Classification Metric Analysis

FY 2016

Author Eric Lewis

TABLE OF CONTENTS

Contents

Introduction

The objective of this machine learning analysis is to generate classification for a classification output data set. Functions are created to evaluate the actual and predicted calculations for the following: accuracy, classification error rate. precision, sensitivity, specificity, F1 score, and ROC curve. The preceding results will be compared with the caret and pROC packages as a comparison. This analysis is outlined as follows: introduction, analysis, results, conclusion, and an appendix of R code.

Analysis

This analysis begins with an initial examination of the classification output dataset, which contains 181 observations, including eleven variables. Pregnant, glucose, diastolic, skinfold. insulin, bmi, pedigree, age, class, scored.class, and scored.probability.

Confusion Matrix

The confusion matrix is created using the table function using the R language and the variables of class and scored.class. This matrix will demonstrate the number of correct and incorrect predictions within this classification model data as compared to the actual outcomes in the data. The results using the table function in R corresponds to the confusion matrix, table 1. The rows do represent the actual class, and the columns do represent the predicted class.

N = 181	Predicted: No {0}	Predicted: Yes {1}	
Actual: No {0}	TN = 119	FP = 5	124
Actual: Yes {1}	FN = 30	TP = 27	57
	149	32	

Table 1: Confusion Matrix

Results

Accuracy

The accuracy is the proportion of the total number of the predictions which are correctly classified. The accuracy is represented by the following formula, which corresponds to the confusion matrix in Table 1. The accuracy calculation is manually calculated which confirms the performance of the R accuracy function which returns the rounded value of 81%.

Accuracy = (TP + TN) / (TP + FP + TN + FN)

Accuracy = (27 + 119) / (27 + 5 + 119 + 30)

Accuracy = 0.8066 or **81%**

Classification Error Rate

The classification error rate is the proportion of the cases that the matrix provided either false positives or false negatives, otherwise known as type I and type II error. The classification error rate is represented by the following formula, which corresponds to the confusion matrix in Table 1. The classification error rate calculation is manually calculated which confirms the performance of the R classification error rate, (CER) function which returns the rounded value of 19%.

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

CER = (5 + 30) / (27 + 5 + 119 + 30)

CER = 0.1934 or **19%**

Validation: CER + Accuracy = 100%, 19% + 81% = **100%**

Precision

The precision is the proportion of the total number of positive cases which are correctly classified. The precision is represented by the following formula, which corresponds to the confusion matrix in Table 1. The precision calculation is manually calculated which confirms the performance of the R precision function which returns the rounded value of 84%.

$Precision = TP \ / \ (TP + FP)$

$Precision = 27 \ / \ (27 + 5)$

$\boldsymbol{Precision} = 0.8438$ or **84%**

Sensitivity

The sensitivity is the proportion of the total number of the actual positive cases which are correctly classified. The sensitivity is represented by the following formula, which corresponds to the confusion matrix in Table 1. The sensitivity calculation is manually calculated which confirms the performance of the R sensitivity function which returns the rounded value of 47%.

$$Sensitivity = TP \;/\; (TP + FN)$$

$$Sensitivity = 27 \;/\; (27 + 30)$$

$$\boldsymbol{Sensitivity} = 0.4737 \text{ or } \mathbf{47\%}$$

Specificity

The specificity is the proportion of the total number of the actual negative cases which are correctly classified. The specificity is represented by the following formula, which corresponds to the confusion matrix in Table 1. The specificity calculation is manually calculated which confirms the performance of the R specificity function which returns the rounded value of 96%.

$$Specificity = TN \;/\; (TN + FP)$$

$$Specificity = 119 \;/\; (119 + 5)$$

$$\boldsymbol{Specificity} = 0.9597 \text{ or } \mathbf{96\%}$$

F1 Score

The F1 Score is the proportion of the weighted average of precision and sensitivity. A F1 score in the range 40% thru 80% is considered a good measure to achieve the maximum values for the classifiers, and this F1 analysis yielded a F1 score of 61%. The F1 score is represented by the following formula, which corresponds to the confusion matrix in Table 1. The F1 score calculation is manually calculated which confirms the performance of the R F1 function which returns the rounded value of 61%.

$$F1\ Score = (2 \times Precision \times Sensitivity) \;/\; (Precision + Sensitivity)$$

$$F1\ Score = (2 \text{ x } 0.8437 \text{ x } 0.4737) \;/\; (0.8437 + 0.4737)$$

$$F1\ Score = 0.7993 \;/\; 1.3174$$

$$\boldsymbol{F1\ Score} = 0.6067 \text{ or } \mathbf{61\%}$$

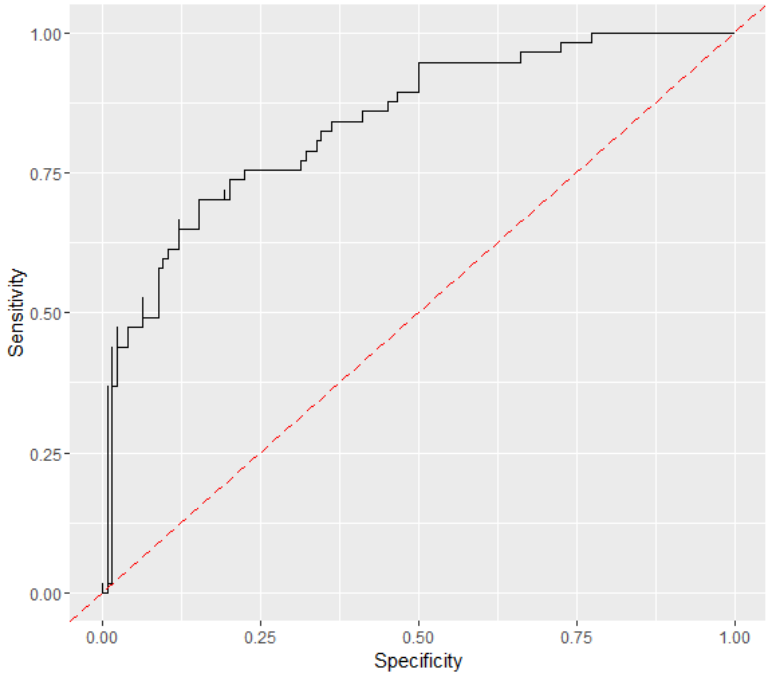
F1 Score Bounds

The bounds of the F1 score will always be greater than 0 and less than or equal to 1 based upon the following:

- Precision and sensitivity are the basis for the F1 score calculation
- Precision and sensitivity will always fall between zero and one because they have smaller numerators than denominators.
- The numerator of the F1 score will always be lower than the denominator based upon the product of precision and sensitivity is always going to be less than the denominator of the addition of precision and sensitivity.

ROC Curve

The area under the ROC curve, AUC is being utilized to measure the quality of this classification model. The red dotted line represents a random classifier which has an area under the curve of 0.5. Classification models will have an AUC range from 0.5 thru 1.0, the greater the AUC, the higher the measure of quality of a given classification model. The ROC curve function returned an area under the curve of 0.8503, which indicates this current model will perform better than a randomly chosen case from the group where the target is between 0 and 85%.



Plot 1: ROC Curve

Caret Package

The Caret package, classification and regression training which is a set of functions that streamline the process for creating predictive models. This analysis is utilizing the confusion matrix function, which generates the confusion matrix and statistics. The functions created previously within this analysis are validated as compared in the results of the Confusion Matrix generated by the Caret Package.

Calculation	Function	Caret Package
Accuracy	0.8066	0.8066
Classification Error Rate	0.1934	
Precision	0.8438	0.8438
Sensitivity	0.4737	0.4737
Specificity	0.9597	0.9597
F1 Score	0.6067	

Table 1: Function vs Caret Package

```
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
```

Table 2: Caret Confusion Matrix

pROC Package

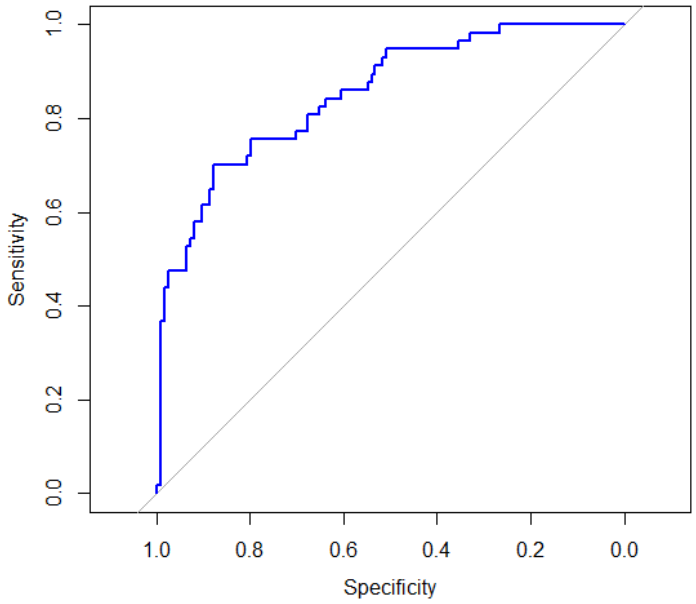
The pROC package contains tools which may be utilized for visualizing, smoothing, and comparing receiver operating characteristic, ROC. The partial area under the curve, AUC may be compared using statistical based testing. The functions created previously within this analysis are validated as compared in the results of the pROC Package generated in table 3, area under the curve: **0.8503**.

```
Call:
plot.roc.default(x = preds$class, predictor = preds$scored.probability)

Data: preds$scored.probability in 124 controls (preds$class 0) < 57 cases
(preds$class 1).

Area under the curve: 0.8503
```

Table 3: pROC Package Results



Plot 2: ROC Curve

Conclusion

The results using the classification output data set demonstrates the objective of this analysis is met through creating individual functions to evaluate the actual and predicted calculations for the following: accuracy, classification error rate. precision, sensitivity, specificity, F1 score, and ROC curve which results match the results of using the caret and pROC packages in R, as demonstrated in table 4 summaries of comparisons.

Calculation	Function	Caret Package	pROC Package
Accuracy	0.8066	0.8066	
Classification Error Rate	0.1934		
Precision	0.8438	0.8438	
Sensitivity	0.4737	0.4737	
Specificity	0.9597	0.9597	
F1 Score	0.6072		
AUC	0.8503		0.8503

Table 4: Summaries

Appendix

 # (1) Load the classification data

```
predict = read.csv('classification-output-data.csv', sep=',', header = TRUE)
attach(predict)
table(class,scored.class)
```

 # (2) Show contents, summary of data

```
summary(predict)
str(predict)
sum(is.na(predict))
```

 # (3) Function Accuracy

```
accuracy <- function(df){
      nm = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol  = table(col)
      result = (tcol[2,2] + tcol[1,1]) / (tcol[2,2] + tcol[1,2] + tcol[1,1] + tcol[2,1])
      return(result)
}
accuracy(predict)
```

 # (4) Function Classification Error Rate

```
cer <- function(df){
      nm   = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol  = table(col)
      result = (tcol[1,2] + tcol[2,1]) / (tcol[2,2] + tcol[1,2] + tcol[1,1] + tcol[2,1])
      return(result)
}
cer(predict)
```

 # (5) Function Precision

```
precision <- function(df){
      nm   = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol = table(col)
      result = (tcol[2,2] / (tcol[2,2] + tcol[1,2]))
      return(result)
}
precision(predict)
```

(6) Function Sensitivity

```
sensitivity <- function(df){
      nm  = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol  = table(col)
      result  = (tcol[2,2] / (tcol[2,2] + tcol[2,1]))
      return(result)
}
sensitivity(predict)
```

(7) Function Specificity

```
specificity <- function(df){
      nm  = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol = table(col)
      result = (tcol[1,1] / (tcol[1,1] + tcol[1,2]))
      return(result)
}
specificity(predict)
```

(8) F1 Score

```
f1score <- function(df){
      nm = c("class", "scored.class")
      col = df[colnames(df) %in% nm]
      tcol = table(col)
      prczn = (tcol[2,2] / (tcol[2,2] + tcol[1,2]))
      spfcty = tcol[2,2] / (tcol[2,2] + tcol[2,1])
      result = (2*prczn*spfcty) / (prczn + spfcty)
      return(result)
}
f1score(predict)
```

 # (10) ROC Curve

```
rocCurve = function(df){
library(ggplot2)
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))
            tcol = table(class.factor, predicted.class.factor)
            sensit = (tcol[2,2] / (tcol[2,2] + tcol[2,1]))
            spfcty = (tcol[1,1] / (tcol[1,1] + tcol[1,2]))
            y[j] = sensit
            x[j] = 1 - spfcty
}
roc.data = data.frame(Specificity= x, Sensitivity = y)
roc.plot = ggplot(roc.data, aes(x= Specificity, y= Sensitivity)) + geom_step()
roc.plot = roc.plot + geom_abline(slope = 1, intercept = c(0,0), col="red", lty=5)
aucCalc <- function(outcome, prb){
 lngth = length(prb)
 lngth_pos = sum(outcome)
 df = data.frame(out = outcome, prob = prb)
 df = df[order(-df$prob),]
 df$above = (1:lngth) - cumsum(df$out)
 return( 1- sum( df$above * df$out ) / (lngth_pos * (lngth-lngth_pos) ) )
}
auc =  aucCalc(predict$class,predict$scored.probability)
result = list("Plot"=roc.plot, auc)
return(result)
}
rocCurve(predict)
```

(12) Caret Package

```
library(caret)
pred.factor = factor(predict$scored.class,levels = c(1,0))
actual.factor = factor(predict$class, levels = c(1,0))
confusionMatrix(pred.factor, actual.factor)
```

(13) pROC Package

```
library(pROC)
plot.roc(predict$class,predict$scored.probability, col="blue")
```