Project 2

Michael Muller

March 12, 2018

(1) Load Data

```
wData = read.csv('cod.csv')
theData = wData[c('class',
                  'scored.class',
                  'scored.probability')]
```

(2) Display data using table()

```
myT = table(theData$scored.class,theData$class)
myT
```

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

119 TP, 5 FP, 27 TP, 30 FN

(3-8) Utilize the following functions

(3) Write a function that computes accuracy
(4) Write a function that computes error rate
(5) Write a function that computes precision
(6) Write a function that computes sensitivity
(7) Write a function that computes specificity
(8) Write a function that computes F1 score

```
classificationMetrics = function(myT){
  TP = myT[1,1]
  FN = myT[1,2]
  FP = myT[2,1]
  TN = myT[2,2]
  accuracy = (TP+TN)/(TP+TN+FP+FN)
  error = (FP+FN)/(TP+FP+TN+FN)
  precision = (TP)/(TP+FP)
  sensitivity = (TP)/(TP+FN)
  specificity = (TN)/(TN+FP)
  f1 = (2*precision*sensitivity)/(precision+sensitivity)
  return(data.frame(accuracy,error,precision,sensitivity,specificity,f1))
}
```

(9) What are the bounds on the F1 score?

F1 equals

$$F1\ Score = (2 \times Precision \times Sensitivity) / (Precision + Sensitivity)$$

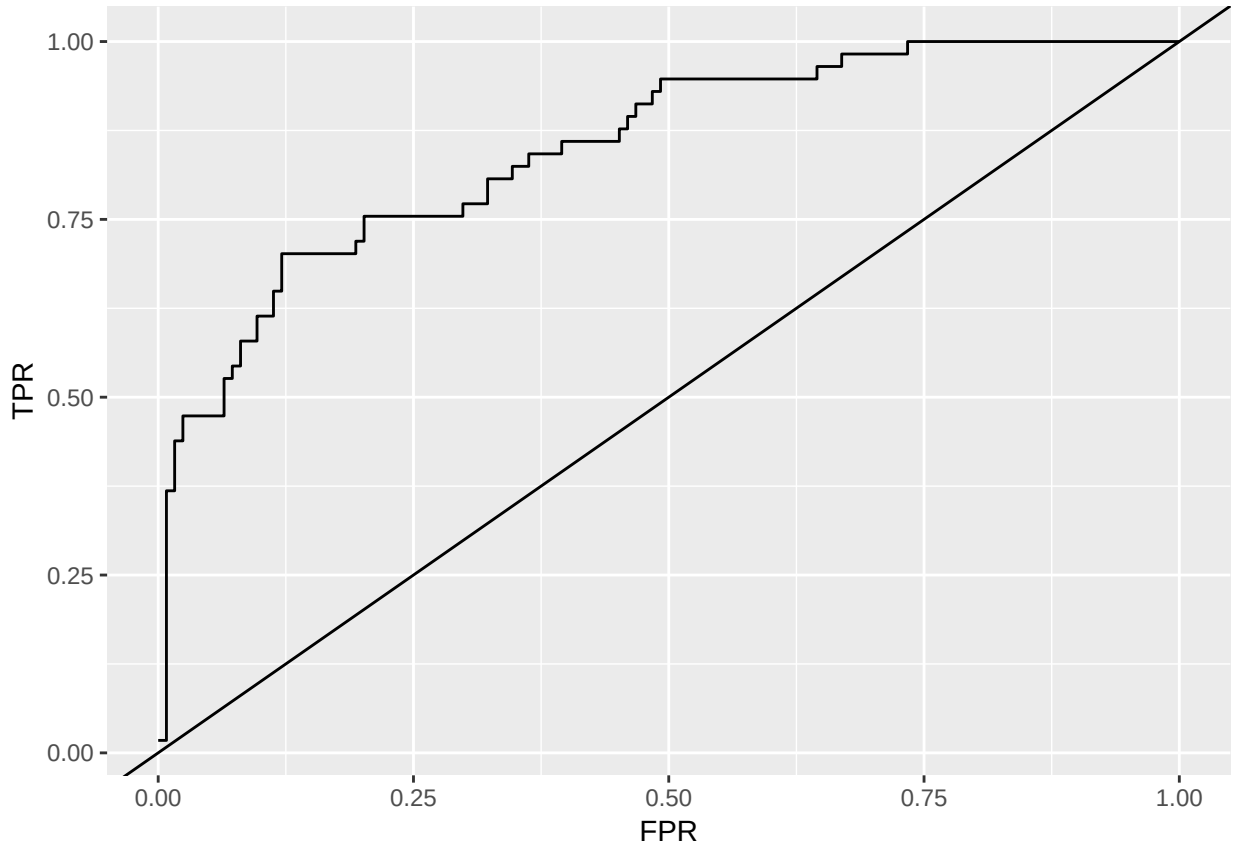
The metrics of Sensitivity and Specificity can never exceed a range of 0 to 1; and the algorithim is set up so that the product of the two is dividing by their summation. The F1 score range is 0 to 1.

(10) Calculate ROC and AUC using class and scored.probability

```
#http://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html
rocauc = function(classification, probability){
  #Sort observed outcomes by probability descending
  classification = classification[order(probability,decreasing=TRUE)]
  #Calculate Sens and Spec
  roc_frame = data.frame(TPR=cumsum(classification)/sum(classification),
                         FPR=cumsum(!classification)/sum(!classification),
                         classification)
  #Calculate AUC below

  #because the thresholds are discrete; we need to calculate the distance between TPR/FPR...We will use
  diffTPR = c(diff(roc_frame$TPR),0)
  diffFPR = c(diff(roc_frame$FPR),0)
  #Now that we have the perimeter measurements of each rectangle under the curve TPR/FPR
  #and our best guess on the area delimited by the actual curve
  #We can compute the area under the curve with a summation of WxL's
  auc = sum(roc_frame$TPR*diffFPR)+sum(diffTPR*diffFPR)/2
  return(list(roc_frame,auc))
}
rocauc = rocauc(theData$class,theData$scored.probability)
```

```
library(ggplot2)
#plot(rocauc[[1]]$TPR~rocauc[[1]]$FPR,xlab=('Specificity'),ylab=('Sensitivity'))
#abline(a = 0, b = 1)
ggplot(data=rocauc[[1]],aes(FPR,TPR)) +geom_line() + geom_abline()
```




```
paste('The AUC is ',rocauc[[2]])
```

```
## [1] "The AUC is  0.850311262026033"
```

(11) Use the function created in 3-8

```
classificationMetrics(myT)
```

accuracy	error	precision	sensitivity	specificity	f1
0.8066298	0.1933702	0.9596774	0.7986577	0.84375	0.8717949

(12) Compare my metrics with the caret package (confusionMatrix(), sensitivity/specificity)

```
library(caret)
```

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

```
#Default CM call outputs their sensitivity and specificity
confusionMatrix(theData$class,theData$scored.class)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   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
##
```

Every metric of confusionMatrix mimics mine. This is due to the fact that these results are not up for variation or debate. It is simply a computation of predefined numbers. The best part of this package (for my sake) is the identification of a positive class automatically.

(13) Investigate the pROC package. Generate an ROC and compare to mine.

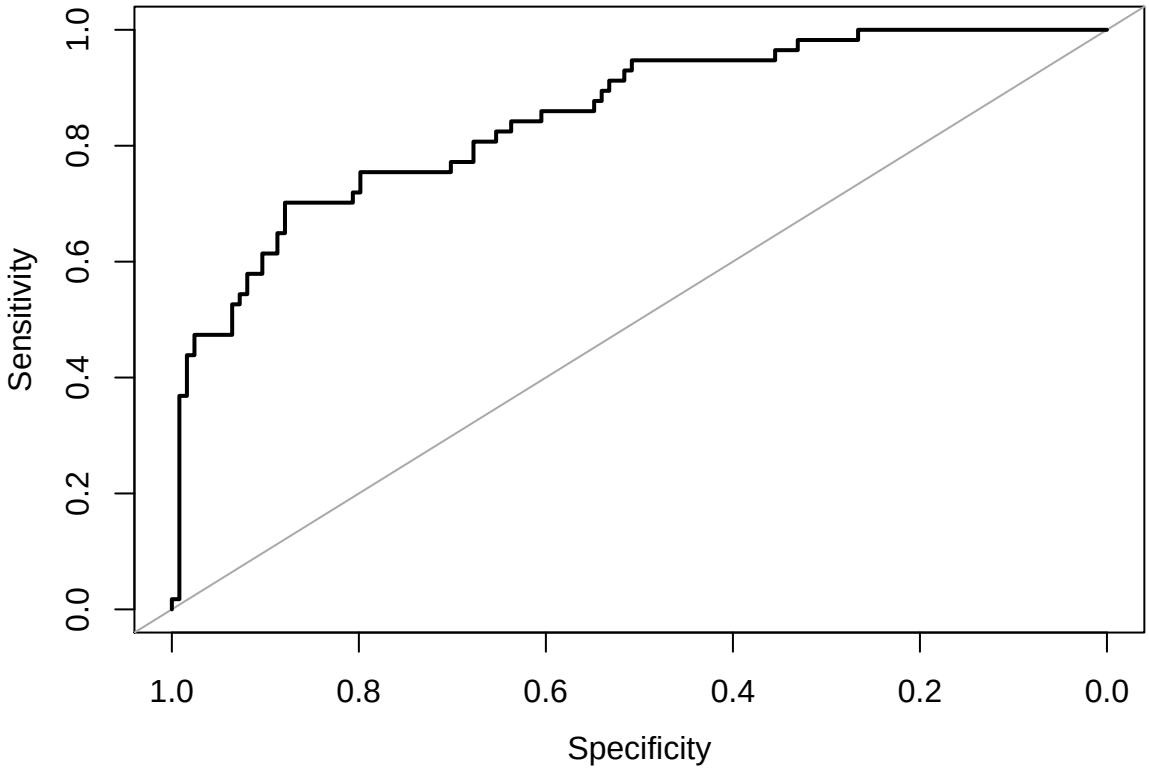
```
library(pROC)
```

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

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

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

```
theRock = roc(class~scored.probability,data=theData)
plot(theRock,asp=NA)
```




```
theRock$auc
```

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

Great, pROC package mimics my ROC (Thanks to a brilliant website) #### http://blog.revolutionanalytics.com/2016/11/calculating-auc.html