hw2-cuny-data621

Business Analytics and Data Mining

Ohannes (Hovig) Ohannessian (ohannes.ohannessian16@spsmail.cuny.edu)

6/24/2018

Contents

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

Question 1

Download the classification output data set (attached in Blackboard to the assignment). Load and analyze dataset.

```
df = read.csv('https://raw.githubusercontent.com/hovig/MSDS_CUNY/master/DATA621/hw2/classification-output-data
attach(df)
```

Question 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

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?

To answer:

- class is the actual class of the dataset.
- Scored class is the model predicted class and
- Probability is the model predicted probability.
- The columns represent "predicted" values and the rows represent the "actual".

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

```
##
##       0   1
##   0 119   5
##   1  30  27
```

Question 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 = (TP + TN) / (TP + FP + TN + FN).

```
prediction_accuracy <- function(df){
        names    = c("class", "scored.class")
        columns  = df[colnames(df) %in% names]
        t        = table(columns)
        accuracy = (t[2,2] + t[1,1]) / (t[2,2] + t[1,2] + t[1,1] + t[2,1])
        accuracy = round(accuracy, 2)
        paste("The prediction accuracy is: ", accuracy)
}
```

Question 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.

Classification Error Rate = (FP + FN) / (TP + FP + TN + FN).

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

```
classification_error_rate <- function(df){
        names   = c("class", "scored.class")
        columns = df[colnames(df) %in% names]
        t       = table(columns)
        classification_error_rate = (t[1,2] + t[2,1]) / (t[2,2] + t[1,2] + t[1,1] + t[2,1])
        classification_error_rate = round(classification_error_rate, 2)
        paste('The classification error rate is: ',classification_error_rate)
}
```

Question 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 = TP / (TP + FP).

```
precision <- function(df){
        names   = c("class", "scored.class")
        columns = df[colnames(df) %in% names]
        t       = table(columns)
        p       = (t[2,2] / (t[2,2] + t[1,2]))
        p       = round(p, 2)
        paste('The precision is: ', p)
}
```

Question 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 = TP / (TP + FN).

```
sensitivity <- function(df){
        names   = c("class", "scored.class")
        columns = df[colnames(df) %in% names]
        t       = table(columns)
        s       = t[2,2] / (t[2,2] + t[2,1])
        s       = round(s,2)
        paste('The sensitiity is: ',s)
}
```

Question 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. Precision = TN / (TN + FP)

```
specificity <- function(df){
        names   = c("class", "scored.class")
        columns = df[colnames(df) %in% names]
        t       = table(columns)
        s       = t[1,1] / (t[1,1] + t[1,2])
        s       = round(s,2)
        paste('The specificity is: ',s)
}
```

Question 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. F1 Score = (2 * Precision * Sensitivity) / (Precision + Sensitivity)

```
f1.score <- function(df){
        names   = c("class", "scored.class")
        columns = df[colnames(df) %in% names]
        t = table(columns)
        p = (t[2,2] / (t[2,2] + t[1,2]))
        s = t[2,2] / (t[2,2] + t[2,1])
        f1 = (2*p*s) / (p + s)
        f1 = round(f1, 2)
        paste("The F1 score is: ", f1)
}
```

Question 9

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.

Maximum senstivitiy occurs if False Negatives are zero and sensitivity is equal to 1. Minimum sensitivity is equal to zero where are no true positives. Maximum precision occurs if False Positives are zero and precision is equal to 1. Minimum precision occurs when there are zero true positives.

F1 will always fall between 0 and 1. Since precision and sensitivity will always be a number between 0 and 1 (because they are derived from always dividing a smaller number by a bigger number), their product will be smaller than both. The denomimator in the F1 score is proportional to the numerator so even if the product of sensitivity and precision times 2 is higher, the denominator will be bigger resulting in a number between 0 and 1 and can be equal to 0 or 1.

Question 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.

The ROC curve is a way to visualize model performance. Each point of the ROC curve consists of a "truth" table, calculated for a given threshold of the scored probability.

Typically, for a ROC curve, the y axis represents the true positive rate or sensitivity. The x axis represents the false positive rate or (1 - specificity).

```
get.rocPlot.auc = function(df){

threshold.grid = seq(0,1, by= 0.01)
predicted.classes = data.frame(row.names = 1:nrow(df))
x = seq_along(threshold.grid)
y = seq_along(threshold.grid)
for (i in threshold.grid) {
        yhat = as.numeric(df$scored.probability>i)
        predicted.classes = cbind(predicted.classes, yhat)
}
for (j in 1:length(threshold.grid)){
                class.factor = factor(df$class,levels = c(0,1))
                predicted.class.factor = factor(predicted.classes[,j], levels = c(0,1))
                t = table(class.factor, predicted.class.factor)
                sensitivity = t[2,2] / (t[2,2] + t[2,1])
                specificity = t[1,1] / (t[1,1] + t[1,2])
```

```
                y[j] = sensitivity
                x[j] = 1 - specificity
}

roc.data = data.frame(fpr = x, tpr = y)
roc.plot = ggplot(roc.data, aes(x=fpr, y=tpr)) + geom_step()
roc.plot = roc.plot + geom_abline(slope = 1, intercept = c(0,0), colour="red", lty=2)

my_auc <- function(outcome, prob){
 N = length(prob)
 N_pos = sum(outcome)
 df = data.frame(out = outcome, prob = prob)
 df = df[order(-df$prob),]
 df$above = (1:N) - cumsum(df$out)
 return( 1- sum( df$above * df$out ) / (N_pos * (N-N_pos) ) )
}

auc1 = my_auc(df$class,df$scored.probability)
results = list("Plot"=roc.plot, "Area under curve"=auc1)
results
}
```

Question 11

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

```
prediction_accuracy(df)
```

```
## [1] "The prediction accuracy is:  0.81"
```

```
classification_error_rate(df)
```

```
## [1] "The classification error rate is:  0.19"
```

```
precision(df)
```

```
## [1] "The precision is:  0.84"
```

```
sensitivity(df)
```

```
## [1] "The sensitiity is:  0.47"
```

```
specificity(df)
```

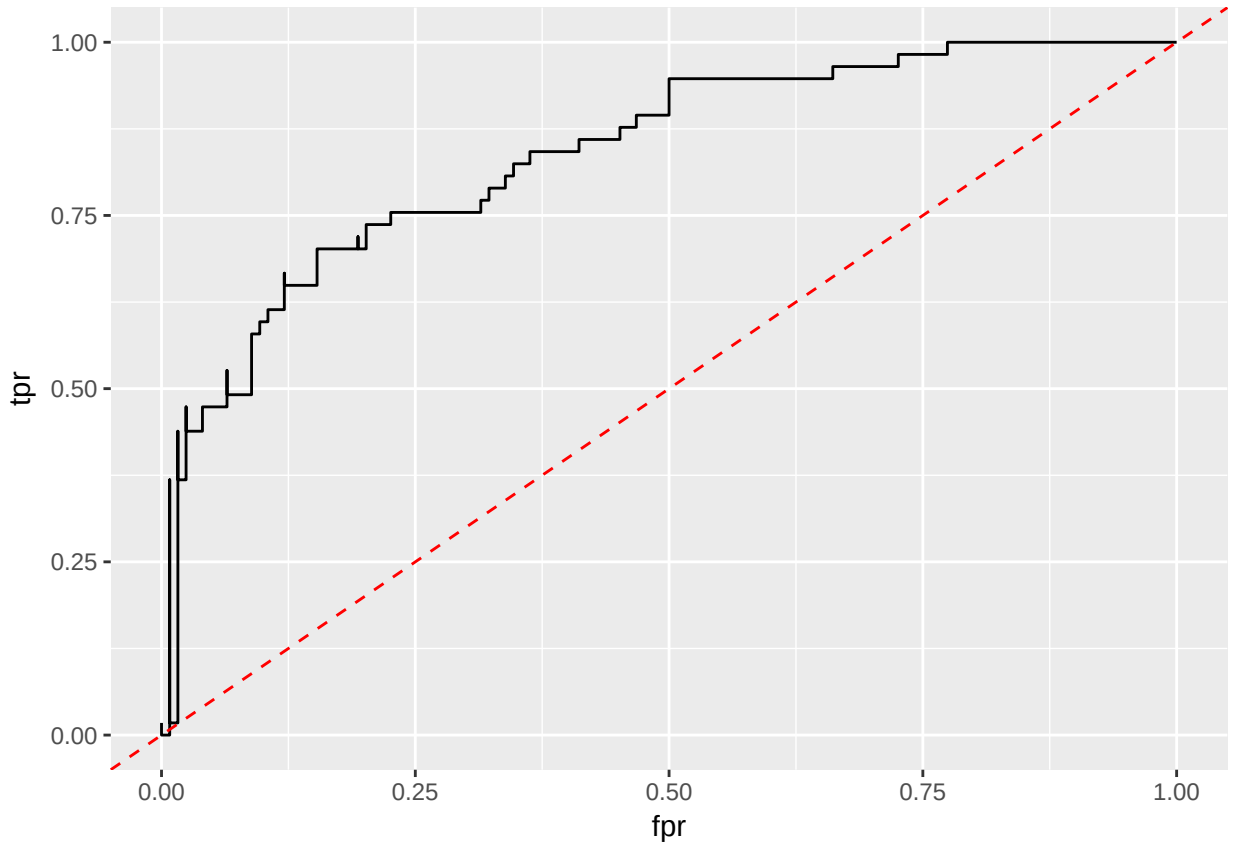
```
## [1] "The specificity is:  0.96"
```

```
f1.score(df)
```

```
## [1] "The F1 score is:  0.61"
```

```
get.rocPlot.auc(df)
```

```
## $Plot
```




```
##
## $`Area under curve`
## [1] 0.8503113
```

Question 12

Investigate the caret package. In particular, consider the functions confusionMatrix, sensitivity, and specificity. Apply the functions to the data set.

```
pred.factor <- factor(df$scored.class,levels = c(1,0))
actual.factor <- factor(df$class, levels = c(1,0))
confusionMatrix(pred.factor, actual.factor)
```

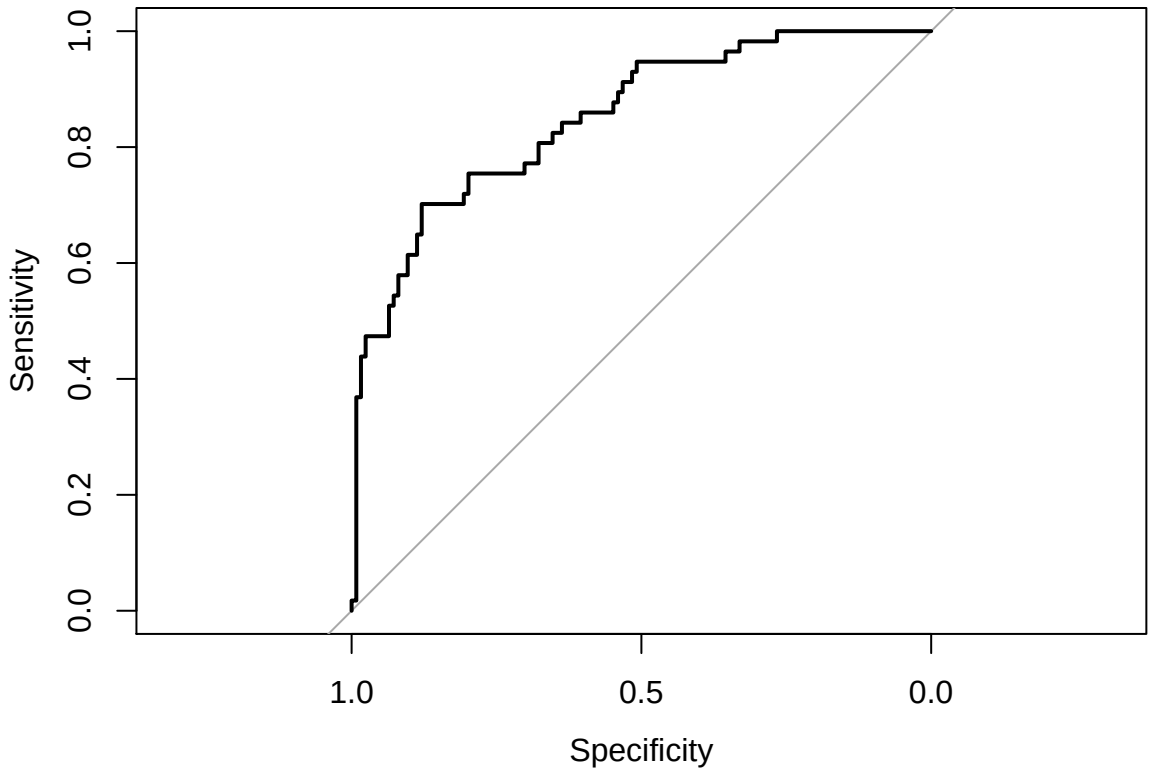
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   0
##          1  27   5
##          0  30 119
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
##
##                   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
##
```

Question 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?

```
plot.roc(class,scored.probability)
```



The plots are quite similar to what has been developed by above.