HW2

Antonio Bayquen

March 12, 2019

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

```
df <- read.csv(url('https://raw.githubusercontent.com/ajbayquen/MSDS/master/c
lassification-output-data.csv'))
class(df)

## [1] "data.frame"

head(df, 10)

##    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
## 7         9      89        62        0       0 22.5    0.142  33     0
## 8         8     120        78        0       0 25.0    0.409  64     0
## 9         1      79        60       42      48 43.5    0.678  23     0
## 10        2     123        48       32     165 42.1    0.520  26     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
## 7             0         0.10711542
## 8             0         0.45994744
## 9             0         0.11702368
## 10            0         0.31536320
```

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 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?

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

##
##       0   1
##   0 119   5
##   1  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.

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

```
accuracy<-function(df){
  tp=nrow(df[df$class==1 & df$scored.class==1,])
  tn=nrow(df[df$class==0 & df$scored.class==0,])
  fp=nrow(df[df$class==0 & df$scored.class==1,])
  fn=nrow(df[df$class==1 & df$scored.class==0,])
  return ((tp+tn)/(tp+fp+tn+fn))
}
acc1 = accuracy(df)
acc1

## [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.

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

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

```
class_error<-function(df){
  tp=nrow(df[df$class==1 & df$scored.class==1,])
  tn=nrow(df[df$class==0 & df$scored.class==0,])
  fp=nrow(df[df$class==0 & df$scored.class==1,])
  fn=nrow(df[df$class==1 & df$scored.class==0,])
  return ((fp+fn)/(tp+fp+tn+fn))
```

```
}
class_err1 = class_error(df)
class_err1

## [1] 0.1933702

#accuracy + classification error must equal to 1
sum1 = acc1 +  class_err1
sum1

## [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(df){
  tp=nrow(df[df$class==1 & df$scored.class==1,])
  fp=nrow(df[df$class==0 & df$scored.class==1,])
  return (tp/(tp+fp))
}
pres1 = precision(df)
pres1

## [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(df){
  tp=nrow(df[df$class==1 & df$scored.class==1,])
  fn=nrow(df[df$class==1 & df$scored.class==0,])
  return (tp/(tp+fn))
}
sensi1 = sensitivity(df)
sensi1

## [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.

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

```
specificity <-function(df){
  tn=nrow(df[df$class==0 & df$scored.class==0,])
```

```
  fp=nrow(df[df$class==0 & df$scored.class==1,])
  return (tn/(tn+fp))
}
spec1 = specificity(df)
spec1

## [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.

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

```
f1_score<-function(df){
  pres2<- precision(df)
  sensi2<- sensitivity(df)
  return (2*pres2*sensi2/(pres2+sensi2))
}
f1_scr1 = f1_score(df)
f1_scr1

## [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).

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

$$F1Score = \frac{2TP}{2TP + FP + FN}$$

Answer - A high TP (true positive) value means that the false positive value is low. Therefore, as the TP limit apporaches 1 (max value), the FP and FN values approach 0 giving us an $F1Score = \frac{2(1)}{2(1) + 0 + 0}$ = 1.

A low TP (true postive) value on the other hand, means that the false positive value is high. Therefore, as the TP limit approaches 0 (min value), $F1Score = \frac{2(0)}{2(0) + 1 + 1}$ = 0.

We can therefore conclude that the F1 Score should be between 0 and 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.

```
ROC <- function(df)
{

}
```