuracy	0.8066298
Classification Error Rate	0.1933702
Precision	0.8437500
Sensitivity	0.4736842
Specificity	0.9597000
F1 Score	0.6067416

```
confusionMatrix(factor(class_output$scored.class), factor(class_output$class), positive = "1")
```

```
## 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
##
```

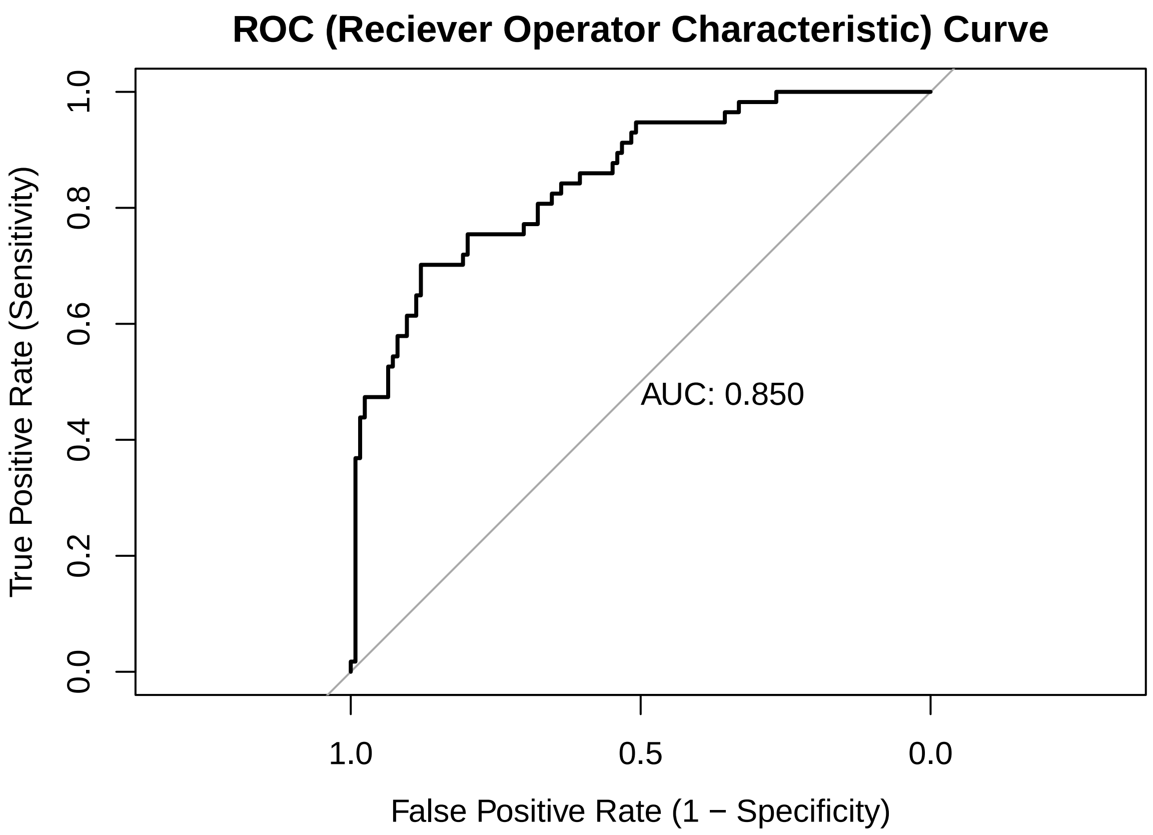
13. Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

Similarly to the caret package, the output from the homegrown ROC function matches the output from the package. The AUC value is slightly lower precision than the homegrown function, at 0.850 vs. 0.8503.

```
plot(roc(class_output$class, class_output$scored.probability), print.auc = TRUE, main = "ROC (Reciever 
    xlab = "False Positive Rate (1 - Specificity)", ylab = "True Positive Rate (Sensitivity)")
```

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

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


ROC (Reciever Operator Characteristic) Curve
True Positive Rate (Sensitivity)
0.0
0.2
0.4
0.6
0.8
1.0
AUC: 0.850
1.0
0.5
0.0
False Positive Rate (1 − Specificity)