DATA621 Homework 3 - Classification Metrics

Erik Nylander

March 8, 2016

1 Introduction

In this analysis, we will be using a data set that contains information about pregnancies and contains the predictors of glucose levels, diastolic blood pressure, skinfold, insulin, body mass index (bmi), pedigree, and age. A model has already been fit to the data and class (scored.class) predictions have been made from this model based on a probability cutoff (scored.probability) of 50%. The data set also contains a class variable (class) that contains the actual classification for each of the individuals. We will be using this data set to work through a series of classification metrics to help us evaluate the model's performance.

2 Creating Classification Metrics

The first thing that we will be doing with this data set is creating our own set of functions to calculate our classification metrics. In the following sections we will lay out the calculation of each of the metrics and the results of our functions. The code for each of the functions can be found in the appendix.

2.1 Confusion Matrix

The confusion matrix gives us an matrix of values that contain the number of true positive results, false positive, false negative, and true negative classifications in the data set. In R, we can use the *table()* function to compute this matrix and we get the results in Table 1, the scored.class is across the columns of the table and the true class are the rows.

Table 1: Confusion Matrix

Class	Negative	Positive
Negative	119	5
Positive	30	27

2.2 Accuracy

The first function that we will create takes the data frame with the columns for class and scored.class identified and returns the accuracy according to the following calculation of accuracy:

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

The function we have created is named *acc()* and running the function on our data set results in an Accuracy = 0.8066.

2.3 Classification Error Rate

The next function that we will create once again takes the data frame with the class and scored.class columns identified and returns the classification error rate according to the following formula:

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

The function that we have created is named *CER()* and returns a classification error rate of 0.1934. We note that the accuracy and classification error rate (0.8066 + 0.1934 = 1) sum to 1.

2.4 Precision

Next we will create a function that calculates the precision based on the following formula:

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

Our function in the appendix is named *prec()* and when it is run on our data the result is a precision of 0.8438.

2.5 Sensitivity

Next we will create a function that calculate the sensitivity based on the following formula:

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

The function in the appendix is named *sens()* and when we run it on our data we get a sensitivity of 0.4737.

2.6 Specificity

The next function that we have created calculates the specificity based on the following formula:

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

The function in the appendix is named *spec()* and when run on our data results in a specificity of 0.9597.

2.7 F1 Score

The next function that we have created calculates the data's F1 Score based on the following formula:

$$\text{F1 Score} = \frac{2(Precision)(Sensitivity)}{Precision + Sensitivity}$$

The function in the appendix is named *F1()* and when run on our data results in a F1 Score of 0.6067.

2.8 F1 always between 0 and 1

Our next task is to show that the F1 score will always be between 0 and 1. We will use the following relationship to show this, if $0 < a < 1$ and $0 < b < 1$ then $ab < a$. Our work follows below:
Given the relationship above

$$\frac{2ab}{a+b} < \frac{2a}{a+b}$$

Dividing both sides by 2 results in:

$$\frac{ab}{a+b} < \frac{a}{a+b}$$

Using some algebra we get that

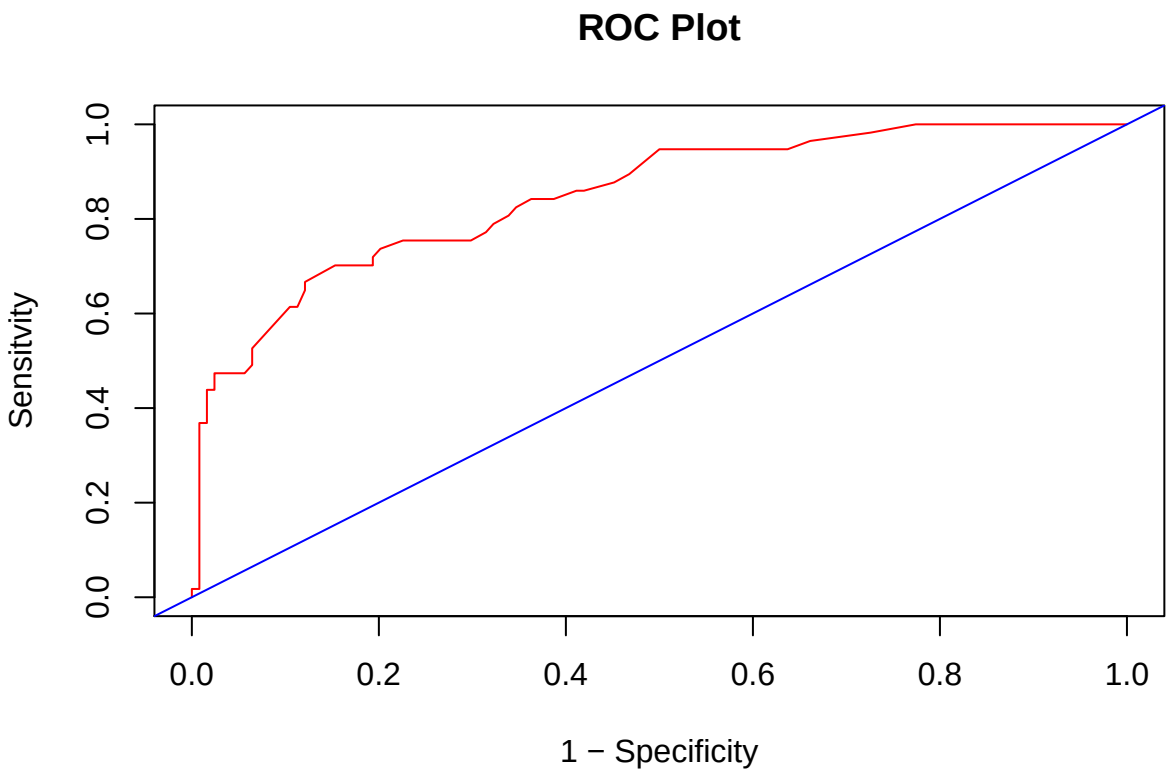
$$\frac{ab}{a+b} < \frac{a}{a+b} = \frac{a}{a(1+b/a)} = \frac{1}{1+b/a} < 1$$

Therefore we have that

$$0 < \frac{2ab}{a+b} < 1$$

2.9 ROC Curve and AUC

The final function that we will create generates the ROC curve for our data. The function also calculates the AUC using the trapezoidal approximation for the area. The roc curve that is generated can be seen in the figure below. We also note that the AUC value that is calculated by our function in AUC = 0.8489.



2.10 Summary of Results

Table 2 contains a summary of the results found above.

Table 2: Summary of Classification Metrics

Metric	Value
Accuracy	0.8066
CER	0.1935
Precision	0.8438
Sensitivity	0.4737
Specificity	0.9597
F1 Score	0.6067
AUC	0.8489

3 caret Package

In the next portion of our analysis we will explore the **caret** package to explore the packages functions for calculating the above classification metrics without needing to write our own function.

3.1 *confusionMatrix()*

The *confusionMatrix()* function takes the data from our data set and calculates the confusion matrix along a number of different statistics. The output of the function can be seen below.

```
## 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
##
```

3.2 *sensitivity()*

The *sensitivity()* function takes the data from our data set and calculates the just the sensitivity of the classification data. When we run the function on our data we get a sensitivity of 0.4737.

3.3 *specificity()*

The *specificity()* function takes the data from our data set and calculates the specificity of the classification data. When we run the function on our data we get a specificity of 0.9597

3.5 *caret* and Our Functions

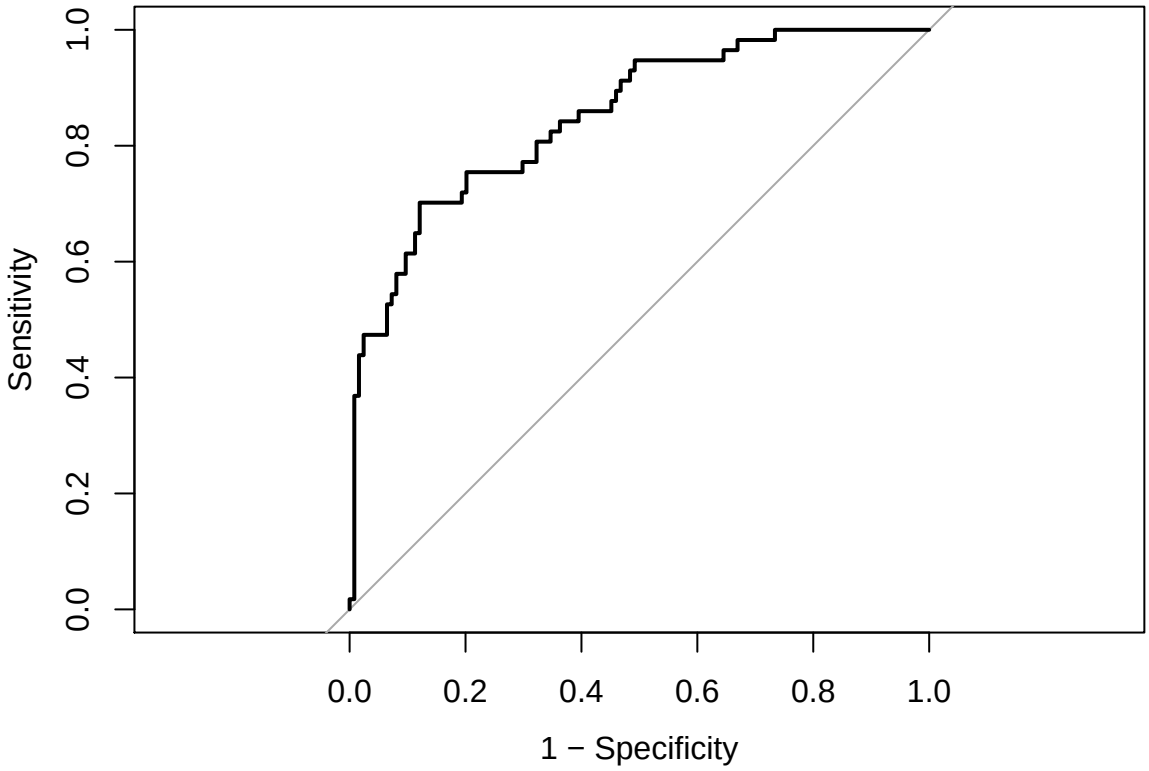
Our exploration of the *caret* function has shown us that this function generates results that are exactly the same as our created functions. The nice thing about this package is the ease of generating all of the classification metrics by calling the *confusionMatrix()* function on our data without having to generate each of the metrics. We also see that the *caret* package does allow for the calling of each individual metric as its own function.

4 *pROC* Package

The final portion of our analysis will use the *pROC* package to generate the ROC curve for our classification data and at the same time it will also generate the AUC.

4.1 ROC Curve

In the figure below we can see the ROC curve that is generated by the *pROC* package. We note that this function appears to be the same at the ROC curve that we generated in our own function.




```
##
## Call:
## roc.default(response = mydata$class, predictor = mydata$scored.probability)
##
## Data: mydata$scored.probability in 124 controls (mydata$class 0) < 57 cases (mydata$class 1).
## Area under the curve: 0.8503
```

We also see that the AUC is reported with the plot of the ROC curve. We do note that the AUC from the *pROC* function gives us a value of 0.8503. This value is very close to the value that we found with our own function of 0.8489. The difference in these two values is likely due to a difference in the way that the two functions calculate the area under the curve.

5 Appendix

In this section, we provide the R code used in the analysis.

```
# Reading in the Data
library(dplyr)
mydata <- read.csv("DATA621/classification-output-data.csv")
head(mydata)

# Finding the Scored Probability cutoff
mydata %>%
  filter(scored.class == 1) %>%
```

```r
  filter(scored.probability == min(scored.probability))

mydata %>%
  filter(scored.class == 0) %>%
  filter(scored.probability == max(scored.probability))

# Test set to figure out the rows and columns
ref = matrix(c("P", "N", "P", "P", "P", "P","N", "N"), ncol=1)
pred = matrix(c("P", "N", "N", "P", "P", "P","P", "P"), ncol=1)
table(ref, pred)

# Creating the confustion matrix using the table() function
# The columns are the predicted values.
#  TN  FP
#  FN  TP
cfmatrix <- table(mydata$class, mydata$scored.class)
cfmatrix

# Creating a function to calculate the accuracy using the data frame and the class columns identified.
acc <- function(df, actual, predicted){
  cf <- table(df[,actual], df[,predicted])
  acc <- (cf[2,2] + cf[1,1]) / sum(cf)
  return(acc)
}

acc(mydata, 9, 10)

# Creating a function to calculate the classification error rate.
CER <- function(df, actual, predicted){
  cf <- table(df[,actual], df[,predicted])
  error <- (cf[1,2] + cf[2,1]) / sum(cf)
  return(error)
}

CER(mydata, 9, 10)

# Testing that the accuracy and classification error rate add up to 1
a + e # equals 1

# Creating a function to calculate the percision.
prec <- function(df, actual, predicted){
  cf <- table(df[,actual], df[,predicted])
  pre <- cf[2,2]/(cf[2,2] + cf[1,2])
  return(per)
}

prec(mydata, 9, 10)

# Creating a function to calculate the sensitivity.
sens <- function(df, actual, predicted){
  cf <- table(df[,actual], df[,predicted])
  sen <- cf[2,2]/(cf[2,2] + cf[2,1])
  return(sen)
```

```
}

sens(mydata, 9, 10)

# Creating a function to calculate the specificity.
spec <- function(df, actual, predicted){
  cf <- table(df[,actual], df[,predicted])
  spec <- cf[1,1]/(cf[1,1] + cf[1,2])
  return(spec)
}

spec(mydata, 9, 10)

# Creating a function to calculate the F1 score for the predictions
F1 <- function(df, actual, predicted){
  p <- perc(df, actual, predicted)
  s <- sens(df, actual, predicted)
  score <- (2*p*s)/(p+s)
  return(score)
}

F1(mydata, 9, 10)

# Calculate the ROC Plot and the AUC
ROC <- function(class, prob){
  roc <- data.frame(threshold = seq(0, 1, 0.01), fpr = NA, sens = NA, area = NA, area2 = NA)
  roc <- roc[order(-roc$threshold),]
  for(i in 1:101){
    roc$fpr[i] <- sum(prob >= roc$threshold[i] & class == 0) / sum(class == 0)
    roc$sens[i] <- sum(prob >= roc$threshold[i] & class == 1) / sum(class == 1)
  }
  auc <- sum(diff(roc$fpr) * (head(roc$sens,-1)+tail(roc$sens,-1)))/2
  p.plot <- plot(roc$fpr, roc$sens, type = "l", main = "ROC Plot",
                 xlab = "1 - Specificity", ylab = "Sensitvity", col = "red") +
    abline(a = 0, b = 1, col = "blue")
  result <- list("rocplot" = p.plot, "AUC" = auc)
  return(result)
}

a <- ROC(mydata$class, mydata$scored.probability)
a[2]

# Exploring the Caret Packagae
library(caret)

# Constructing the confusion matrix
confusionMatrix(data = as.factor(mydata$scored.class),
                reference = as.factor(mydata$class),
                positive = "1")

# Calculating the sensitivity
sensitivity(data = as.factor(mydata$scored.class),
            reference = as.factor(mydata$class),
```

```
            positive = "1")

# Calculating the specificity
specificity(data = as.factor(mydata$scored.class),
            reference = as.factor(mydata$class),
            negative = "0")

# Exploring the pROC function
library(pROC)
rocCurve <- roc(response = mydata$class,
                predictor = mydata$scored.probability)
auc(rocCurve)
plot(rocCurve, legacy.axes = TRUE)
```