Data 621: Homework 2

Classification Metrics

Aaron Grzasko

March 17, 2018

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.

1. Read Data

Download the classification output data set.

```
# read in csv file provided for assignment
myurl <- "https://raw.githubusercontent.com/spitakiss/Data621/master/Homework2/
classification-output-data.csv"

mydata <- read.csv(myurl)
```

2. Confusion Matrix

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?

```
# subset original data set
cols <- c("scored.class","class")
classdata <- mydata[cols]

# create confusion matrix
cm <- table(classdata)
cm <- cm[order(rownames(cm),decreasing = T), order(colnames(cm),decreasing = T)]
rownames(cm) <- c("Event","Nonevent")
colnames(cm) <- c("Event","Nonevent")
knitr:: kable(cm)
```

	Event	Nonevent
Event	27	5
Nonevent	30	119

The rows in the confusion matrix represent the predicted classes, "Event" and "Nonevent", from the classification model. The columns represent the actual classes represented in the data set.

Note: we code coded '1' values in the raw data set as "Event" (i.e. true value) and '0' values as "NonEvent".

3. Accuracy

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

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

Before we write the accuracy function, let's create a helper function that calculates the various elements of the confusion matrix. This helper function will be used in multiple questions of the assignment.

```r
# helper function to calculate values in 2x2 confusion matrix
cm_calc <- function(df, pcol, acol,truval = 1) {

  # count true pos, true neg, false pos, false neg
  tp <- sum(ifelse(pcol == truval & acol == truval,1,0))
  tn <- sum(ifelse(pcol != truval & acol != truval,1,0))
  fp <- sum(ifelse(pcol == truval & acol != truval,1,0))
  fn <- sum(ifelse(pcol != truval & acol == truval,1,0))

  # output each element of cm
  c(tp=tp,tn=tn,fp=fp,fn=fn)
}
```

Here is the accuracy function:

```r
# accuracy function.
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

acc <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate error rate
  round(sum(cm[c("tp","tn")]) / sum(cm[c("tp","tn","fp","fn")]),digits)

}
```

4. Error Rate

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.

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

```
# error function
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

err_rt <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate error rate
  round(sum(cm[c("fp","fn")]) / sum(cm[c("tp","tn","fp","fn")]),digits)

}
```

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

```
# verify that acc + err = 1 for supplied data set
myacc <- acc(mydata, scored.class, class)
myerr_rt <- err_rt(mydata, scored.class, class)
myacc + myerr_rt
```

```
[1] 1
```

5. Precision

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

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

```
# precision function
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

prec <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate precision rate
  round(sum(cm[c("tp")]) / sum(cm[c("tp","fp")]),digits)

}
```

6. Sensitivity

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.

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

```r
# sensitivity function
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

sens <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate sensitivity rate
  round(sum(cm[c("tp")]) / sum(cm[c("tp","fn")]),digits)

}
```

7. Specificity

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

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

```r
# specificity function
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

spec <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate specificity rate
  round(sum(cm[c("tn")]) / sum(cm[c("tn","fp")]),digits)

}
```

8. F1 Score

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.

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

```r
# F1 function
# arguments: dataframe, predicted column name, actual column name,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

F1 <- function(df, pcol, acol,truval=1, digits = 3) {

  # produce confusion matrix elements
  pcol <- eval(substitute(pcol),df, parent.frame())
  acol <- eval(substitute(acol),df, parent.frame())
  cm <- cm_calc(df,pcol,acol,truval)

  # calculate precision and sensitivity
  myprec <- prec(df, pcol, acol, truval, digits + 3)
  mysens <- sens(df, pcol, acol, truval, digits + 3)

    # calculate F1 rate
  round((2*myprec*mysens) / (mysens + myprec),digits)

}
```

9. Bounds on F1

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 $ab < a$).

Proof 1

Let's expand the F1 equation using the definitions of precision and sensitivity.

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

$$= \frac{2 \times TP \times TP}{(TP + FP)(TP + FN)} \times \frac{1}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

$$= \frac{2(TP)^2}{(TP + FP)(TP + FN)} \times \frac{(TP + FP)(TP + FN)}{TP(TP + FN) + TP(TP + FP)}$$

$$= \frac{2(TP)^2}{TP(2TP + FN + FP)}$$

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

In the equation above, $TP, FN, FP \in \mathbb{N}$ where $\mathbb{N} = \{0, 1, 2, 3...\}$

Given these restrictions, both the numerator and denominator assume non-negative, integer values.

Also, $2TP \leq 2TP + FN + FP$. This implies that the numerator is less than or equal to the denominator (i.e $F1 \leq 1$).

The expression assumes a maximum value of 1 when $FN = FP = 0$ and $TP > 0$

Furthermore, the fraction $= \frac{2TP}{2TP+FN+FP}$assumes a minimum value of zero when the numerator is minimized (i.e. $TP = 0$).

Finally, the expression is undefined when $TP = FN = FP = 0$.

We have shown that $0 \leq F1 \leq 1$ or is undefined.

Proof 2

Let $0 \leq p \leq 1$ and $0 \leq s \leq 1$, where $p =$precision and $s =$sensitivity.

Then $0 \leq p + s \leq 2$ and $2ps \leq 2$.

Therefore, $\frac{2ps}{p+s} \geq 0$ or is undefined when $p = s = 0$.

Now, let's assume $\frac{2ps}{p+s} > 1$.

Then

$$\frac{p+s}{2ps} < 1$$

$$\frac{1}{2s} + \frac{1}{2p} < 1$$

$$\frac{1}{s} + \frac{1}{p} < 2$$

Given the constraints on p and s, the expression on the left side of the inequality is minimized when $p = s = 1$.

But when we substitute $p = 1$ and $s = 1$ we have:

$$2 < 2$$

This is a contradiction, as $2 = 2$; so it is not true that $\frac{2ps}{p+s} > 1$.

We have shown that $0 \leq \frac{2ps}{p+s} \leq 1$ or is undefined.

10. ROC Curve

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.

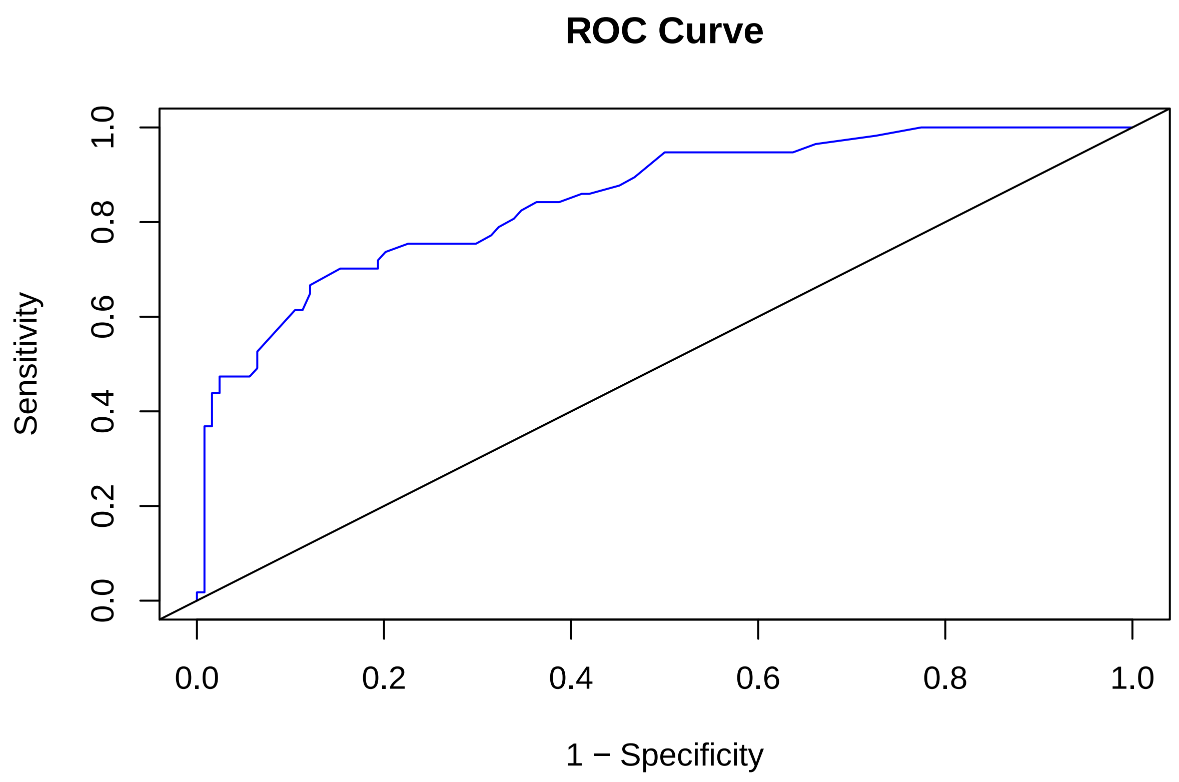
```r
# ROC function
# arguments: dataframe, actual class values, model class probabilities,
#            value in df indicating "true" (i.e. event took place),
#            number of digits to display in answer

myROC <- function(df,acol,prob,truval=1, digits = 3) {

  # actual class values, adjust actual values to 1s and 0s if data is text
  acol <- eval(substitute(acol),df, parent.frame())
  acol <- ifelse(acol == truval,1,0)

  # model class probabilities
  prob <- eval(substitute(prob),df, parent.frame())

  # define sensitivity, complement of specificity, and area
  senst <- 1
  spec_c <- 1
  area_vec <- numeric()

  for (i in seq(0.01,1,0.01)) {
    class_predict <- ifelse(prob > i, 1, 0)
    senst <- c(senst, sens(df,class_predict,acol,digits=5))
    spec_c <- c(spec_c, 1- spec(df,class_predict,acol,digits=5))

    # calculate area of rectangle using midpoint method
    area_vec <- c(area_vec,(0.5*(senst[length(senst)] + senst[length(senst)-1])) *
      (spec_c[length(spec_c)-1]-spec_c[length(spec_c)]))

  }

  # make plot
  p <- plot(spec_c,senst, xlim = c(0, 1), ylim = c(0, 1), type = "l",
         xlab = "1 - Specificity", ylab = "Sensitivity", col = 'blue',
         main="ROC Curve")
     abline(0,1, col="black")

  # return list containing plot, and area under curve
  list(plot=p, AUC=round(sum(area_vec), digits))

}
```

11. Classification Metrics Output

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

```r
# classification metrics
a <- acc(mydata,scored.class, class) # accuracy
e <- err_rt(mydata, scored.class, class)  # error rate
p <- prec(mydata, scored.class, class) # precision
st <- sens(mydata, scored.class, class) # sensitivity
sp <- spec(mydata, scored.class, class) # specificity
f1 <- F1(mydata, scored.class, class) # F1
```

```r
r <- myROC(mydata,class,scored.probability) # ROC Curve and AUC
```



```r
# display metrics
list(accuracy=a, error_rt = e, precision=p, sensitivity=st, specificity=sp, F1=f1, AUC=r$AUC)
```

```
$accuracy
[1] 0.807

$error_rt
[1] 0.193

$precision
[1] 0.844

$sensitivity
[1] 0.474

$specificity
[1] 0.96

$F1
[1] 0.607

$AUC
[1] 0.849
```

12. Caret Package

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?

```r
library(caret)

# confusion matrix
cm <- confusionMatrix(data=mydata$scored.class,
                      reference=mydata$class, positive='1')

# print confusion matrix
knitr::kable(cm$table)
```

	0	1
0	119	30
1	5	27

```r
# calculate metrics using caret package
a <- cm$overall["Accuracy"]; names(a) <- NULL # accuracy
e <- 1 - a; names(e) <- NULL # error rate
p <- cm$byClass["Precision"]; names(p) <- NULL  # precision
st <- cm$byClass["Sensitivity"]; names(st) <- NULL  # sensitivity
sp <- cm$byClass["Specificity"]; names(sp) <- NULL # specificity
f1 <- cm$byClass["F1"] ; names(f1) <- NULL # F1

# display metrics
list(accuracy=a, error_rt = e, precision=p, sensitivity=st, specificity=sp, F1=f1)
```

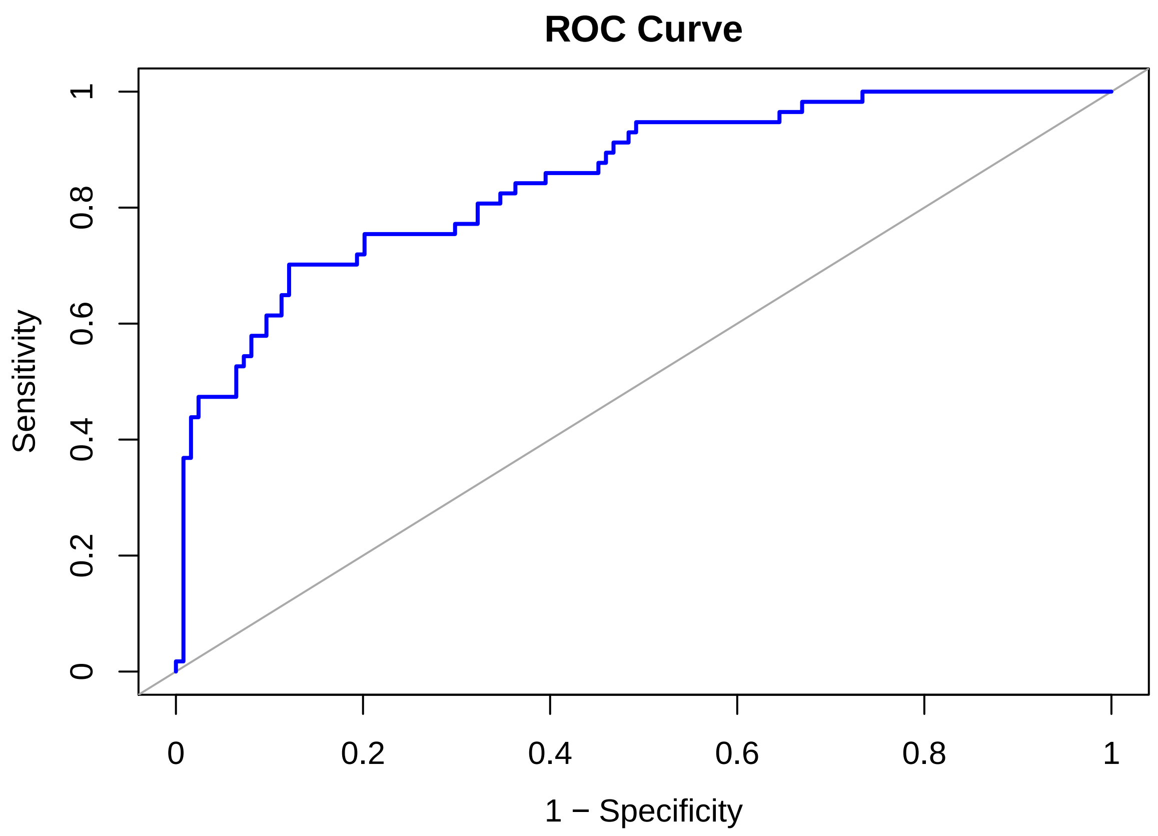
```
$accuracy
[1] 0.8066298

$error_rt
[1] 0.1933702

$precision
[1] 0.84375

$sensitivity
[1] 0.4736842

$specificity
[1] 0.9596774

$F1
[1] 0.6067416
```

All of the metrics calculated in the caret package have values that are consistent with our manually calculated values.

13. pROC Package

Investigate the *pROC* package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```r
library(pROC)

# plot ROC curve
myroc <- roc(mydata$class, mydata$scored.probability, plot=T, asp=NA,
                legacy.axes=T, main = "ROC Curve", ret="tp", col="blue")
```



```r
# calculate area under curve
myroc["auc"]
```

```
$auc
Area under the curve: 0.8503
```

The ROC curve and AUC values from the pROC package are consistent with our earlier, manual calculations.