D621_HW2_ClassificationModel

Coffy Andrews-Guo

2022-10-02

Class Predictions

Classification models produce a continuous valued predication, that is usually in the form of a probability focusing on discrete prediction rather than continuous prediction. The classification model will predict values of class membership for any individual sample between 0 and 1 and sum to 1.

1. Download the classification output data set.

Load data set Loading `classification-output-data.csv` file from Github repositories.

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

The `table()` function creates a `raw confusion`tabular summary of the 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?

The raw confusion matrix is a simple table used to summarise the performance of a classification algorithm. The concept of a confusion matrix involves: **false positives** and **false negatives**. The rows represents the **actual** class and the columns represents the **predicated** class.

Raw Confusion Tubular Summary

The confusion matrix for a binary classifier:

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

The `table() function` (raw confusion matrix) has made a total of **181** predictions made (146 are correct and 35 are wrong).

The table shows the observations as a binary classifer where: `0-0` is `TRUE POSITIVE (TP)`, `0-1` is `FALSE POSITIVE (FP)`, `1-0` is `FALSE NEGATIVE (FN)`, and `1-1` is `TRUE NEGATIVE (TN)` : - there were 119 cases in which the algorithm predicted a 0 and the actual label was 0 (correct). - there were 5 cases in which the algorithm predicted a 1 and the actual label was 0 (incorrect). - there were 30 cases in which the algorithm predicted a 0 and the actual label was 1 (incorrect). - there were 27 cases in which the algorithm predicted a 1 and the actual label was 1 (correct)

We can interpret `0` as being negative, while `1` as being positive.

```r
#defining some basic variables that will be needed to compute evaluation metrics.
n = sum(raw_cm)    # number of instances
nc = nrow(raw_cm)  # number of classes
diag = diag(raw_cm)  #number of correctly classified instances per class
rowsums = apply(raw_cm, 1, sum) #number of instances per class
colsums = apply(raw_cm, 2, sum) # number of predictions per class
p = rowsums / n    # distribution of instances over the actual class
q = colsums / n    # distribution of instances over the predicted classes
```

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.

The **accuracy** of the confusionMatrix is the proportion of correct classifications and calculated using the formula:

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

The accuracy score reads as ", 80.66,"% for the given data and observations."

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.

The **total classification error rate,** calculates the percentage of total incorrect classifications made by the model as follows:

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

The *total classification error* rate reads as ", 19.34,"% for the given data and observations.

In this case, the error rate is low and indicates that the model has a high success prediction rate on `TRUE POSITIVE` and `FALSE POSITIVE`.

The summation on *accuracy score* and the *total classification error rate* is ", 100,"% or ", 1," for the given data and observations.

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 is the proportion of correct predictions given the prediction was positive:

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

The metric of precision is important in cases when you want to minimise false positives and maximise true positives. The binary classifier will predict whether values are negative (0) or positive (1). In the task we will look at the values of the positive class (1) for the reporting metrics.

The positive class **precision** rate on correct predictions is ", 79.87, 84.38,"% for the given data and observations.

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 (or recall) is the proportion of correct predictions given that the actual labels are positive:

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

```
## [1] 0.4736842
```

In this task we will look at the values of the positive class (1) for the reporting metrics.

```
##         0         1
## 0.9596774 0.4736842
```

The positive class **sensitivity rate (recall)** of the predictions is able to correctly detect ", 47.37,"% of all fraudulent cases in the given data and observations.

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 represents the proportion of correct predictions for negative labels:

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

```
## [1] 0.9596774
```

In this task we will look at the values of the negative class (0) for the reporting metrics.

```
##         0         1
## 0.9596774 0.4736842
```

The **specificity** rate of the predictions reads as ", 95.97,"% for the given data and observations.

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.

F-score is a metric that combines both the precision and sensitivity (recall) into a single value:

$$F = \frac{2 * Precision * Sensitivity(Recall)}{Precision + Sensitivity(Recall)}$$

In this binary classification task, we will look at the values of the positive class (1) for reporting metrics.

```
##         0         1
## 0.8717949 0.6067416
```

The positive class F1-score of the predictions reads as ", 60.67,"% for the given data and observations. An F1-score in the range of 0.5 - 0.8 is considered a good performance measure for a classification model to predict each observation correctly.

9. 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.)

F1-score represents the harmonic mean of the precision and recall. The bounds on the **F1-score** range from 0.47 to 0.84, with 1 representing a model that perfectly classifies each observation into the correct class and 0 representing a model that is unable to classify any observation in to the correct class.

In the `raw confusion matrix` model, we can show that the F-1 score is always between 0 and 1 as follows:

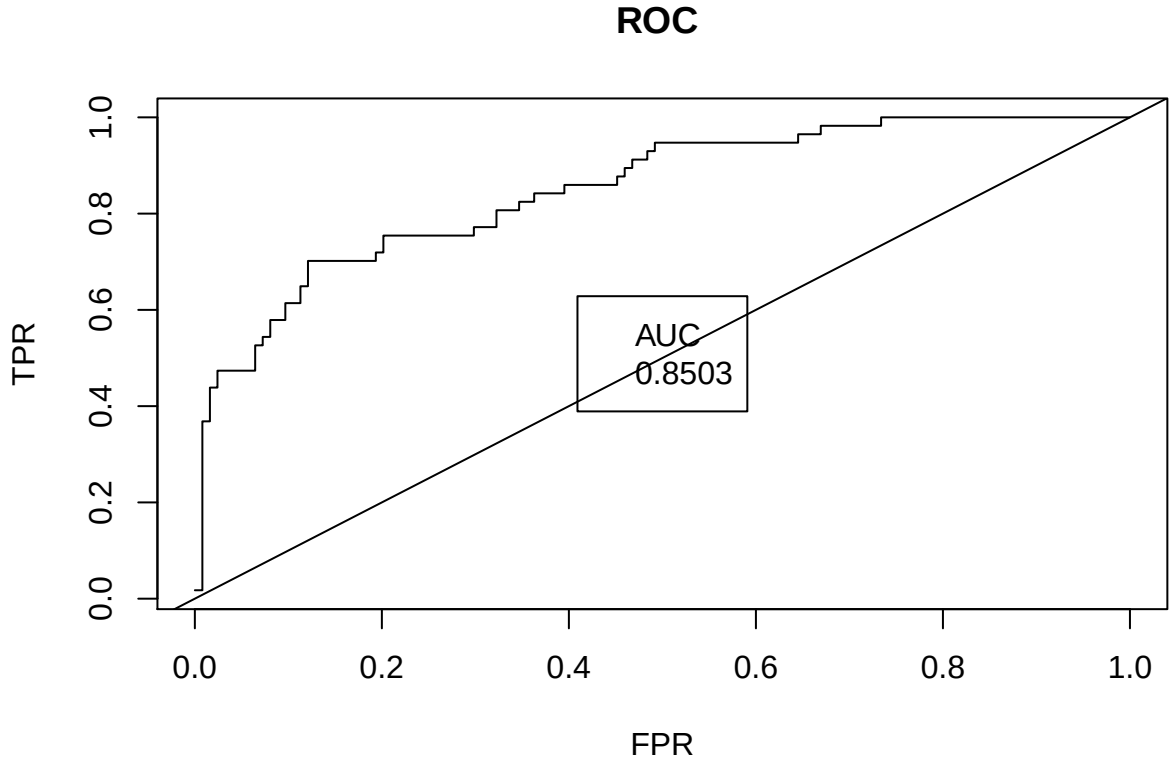
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{27}{27+5} = \frac{27}{32} = 0.84$$

$$Recall = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{27}{27+30} = \frac{27}{57} = 0.47$$

Now, we will calculate the F1 Score for each threshold 0.01, 0.02, 0.03,... 0.99. The threshold that gives the optimal cutoff (optimal F1 Score) is:

```
## [1] 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.



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

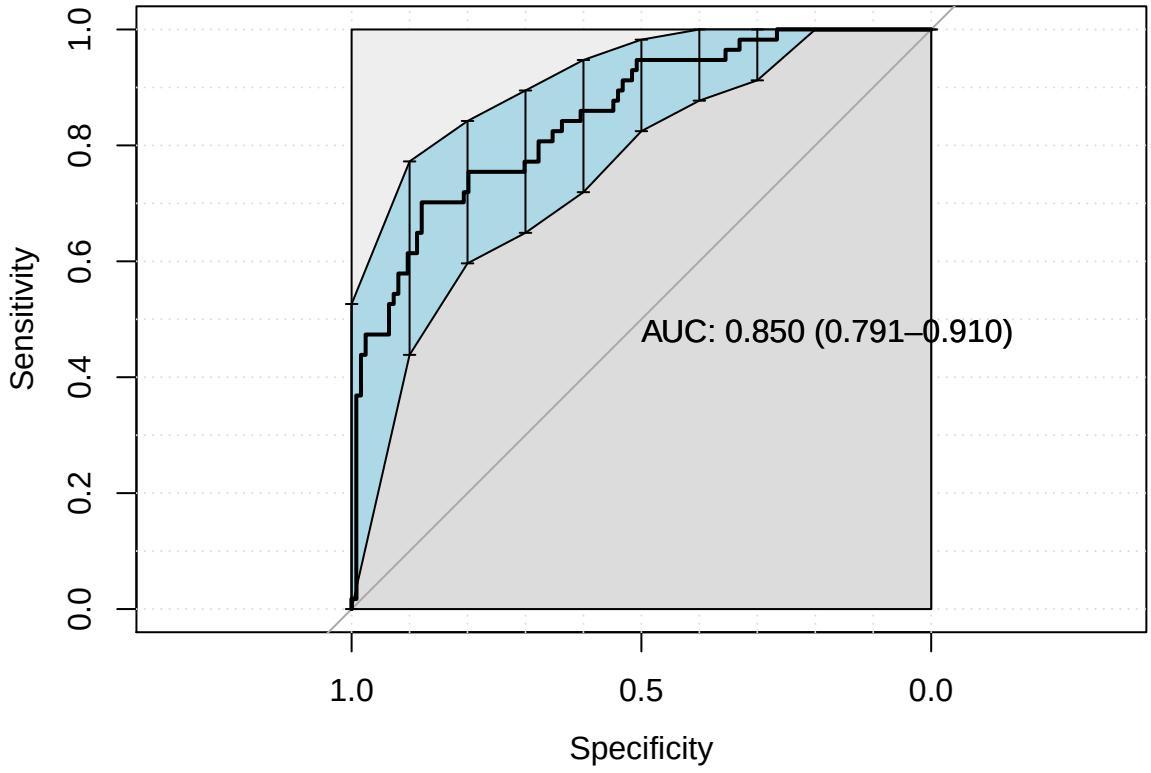
```
##    accuracy precision   recall.   specif.    fscore
## 0 0.8066298 0.7986577 0.9596774 0.9596774 0.8717949
## 1 0.8066298 0.8437500 0.4736842 0.4736842 0.6067416
```

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?

Convert data value and reference value to factors with the same levels.

```
## 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
##               Precision : 0.8438
##                  Recall : 0.4737
##                      F1 : 0.6067
##              Prevalence : 0.3149
##          Detection Rate : 0.1492
##    Detection Prevalence : 0.1768
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 1
##
```

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?



Appendix: All code for this report

```
knitr::opts_chunk$set(fig.pos = 'h')
knitr::opts_chunk$set(echo = FALSE)
#load required libraries
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library("InformationValue")) #measure predictive capability
suppressPackageStartupMessages(library(cutpointr))
suppressPackageStartupMessages(library(MLmetrics))
suppressPackageStartupMessages(library("caret"))
suppressPackageStartupMessages(library("pROC"))
#df <- read.csv("url", encoding = "utf-8")
df <- read.csv("classification-output-data.csv")
#matrix = confusion_matrix(true = actual_labels, pred = predicted_labels)
raw_cm <- as.matrix(table(Actual = df$class, Predicted = df$scored.class)) #created the confusion matri
raw_cm
#defining some basic variables that will be needed to compute evaluation metrics.
n = sum(raw_cm)     # number of instances
nc = nrow(raw_cm)  # number of classes
diag = diag(raw_cm)  #number of correctly classified instances per class
rowsums = apply(raw_cm, 1, sum) #number of instances per class
```

```
colsums = apply(raw_cm, 2, sum) # number of predictions per class
p = rowsums / n    # distribution of instances over the actual class
q = colsums / n    # distribution of instances over the predicted classes
Accuracy = (119+27) / (119+5+30+27)
accuracy = sum(diag) / n
#accuracy
Error_rate = (5+30)/(119+5+30+27)
#Error_rate
error_rate = 1 - (sum(diag)/n)
#error_rate
Precision = 27/(5+27)
#Precision
#The precision contains 2 values corresponding to the classes 0, and 1.
#In binary classification tasks, we will look at the values of the positive class (1) for reporting met
precision = diag / colsums
#precision
#Sensitivity = TP/(TP+FN) or TP/Overall Positives
recall = (27)/(30+27)
recall
#The sensitivity/recall contains 2 values corresponding to the classes 0, and 1.
#In binary classification tasks, we will look at the values of the positive class (1) for reporting met
recall. = diag / rowsums
recall.
specif = (119/(119+5))
specif
specif. = diag / rowsums
specif.
fscore = 2 * precision * recall. / (precision + recall.)
fscore
f1Scores <- sapply(seq(0.01, 0.99, .01), function(thresh) F1_Score(df$class, ifelse(df$scored.class >= t

which.max(f1Scores)
ROC <- function(x, y){
  x <- x[order(y, decreasing = TRUE)]
  TPR <- cumsum(x) / sum(x)
  FPR <- cumsum(!x) / sum(!x)
  df <- data.frame(TPR, FPR, x)

  FPR_df <- c(diff(df$FPR), 0)
  TPR_df <- c(diff(df$TPR), 0)
  area_under_curve <- sum(df$TPR * FPR_df) + sum(TPR_df * FPR_df)/2

  plot(df$FPR, df$TPR, type = "l",
       main = "ROC ",
       xlab = "FPR",
       ylab = "TPR")
  abline(a = 0, b = 1)
  legend("center", legend= c("AUC", round(area_under_curve, 4)))

}
ROC(df$class,df$scored.probability)
data.frame(accuracy, precision, recall., specif., fscore)
#create vectors having data points
```

```
actual_value <- factor(df$class)
predicted_value <- factor(df$scored.class)
set.seed(123)
#creating confusion matrix
cm <- confusionMatrix(data = as.factor(df$scored.class), reference = as.factor(df$class), mode = "every
cm
pROC_obj <- roc(df$class, df$scored.probability, smoothed = TRUE,
                # arguments for ci
                ci = TRUE, ci.alpha = 0.9, stratified = FALSE,
                # arguments for plot
                plot = TRUE, auc.polygon = TRUE, max.auc.polygon = TRUE, grid = TRUE,
                print.auc = TRUE, show.thres = TRUE)

sensi.ci <- ci.se(pROC_obj)
plot(sensi.ci, type = "shape", col = "lightblue")

plot(sensi.ci, type = "bars")
```