DATA 621 Assignment 2

Darwhin Gomez

Table of contents


```
1 library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
1 library(caret)
```

```
Loading required package: ggplot2

Loading required package: lattice
```

```
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
library(ggplot2)
```

1- Loading Data

```
data <- read.csv("classification-output-data.csv")
```

```
head(data,5)
```

```
  pregnant glucose diastolic skinfold insulin  bmi pedigree age class
1        7     124        70       33     215 25.5    0.161  37     0
2        2     122        76       27     200 35.9    0.483  26     0
3        3     107        62       13      48 22.9    0.678  23     1
4        1      91        64       24       0 29.2    0.192  21     0
5        4      83        86       19       0 29.3    0.317  34     0
  scored.class scored.probability
1            0         0.32845226
2            0         0.27319044
3            0         0.10966039
4            0         0.05599835
5            0         0.10049072
```

The class column represents the actual class whilst the scored.class represents the predicted class.

2- Table function

```r
conf_matrix<-table(data$class, data$scored.class)
conf_matrix
```

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

Interpretation of the Values:

- True Positives (TP) = 119: The model *correctly* predicted Positive cases.
- True Negatives (TN) = 27: The model *correctly* predicted Negative cases.
- False Positives (FP) = 30: The model *incorrectly* predicted Positive (Type I Error or "False Alarm"). The actual case was Negative.
- False Negatives (FN) = 5: The model *incorrectly* predicted Negative (Type II Error or "Miss"). The actual case was Positive.

3- Accuracy function

```r
acc_func <- function(df, actual, predicted){
  cm <- table(df[[actual]], df[[predicted]])
  tp <- cm[2,2]
  tn <- cm[1,1]
  fp <- cm[1,2]
  fn <- cm[2,1]

  (tp + tn) / (tp + tn + fp + fn)



}
computed_acc<- acc_func(data, "class", "scored.class")
print(paste("The calculated accuracy of the model is:",
            round(acc_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated accuracy of the model is: 0.8066"
```

4- Classification Error Rate

```
error_func <- function(df, actual, predicted){
  cm <- table(df[[actual]], df[[predicted]])
  tp <- cm[2,2]
  tn <- cm[1,1]
  fp <- cm[1,2]
  fn <- cm[2,1]

(fp + fn)/(tp + tn + fp +fn)



}
computed_err <- error_func(data, "class", "scored.class")
print(paste("The calculated classification error rate of the model is:",
            round(error_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated classification error rate of the model is: 0.1934"
```

We can verify by adding the accuracy and the error rate which should sum to 1

```
print(computed_acc + computed_err)
```

```
[1] 1
```

5- Precision

```
precision_func <- function(df, actual, predicted) {
  cm <- table(df[[actual]], df[[predicted]])
  TP <- cm[2,2]; FP <- cm[1,2]
  TP / (TP + FP)
}
 calculated_prec<- precision_func(data, "class", "scored.class")
 print(paste("The calculated precision of the model is:",
             round(precision_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated precision of the model is: 0.8438"
```

6- Sensitivity (Recall)

```
sensitivity_func <- function(df, actual, predicted) {
  cm <- table(df[[actual]], df[[predicted]])
  TP <- cm[2,2]; FN <- cm[2,1]
  TP / (TP + FN)
}
calculated_sensitivity<-sensitivity_func(data, "class", "scored.class")
print(paste("The calculated  Sensiticity (recall rate) of the model is:",
            round(sensitivity_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated  Sensiticity (recall rate) of the model is: 0.4737"
```

7- Specificity

```
specificity_func <- function(df, actual, predicted) {
  cm <- table(df[[actual]], df[[predicted]])
  TN <- cm[1,1]; FP <- cm[1,2]
  TN / (TN + FP)
}
 calculated_specif<-specificity_func(data, "class", "scored.class")
 print(paste("The calculated  Specificity of the model is:",
             round(specificity_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated  Specificity of the model is: 0.9597"
```

8- F1 Score

```
f1_score_func <- function(df, actual, predicted) {
  p <- precision_func(df, actual, predicted)
  r <- sensitivity_func(df, actual, predicted)
  2 * (p * r) / (p + r)
}
calculated_f1<-f1_score_func(data, "class", "scored.class")
 print(paste("The calculated  F1 score of the model is:",
             round(f1_score_func(data, "class", "scored.class"),4)))
```

```
[1] "The calculated  F1 score of the model is: 0.6067"
```

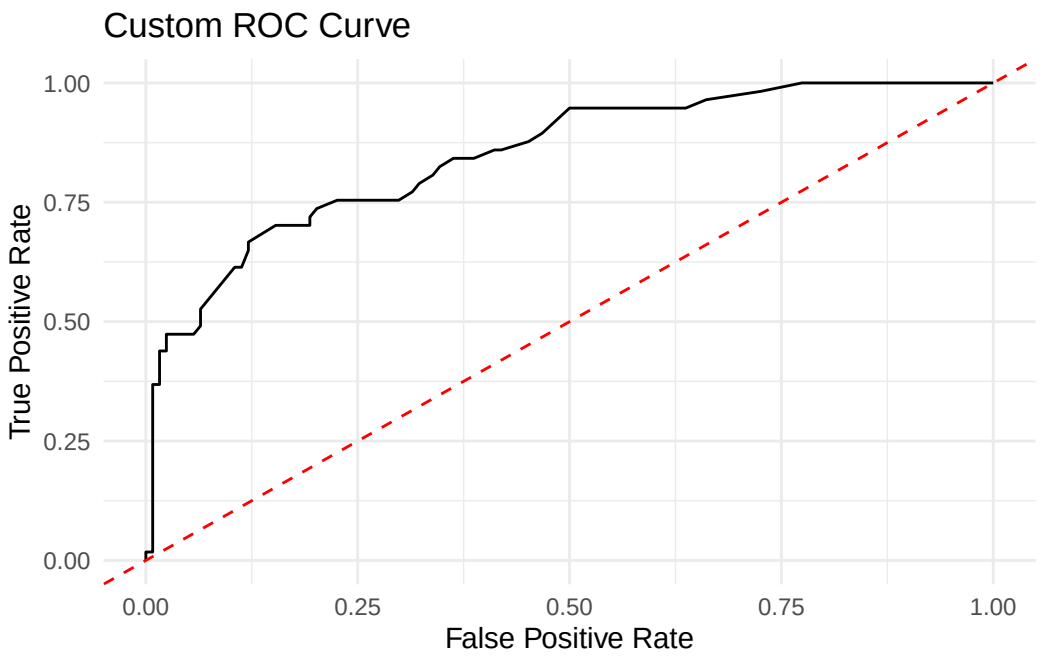
The F1-score will always be between 0 and 1 because it is based on precision and recall, which are both proportions that range from 0 to 1. When either precision or recall is 0, the F1-score is also 0. When both are 1, the F1-score reaches its maximum value of 1. Therefore, since it is the harmonic mean of two values that cannot be less than 0 or greater than 1, the F1-score will always fall between 0 and 1.

10- ROC Curve Function and plot

```r
roc_curve_custom <- function(df, actual, prob_col) {
  thresholds <- seq(1, 0, -0.01)

  TPR <- FPR <- numeric(length(thresholds))

  for (i in seq_along(thresholds)) {
    t <- thresholds[i]
    df$pred <- ifelse(df[[prob_col]] >= t, 1, 0)


    cm <- table(factor(df[[actual]], levels = c(0,1)),
                factor(df$pred, levels = c(0,1)))

    TP <- cm[2,2]; TN <- cm[1,1]; FP <- cm[1,2]; FN <- cm[2,1]
    TPR[i] <- TP / (TP + FN)
    FPR[i] <- FP / (FP + TN)
  }

  auc <- sum(diff(FPR) * (head(TPR, -1) + tail(TPR, -1)) / 2)

  p<-ggplot(data.frame(FPR, TPR), aes(x = FPR, y = TPR)) +
    geom_line(color = "black") +
    geom_abline(linetype = "dashed", color = "red") +
    labs(title = "Custom ROC Curve", x = "False Positive Rate",
         y = "True Positive Rate") +
    theme_minimal()


  return(list( Plot=p,AUC = auc))
}
```

```
roc_curve_custom(data, "class", "scored.probability")
```

```
$Plot
```




```
$AUC
[1] 0.8488964
```

11- All Metrics Comparison Caret

```
data.frame(
  Accuracy = computed_acc,
  Error_Rate = computed_err,
  Precision = calculated_prec,
  Sensitivity = calculated_sensitivity,
  Specificity = calculated_specif,
  F1_Score = calculated_f1
)
```

```
   Accuracy Error_Rate Precision Sensitivity Specificity  F1_Score
1 0.8066298  0.1933702   0.84375   0.4736842   0.9596774 0.6067416
```

```r
caret_conf <- confusionMatrix(
  factor(data$scored.class),
  factor(data$class),
  positive = "1"
)
caret_conf
```

```
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
             Prevalence : 0.3149
         Detection Rate : 0.1492
   Detection Prevalence : 0.1768
      Balanced Accuracy : 0.7167

       'Positive' Class : 1
```

The functions match output from the caret package indicating caret can be used to calculate these metrics reliably and easily.

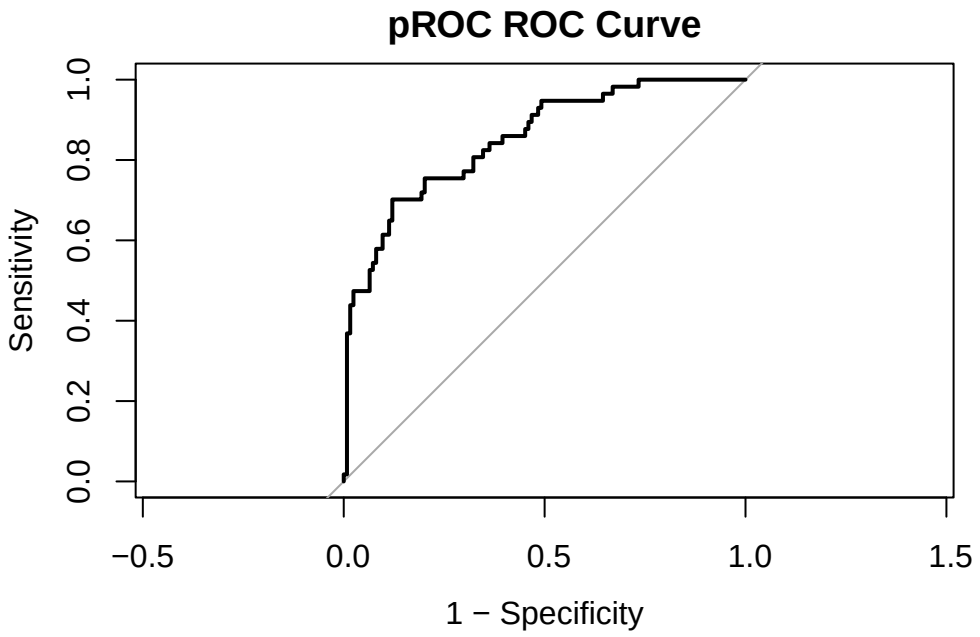
12-13 pROC

```
roc_obj <- roc(data$class, data$scored.probability)
```

```
Setting levels: control = 0, case = 1
```

```
Setting direction: controls < cases
```

```
plot(roc_obj, col="black", main="pROC ROC Curve",legacy.axes = TRUE)
```




```
auc(roc_obj)
```

```
Area under the curve: 0.8503
```

The computed ROC plot and the pROC plot display very similar curves, with the custom AUC at 0.8488 and the pROC AUC at 0.8503, indicating highly consistent results in the calculation of the area under the curve. Overall, both the ROC plots and AUC values demonstrate that the model is performing well, particularly in accurately identifying positive cases.

The packages work well and should be used to produce classification metrics efficiently and reliably.