BHao_HW2

```r
# 1-2.
setwd("~/Google Drive/CUNY/git/DATA621/HW2")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
df = read.csv('classification-output-data.csv', header = TRUE)
str(df)
```

```
## 'data.frame':    181 obs. of  11 variables:
##  $ pregnant          : int  7 2 3 1 4 1 9 8 1 2 ...
##  $ glucose           : int  124 122 107 91 83 100 89 120 79 123 ...
##  $ diastolic         : int  70 76 62 64 86 74 62 78 60 48 ...
##  $ skinfold          : int  33 27 13 24 19 12 0 0 42 32 ...
##  $ insulin           : int  215 200 48 0 0 46 0 0 48 165 ...
##  $ bmi               : num  25.5 35.9 22.9 29.2 29.3 19.5 22.5 25 43.5 42.1 ...
##  $ pedigree          : num  0.161 0.483 0.678 0.192 0.317 0.149 0.142 0.409 0.678 0.52 ...
##  $ age               : int  37 26 23 21 34 28 33 64 23 26 ...
##  $ class             : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ scored.class      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ scored.probability: num  0.328 0.273 0.11 0.056 0.1 ...
```

```r
table(pred = df$scored.class, actual = df$class)
```

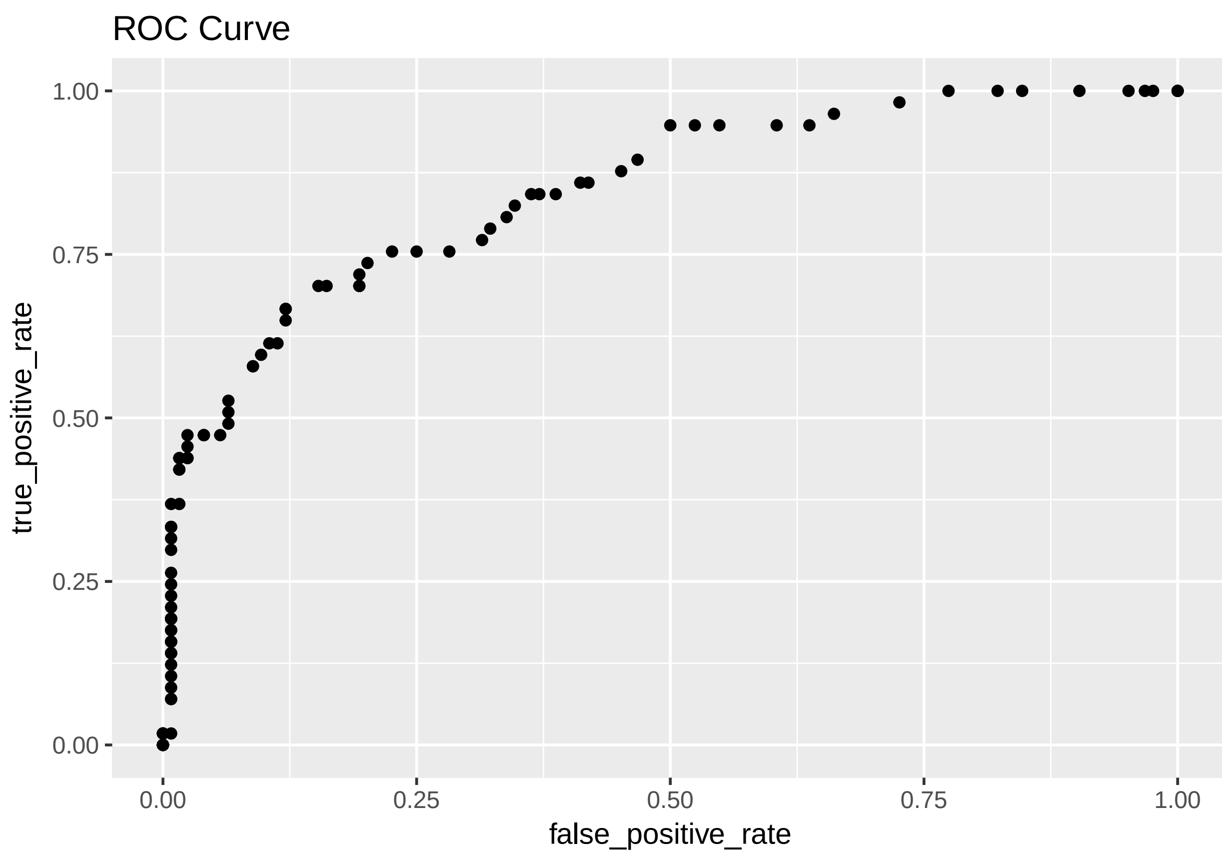
```
##     actual
## pred   0   1
##    0 119  30
##    1   5  27
```

```r
# 3-8.
# accuracy = % of true positives and true negatives
pred_accuracy = function(df) {
  as.numeric(df %>% filter(class == scored.class) %>% summarise(n = n()) /
    nrow(df))
}

# error rate = % of false positives and false negatives
pred_error_rate = function(df) {
  as.numeric(df %>% filter(class != scored.class) %>% summarise(n = n()) /
    nrow(df))
}

# precision = % of all positive predictions that were actually positives
pred_precision = function(df) {
  as.numeric(df %>% filter(class == 1 & scored.class == 1) %>% summarise(n = n()) /
```

```r
    df %>% filter(scored.class == 1) %>% summarise(n = n()))
}

# sensitivity/recall = % of all actual positives that were predicted positives
pred_sensitivity = function(df) {
  as.numeric(df %>% filter(class == 1 & scored.class == 1) %>% summarise(n = n()) /
    df %>% filter(class == 1) %>% summarise(n = n()))
}

# specificity = % of all actual negatives that were predicted as negatives
pred_specificity = function(df) {
  as.numeric(df %>% filter(class == 0 & scored.class == 0) %>% summarise(n = n()) /
    df %>% filter(class == 0) %>% summarise(n = n()))
}

# F1 score
pred_F1 = function(df) {
  (2 * pred_precision(df) * pred_sensitivity(df)) /
    (pred_precision(df) + pred_sensitivity(df))
}
```

```r
# 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.

# Since precision and sensitivity are both between 0 and 1, their product is less than their sum.
# As both approach zero, the numerator approaches zero faster than the denominator; thus the ratio
# approaches zero. As both approach 1, the ratio approaches 1/2. The multiplier 2 then scales the
# ratio to range between 0 and 1.
```

```r
# 10.
plot_ROC = function(df, scale = 100) {
  library(ggplot2)
  true_positive_rate = rep(0, scale+1)
  false_positive_rate = rep(0, scale+1)
  for (i in seq(0, 1, 1 / scale)) {
    df$scored.class = ifelse(df$scored.probability > i, 1, 0)
    true_positive_rate[i*scale+1] = pred_sensitivity(df)
    false_positive_rate[i*scale+1] = 1 - pred_specificity(df)
  }
  # plot(x = false_positive_rate, y = true_positive_rate)
  data.frame(false_positive_rate = false_positive_rate, true_positive_rate = true_positive_rate) %>%
    ggplot(aes(x = false_positive_rate, y = true_positive_rate)) + geom_point() +
    ggtitle('ROC Curve')
}

plot_ROC(df)
```




```
# 11.
# accuracy = % of true positives and true negatives
pred_accuracy(df)
```

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

```
# error rate = % of false positives and false negatives
pred_error_rate(df)
```

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

```
# precision = % of all positive predictions that were actually positives
pred_precision(df)
```

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

```
# sensitivity/recall = % of all actual positives that were predicted positives
pred_sensitivity(df)
```

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

```
# specificity = % of all actual negatives that were predicted as negatives
pred_specificity(df)
```

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

```
# F1 score
pred_F1(df)
```

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

```r
# 12. compare to caret function outputs
library(caret)
```

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

```r
# IMPORTANT to add positive = '1' to confusionMatrix since according to ?confusionMatrix
# "If there are only two factor levels, the first level will be used as the 'positive' result."
confusionMatrix(df$scored.class, df$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
##
```

```r
# The caret results tie with the custom functions built above.
```

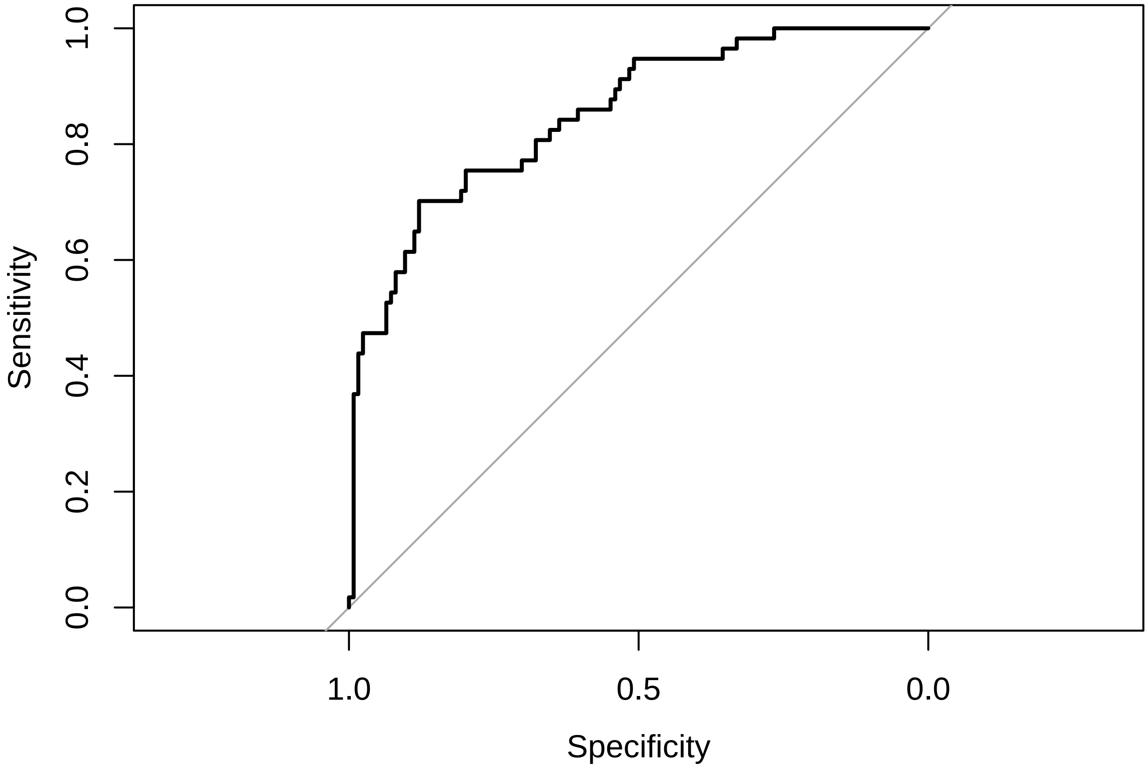
```r
# 13. All of the produced ROC curves are essentially the same
library(pROC)
```

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

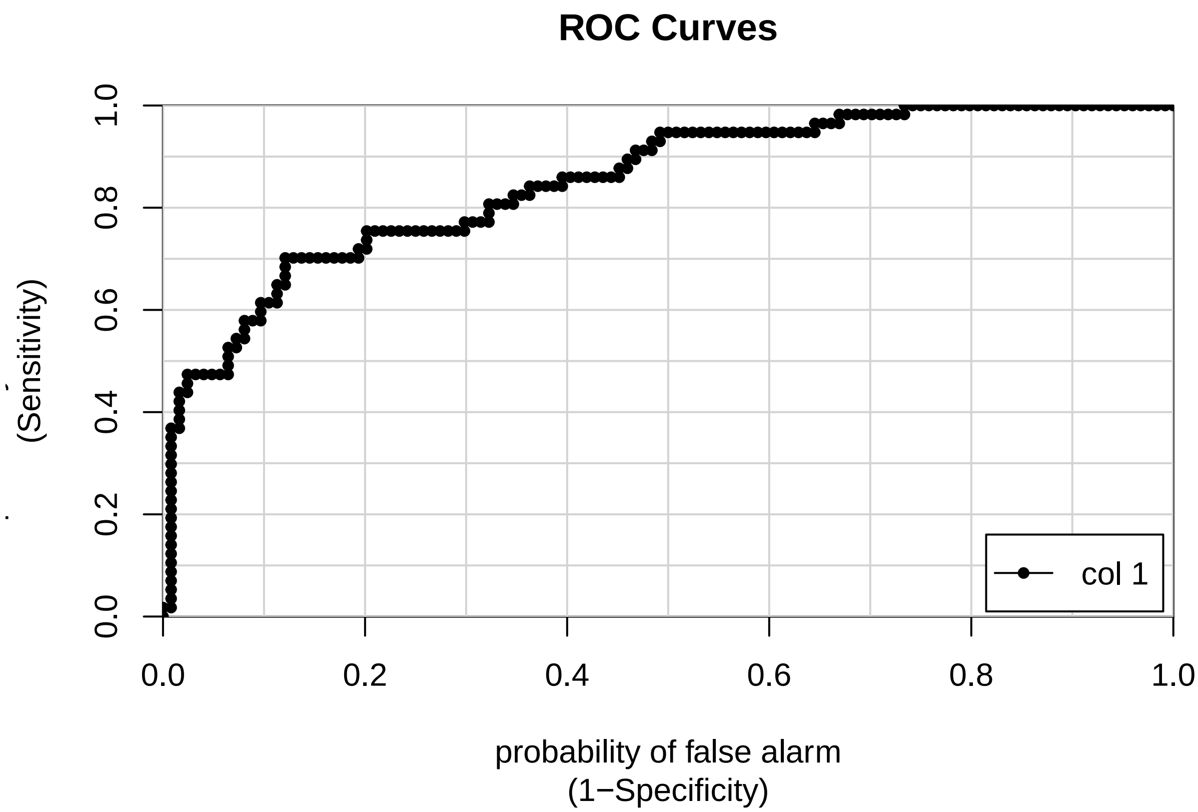
```
##
## Attaching package: 'pROC'
```

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

```r
plot.roc(df$class, df$scored.probability)
```




```r
# I prefer the caTools colAUC output to the pROC curve
library(caTools)
colAUC(df$scored.probability, df$class, plotROC = TRUE)
```




```
##              [,1]
## 0 vs. 1 0.8503113
```