DATA 621 Assignment 2

Kai Lukowiak

2018-03-14

Import Data

```
library(tidyverse)
library(knitr)
df <- read_csv('~/DATA621/Assignments/Assignment2/classification-output-data.csv')
sample_n(df, size = 5) %>% kable()
```

pregnant	glucose	diastolic	skinfold	insulin	bmi	pedigree	age	class	scored.class	scored.probability
3	84	72	32	0	37.2	0.267	28	0	0	0.1086797
3	158	70	30	328	35.5	0.344	35	1	1	0.5919838
5	144	82	26	285	32.0	0.452	58	1	1	0.6764516
1	90	68	8	0	24.5	1.138	36	0	0	0.1070070
1	128	88	39	110	36.5	1.057	37	1	0	0.4590950

Confusion Matrix

R's table function can be used to create a confusion matrix. For an more indepth explenation of this please see this excelent website.

```
x <- table(df$class, df$scored.class)
colnames(x) <- c('Actual Negative ', 'Actual Positive')
rownames(x) <- c("Predicted Negative", 'Predicted Positive')
x %>% kable()
```

	Actual Negative	Actual Positive
Predicted Negative	119	5
Predicted Positive	30	27

The sum of the rows and columns can give insight into model performance. The rows represent the predicted values while the columns represent the actual values.

Accuracy

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

```
confusionFunction <- function(df, actual, predicted, metric){
  x <- table(df[[actual]], df[[predicted]])
  TN <- x[2, 2]; FN <- x[1, 2]; FP <- x[2, 1]; TP <- x[2, 2]
  # Values.
```

```r
  if (metric == 'Accuracy'){
    Accuracy <- (TP + TN) / (TN + FN + FP + TP)
    return(Accuracy)
    }

  else if (metric == 'ClassificationErrorRate'){
    ClassificationErrorRate <- (FP + FN) / (TN + FN + FP + TP)
    return(ClassificationErrorRate)
  }

  else if (metric == 'Precicion'){
     Precicion <- TP / (TP + FP)
     return( Precicion)
  }

  else if (metric == "Sensitivity"){
    Sensitivity <- TP / (TP + FN)
    return(Sensitivity)
  }

  else {
    Specificity <- TN / (TN + FP)
    return(Specificity)
  }
}

confusionFunction(df, 9, 10, "Accuracy")
```

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

Classification Error Rate

```r
confusionFunction(df, 9, 10, 'ClassificationErrorRate')
```

```
## [1] 0.3932584
```

To verify that these sum to one:

```r
confusionFunction(df, 9, 10, 'ClassificationErrorRate') +
  confusionFunction(df, 9, 10, 'Accuracy')
```

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

This test is passed.

Sensitivity

```r
confusionFunction(df, 9, 10, 'Sensitivity')
```

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

Precision

```
confusionFunction(df, 9, 10, 'Precicion')
```

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

Specificity

```
confusionFunction(df, 9, 10, 'Specificity')
```

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

```
Prec <- confusionFunction(df, 9, 10, 'Precicion')
Prec
```

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

```
ACC=   confusionFunction(df, 9, 10, 'Accuracy')
ACC
```

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

F1 Score

Write a function to calculate the F1 score.

```
f1 <- function(df, actual, predicted){
  f1Tab <- table(df[[actual]], df[[predicted]])
  sens <- confusionFunction(df, actual, predicted, 'Sensitivity')
  prec <- confusionFunction(df, actual, predicted, 'Precision')
  f1Score <- 2 * sens * prec / (prec + sens)
  return(f1Score)
}

f1(df, 9, 10)
```

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

Bounds of F1

The `F1` score is bouned between zero and 1.

$$F1_{Score} = \frac{2 * Precicion * Sensitivity}{Precicion + Sensitivity}$$

For values of a, b $0 < a < 1$ and $0 < b < 1$ $ab < a$ and $ab < b$. Therefore the numerator is strictly less than the demominator of the above fraction.

ROC Curve Function

```r
rocFunc <- function(values, predictions){
  # Returns a df of FPR and TPR and a tufte style graph of the AUC.
  # Special thanks too: http://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.htm
  values <- values[order(predictions, decreasing=TRUE)]
  df <- data.frame(TPR=cumsum(values)/sum(values),
             FPR=cumsum(!values)/sum(!values))
  p <- ggplot(df, aes(FPR, TPR)) +
    geom_line() +
    ggtitle('AUC Curve') +
    geom_abline(slope = 1) +
    ggthemes::theme_tufte()
  auc <- df %>%
    mutate(AUC = FPR * lead(FPR) * TPR) %>%
    select(AUC)
  return(list(auc, p))
}

Temp <- rocFunc(df$class, df$scored.probability)
x <- Temp[1]
data.frame(AUC = matrix(unlist(x))) %>% head() %>% kable()
```

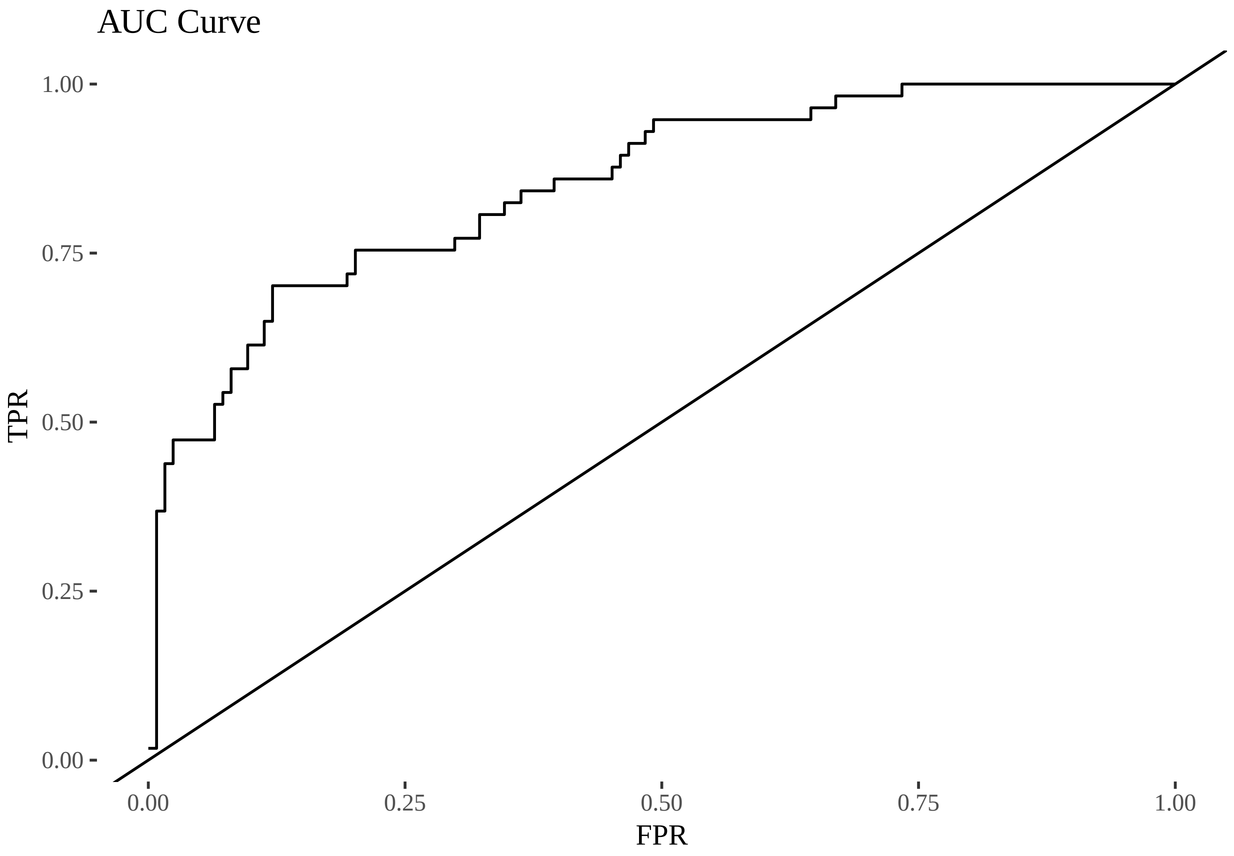
AUC
0.0e+00
1.1e-06
2.3e-06
3.4e-06
4.6e-06
5.7e-06

```r
x  %>% unlist() %>% matrix() %>% data.frame() %>% head()
```

```
##              .
## 1 0.000000e+00
## 2 1.140990e-06
## 3 2.281980e-06
## 4 3.422969e-06
## 5 4.563959e-06
## 6 5.704949e-06
```

```r
Temp[2]
```

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



Investigate the caret package

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018c.
## 1.0/zoneinfo/America/Edmonton'
```

```
confusionMatrix(df$class, df$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
##
```

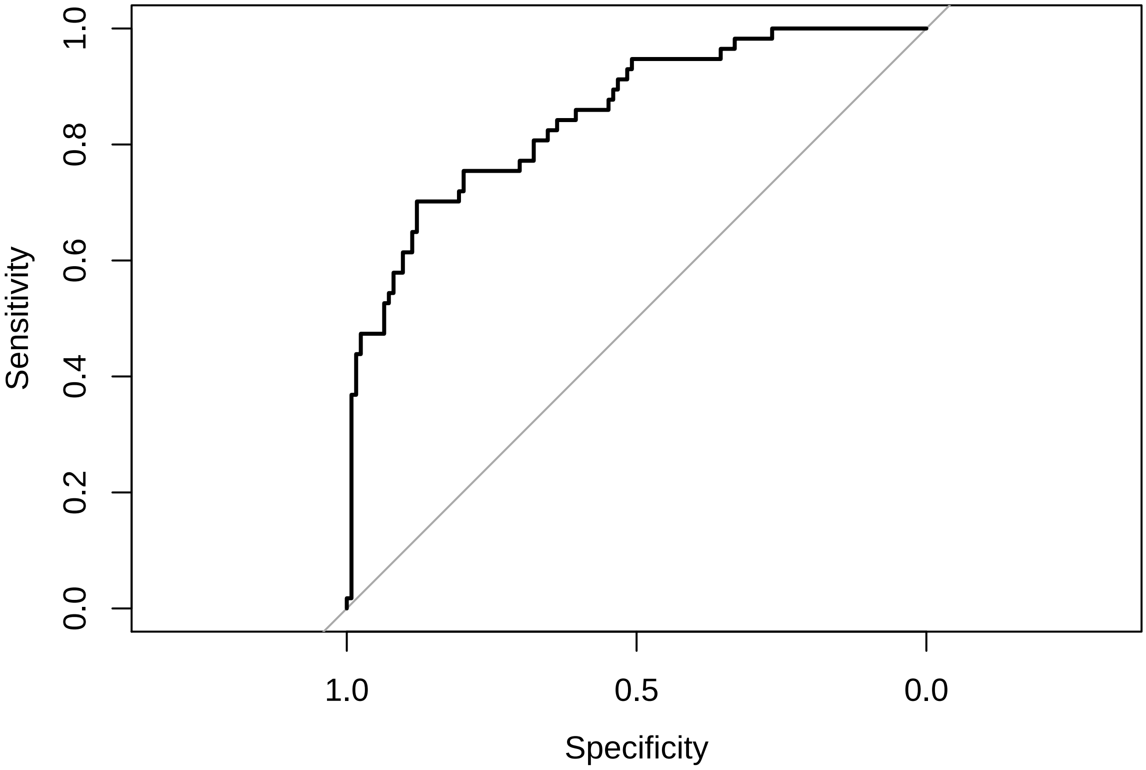
We can see that this is a much more concise way to get many of the values that our function got. Given that it is probably written in C++ it will also be faster.

Investigate the pROC package

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.4.4
```

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



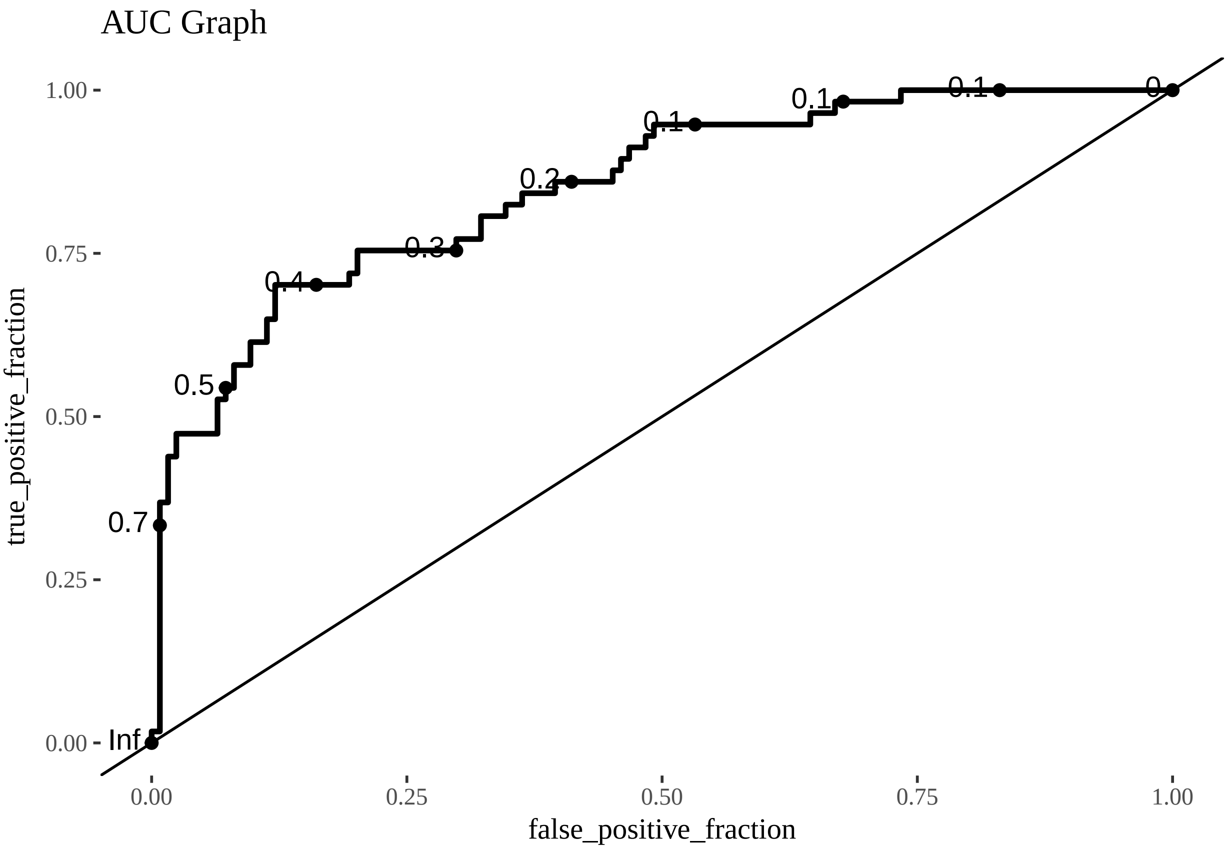
```
##
## Call:
## roc.default(response = df$class, predictor = df$scored.probability,     plot = TRUE)
##
## Data: df$scored.probability in 124 controls (df$class 0) < 57 cases (df$class 1).
## Area under the curve: 0.8503
```

This is also a much more concise way to perform the analysis, however, in my huble opinion, my graph looks better.

There is also the `plotROC` package which performs well:

```r
#devtools::install_github("sachsmc/plotROC")
library(plotROC)


ggplot(df, aes(d = class, m = scored.probability)) +
  geom_roc() +
  ggtitle('AUC Graph')+
  geom_abline()+
  ggthemes::theme_tufte()
```



This produces an even better graph.