DATA 621 Homework #2

Critical Thinking Group 4

2/20/2020

```
library(tidyverse)
```

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.

Supplemental Material

- Applied Predictive Modeling, Ch. 11 (provided as a PDF file).
- Web tutorials: http://www.saedsayad.com/model_evaluation_c.htm

Deliverables (100 Points)

- 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.

Instructions

Complete each of the following steps as instructed:

1. Download the classification output data set (attached in Blackboard to the assignment).

```
data_raw <- read_csv('https://raw.githubusercontent.com/oggyluky11/DATA621_CR4/master/classification-ou
```

```
## Parsed with column specification:
## cols(
##   pregnant = col_double(),
##   glucose = col_double(),
##   diastolic = col_double(),
##   skinfold = col_double(),
##   insulin = col_double(),
##   bmi = col_double(),
##   pedigree = col_double(),
##   age = col_double(),
##   class = col_double(),
```

```
##   scored.class = col_double(),
##   scored.probability = col_double()
## )
```

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

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

```
## # A tibble: 181 x 3
##    class scored.class scored.probability
##    <dbl>        <dbl>              <dbl>
##  1     0            0             0.328
##  2     0            0             0.273
##  3     1            0             0.110
##  4     0            0             0.0560
##  5     0            0             0.100
##  6     0            0             0.0552
##  7     0            0             0.107
##  8     0            0             0.460
##  9     0            0             0.117
## 10     0            0             0.315
## # ... with 171 more rows
```

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
```

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.

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

```
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
```

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.

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

```
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.

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

```
## [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.

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

```
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
```

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.

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

```
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
```

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.

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

```
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
```

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.

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

```
func_f1score <- function(data){
  fp <- sum(data$class == 0 & data$scored.class ==1)
  tp <- sum(data$class == 1 & data$scored.class ==1)
  fn <- sum(data$class == 1 & data$scored.class ==0)
  prec <- tp/(tp+fp)
  sens <- tp/(tp+fn)
  return((2*prec*sens)/(prec+sens))
}

func_f1score(data)
```

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

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 $ab < a$.)

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

$$\because 0 < \alpha < 1 \text{ and } 0 < \beta < 1$$

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

$$\text{and } \because 0 < \alpha < 1 \text{ and } 0 < \beta < 1 \text{ then } \alpha\beta < \alpha$$

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

$$\therefore 0 < \gamma < 1$$

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.