

# DATA621 HW2 combo

Misha Kollontai, Matthew Baker, Erinda Budo, Subhalaxmi Rout, Don Padmaperuma

9/28/2020

## Overview

In this homework assignment, we will work through various classification metrics, providing multiple approaches to some problems. We will create functions in R to carry out the various calculations. We will also investigate some functions in packages that will let us obtain the equivalent results. Finally, we will create graphical output that also can be used to evaluate the output of classification models, such as binary logistic regression.

Reference:

[http://www.saedsayad.com/model\\_evaluation\\_c.htm](http://www.saedsayad.com/model_evaluation_c.htm)

Chapter 11- Measuring Performance in Classification Models

## Libraries

```
library(dplyr)
library(ggplot2)
library(reshape2)
library(kableExtra)
library(DT)
```

## 1. Downloading the Data

```
df <- read.csv("https://raw.githubusercontent.com/mkollontai/DATA621_GroupWork/master/HW2/classification-output-data.csv")
df_hw2 <- df[,c('class', 'scored.class', 'scored.probability')]

class_df <- read.csv("https://raw.githubusercontent.com/gpadmaperuma/DATA621/main/classification-output-data%20(2).csv", header =
head(class_df)
```

```
##   pregnant glucose diastolic skinfold insulin  bmi pedigree age class
## 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
##   scored.class scored.probability
## 1           0         0.32845226
## 2           0         0.27319044
## 3           0         0.10966039
## 4           0         0.05599835
## 5           0         0.10049072
## 6           0         0.05515460
```

The df\_hw2 dataframe is limited to the 3 columns:

- class
- scored.class
- scored.probability

## 2. Get to know the data set

```
summary(class_df)
```

```
##   pregnant      glucose      diastolic      skinfold
##  Min.   : 0.000   Min.   : 57.0   Min.   : 38.0   Min.   : 0.0
## 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.0   1st Qu.: 0.0
##  Median : 3.000   Median :112.0   Median : 70.0   Median :22.0
##   Mean    : 3.862   Mean    :118.3   Mean    : 71.7   Mean    :19.8
```

```
## 3rd Qu.: 6.000 3rd Qu.:136.0 3rd Qu.: 78.0 3rd Qu.:32.0
## Max. :15.000 Max. :197.0 Max. :104.0 Max. :54.0
## insulin bmi pedigree age
## Min. : 0.00 Min. :19.40 Min. :0.0850 Min. :21.00
## 1st Qu.: 0.00 1st Qu.:26.30 1st Qu.:0.2570 1st Qu.:24.00
## Median : 0.00 Median :31.60 Median :0.3910 Median :30.00
## Mean : 63.77 Mean :31.58 Mean :0.4496 Mean :33.31
## 3rd Qu.:105.00 3rd Qu.:36.00 3rd Qu.:0.5800 3rd Qu.:41.00
## Max. :543.00 Max. :50.00 Max. :2.2880 Max. :67.00
## class scored.class scored.probability
## Min. :0.0000 Min. :0.0000 Min. :0.02323
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.11702
## Median :0.0000 Median :0.0000 Median :0.23999
## Mean :0.3149 Mean :0.1768 Mean :0.30373
## 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:0.43093
## Max. :1.0000 Max. :1.0000 Max. :0.94633
```

2.1 Confusion Matrix

Approach 1

```
table(df_hw2$class, df_hw2$scored.class)
```

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

This confusion matrix shows us the number of entries of each class against what they were predicted to be. In this table the row indicates the actual class, whereas the column signifies the predicted value (scored.class). We can therefore deduce that of the 57 total cases of class ‘1’, 27 were predicted to be class ‘1’ correctly. The other 30 were mistakenly predicted to be of class ‘0’. For the actual class ‘0’ cases, 5 were mistakenly predicted to be class ‘1’, whereas 119 were correctly identified.

In order to calculate all of the following evaluations of the predictions made within this dataframe we need to identify the number of:

- True Positives - Correctly identified class ‘1’ cases
- False Positives - Class ‘0’ cases incorrectly identified as class ‘1’
- True Negatives - Correctly identified class ‘0’ cases
- False Negatives - Class ‘1’ cases incorrectly identified as class ‘0’

Approach 2

```
conf_matrix <- with(class_df, table("Actual" = class, "Predicted" = scored.class))
addmargins(conf_matrix)
```

```
##      Predicted
## Actual    0    1 Sum
##    0    119    5 124
##    1     30   27  57
##    Sum   149   32 181
```

Confusion matrix is all about number of correct and incorrect predictions compared to actual outcomes in the dataset. Rows in this matrix represent the number of actual classes (0 and 1) of the data set and columns represent the number of predicted classes (0 and 1) of the dataset.

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.

Approach 1

```
outcomes <- function(df){
  outs <- aggregate(df, by = list(df[,1], df[,2]), FUN=length) %>%
    select(1:3) %>%
    `colnames<-` (c('class','prediction','count'))
  return(outs)
  #Within this df, in the 'count' column:
  #the first row is our True Negative (TN),
  #the second row is False Negatives (FN),
  #third row is False Positives (FP) and
  #final row is the True Positives (TP).
}
outcomes(df_hw2) %>%
  kable() %>%
  kable_paper('hover', full_width = F)
```

class	prediction	count
0	0	119
1	0	30
0	1	5
1	1	27

To calculate accuracy we must use the formula

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

```
acc <- function(df){
  outs <- outcomes(df)
  TN <- outs$count[1]
  FN <- outs$count[2]
  TP <- outs$count[4]
  FP <- outs$count[3]
  acc <- (TP+TN)/(TP+FP+TN+FN)
  return(acc)
}
acc(df_hw2)
```

## [1] 0.8066298

### Approach 2

```
# Accuracy
accuracy <- function(df)
{
  cmat <- table(df$class, df$scored.class)
  acc <- (cmat[2,2]+cmat[1,1])/sum(cmat)
  return(acc)
}
accuracy(class_df)
```

## [1] 0.8066298

## 4. Classification Error Rate Calculation

### Approach 1

To calculate classification error we must use the formula

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

```
cer <- function(df){
  outs <- outcomes(df)
  TN <- outs$count[1]
  FN <- outs$count[2]
  TP <- outs$count[4]
  FP <- outs$count[3]
  cer <- (FP+FN)/(TP+FP+TN+FN)
  return(cer)
}
cer(df_hw2)
```

## [1] 0.1933702

The calculations for these two measures are such that summing the two should yield 1:

```
acc(df_hw2) + cer(df_hw2)
```

## [1] 1

## Approach 2

```
classification_error <- function(df){  
  
  cmat <- table(df$class, df$scored.class)  
  error <- (cmat[1,2]+cmat[2,1])/sum(cmat)  
  return(error)  
}  
classification_error(class_df)
```

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

```
# Check to see if accuracy and classification error adds up to 1  
sum <- accuracy(class_df)+classification_error(class_df)  
sum
```

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

---

## 5. Precision Calculation

### Approach 1

To calculate precision we must use the formula

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

```
prec <- function(df){  
  outs <- outcomes(df)  
  TP <- outs$count[4]  
  FP <- outs$count[3]  
  prec <- (TP)/(TP+FP)  
  return(prec)  
}  
prec(df_hw2)
```

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

### Approach 2

```
precision <- function(df){  
  cmat <- table(df$class, df$scored.class)  
  prec <- (cmat[2,2])/(cmat[2,2]+cmat[1,2])  
  return(prec)  
}  
precision(class_df)
```

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

---

## 6. Sensitivity Calculation

To calculate sensitivity we must use the formula

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

```
sens <- function(df){  
  FN <- sum(df$class == 1 & df$scored.class ==0)  
  TP <- sum(df$class == 1 & df$scored.class ==1)  
  sens <- (TP)/(TP+FN)  
  return(sens)  
}  
sens(df_hw2)
```

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

---

## 7. Specificity Calculation

To calculate specificity we must use the formula

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

```
spec <- function(df){
  TN <- sum(df$class == 0 & df$scored.class ==0)
  FP <- sum(df$class == 0 & df$scored.class ==1)
  spec <- (TN)/(TN+FP)
  return(spec)
}
spec(df_hw2)
```

## [1] 0.9596774

### Specificity Calculation approach 2

```
specificity <- function(df){
  cmat <- table(df$class, df$scored.class)
  spec <- (cmat[1,1])/(cmat[1,1]+cmat[1,2])
  return(spec)
}
specificity(class_df)
```

## [1] 0.9596774

## 8. F1 Calculation

To calculate F1 we must use the formula

$$F1 = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

```
f1 <- function(df){
  prec <- prec(df)
  sens <- sens(df)
  f1 <- (2*prec*sens)/(prec+sens)
  return(f1)
}
f1(df_hw2)
```

## [1] 0.6067416

## 9. Bounds of F1

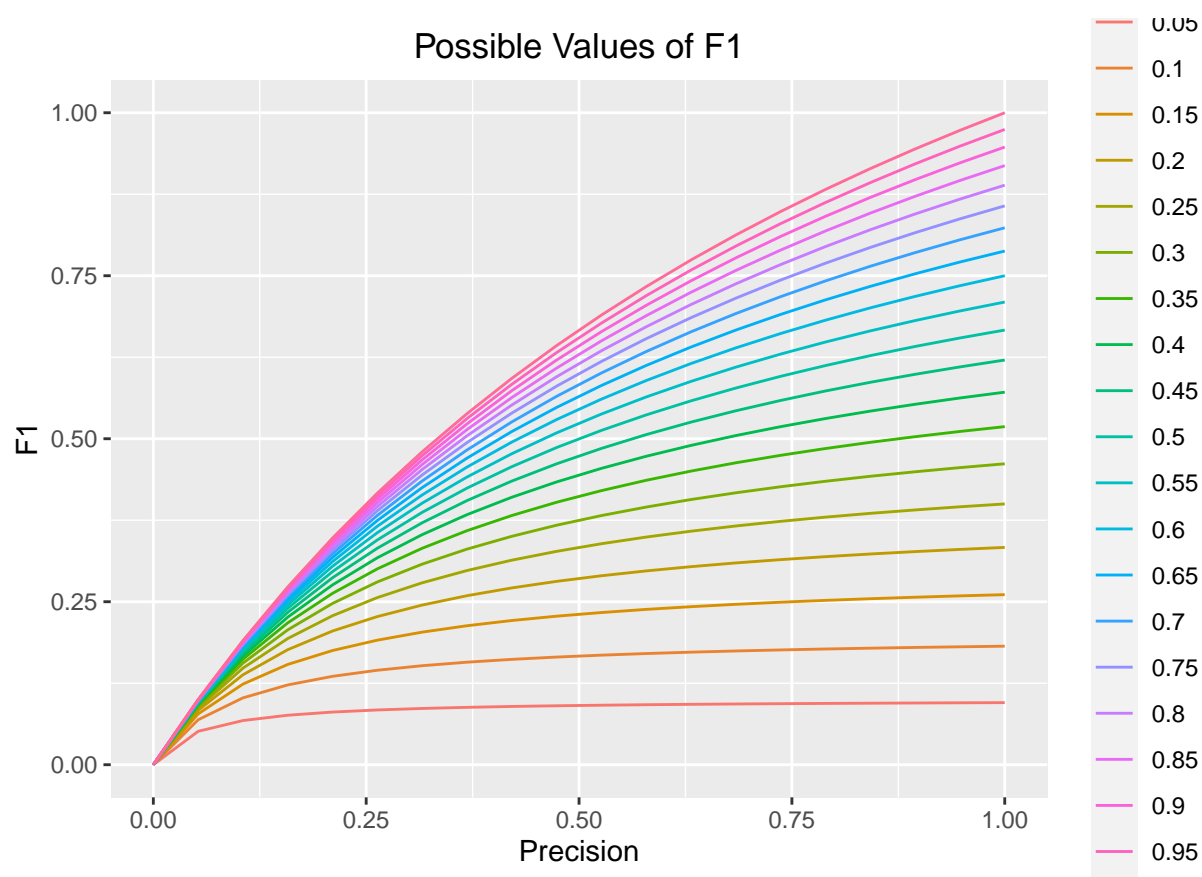
### Approach 1

To show the bounds of F1, let’s simplify the euation to a more manageable form:

$$F1 = \frac{2ps}{p + s}$$

The following code generates ‘l’ evenly-spaced values between 0 and 1 for the Precision. It then uses a loop to generate the calculated F1 values through the same range of Sensitivity values corresponding to each of the Precision values. The plot below shows calculated F1 on the y-axis, with each line corresponding to the calculation associated with a particular Sensitivity value.

```
#Partially used approach from (https://stackoverflow.com/questions/14704742/use-for-loop-to-plot-multiple-lines-in-single-plot-
l <- 20
p <- seq(0,1, length.out = l)
df <- NULL
for(i in 1:l){
  j <- i/l
  temp_df <- data.frame(x = p, y = (2*p*j)/(p+j), col = rep(j:j,each = l))
  df <- rbind(df, temp_df)
}
ggplot(df,aes(x=x,y=y,group=col,color=factor(col))) +
  geom_line() +
  ggtitle('Possible Values of F1') +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab('F1') +
  xlab('Precision') +
  ylim(0,1)
```



As we can clearly see, **the range of possible values for F1 is between 0 and 1**, as it was for both the Precision and the Sensitivity. We can clearly see that higher values of both Precision and Sensitivity result in higher F-1 values, with the highest value of 1 resulting from both Precision and Sensitivity being 1.

## Approach 2

F1 score is the harmonic mean of Precision and Sensitivity (Recall). This takes both false positives and false negatives into account. The highest possible value of F1 score is 1, and the lowest possible F1 score is 0.

```
# assume a is precision and b is sensitivity
```

```
a <- runif(100, min=0, max=1)
```

```
b <- runif(100, min=0, max=1)
```

```
f1d <- (2*a*b)/(a+b)
```

```
summary(f1d)
```

```
##      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
## 0.0002302 0.2338813 0.4037832 0.4190379 0.5980670 0.9556549
```

## 10. Function for ROC curve

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.

The ROC curve shows the trade-off between sensitivity and specificity. It is plotted with True Positive Rate and False Positive Rate at all possible pairings of those values (from 0 to 1)

True Positive Rate = Sensitivity

False positive Rate = 1 - Specificity

```
roc <- function(df){

  for (threshold in seq(0,1,0.01))
  {
    #create dataset for each threshold
    temp1 <- data.frame(class = df[,1], scored.class = if_else(df[,3] >= threshold,1,0),
                        scored.probability = df[,3])

    #create vectors to store TPR & FPR for all datasets
    if(!exists('TPR') & !exists('FPR'))
    {
      TPR <- sens(temp1)
      FPR <- 1- spec(temp1)
    }
    else
    {
      TPR <- c(TPR,sens(temp1))
      FPR <- c(FPR, 1- spec(temp1))
    }
  }
}
```

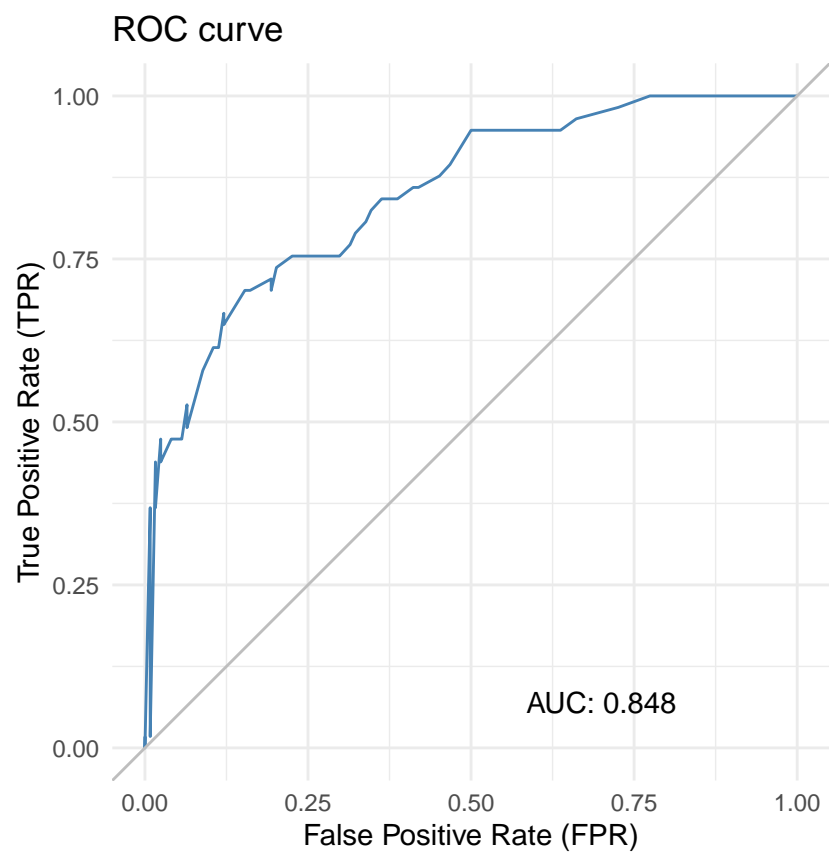
```
roc_df1 <- data.frame(TPR, FPR) %>% arrange(FPR)

# AUC calculation
diff_FPR <- c(diff(roc_df1$FPR), 0)
diff_TPR <- c(diff(roc_df1$TPR), 0)
AUC <- round(sum(roc_df1$TPR * diff_FPR) + sum(diff_TPR * diff_FPR)/2, 3)

#Create plot
plot <- ggplot(roc_df1) +
  geom_line(aes(FPR, TPR), color = "steel blue") +
  ggtitle("ROC curve") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("False Positive Rate (FPR)") +
  ylab("True Positive Rate (TPR)") +
  theme_minimal() +
  annotate(geom = "text", x = 0.7, y = 0.07, label = paste("AUC:", round(AUC, 3))) +
  geom_abline(intercept = 0, slope = 1, linetype = "solid", col = "gray") +
  coord_equal(ratio = 1)

q10_list <- list(plot,AUC)
return(q10_list)
}
q10 <- roc(df_hw2)
```

## [[1]]



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

```
df_detail <- c(acc(df_hw2), cer(df_hw2), f1(df_hw2), prec(df_hw2), sens(df_hw2), spec(df_hw2))
names(df_detail) <- c("Accuracy", "Classification Error Rate", "F1-Score",
  "Percision", "Sensitivity", "Specificity")
kable(df_detail, col.names = "Values") %>%
  kable_paper('hover', full_width = F)
```

	Values
Accuracy	0.8066298
Classification Error Rate	0.1933702
F1-Score	0.6067416
Percision	0.8437500
Sensitivity	0.4736842
Specificity	0.9596774



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?

```
library(caret)
library(e1071)
cf<- confusionMatrix(as.factor(df_hw2$scored.class), as.factor(df_hw2$class), positive = '1')
print(cf)

## 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
##
```

```
# caret package confusion matrix
cf$table
```

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

```
# created function for confusion matrix
df_hw2 %>% select(scored.class, class) %>% table()
```

```
##              class
## scored.class    0    1
##              0 119  30
##              1   5  27
```

```
caret_fn <- c(cf$byClass["Sensitivity"], cf$byClass["Specificity"])
user_fn <- c(sens(df_hw2), spec(df_hw2))
tble <- cbind(caret_fn, user_fn)
kable(tble) %>%
  kable_paper('hover', full_width = F)
```

	caret_fn	user_fn
Sensitivity	0.4736842	0.4736842
Specificity	0.9596774	0.9596774

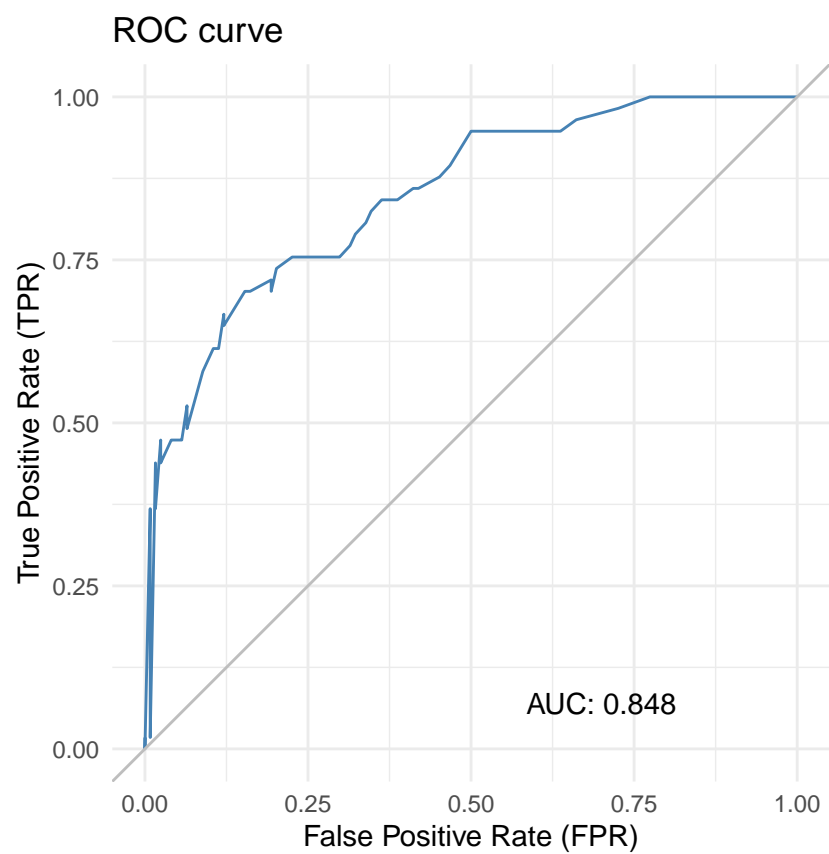
The *caret* package yields the same confusion matrix as the *confusionMatrix* function. We can also see that it yields the same values for **sensitivity** and **specificity** as were generated by our custom functions.

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?

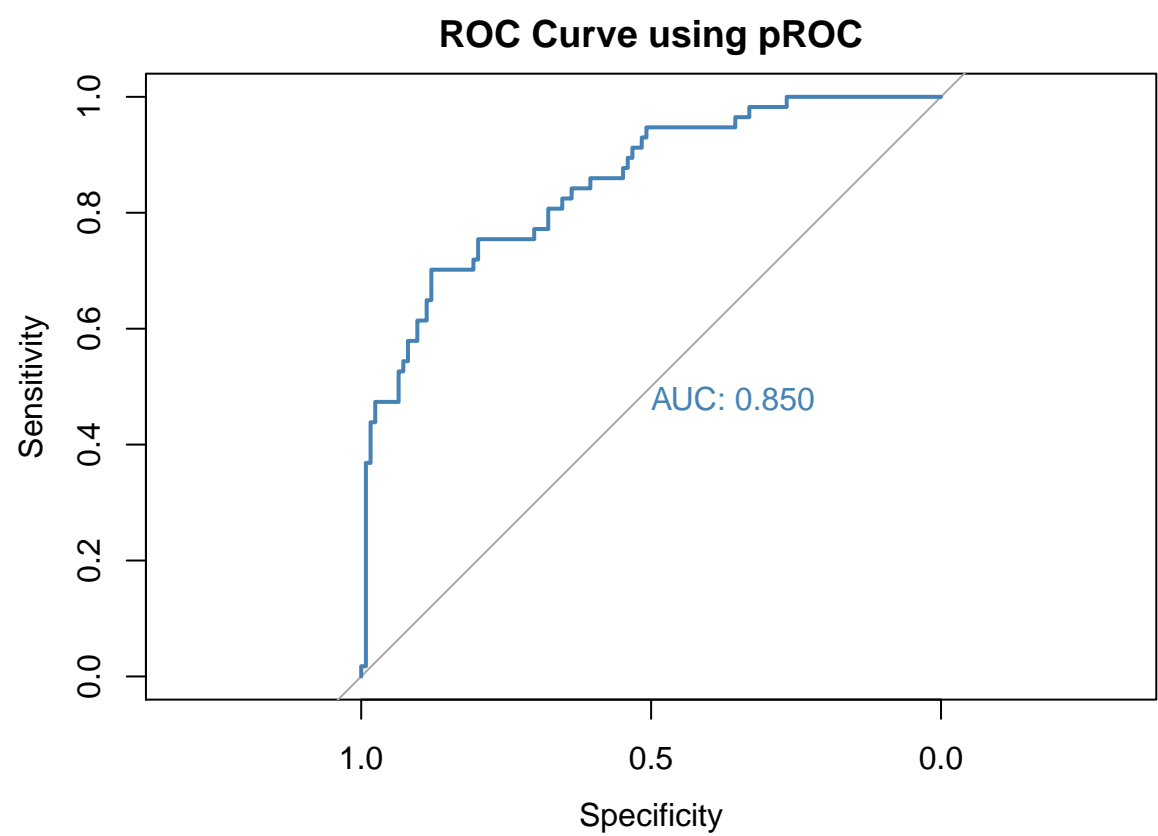


```
require(pROC)
q10[1]
```

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



```
pROC::roc(response = df_hw2$class,
  predictor = df_hw2$scored.probability,
  plot = TRUE,
  print.auc = TRUE,
  col="steel blue",
  bg="grey",
  main = "ROC Curve using pROC")
```



```
##
## Call:
## roc.default(response = df_hw2$class, predictor = df_hw2$scored.probability,      plot = TRUE, print.auc = TRUE, col = "steel blu
##
## Data: df_hw2$scored.probability in 124 controls (df_hw2$class 0) < 57 cases (df_hw2$class 1).
## Area under the curve: 0.8503
```

The two plots above show that results are very similiar in the pROc package and our custom roc function for both the curve and AUC calculation.