DATA 621 HW 2

Vanita Thompson, David Moste, Sadia Perveen

March 17, 2021

Overview 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.

Applied Predictive Modeling, Ch. 11 (provided as a PDF file).

```
install.packages("tidyverse",repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/43/xlmw9f817zzfspr8glz_vdyh0000gn/T//Rtmpf2LemX/downloaded_packages
```

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

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?

The rows represent the predicted, the columns represent the actual classes.

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

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

```
accuracy_calc <- function(df){
TP <- sum(class_df$class == 1 & class_df$scored.class == 1)
TN <- sum(class_df$class == 0 & class_df$scored.class == 0)
(TP + TN)/nrow(df)
}
accuracy_calc(class_data)
```

```
## [1] 0.8066298
```

```
accuracy_var <- accuracy_calc(class_data)
```

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){
  FP <- sum(class_df$class == 0 & class_df$scored.class == 1)
  FN <- sum(class_df$class == 1 & class_df$scored.class == 0)
  (FP + FN)/nrow(df)
}
error_calc(class_data)
```

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

```
error_var <- error_calc(class_data)
```

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){
  TP <- sum(class_df$class == 1 & class_df$scored.class == 1)
  FP <- sum(class_df$class == 0 & class_df$scored.class == 1)
  TP/(TP + FP)
}
precision_calc(class_data)
```

```
## [1] 0.84375
```

```
percision_var <- precision_calc(class_data)
```

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){
  TP <- sum(class_df$class == 1 & class_df$scored.class == 1)
  FN <- sum(class_df$class == 1 & class_df$scored.class == 0)
  TP/(TP + FN)
}
sensitivity_calc(class_data)
```

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

```
sensitivity_var <- sensitivity_calc(class_data)
```

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){
  TN <- sum(class_df$class == 0 & class_df$scored.class == 0)
  FP <- sum(class_df$class == 0 & class_df$scored.class == 1)
  TN/(TN + FP)
}
spec_calc(class_data)
```

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

```
spec_var <- spec_calc(class_data)
```

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 prediction.

```
f1_score <- (2 * percision_var * sensitivity_var) / (percision_var + sensitivity_var)
f1_score
```

```
## [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.

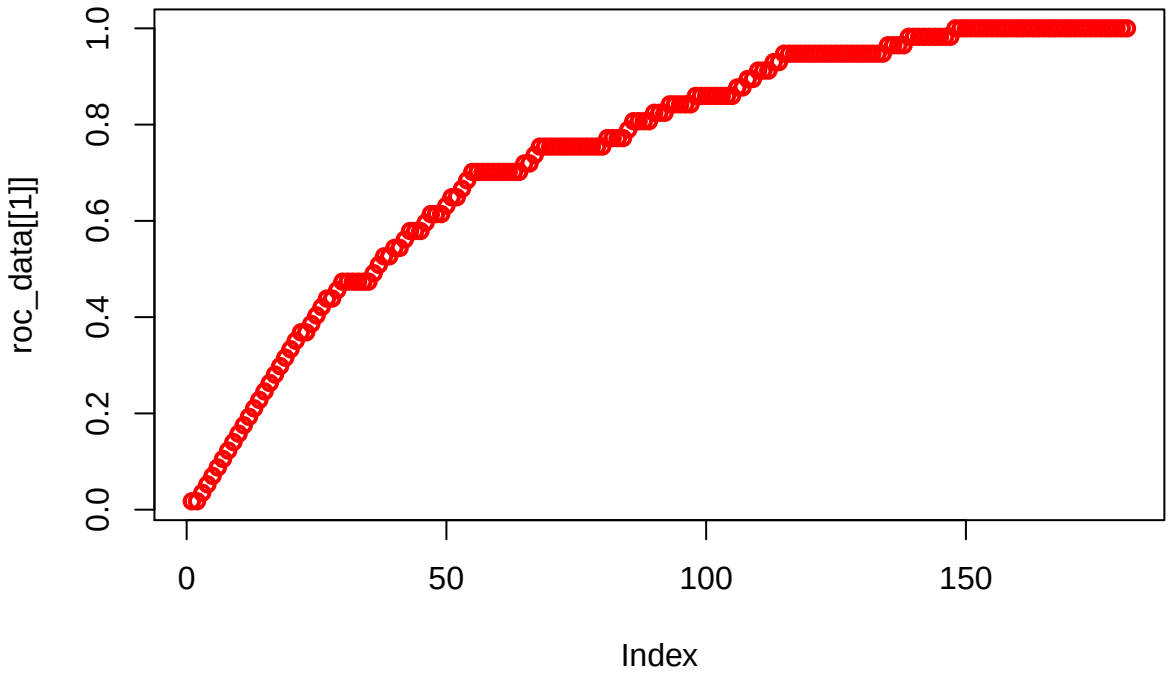
If the FP and FN are close to 0 (small), then the f1 score would be close to 1. If the FP or FN is close to 1, the f1 score will be close to 0. Therefore, the score will always be close to 0 and 1.

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

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




```
kable(roc_data$auc)
```

x
0.8503113

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

```
df1 <- c(accuracy_var, error_var, f1_score, percision_var, sensitivity_var, spec_var)
names(df1) <- c("Accuracy", "Error", "F1", "Percision", "Sensitivity", "Spec")
kable(df1)
```

	x
Accuracy	0.8066298
Error	0.1933702
F1	0.6067416
Percision	0.8437500
Sensitivity	0.4736842
Spec	0.9596774

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?

```
class_df$scored.class <- as.factor(class_df$scored.class)
class_df$class <- as.factor(class_df$class)
confusionMatrix(class_df$scored.class, class_df$class, mode = 'everything')
```

```
## 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 : 0.9597
##             Specificity : 0.4737
##          Pos Pred Value : 0.7987
##          Neg Pred Value : 0.8438
##               Precision : 0.7987
##                  Recall : 0.9597
##                      F1 : 0.8718
##              Prevalence : 0.6851
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.8232
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
##
```

```
caret::sensitivity(class_df$scored.class, class_df$class)
```

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

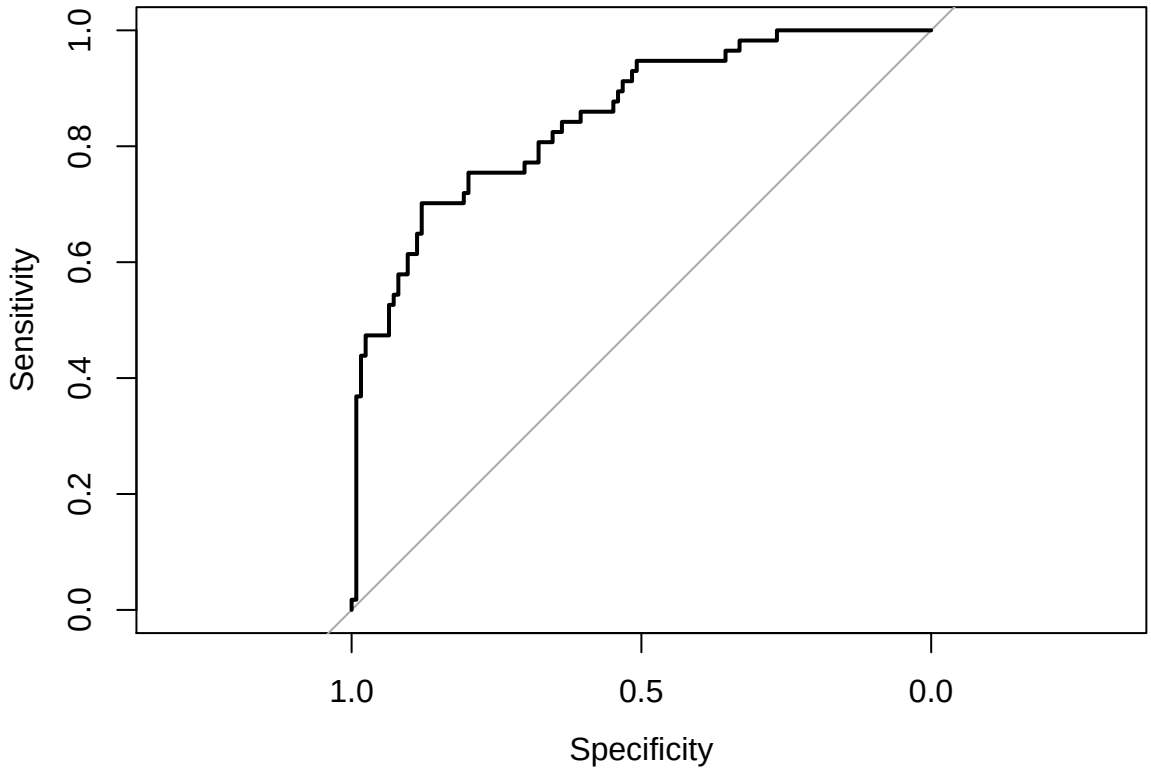
```
caret::specificity(class_df$scored.class, class_df$class)
```

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

The values from the built in function are almost exactly the same to the hand created ones.

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**

```
library(pROC)
plot(roc(class_data$class, class_data$scored.probability),
     colorize=TRUE, print.cutoffs.at=seq(0.1,by=0.1))
```




```
auc(roc(class_data$class, class_data$scored.probability))
```

```
## Area under the curve: 0.8503
```

Both the built in function and hand-made functions are very similar to each other.