# DATA621: Business Analytics and Data Mining

Assignment 2: Applied Predictive Modeling on Classification Output Dataset

Irene Jacob

## 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.

### 1. Download the classification output data set

Q: Download the classification output data set.

I downloaded the dataset and read it from my github using read.csv(). Below is the first 5 rows of the dataset.

```
class_df <- read.csv("https://raw.githubusercontent.com/irene908/DATA621/main/classification-output-data.csv")
head(class_df)
```

| | pregnant <int> | glucose <int> | diastolic <int> | skinfold <int> | insulin <int> | bmi <dbl> | pedigree <dbl> | age <int> | class <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 124 | 70 | 33 | 215 | 25.5 | 0.161 | 37 | 0 |
| 2 | 2 | 122 | 76 | 27 | 200 | 35.9 | 0.483 | 26 | 0 |
| 3 | 3 | 107 | 62 | 13 | 48 | 22.9 | 0.678 | 23 | 1 |
| 4 | 1 | 91 | 64 | 24 | 0 | 29.2 | 0.192 | 21 | 0 |
| 5 | 4 | 83 | 86 | 19 | 0 | 29.3 | 0.317 | 34 | 0 |
| 6 | 1 | 100 | 74 | 12 | 46 | 19.5 | 0.149 | 28 | 0 |

6 rows | 1-10 of 12 columns

## 2. Confusion Matrix

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

Q: 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?

Below is the confusion matrix:

```
class_scored <- class_df[,c('class', 'scored.class','scored.probability')]

table(class_scored$class,class_scored$scored.class)
```

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

Confusion matrix gives the number of entries in each class along with the prediction.

From the above table it is understood that there was a total of 57 cases of class '1' and 27 of them were correctly predicted as '1' whereas the rest 30 were predicted to be '0' which is wrong. The class '0' category has a total of 124 cases and 5 of them were predicted as '1' which is wrong whereas the rest 119 were correctly predicted.

## 3. Accuracy

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

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy measures the closeness of the predicted value to the actual value.

Below is the accuracy of the classification dataset.

tn: true negative

tp: true positive

```
accuracyfn <- function(df){
  s <- nrow(df)
  tn <- sum(df$class == 0 & df$scored.class ==0)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return((tn+tp)/s)
}

print(accuracyfn(class_scored))
```

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

It can be seen that the accuracy for this dataset is around 80%.

## 4. Classification error rate

Q: 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.

$$Classification\ Error\ Rate = \frac{FP + FN}{TP + FP + TN + FN}$$

Verify that you get an accuracy and an error rate that sums to one.

Classification error rate is the ratio of the total number of erroneous values to the total number of entries in the dataset or it is *1-Accuracy*.

Below is the error rate of the classification dataset.

fn: false negative

fp: false positive

```
errorfn <- function(df){
  s <- nrow(df)
  fn <- sum(df$class == 1 & df$scored.class ==0)
  fp <- sum(df$class == 0 & df$scored.class ==1)
  return((fn+fp)/s)
}

print(errorfn(class_scored))
```

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

Below is the sum of accuracy and error rate. It can be seen that the sum is 1.

```
accuracyfn(class_scored)+errorfn(class_scored)
```

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

## 5. Precision

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

$$Precision = \frac{TP}{TP + FP}$$

Precision is the measure of how close each value is to each other.

Below is the precision of this dataset.

fp: false positive

tp: true positive

```
precisionfn <- function(df){
  fp <- sum(df$class == 0 & df$scored.class ==1)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return(tp/(tp+fp))
}

print(precisionfn(class_scored))
```

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

The precision is around 84%.

## 6. Sensitivity

Q: 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.

$$Sensitivity = \frac{TP}{TP + FN}$$

Sensitivity is also known as the true positive rate which means the proportion of the dataset that was correctly predicted as positive.

Below is the sensitivity of this dataset.

fn: false negative

tp: true positive

```
sensitivityfn <- function(df){
  fn <- sum(df$class == 1 & df$scored.class ==0)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return(tp/(tp+fn))
}

print(sensitivityfn(class_scored))
```

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

The sensitivity is around 47%.

## 7. Specificity

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

$$Specificity = \frac{TN}{TN + FP}$$

Specificity is also known as the true negative rate which means the proportion of the dataset that was correctly predicted as negative.

Below is the specificity of this dataset.

tn: true negative

fp: false positive

```
specificityfn <- function(df){
  tn <- sum(df$class == 0 & df$scored.class ==0)
  fp <- sum(df$class == 0 & df$scored.class ==1)
  return(tn/(tn+fp))
}

print(specificityfn(class_scored))
```

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

The specificity is around 96%.

## 8. F1 Score

Q: 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.

$$F1\ Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}$$

F1 score is the harmonic mean of sensitivity and precision. It is a measure of how accurate the test is.

Below is the F1 score of this dataset.

```
f1scorefn <- function(df){
  precision <- precisionfn(df)
  sensitivity <- sensitivityfn(df)
  return((2*precision*sensitivity)/(precision+sensitivity))
}

print(f1scorefn(class_scored))
```

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

F1 score is around 61%.

## 9. Prove F1 score will always be between 0 and 1

Q: 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. (Hint: If 0 < $a$ < 1 and 0 < $b$ < 1 then $ab$ < $a$.)

F1 score is:

$$F1 = \frac{2 * precision * sensitivity}{precision + sensitivity}$$

precision bound is $[0 < precision < 1]$

sensitivity bound is $[0 < sensitivity < 1]$

So, $precision * sensitivity$ will also have the bound between 0 and 1.

$$\frac{2*Precision*Sensitivity}{Precision+Sensitivity} = \frac{Precision*Sensitivity}{Precision+Sensitivity} + \frac{Precision*Sensitivity}{Precision+Sensitivity} < \frac{Precision}{Precision+Sensitivity} + \frac{Sensitivity}{Precision+Sensitivity}$$
$$= \frac{Precision+Sensitivity}{Precision+Sensitivity}$$

$ = 1$

$$0 < F1 < 1$$

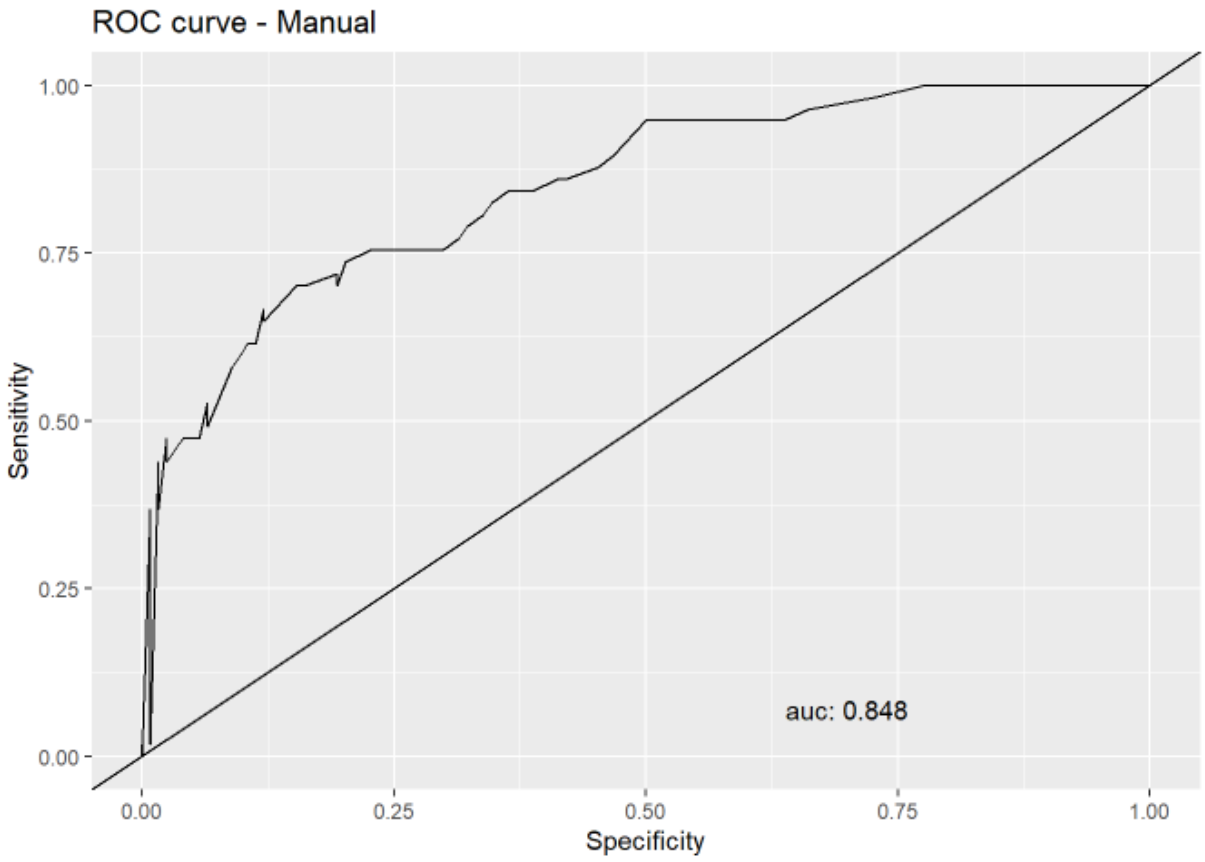
Therefore, the sum will be greater than product which proves that F1 will always be between 0 and 1.

## 10. ROC Curve and AUC

Q: 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.

A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve for our dataset is as follows:




## 11. All Classification Metrics

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

Below is the table containing all the classification metrics discussed above:

```
class_metrics <- c(accuracyfn(class_scored), errorfn(class_scored), f1scorefn(class_scored), precisi
onfn(class_scored), sensitivityfn(class_scored), specificityfn(class_scored))
names(class_metrics) <- c("Accuracy", "Error Rate", "F1 Score", "Percision", "Sensitivity", "Specifi
city")

print(class_metrics,col.names = "Metric Values")
```

```
##    Accuracy  Error Rate    F1 Score   Percision Sensitivity Specificity
##   0.8066298   0.1933702   0.6067416   0.8437500   0.4736842   0.9596774
```

## 12. 'caret' package

Q: 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?

The caret package (short for Classification And REgression Training) contains functions to streamline the model training process for complex regression and classification problems. The package utilizes a number of R packages but tries not to load them all at package start-up. The package “suggests” field includes 32 packages. caret loads packages as needed and assumes that they are installed. If a modeling package is missing, there is a prompt to install it.

I have used sensitivity, specificity and confusionmatrix functions of caret package below:

```r
library("caret")
```

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

```r
sensitivity(as.factor(class_scored$scored.class), as.factor(class_scored$class), positive='1')
```

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

```r
specificity(as.factor(class_scored$scored.class), as.factor(class_scored$class), negative='0')
```

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

```r
confusionMatrix(as.factor(class_scored$scored.class), as.factor(class_scored$class), positive='1')
```

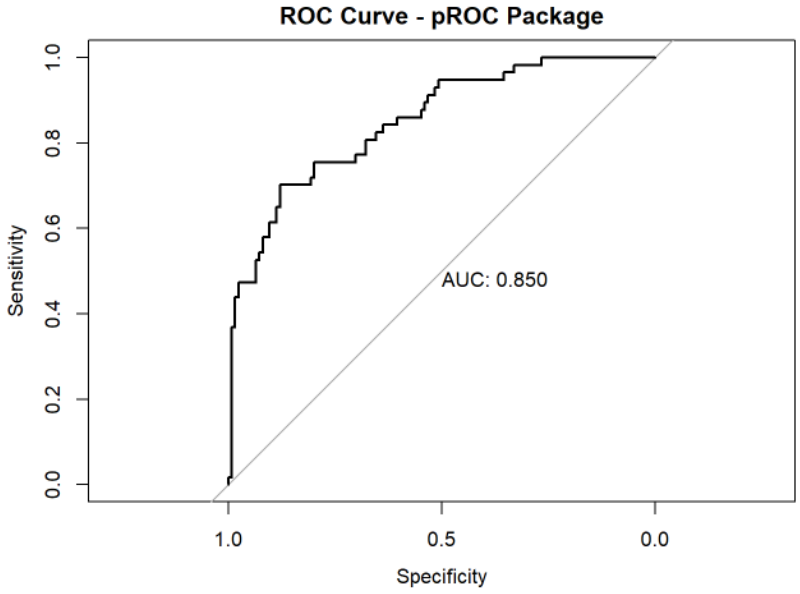
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
##
##                   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
##
```

## 13. 'pROC' package

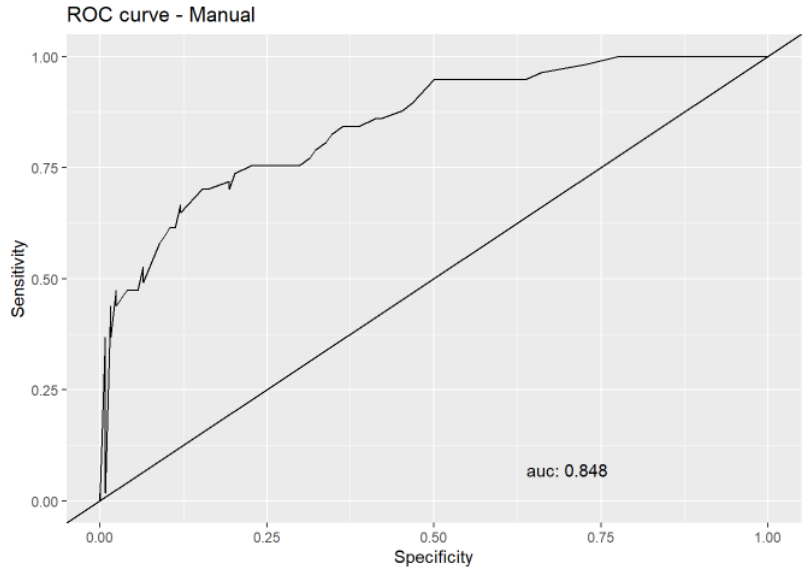
Q: Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

pROC is a package to visualize, smooth and compare receiver operating characteristic (ROC curves).

Below is the visualization of ROC using pROC package.




Below is the ROC visualization using the function I created.



The ROC visualization using pROC package and my function look similar. There is a minor difference in AUC which can be ignored. This difference could be due to some difference in the calculations. The other difference is that the pROC plot x axis is from 1 to 0 whereas my function plotted it from 0 to 1.

## 14. Annexure

R code:

```
---
title: "DATA621 Assignment 2"
author: "Irene Jacob"
date: "10/10/2021"
output:
  html_document:
    df_print: paged
    toc: yes
    toc_collapsed: yes
    toc_float: yes
  rmdformats::readthedown:
    code_folding: hide
    df_print: paged
    highlight: tango
    number_sections: no
    smooth_scroll: yes
    theme: united
    toc_collapsed: yes
    toc_depth: 5
    toc_float: yes
theme: readthedown
number_sections: yes
toc_depth: 3
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(ggplot2)
library(DT)
```
```

# 1. Download the classification output data set

```{r}
class_df <-
read.csv("https://raw.githubusercontent.com/irene908/DATA621/main/classification-output-data.csv")
head(class_df)
```

# 2. Confusion matrix

```{r}

class_scored <- class_df[,c('class', 'scored.class','scored.probability')]

table(class_scored$class,class_scored$scored.class)

```

# 3. Accuracy

```{r}

accuracyfn <- function(df){
  s <- nrow(df)
  tn <- sum(df$class == 0 & df$scored.class ==0)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return((tn+tp)/s)
}

print(accuracyfn(class_scored))

```

# 4. Classification Error Rate

```{r}

errorfn <- function(df){
  s <- nrow(df)
  fn <- sum(df$class == 1 & df$scored.class ==0)
  fp <- sum(df$class == 0 & df$scored.class ==1)
  return((fn+fp)/s)
}
```

```
print(errorfn(class_scored))

```


```{r}

accuracyfn(class_scored)+errorfn(class_scored)

```

# 5. Precision

```{r}

precisionfn <- function(df){
  fp <- sum(df$class == 0 & df$scored.class ==1)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return(tp/(tp+fp))
}

print(precisionfn(class_scored))

```

# 6. Sensitivity

```{r}

sensitivityfn <- function(df){
  fn <- sum(df$class == 1 & df$scored.class ==0)
  tp <- sum(df$class == 1 & df$scored.class ==1)
  return(tp/(tp+fn))
}

print(sensitivityfn(class_scored))

```

# 7. Specificity

```{r}
```

```
specificityfn <- function(df){
  tn <- sum(df$class == 0 & df$scored.class ==0)
  fp <- sum(df$class == 0 & df$scored.class ==1)
  return(tn/(tn+fp))
}

print(specificityfn(class_scored))
```

# 8. F1 Score

```{r}

f1scorefn <- function(df){
  precision <- precisionfn(df)
  sensitivity <- sensitivityfn(df)
  return((2*precision*sensitivity)/(precision+sensitivity))
}

print(f1scorefn(class_scored))

```

# 9. Show that the F1 score will always be between 0 and 1

F1 score is:

$F1~=~\frac{2~*~precision~*~sensitivity}{precision~+~sensitivity}$

precision bound is ${[0~<~precision~<~1]}$

sensitivity bound is ${[0~<~sensitivity~<~1~]}$

So, $precision*sensitivity$ will also have the bound between 0 and 1.

$\frac{2 * Precision * Sensitivity}{Precision+Sensitivity} = \frac{Precision * Sensitivity}{Precision+Sensitivity}+\frac{Precision * Sensitivity}{Precision+Sensitivity}< \frac{Precision}{Precision+Sensitivity}+\frac{Sensitivity}{Precision+Sensitivity}$\
$= \frac{Precision+Sensitivity}{Precision+Sensitivity}$\
$ = 1$

 $0<F1<1$

Therefore, the sum will be greater than product which proves that F1 will always be between 0 and 1.

# 10. ROC Curve and AUC

```{r}

rocfn <- function(df){

  for (t in seq(0,1,0.01))
    {
    #create dataset for each threshold
    x <- data.frame(class = df[,1], scored.class = if_else(df[,3] >= t,1,0), scored.probability = df[,3])

    #create vectors to store sens & speci for all datasets
    if(!exists('sens') & !exists('speci'))
      {
      sens <- sensitivityfn(x)
      speci <- 1- specificityfn(x)
    }
    else
      {
      sens <- c(sens,sensitivityfn(x))
      speci <- c(speci, 1- specificityfn(x))
    }
  }

  df_roc <- data.frame(sens, speci) %>% arrange(speci)


  # AUC calculation
  speci_df <- c(diff(df_roc$speci), 0)
  sens_df <- c(diff(df_roc$sens), 0)
  auc <- round(sum(df_roc$sens * speci_df) + sum(sens_df * speci_df)/2, 3)


  #Create plot
  ggplot(df_roc) + geom_line(aes(speci, sens)) + ggtitle("ROC curve - Manual") +
xlab("Specificity") + ylab("Sensitivity") +  annotate(geom = "text", x = 0.7, y = 0.07,label =
paste("auc:", auc)) +  geom_abline(intercept = 0, slope = 1)
  }

rocfn(class_scored)
```

```

# 11. All Classification Metrics

```{r}

class_metrics <- c(accuracyfn(class_scored), errorfn(class_scored), f1scorefn(class_scored), precisionfn(class_scored), sensitivityfn(class_scored), specificityfn(class_scored))
names(class_metrics) <- c("Accuracy", "Error Rate", "F1 Score", "Percision", "Sensitivity", "Specificity")

print(class_metrics,col.names = "Metric Values")

```

# 12. *caret* package

```{r, warning=FALSE}

library("caret")

sensitivity(as.factor(class_scored$scored.class), as.factor(class_scored$class), positive='1')

specificity(as.factor(class_scored$scored.class), as.factor(class_scored$class), negative='0')

confusionMatrix(as.factor(class_scored$scored.class), as.factor(class_scored$class), positive='1')
```

# 13. *pROC* package

```{r,warning=FALSE}

library("pROC")

plot(roc(class_scored$class, class_scored$scored.probability), print.auc = TRUE, main = 'ROC Curve - pROC Package')

rocfn(class_scored)

```
```