Data_621_HW2

Mohamed Hassan-El Serafi, Chun Shing Leung, Keith Colella, Yina Qiao, Eddie Xu

2024-09-21

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.

```
library(tidyverse)
library(pROC)
library(knitr)
```

1. Download the classification output data set

```
df = read.csv("https://raw.githubusercontent.com/tonyCUNY/DATA_621/refs/heads/main/classifica
tion-output-data.csv")

head(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
```

2. 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

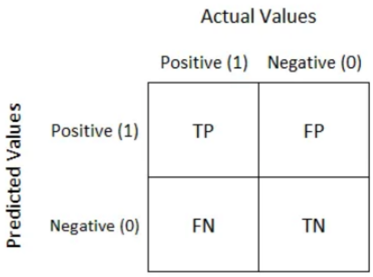
Use the table() function to get the raw confusion matrix for this scored dataset. Make sure you understand the output. In particular, do the rows represent the actual or predicted class? The columns?

```
matrix_df <- df |>
              select(class, scored.class, scored.probability)

Actual_value = matrix_df$class
Predicted_value = matrix_df$scored.class

confus_matrix <- table(Predicted_value, Actual_value)
confus_matrix
```

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

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. Rows represent the Actual class while Column represent the Predicted class.

```
knitr::include_graphics('C_matrix.png')
```



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

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.

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

```
tp = confus_matrix[2,2]
fn = confus_matrix[1,2]
fp = confus_matrix[2,1]
tn = confus_matrix[1,1]

fn_accurary <- function(m){
  return((tp+tn)/nrow(m))
}
fn_accurary(matrix_df)
```

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

4. 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.

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

```
CER <- function(m){
  return((fp+fn)/nrow(m))
}

CER(matrix_df)
```

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

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

```
fn_accurary(matrix_df)+CER(matrix_df)
```

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

5.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 <- function(m){
  return(tp/(tp+fp))
}

precision(matrix_df)
```

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

6. 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 <- function(m){
  return(tp/(tp+fn))
}

sensitivity(matrix_df)
```

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

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

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

```r
specificity <- function(m){
  return(tn/(tn+fp))
}

specificity(matrix_df)
```

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

8. 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.

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

```r
f1score <- function(m){
  return((2*precision(m)*specificity(m))/(precision(m)+specificity(m)))
}

f1score(matrix_df)
```

```
## [1] 0.8979877
```

9. 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.)

Let a = Precision and b = Sensitivity

$$\begin{aligned} F1Score &= \frac{2 * Precision * Sensitivity}{Precision + Sensitivity} \\ &= \frac{2 * a * b}{a + b} \end{aligned}$$

Lower bound:

Given 0 < a < 1 and 0 < b < 1,

a > 0

b > 0

$$\frac{2 * a * b}{a + b} > 0$$

Upper bound:

When a = 1 and b = 1,

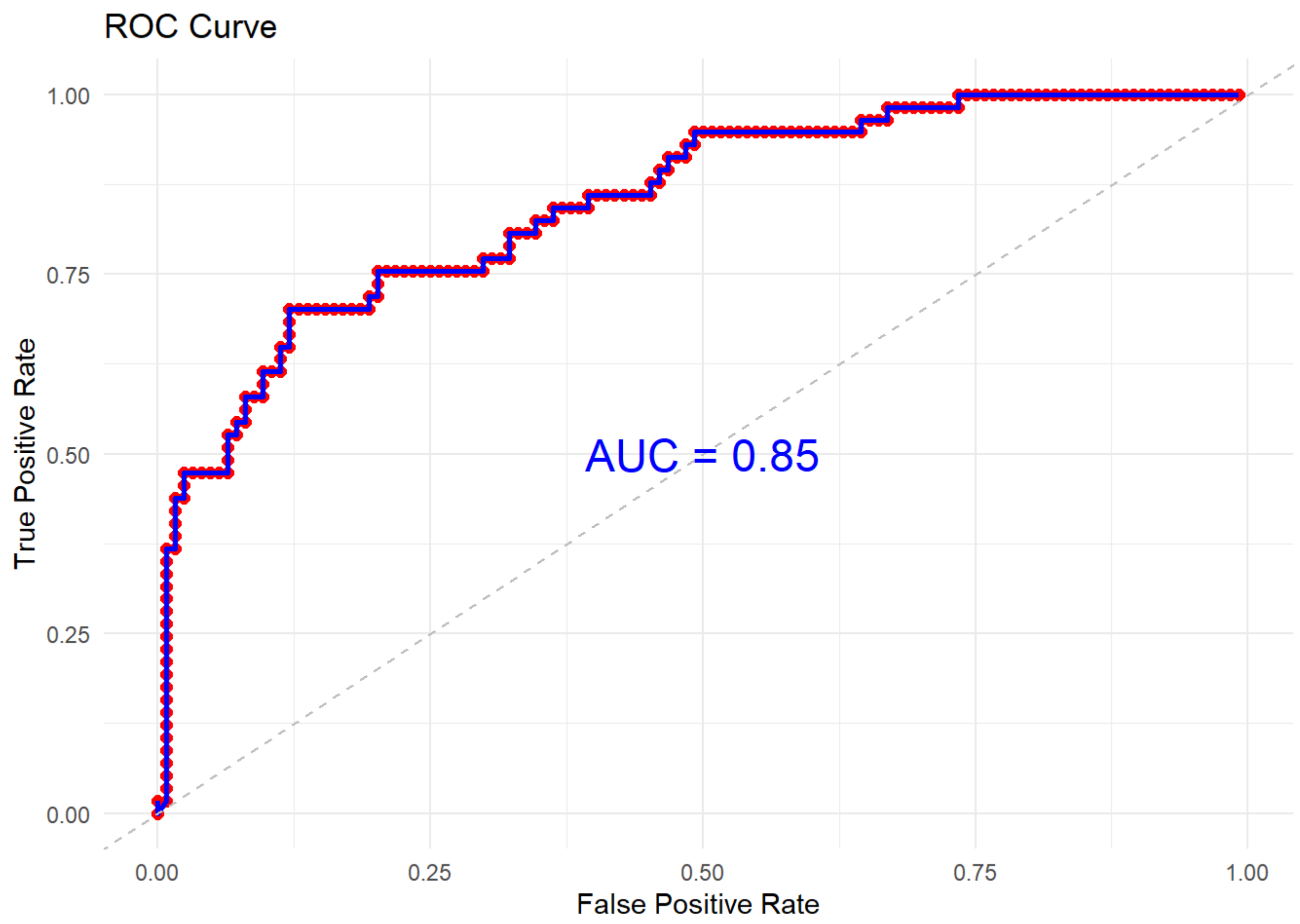
$$F1Score = \frac{2 * 1 * 1}{1 + 1} = \frac{2 * 1}{2} = 1$$

Since a < 1 and b < 1, F1Score will not exceed 1 and we can conclude F1 score will always be between 0 and 1:

$$0 < F1 < 1$$

10. 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.

```
plot_roc_curve <- function(predicted_probs, actual_labels) {

  thresholds <- unique(predicted_probs)
  thresholds <- sort(thresholds, decreasing = TRUE)

  # Initialize vectors for TPR and FPR
  tpr <- numeric(length(thresholds))
  fpr <- numeric(length(thresholds))

  # Calculate TPR and FPR for each threshold
  for (i in seq_along(thresholds)) {
    threshold <- thresholds[i]

    # Classify observations based on the current threshold
    predicted_classes <- as.integer(predicted_probs >= threshold)

    # Create a confusion matrix
    cm <- table(Predicted = predicted_classes, Actual = actual_labels)

    # Calculate TPR and FPR
    if (sum(dim(cm)) == 4) {
      tpr[i] <- cm["1", "1"] / (cm["1", "1"] + cm["0", "1"])
      fpr[i] <- cm["1", "0"] / (cm["1", "0"] + cm["0", "0"])
    }
  }

  # Create a data frame for ROC curve
  roc_df <- data.frame(Threshold = thresholds, TPR = tpr, FPR = fpr)
  auc <- auc(actual_labels, predicted_probs)
  # Plot the ROC curve
  roc_plot <- ggplot(roc_df, aes(x = FPR, y = TPR)) +
    geom_point(size = 2, color = "red") +  # Add points (dots)
    geom_line(color = "blue", size = 1) +  # Connect the points with lines
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "gray") +
    labs(
      title = "ROC Curve",
      x = "False Positive Rate",
      y = "True Positive Rate"
    ) +
    theme_minimal() +
    annotate("text", x = 0.5, y = 0.5, label = paste("AUC =", round(auc, 2)), size = 6, color
= "blue")

  print(roc_plot)
}


plot_roc_curve(matrix_df$scored.probability, matrix_df$class)
```



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

```
m <- c(fn_accurary(matrix_df), CER(matrix_df), precision(matrix_df), sensitivity(matrix_df),
specificity(matrix_df), f1score(matrix_df))
names(m) <- c("Accuracy", "Classification Error Rate", "Precision", "Sensitivity", "Specifici
ty", "F1 Score")
kable(m, col.names = "Classification Metrics")
```

	Classification Metrics
Accuracy	0.8066298
Classification Error Rate	0.1933702
Precision	0.8437500
Sensitivity	0.4736842
Specificity	0.9596774
F1 Score	0.8979877

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?

The results are the same as the function we created before.

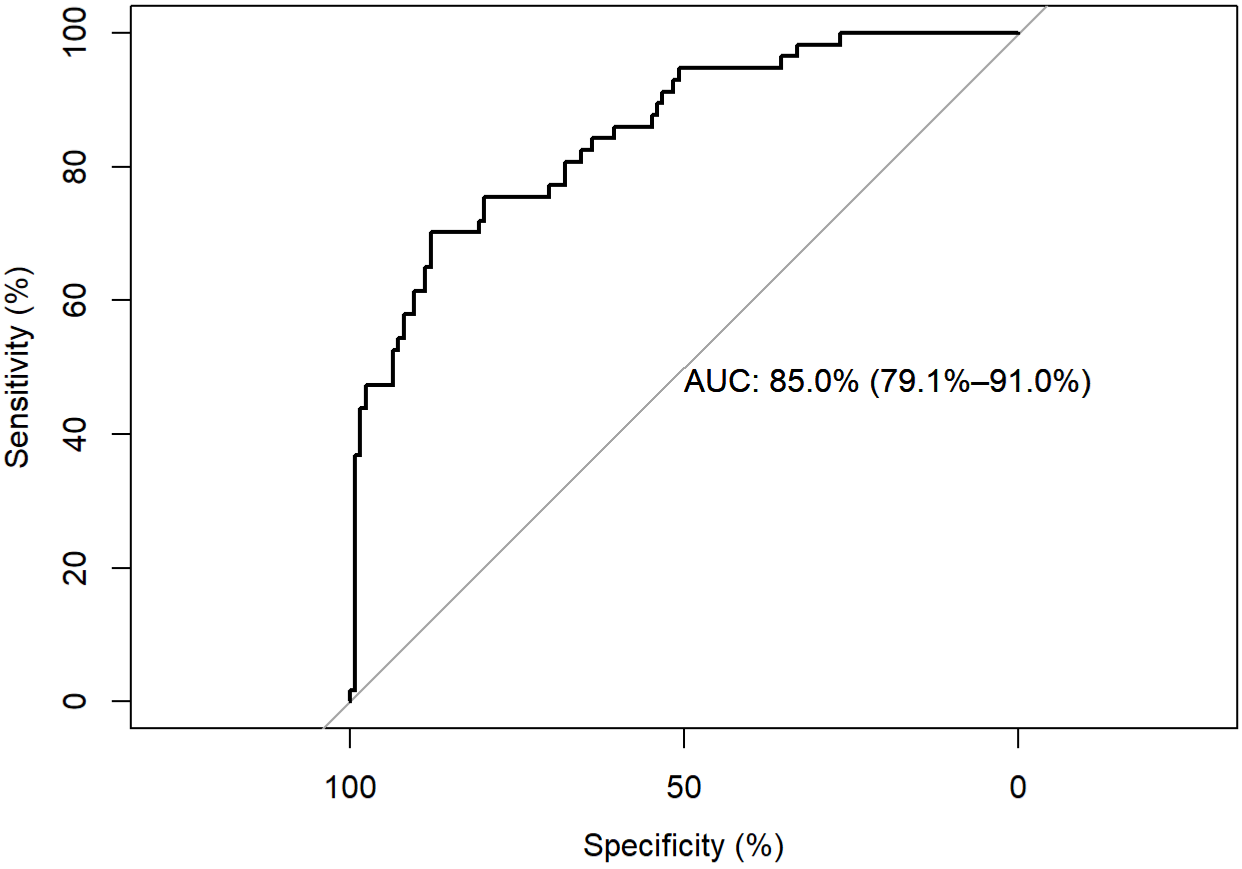
```
library(caret)

actual_v <- factor(c(df %>% pull(class)))
predicted_v <- factor(c(df %>% pull(scored.class)))

confusionMatrix(data=predicted_v, reference = actual_v, 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. Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

It looks the about the same.

```
roc(df$class,df$scored.probability, percent=TRUE, plot=TRUE, ci=TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = df$class, predictor = df$scored.probability,     percent = TRUE, ci
= TRUE, plot = TRUE, print.auc = TRUE)
##
## Data: df$scored.probability in 124 controls (df$class 0) < 57 cases (df$class 1).
## Area under the curve: 85.03%
## 95% CI: 79.05%-91.01% (DeLong)
```

```
```