DATA 621 - Homework 2 - Classification Metric Calculations

Coffy Andrews-Guo, Krutika Patel, Alec McCabe, Ahmed Elsaeyed, Peter Phung

2022-10-09

Class Predictions

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

Data Set Exploration

There are three columns in the dataset that were used throughout this report for analysis:

- `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 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 **predicted** 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.

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 confusion matrix is the proportion of correct classifications and calculated using the formula:

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

Here we calculate the accuracy manually using the above definition for accuracy, and below we create a function that takes a data frame and calculates the True Positive and True Negative values and outputs the accuracy.

Using the output of the confusion matrix:

$$Accuracy = \frac{119 - 27}{119 + 5 + 30 + 27} = 0.8066$$

The accuracy score is 80.66% for the given data and observations. The R function that was used to generate the accuracy score is shown in the appendix.

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:

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

Similarly we manually calculate the error rate as defined above and then again have a function to do this given any dataframe.

Using the output of the confusion matrix:

$$ClassificationErrorRate = \frac{5 + 30}{119 + 5 + 30 + 27} = 0.1934$$

The *total classification error* rate is 19.34% for the given data and observations. The R function used to generate this value is shown in the appendix.

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 minimize false positives and maximize 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.

Using the output of the classification matrix:

$$Precision = \frac{27}{5 + 27} = 0.8438$$

The positive class **precision** rate on correct predictions is 84.38% for the given data and observations. The R function that was used to generate this value is shown in the Appendix section.

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. The sensitivity is also sometimes considered the *true positive rate* since it measures the accuracy in the event population. The equation for sensitivity is:

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

In this task we will look at the values of the positive class (1) for the reporting metrics. With that being said, after plugging in the values from the confusion matrix:

$$Sensitivity = \frac{27}{30 + 27} = 0.4737$$

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. The *false-positive rate* is defined as one minus the specificity:

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

In this task we will look at the values of the negative class (0) for the reporting metrics. With that being said, using the values from the confusion matrix:

$$Specificity = \frac{119}{119 + 5} = 0.9597$$

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. It is also known as the harmonic mean of the precision and sensitivity. It is expressed through the following expression:

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

F-score is a function of precision and sensitivity, which we have calculated previously as 0.84 and 0.47. In this binary classification task, we will look at the values of the positive class (1) for reporting metrics. With that being said, using the precision and recall values calculated earlier:

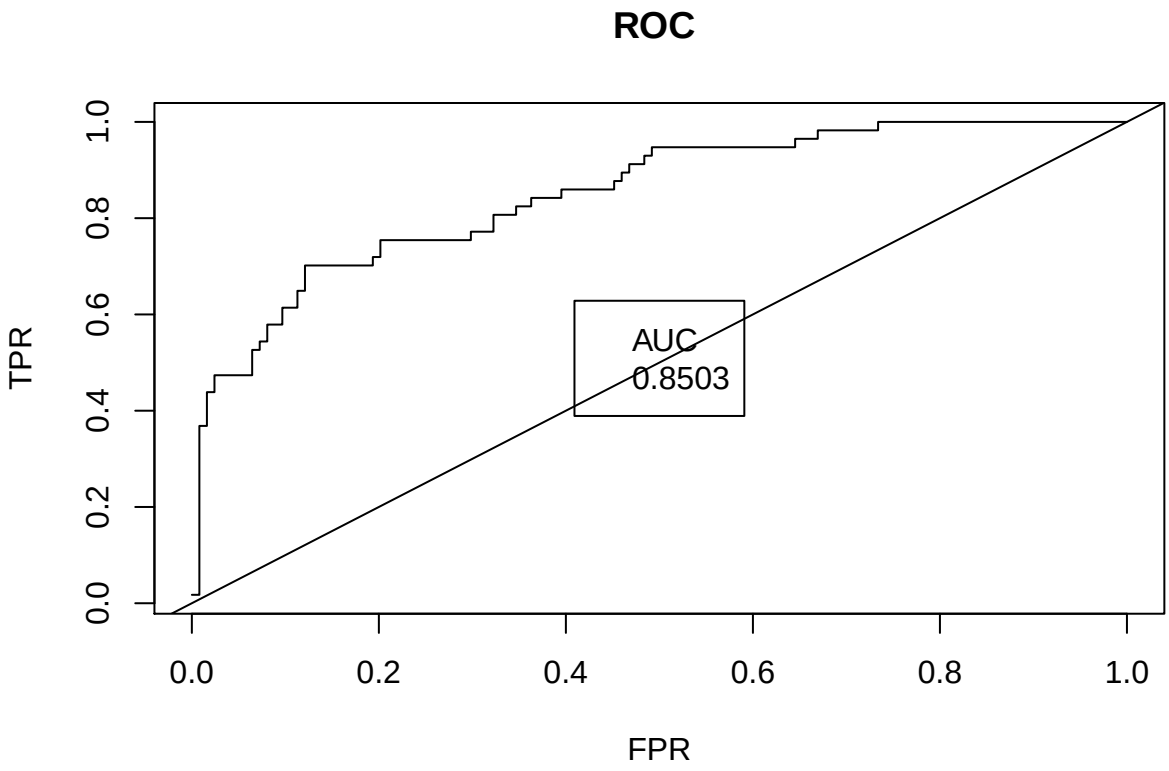
$$F = \frac{2 \times 0.84 \times 0.47}{0.84 \times 0.47} = 0.6068$$

The positive class F1-score of the predictions reads as 60.6767% 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.)

Based on the bounds for precision and sensitivity in the formula above ($0 < precision < 1$ and $0 < sensitivity < 1$) we can see that the numerator and denominator both have bounds of 0-2. Therefore the F-score itself will have bounds of 0 to 1. An F-score of 1 would indicate perfect precision and recall, while the lowest possible value is 0.

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.



`AUC` is a quantification how well a classification model does at classifying data. The area under the diagonal shown above would be 0.5. Since the ROC curve is above this diagonal, we can assume based on the data that the classification model is performing better than a model that performs random guessing. Hosmer and Lemeshow indicate that:

- 0.5 = No discrimination
- $0.5 - 0.7$ = Poor discrimination
- $0.7 - 0.8$ = Acceptable discrimination
- $0.8 - 0.9$ = Excellent discrimination
- > 0.9 = Outstanding discrimination

Therefore, the classification model results indicate excellent discrimination.

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

All of the created R functions are shown in the Appendix. The classification metrics are shown below.

accuracy_score	precision_score	recall_score	specif_score	fscore_score
0.8066	0.8438	0.4737	0.9597	0.6067675

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?

The `caret` package revealed the following values when using the data stored in the three key columns outlined in the *Data Set Exploration* section. The `confusionMatrix` function prints out not only the confusion matrix, but also the sensitivity and specificity, including other values.

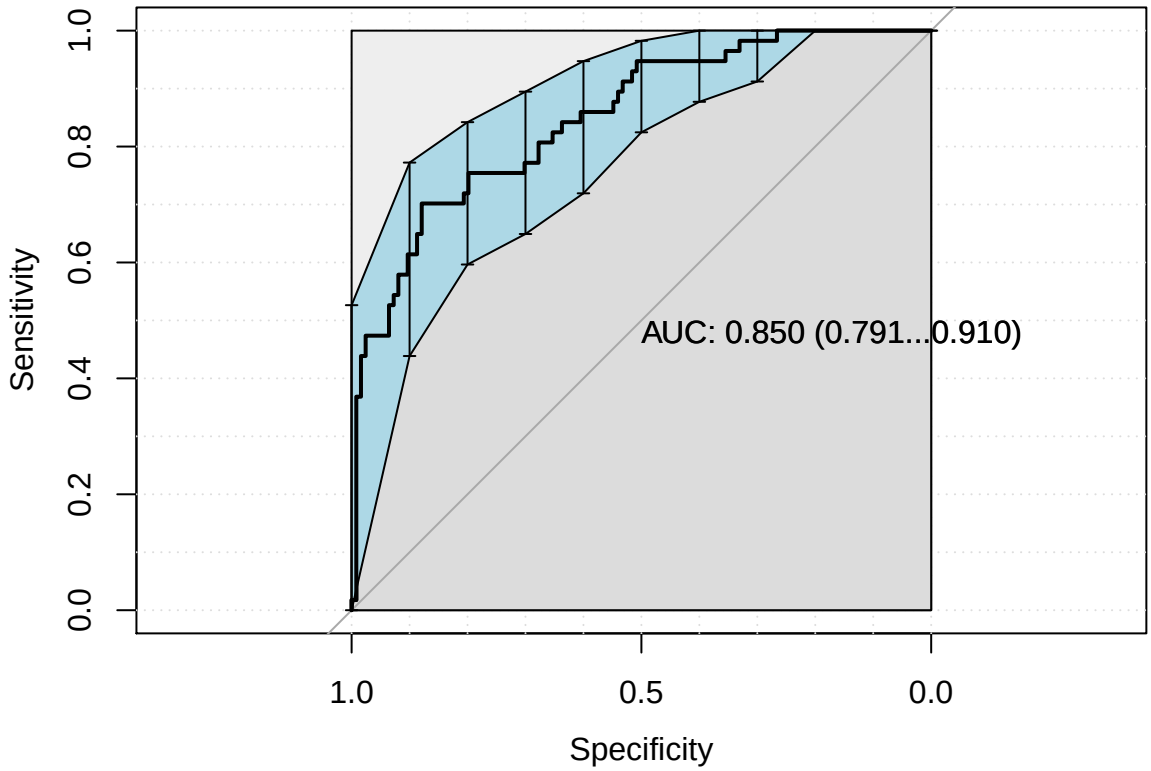
```
Confusion Matrix and Statistics

          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
```

In the context of binary classification, sensitivity is known alternatively as recall, while precision is known as the positive predictive value. Sensitivity comes from the world of medical tests; if one has a very *sensitive* test, than most or all of the people who have the disease from a pool will test positive. In the world of information retrieval, a query that retrieves all or most of the relevant documents is said to have a high *recall*. A modeling technique that "retrieves" most or all of the people that have a certain disease works the same way as a query that retrieves most or all of the relevant documents. This illustrates why the terms for `Sensitivity` and `Recall` are used interchangeably in the context of binary classification.

Positive predictive values are expressed as the proportion of positive identifications that were actually correct. Precision also uses this same definition, which is why the two values for `Pos Pred Value` and `Precision` are the same.

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?

Using the `pROC` package results in the ROC curve shown below. The blue area is the confidence interval for the ROC curve. Note that the calculated AUC is the same as the AUC that was calculated using the function that was written to generate the ROC curve in step 10.



Appendix: All code for this report

```
knitr::opts_chunk$set(fig.pos = 'h')
knitr::opts_chunk$set(echo = FALSE)
rm(list = ls())
if(!is.null(dev.list()))dev.off()

#Load required libraries
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library("InformationValue")) #measure predictive capability
suppressPackageStartupMessages(library(cutpointr))
suppressPackageStartupMessages(library(MLmetrics))
suppressPackageStartupMessages(library("caret"))
suppressPackageStartupMessages(library("pROC"))
```

```r
# Load data
#df <- read.csv("url", encoding = "utf-8")
data <- read.csv("classification-output-data.csv")

# Create confusion matrix
#matrix = confusion_matrix(true = actual_labels, pred = predicted_labels)

#creates the confusion matrix
raw_cm <- as.matrix(table(Actual = data$class, Predicted = data$scored.class))
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 calculation
Accuracy = (119+27) / (119+5+30+27)
Accuracy

# Accuracy Function
accuracy_predictions <- function(x){
  TP <- sum(x$class == 1 & x$scored.class == 1)
  TN <- sum(x$class == 0 & x$scored.class == 0)
  round((TP + TN)/nrow(x), 4)
}
accuracy_score = accuracy_predictions(data)
accuracy_score

## Classification error rate calculation
Error_rate = (5+30)/(119+5+30+27)
Error_rate

## Classification error rate function
error_predictions <- function(x){
  FP <- sum(x$class == 0 & x$scored.class == 1)
  FN <- sum(x$class == 1 & x$scored.class == 0)
  round((FP + FN)/nrow(x), 4)
}
error_prediction = error_predictions(data)
error_prediction

## Precision calculation
Precision = 27/(5+27)
Precision

## Precision function
##The precision contains 2 values corresponding to the classes 0, and 1.
##In binary classification tasks, we will look at the values of the
```

```r
##positive class (1) for reporting metrics.
precision <- function(x){
  TP <- sum(x$class == 1 & x$scored.class == 1)
  FP <- sum(x$class == 0 & x$scored.class == 1)

  round(TP/(TP+FP), 4)
}
precision_score = precision(data)

precision_score

## Sensitivity calculation
#Sensitivity = TP/(TP+FN) or TP/Overall Positives
recall = (27)/(30+27)
recall

## Sensitivity function
##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 metrics.
recall <- function(x){
  TP <- sum(x$class == 1 & x$scored.class == 1)
  FN <- sum(x$class == 1 & x$scored.class == 0)

  round(TP/(TP+FN), 4)
}
recall_score = recall(data)

recall_score

## Specificity calculation
specif = (119/(119+5))
specif

## Specificity function
specif2 <- function(x){
  TN <- sum(x$class == 0 & x$scored.class == 0)
  FP <- sum(x$class == 0 & x$scored.class == 1)

  round(TN/(TN+FP), 4)
}
specif_score = specif2(data)

specif_score

## F-score calculation
(2*0.84*0.47)/(0.84+0.47)

## F-score function
fscore <- function(x) {
  precision_score = precision(x)
  recall_score = recall(x)
  (2* precision_score * recall_score)/(precision_score + recall_score)
```

```
}

fscore_score = fscore(data)
fscore_score

## ROC function for step 10
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(data$class,data$scored.probability)

## Creating classification metrics dataframe
classification_metrics_df <- data.frame(accuracy_score,
                                        precision_score,
                                        recall_score,
                                        specif_score,
                                        fscore_score)
knitr::kable(classification_metrics_df)

##creating confusion matrix
set.seed(123)
cm <- confusionMatrix(data = as.factor(data$scored.class),
                      reference = as.factor(data$class),
                      mode = "everything",
                      positive = "1")
cm

##Investigating pROC package
pROC_obj <- roc(data$class, data$scored.probability, smoothed = TRUE,
                # arguments for ci
                ci = TRUE, ci.alpha = 0.95, 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")
```

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