Week 2 Assignment

By Brian Weinfeld

June 22nd, 2018

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

```
data <- read_csv('C:\\Users\\Brian\\Desktop\\GradClasses\\Summer18\\621\\621week2\\classification-output
  mutate(class = factor(class) %>% relevel(ref='1'),
         scored.class = factor(scored.class) %>% relevel('1')
         )
```

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?

```
table <- table(predicted = data$scored.class, actual = data$class)
table
```

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

The actual and predicted rows and columns are labeled. 1 indicates positive and 0 indicates negative.

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 <- function(data, actual, predicted){
  table <- table(predicted = unlist(data[, predicted]), actual = unlist(data[, actual]))
  (table[1] + table[4]) / sum(table)
}
```

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.Rate <- function(data, actual, predicted){
  table <- table(predicted = unlist(data[, predicted]), actual = unlist(data[, actual]))
  (table[2] + table[3]) / sum(table)
}
```

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 <- function(data, actual, predicted){
  table <- table(predicted = unlist(data[, predicted]), actual = unlist(data[, actual]))
  table[1] / (table[1] + table[3])
}
```

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 <- function(data, actual, predicted){
  table <- table(predicted = unlist(data[, predicted]), actual = unlist(data[, actual]))
  table[1] / (table[1] + table[2])
}
```

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.

```
Specificity <- function(data, actual, predicted){
  table <- table(predicted = unlist(data[, predicted]), actual = unlist(data[, actual]))
  table[4] / (table[4] + table[3])
}
```

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(data, actual, predicted){
  prec <- Precision(data, actual, predicted)
  sens <- Sensitivity(data, actual, predicted)
  (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.???.)

Prove: $0 \leq \frac{2\times a\times b}{a+b} \leq 1$

Use: $0 < a < 1, 0 < b < 1 \therefore a \times b < a$

Then: $0 \leq \frac{2\times a\times b}{a+b} < \frac{a}{a+b} \leq 1$

Conclusion: As $a + b > a$ then $\frac{a}{a+b} < 1$ and as $a > 0, b > 0$ then $\frac{2\times a\times b}{a+b} > 0$

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).

```
ROC <- function(data, actual, predicted.prob){
  sens <- numeric(100)
  spec <- numeric(100)
  for(i in seq(0, 100, 1)){
    data$myPredict <- data[, predicted.prob] <= (i / 100)
    sens[i] <- Sensitivity(data, actual, 12)
    spec[i] <- 1 - Specificity(data, actual, 12)
  }
  plot <- ggplot(data_frame(spec=spec, sens=sens), aes(spec, sens)) +
    geom_point() +
    geom_abline(slope=1, intercept=0, color='red') +
    geom_line() +
    labs(x = 'False Positive Rate',
         y = 'True Positive Rate')
  auc <- sintegral(spec, sens, n.pts=1)$cdf$y
  return(list(plot, auc))
}
```

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

```
Accuracy(data, 9, 10)
```

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

```
Error.Rate(data, 9, 10)
```

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

```
Precision(data, 9, 10)
```

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

```
Sensitivity(data, 9, 10)
```

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

```
Specificity(data, 9, 10)
```

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

```
F1.Score(data, 9, 10)
```

```
## [1] 0.6067416
```

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?

```
caret::confusionMatrix(table, mode='everything')
```

```
## Confusion Matrix and Statistics
##
##          actual
## predicted   1   0
##         1  27   5
##         0  30 119
##
##                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
##               Precision : 0.8438
##                  Recall : 0.4737
##                      F1 : 0.6067
##              Prevalence : 0.3149
##          Detection Rate : 0.1492
##    Detection Prevalence : 0.1768
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 1
##
```

```
caret::sensitivity(table)
```

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

```
caret::specificity(table)
```

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

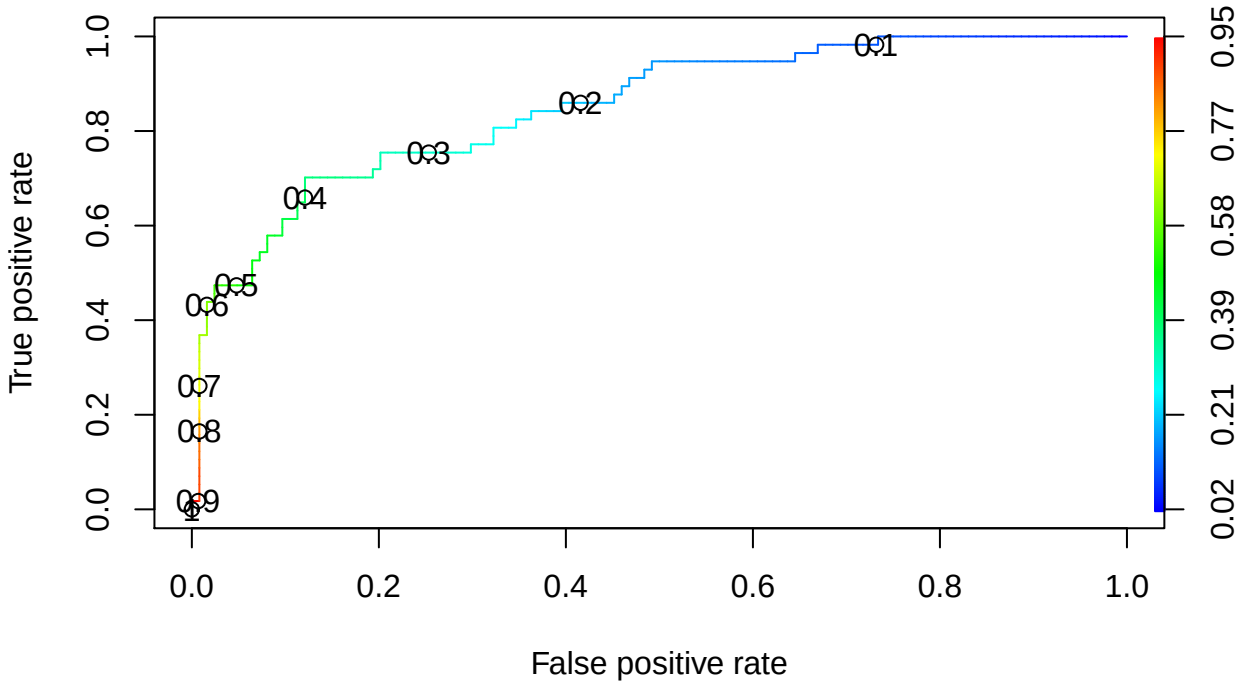
My answers were initially different from those provided by caret. After inspection I discovered that caret and I were placing prediction/actual in different rows/columns. In order to bring the answers, and code, into alignment, I altered all of my methods to match caret. Afterwards, all of the answers matched.

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?

```
ROCRPred <- prediction(data[, 11], data[, 9])
ROCRPref <- performance(ROCRPred, 'tpr', 'fpr')
sintegral(unlist(ROCRPref@x.values), unlist(ROCRPref@y.values), n.pts=1)$cdf$y
```

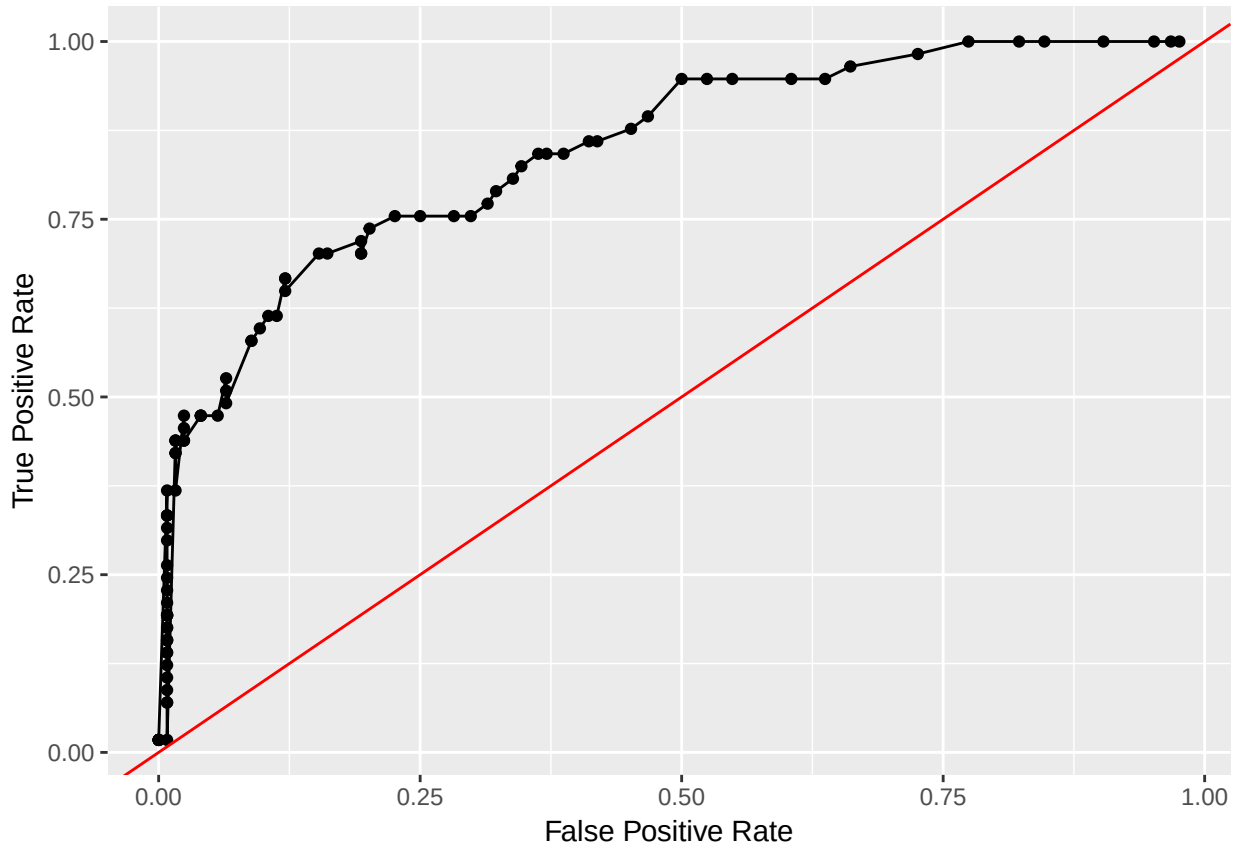
```
## [1] 0.7997076
```

```
plot(ROCRPref, colorize=TRUE, print.cutoffs.at = seq(0.1, by=0.1))
```




```
ROC(data, 9, 11)
```

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




```
##
## [[2]]
## [1] 0.7689469
```

The results are nearly identical between my graph and the one from the ROCR package. There are slight differences that appear to be due to my choice of sequencing only 100 equally spaces points between 0 and 1. This is represented in the incredibly small different in the AUC calcations from my graph compared to the ROCR package. A difference of only ≈ 0.03 indicates that our answers are in alignment.