DATA 621 Homework 2 - Classification Metrics

Cheryl Bowersox, Christopher Martin, Robert Sellers, Edwige Talla Badjio

June 26, 2016

2. Using 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?

Reference Confusion Matrix

	p0	**p1**
a0	TN	FN
a1	FP	TP

Class Confusion Matrix

	p0	**p1**
a0	119	30
a1	5	27

This tabulation represents the success of the models predictions *scored.class/a* versus the actual results *class/p*. The *scored.probability* represents the threshold value which determines if the predicted value is 0 (*scored.probability* < 50%) or 1 (*scored.probability* > 50%).

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.

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

```
the.accuracy <- function(df, predicted, actual="class"){
   #computing the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the accuracy
      myaccuracy = (TP + TN) / (TP + FN + FP + TN)
      return(as.numeric(myaccuracy))
}
```

The accuracy for the provided data is 0.8066. This means that 81% of the number of predictions from the model (postiive or negative) were correct.

---

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

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

```
the.classif.err.rate <- function(df, predicted, actual="class"){
   #obtaining the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the classification error rate
      myclasserrate <- (FP + FN) / (TP + FP + TN + FN)
      return(as.numeric(myclasserrate))
}
```

The classification error rate for the provided data is 0.1934 and the sum of the error rate and accuracy correctly sum to 1. The classification error rate indicates about 19% of the model predictions are not correct.

---

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.

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

```
the.precision <- function(df, predicted, actual="class")
{
   #obtaining the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the precision
      myprecision <- TP / (TP + FP)
      return(as.numeric(myprecision))
}
```

The precision for the provided data is 0.8438. This is the percentage of the positive predictions that are actually positive.

---

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.

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

```
the.sensitivity <- function(df, predicted, actual="class")
{
   #obtaining the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the sensitivity
      mysensitiv <- TP / (TP + FN)
      return(as.numeric(mysensitiv))
}
```

The sensitivity for the provided data is 0.4737. This precentage shows how many of the actual positive outcomes were predicted as positive in the model.

---

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.

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

```
the.specificity <- function(df, predicted, actual="class"){
   #obtaining the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the specificity
      myspecif <- TN / (TN + FP)
      return(as.numeric(myspecif))
}
```

The specificity for the provided data is 0.9597. This is the percentage of negative outcomes that were correctly predicted by the model.

---

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.

$$F1Score = \frac{2 \quad Precision \quad Sensitivity}{}$$

$$Precision + Sensitivity$$

```
the.f1score <- function(df, predicted, actual="class"){
   #obtaining the confusion matrix
      conf.mat <- the.conf.matrix(df, predicted)
   # Assigning TP, FN, FP and TN using the confusion matrix
      TN <- conf.mat[1,1]
      FN <- conf.mat[1,2]
      FP <- conf.mat[2,1]
      TP <- conf.mat[2,2]
   #computing the specificity
      mysensitiv <- TP / (TP + FN)
      myprecision <- TP / (TP + FP)
      myf1score <- (2 * myprecision * mysensitiv)/(myprecision + mysensitiv)
      return(as.numeric(myf1score))
}
```

The F1 score for the provided data is 0.6067 measuring the combined sensitivity and precision of the model.

---

9. Before we move on, letâs 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 $ab < a$.)

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

The F1 score is quantified by precision and sensitivity.

$$Precision = \frac{TruePositives}{AllActualPositives}$$

Precision will always have a lower bound of 0 (if true positives are 0) and an upper bound of 1 (if true positives are equal to all actual positives).

$$Sensitivity = \frac{TruePositives}{AllPredictedPositives}$$

Sensitivity will always have a lower bound of 0 (if true positives are 0) and an upper bound of 1 (if true positives are equal to all predicted positives).

If either Precision or Sensitivity equals 0, the F1 score will have a numerator of 0 and is therefore lower bound by 0. If both variables are equal to 0, the F1 score denominator will be 0 and be indeterminant/invalid. At the other end of the spectrum, if the Precision and Sensitivity are equal to their maximum value (1), the numerator will be 2 (2 * 1 * 1) and the denominator will be 2 (1 + 1), giving the F1 score itâs maximum value of 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

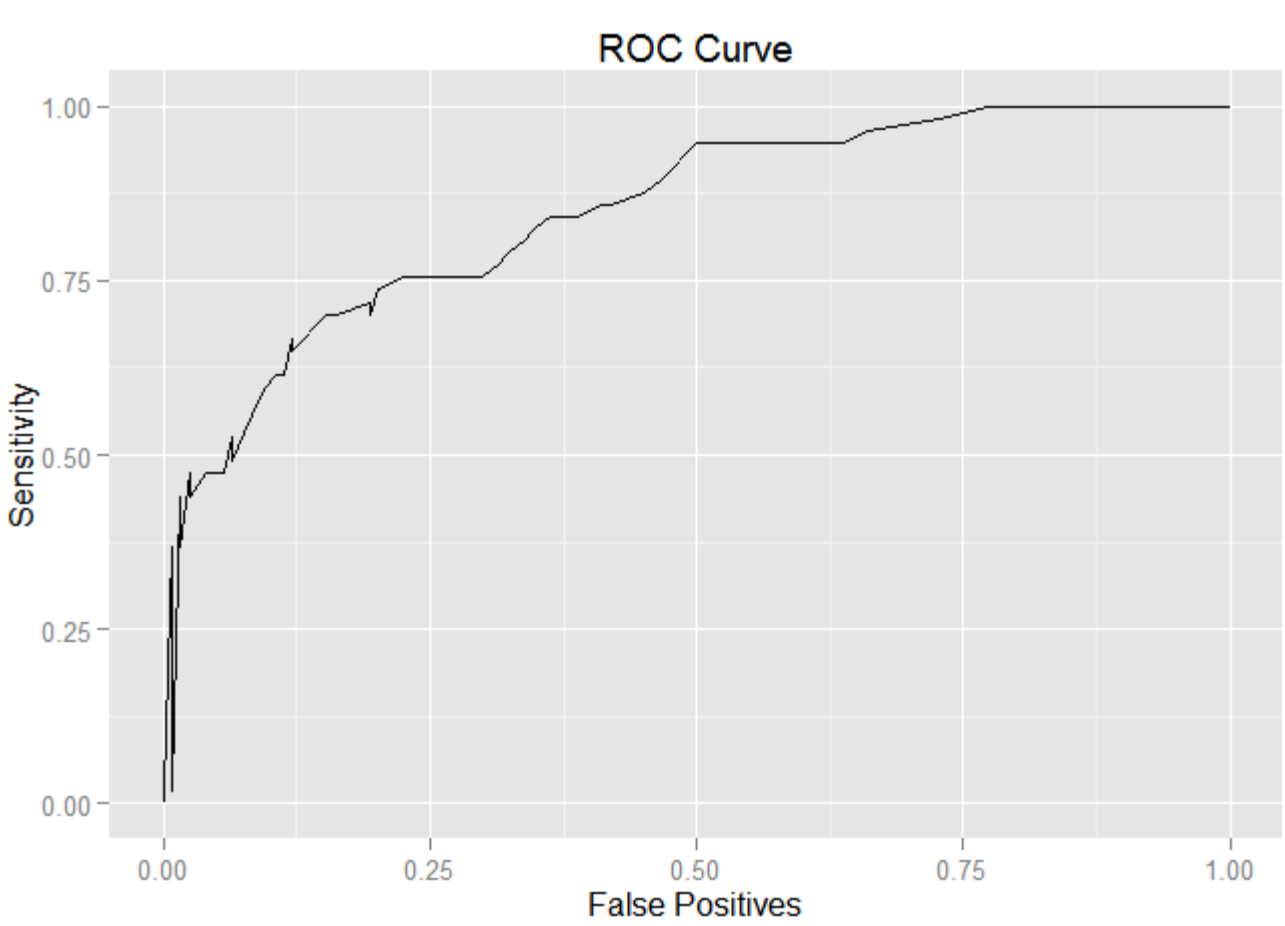
```
roc_func2<- function(df,actual) {
```

```
  #create empty dfs
  new.sens <- c()
  new.spec <- c()
  seq <- c()
  sequence <- seq(0.01,1,0.01)
  #loop to create the specificity and sensitivity results and binary scores
  for(i in sequence) {
        seq <- append(seq,i)
        if(i < min(df$scored.probability)){ #all 0
          new.sens <- append(new.sens,0)
          new.spec <- append(new.spec, 1)
        }else if(i > max(df$scored.probability)){
          new.sens <- append(new.sens,1)
          new.spec <- append(new.spec, 0)
        }else{
          df$new.score <- ifelse(df$scored.probability < i, 0, 1)
          new.spec <- append(new.spec,the.specificity(df, predicted = "new.score", actual = "c
lass"))
          new.sens <- append(new.sens,the.sensitivity(df, predicted = "new.score", actual = "c
lass"))
        }
  }

   df_roc <- data.frame(threshold=seq, sensitivity = new.sens, specificity = new.spec, falsepo
s = 1 - new.spec)
  return(df_roc)
}
```

```
rocdata <- roc_func2(originaldata, actual = "class")

rocArea = trapz(rocdata$sensitivity, rocdata$falsepos)
```



The area under the ROC curve is 0.8280277.

So close!

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

Summary Output

Class Confusion Matrix

	p0	p1
a0	119	30
a1	5	27

Accuracy: 80.66%

Classification Error Rate: 19.34%

Precision: 84.38%

Sensitivity: 47.37%

Specificity: 95.97%

F1 Score: 60.67%

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?

Caret Confusion Matrix Results

	0	**1**
0	119	30
1	5	27

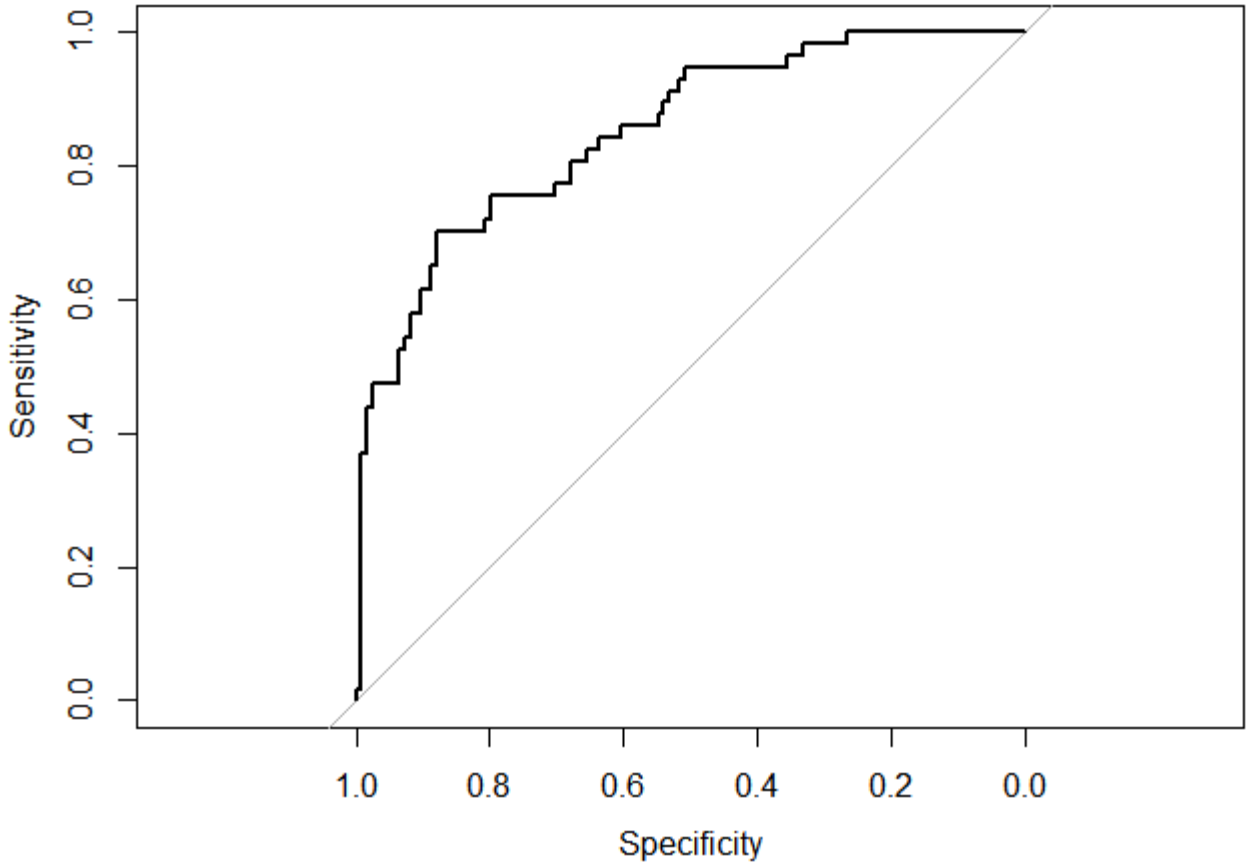
Comparison Table

	Sensitivity	**Specificity**
Caret Package	0.4737	0.9597
Custom Functions	0.4737	0.9597

The results from the Caret package are identical to those obtained from our custom functions.

Where is your code for these three Caret functions? -3

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?




```
##
```

```
## Call:
## roc.formula(formula = class ~ scored.probability, data = classificationdata)
##
## Data: scored.probability in 124 controls (class 0) < 57 cases (class 1).
## Area under the curve: 0.8503
```

The area under the curve found by using the roc function in the pROC package is .8503. Using our custom function the area was found to be 0.8280277, a difference of 0.0222835 or around 2.69%.
This difference could be caused by a few different things. In the custom function we used the recommended binning from 0 to 1 at intervals of .01 and evaluated the specificity and sensitivity for 100 probability levels. The roc function may use different or more specific binning or evaluate a different number of points, resulting in a slightly different curve and subsequent area.