Critical Thinking Group 4: DATA621 Homework 2

Table of Contents

TEAM Members:

Rajwant Mishra

Priya Shaji

Debabrata Kabiraj

Isabel Ramesar

Sin Ying Wong

Fan Xu

---

1 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.

2 Deliverables

Upon following the instructions below, use your created R functions and the other packages to generate the classification metrics for the provided data set. A write-up of your solutions submitted in PDF format.

3 Task 1: Download Data Set

```
data_raw <- read_csv('https://raw.githubusercontent.com/Rajwantmishra/DATA621_CR4/master/HW2/classification-output-data.csv?_sm_au_=iVVW2ql3rPKlbr26kRvMGK3JRp2ft')

data_raw
```

pregnant <dbl>	glucose <dbl>	diastolic <dbl>	skinfold <dbl>	insulin <dbl>	bmi <dbl>	pedigree <dbl>	a... <dbl>	class <dbl>
7	124	70	33	215	25.5	0.161	37	0
2	122	76	27	200	35.9	0.483	26	0
3	107	62	13	48	22.9	0.678	23	1
1	91	64	24	0	29.2	0.192	21	0
4	83	86	19	0	29.3	0.317	34	0
1	100	74	12	46	19.5	0.149	28	0
9	89	62	0	0	22.5	0.142	33	0
8	120	78	0	0	25.0	0.409	64	0
1	79	60	42	48	43.5	0.678	23	0
2	123	48	32	165	42.1	0.520	26	0

1-10 of 181 rows | 1-9 of 11 columns Previous 1 2 3 4 5 6 ... 19 Next

4 Task 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

```
data <- data_raw %>%
  select(class, scored.class, scored.probability)
data
```

class <dbl>	scored.class <dbl>	scored.probability <dbl>
0	0	0.32845226
0	0	0.27319044
1	0	0.10966039
0	0	0.05599835
0	0	0.10049072
0	0	0.05515460
0	0	0.10711542
0	0	0.45994744
0	0	0.11702368
0	0	0.31536320

1-10 of 181 rows Previous **1** 2 3 4 5 6 ... 19 Next

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?

Answer: the field class (the rows) represent the actual class, and the field scored.class (the columns) represent the predicted class.

```
data %>%
  select(class, scored.class) %>%
  mutate(class = recode(class,
                        '0' = 'Actual Negative',
                        '1' = 'Actual Positive'),
         scored.class = recode(scored.class,
                               '0' = 'Predicted Negative',
                               '1' = 'Predicted Positive')) %>%
  table()
```

```
##                  scored.class
## class             Predicted Negative Predicted Positive
##   Actual Negative                119                  5
##   Actual Positive                 30                 27
```

As a reference to the following questions (#3 - #13), a a detail description of all variables in the formula (and all the formulas in the question below) are listed below:

```
a) TP: True Positive
b) TN: True Negative
c) FP: False Positive
d) FN: False Negative
e) Accuracy: The closeness of the measurements to a specific value
f) Classification Error Rate: The ratio of total number of units in error to the total
population, or can be calculated as 1-Accuracy
g) Precision: The closeness of the measurements to each other.
h) Sensitivity: The proportion of actual positives that are correctly identified as suc
h, AKA true positive rate
i) Specificity: the proportion of actual negatives that are correctly identified as suc
h, AKA true negative rate
j) F1 score: a measure of a test's accuracy, and is calucalted as the harmonic mean of
the precision and Sensitivity
```

5 Task 3: 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.

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

Answer: a function named `func_accuracy` to represent the formula of Accuracy.

```
func_accuracy <- function(data){
  total <- nrow(data)
  tn <- sum(data$class == 0 & data$scored.class ==0)
  tp <- sum(data$class == 1 & data$scored.class ==1)
  return((tn+tp)/total)
}

func_accuracy(data)
```

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

6 Task 4: Classification Error Rate

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.

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

Answer: a function named `func_Error_Rate` to represent the formula of Classification Error Rate.

```
func_Error_Rate <- function(data){
  total <- nrow(data)
  fn <- sum(data$class == 1 & data$scored.class ==0)
  fp <- sum(data$class == 0 & data$scored.class ==1)
  return((fn+fp)/total)
}

func_Error_Rate(data)
```

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

Verify that you get an accuracy and an error rate that sums to one. Answer: verifed the output of functions `func_accuracy` and `func_Error_Rate` add up to 1.

```
func_accuracy(data)+func_Error_Rate(data)
```

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

7 Task 5: Precision

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

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

Answer: a function named `func_precision` to represent the formula of Precision.

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

func_precision(data)
```

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

8 Task 6: Sensitivity

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.

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

Answer: a function named `func_sensitivity` to represent the formula of Sensitivity.

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

func_sensitivity(data)
```

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

9 Task 7: Specificity

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

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

Answer: a function named `func_specificity` to represent the formula of Specificity.

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

func_specificity(data)
```

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

10 Task 8: F1 Score

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.

$$F1Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}$$

Answer: a function named `func_f1score` to represent the formula of F1 score. Precision and Sensitivity are used to compute F1 score, therefore the function `func_precision` and 'func_sensitivity' defined above are reused in this question.

```
func_f1score <- function(data){
  prec <- func_precision(data)
  sens <- func_sensitivity(data)
  return((2*prec*sens)/(prec+sens))
}

func_f1score(data)
```

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

11 Task 9: Prove 0 < F1Score < 1

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.)

Answer: let $\alpha = Precision, \beta = Sensitivity, \gamma = F1Score = \frac{2 \times \alpha \times \beta}{\alpha + \beta}$

$\because 0 < \alpha < 1$ and $0 < \beta < 1$

$\therefore \frac{2 \times \alpha \times \beta}{\alpha + \beta} > 0$

and $\because 0 < \alpha < 1$ and $0 < \beta < 1$ then $\alpha\beta < \alpha$

$\therefore \frac{2 \times \alpha \times \beta}{\alpha + \beta} = \frac{\alpha\beta}{\alpha + \beta} + \frac{\alpha\beta}{\alpha + \beta} < \frac{\alpha}{\alpha + \beta} + \frac{\beta}{\alpha + \beta} = \frac{\alpha + \beta}{\alpha + \beta} = 1$

$\therefore 0 < \gamma < 1$

12 Task 10: 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.

Answer: ROC curve (short form of Receiver Operating Characteristic curve), is a graphical phot that illustrates the diagonostic ability of a binary classifier system as its discrimination threshold is varied (Reference: Wikipedia).

The ROC curve is created by plotting the true positive rate (TPR, or a.k.a Senstivity) against the false positive rate (FPR, can be calculated as (1-Specificity)) at various threshold settings.

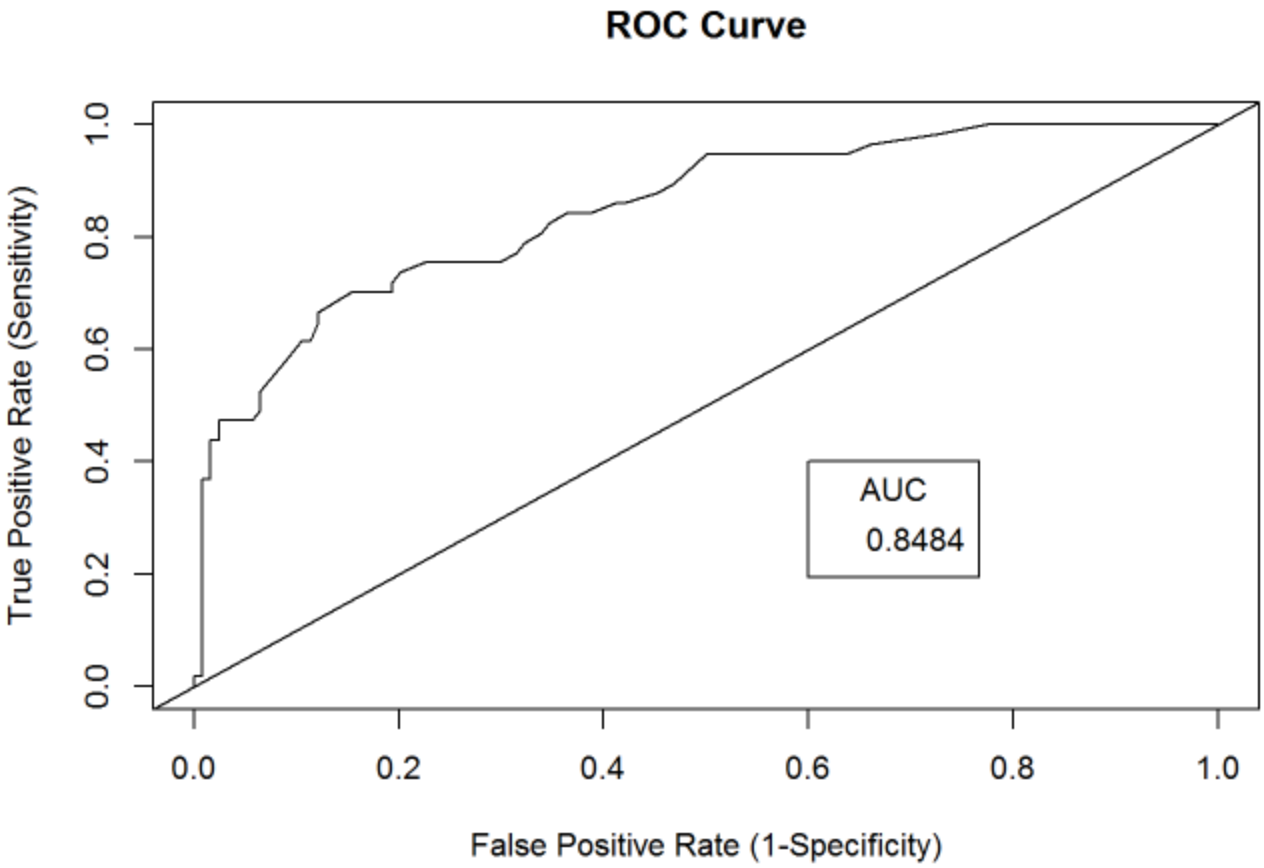
```
library(grid)
func_roc <- function(x,p){
  for (threshold in seq(0,1,0.01)){
    #create dataset for each threshold
    temp <- data.frame(class = x,
                scored.class = if_else(p >= threshold,1,0),
                scored.probability = p)

    #create vectors to store TPR & FPR for all datasets
    if(!exists('TPR') & !exists('FPR')){
      TPR <- func_sensitivity(temp)
      FPR <- 1- func_specificity(temp)
    }
    else{
      TPR <- c(TPR,func_sensitivity(temp))
      FPR <- c(FPR, 1- func_specificity(temp))
    }
  }
  roc_df <- data.frame(TPR, FPR) %>% arrange(FPR)

  #Compute AUC
  AUC <- round(sum(roc_df$TPR * c(diff(roc_df$FPR),0)) + sum(c(diff(roc_df$TPR),0) * c(
diff(roc_df$FPR),0))/2, 4)

  #Create plot
  plot(FPR, TPR, 'l',
     main = 'ROC Curve',
     xlab = 'False Positive Rate (1-Specificity)',
     ylab = 'True Positive Rate (Sensitivity)')
  abline(a=0,b=1)
  legend(0.6,0.4, AUC, title = 'AUC')
}

func_roc(data$class, data$scored.probability)
```



13 Task 11: Produce All Metrics

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

```r
library(knitr)
createdfunctions <- c(func_accuracy(data), func_Error_Rate(data), func_precision(data),
 func_sensitivity(data), func_specificity(data), func_f1score(data))
names(createdfunctions) <- c("Accuracy", "Classification Error Rate", "Precision", "Sen
sitivity", "Specificity", "F1 Score")
kable(createdfunctions, col.names = "Created Functions")
```

	Created Functions
Accuracy	0.8066298
Classification Error Rate	0.1933702
Precision	0.8437500
Sensitivity	0.4736842
Specificity	0.9596774
F1 Score	0.6067416

14 Task 12: Package: Caret

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)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
b <- data_raw %>%
  select(scored.class, class) %>%
  mutate(scored.class = as.factor(scored.class),
         class = as.factor(class))

c <- confusionMatrix(b$scored.class, b$class, positive = "1")

caret_package <- c(c$overall["Accuracy"], c$byClass["Sensitivity"], c$byClass["Specific
ity"])

createdfunctions2 <- c(func_accuracy(data), func_sensitivity(data), func_specificity(da
ta))

d <- cbind(caret_package, createdfunctions2)
kable(d, col.names = c("Caret Package","Created Functions"))
```

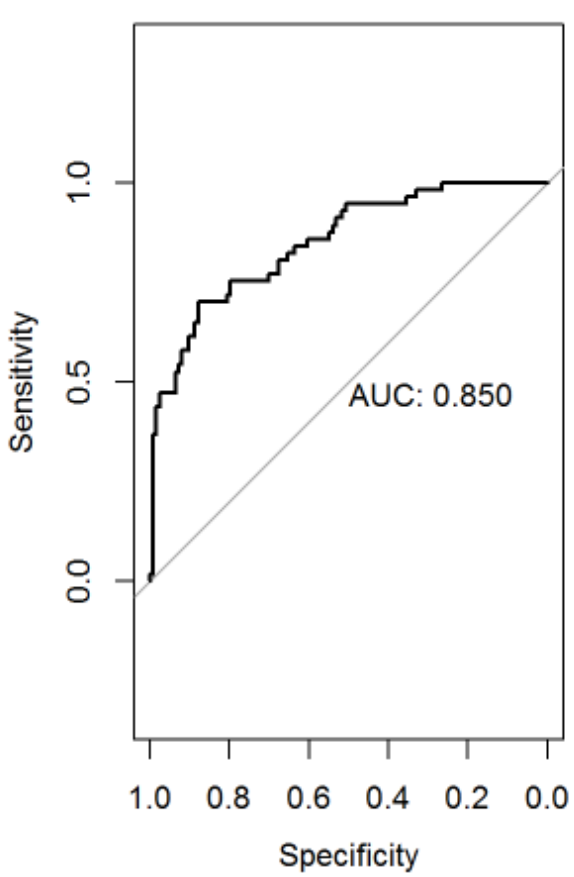
	Caret Package	Created Functions
Accuracy	0.8066298	0.8066298
Sensitivity	0.4736842	0.4736842
Specificity	0.9596774	0.9596774

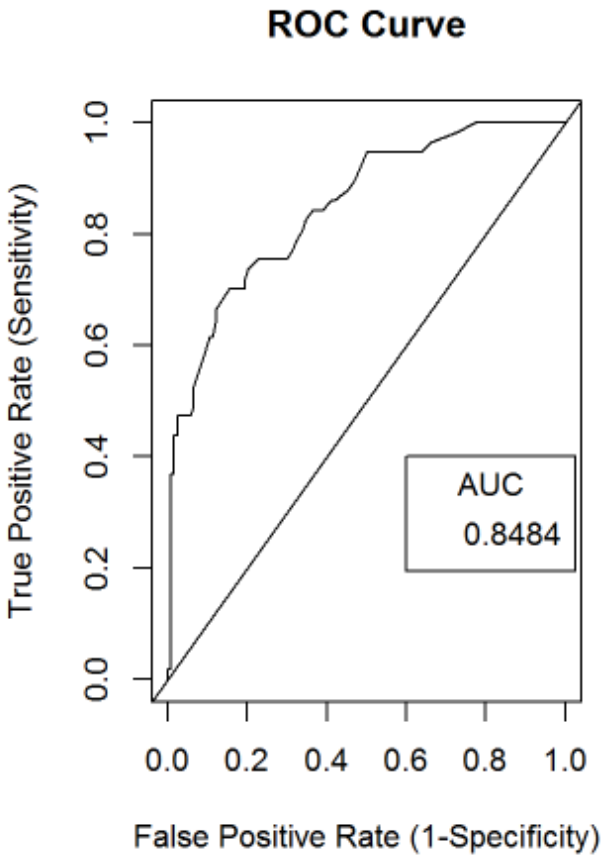
The results from the Caret package and the functions confusionMatrix, sensitivity, and specificity are the same.

15 Task 13: Package: pROC

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

```
library(pROC)
par(mfrow = c(1, 2))
plot(roc(data_raw$class, data_raw$scored.probability), print.auc = TRUE)
func_roc(data$class, data$scored.probability)
```





It appears that our results are similiar to that of the ROC curve for the data set.

Appendix

https://github.com/Rajwantmishra/DATA621_CR4/blob/master/HW2/DATA%20621%20Homework%20%232%20-%20Ver%203.Rmd

Thank you