Data621_HW2

Moscoe, Telab, Greenlee, Biguzzi, Wright, Sookal, Connin

10/6/2021

This assignment is a study in classification metrics and related data graphics in R. Our dataset includes class membership scores for a target variable (var = “class”) as well as predicted membership scores (var = “scored.class”) for each observation. The predictions derive from model probability estimates (var = “scored.probability”) of class membership based on predictor variables. The model form, target variable, and predictors are not defined.

Observations for class (actual) and scored.class (predicted) are scored as 1 or 0 to indicate membership or not with respect to our target. A score of 1 indicates membership and is considered a “positive” result for our purposes. An score of 0 indicates non-membership and is considered a “negative” result.

Comparing actual and predicted scores, we distingish membership classification for the latter as follows:

- True Positives - predictions correctly classified as members
- True Negatives - predictions correctly classified as non-members
- False Positives - predictions mistakenly classified as members
- False Negatives - predictions mistakenly classified as non-members

It is important to note that class predictions reflect the probability of membership in relation to an assigned cutoff (threshold) value. And that model probabilities and thresholds can range in value from 0 to 1.

A probabilty that equals or exceeds an assigned threshold value results in a positive classification. A negative classification results when a probability is less than a threshold value. Misclassification (False Positive or False Negative) occurs when a membership prediction is incorrect.

Our ability to discriminate membership is, in essence, a balance between the cost:benefit of making False vs. True predictions. With consequences that are domain specific - e.g., monetary, medical, human resources, etc. It is incumbent upon the investigator, therefore, to develop a classifier (model and threshold) that optimizes cost:benefit decision-making for her/his purpose.

This assignment asks use to derive classification metrics for membership predictions produced using a threshold value of 0.5.

1. Download the classification output data set.

```
df <-read_csv("https://raw.githubusercontent.com/sconnin/621_Business_Analytics/main/HW2_GRP/classification-output-data.csv"
)

# select columns that we will use for this project

df%<>%dplyr::select(class, scored.class, scored.probability)

# assess dataframe dimensions, features, and dtypes

glimpse(df)
```

2. Use the table() function to get the raw confusion matrix for this scored dataset.

```
# compile the confusion matrix in table form

cfm<-table(df$class, df$scored.class, dnn =c("Actual Class", 'Predicted Class'))

cfm
```

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

We can identify our table counts as follows:

```
## True Negative Counts (correctly classified as not the target class) = 119
```

```
## True Positives Counts (correctly classified as the target class) = 27
```

```
## False Positive Counts (incorrectly classified as the target class) = 5
```

```
## False Negative Counts (incorrectly classified as not the target class) = 30
```

Note: for questions 3-8 we construct a general function that can be called to calculate our classification metrics. This ensures consistent class assignments for each combination of target and prediction value. Because class assignments are based on a cut-off threshold set a-priori (i.e., 0.5), we include this argument in our function call.

```
#First we write a general function that can be repurposed

class_metric <- function(dframe, actual, predicted, threshold, metric){
    true_pos <- sum(dframe[,actual] == 1 & dframe[,predicted] >= threshold)
    true_neg <- sum(dframe[,actual] == 0 & dframe[,predicted] < threshold)
    false_pos <- sum(dframe[,actual] == 0 & dframe[,predicted] >= threshold)
    false_neg <- sum(dframe[,actual] == 1 & dframe[,predicted] < threshold)

    if(metric == "accuracy"){
        return((true_pos + true_neg) / nrow(dframe))
    } else if(metric == "error"){
        return((false_pos + false_neg) / nrow(dframe))
    } else if(metric == "precision"){
        return(true_pos / (true_pos + false_pos))
    } else if(metric == "sensitivity" | metric == "recall"){
        return(true_pos / (true_pos + false_neg))
    } else if(metric == "specificity"){
        return(true_neg / (true_neg + false_pos))
    } else if(metric == "F1"){
        return(2 * true_pos / (2 * true_pos + false_pos + false_neg))
    } else {
        return("oops.")
    }
}
```

3. Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the accuracy of these predictions.

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

```
# Accuracy is a measure (percentage) of correct predictions

class_metric(df, "class", "scored.class", 0.5, 'accuracy')
```

```
## [1] 0.8066298
```

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.

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

```
#The error rate is measure (percentage) of incorrect predictions

class_metric(df, "class", "scored.class",.05, 'error')
```

```
## [1] 0.1933702
```

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.

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

```
class_metric(df, "class", "scored.class",.05, 'precision')
```

```
## [1] 0.84375
```

We can verify that our accuracy and error rate sum to 1.

```
class_metric(df, "class", "scored.class", 0.5, 'accuracy')+class_metric(df, "class", "scored.class", 0.5, "error")
```

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

We can also demonstrate that accuracy and error rates sum to 1 across a range of thresholds

```
#Set 101 threshold values between 0-1:

thresholds <- seq(0,1,0.01)

accumulator <- 0
for(i in thresholds){
  acc <- class_metric(df, "class", "scored.class", i, "accuracy")
  err <- class_metric(df, "class", "scored.class", i, "error")
  accumulator <- accumulator + (acc + err == 1)
}

glue("Given 101 threshold levels between 0-1, we find that the sum of accuracy + error = 1 results {accumulator} times.")
```

```
## Given 101 threshold levels between 0-1, we find that the sum of accuracy + error = 1 results 101 times.
```

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.

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

Sensitivity: The probability that the model predicts a positive outcome for an observation when indeed the outcome is positive

```
class_metric(df, "class", "scored.class", 0.5, 'sensitivity')
```

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

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.

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

Specificity: The probability that the model predicts a negative outcome for an observation when indeed the outcome is negative.

```
class_metric(df, "class", "scored.class", 0.5, 'specificity')
```

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

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.

Eqn 1. $F1 = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$

Another way to write this is:

Eqn 2. $F1 = 2 * \frac{precision*recall}{precision+recall}$

Or

Eqn 3. $F1 = 2 * \frac{precision*sensitivity}{precision+sensitivity}$

```
class_metric(df, "class", "scored.class", 0.5, 'F1')
```

```
## [1] 0.6067416
```

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.)

Recall that:

- precision can range from 0 to 1
- sensitivity can range from 0 to 1

The F1 score simplifies to $\frac{2TP}{2TP+FP+FN}$. All variables are >= 0.

Then:

$2TP + FP + FN \geq 2TP$. Then $0 \leq F1 \leq 1$.

10. Write a function that generates an ROC curve from a data set with a true classification column (*class*) and a probability column (*scored.probability*).

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.

ROC (Receiver Operating Characteristics) CURVES

The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for selected probability thresholds.

The FPR can be calculated as follows:

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

FPR is also equivalent to 1 - model *specificity*.

The TPR can be calculated as follows:

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

TPR is equivalent to model *sensitivity* and/or *recall*.

Binary classification models rarely (if ever) predict class membership correctly for every observation. As such, they represent a trade-off for decision-makers seeking to optimize the number of target observations that are correctly classified (TPR) while minimizing the number of non-target observations that are misclassified (FPR).

By reducing the probability threshold, we can increase the number of TP classifications. However, the number number of FP classifications also increase simultaneously. The opposite pattern results when we increase the probability threshold.

An ROC curve can be used to assess the balance of TP and FP classifications at different threshold values. Thus providing a diagnostic by which to optimize a classifier to best meet model/project goals.

AUC Calculations

The AUC (area under the ROC curve) provides an additional measure of a model's ability to correctly discriminate between target and non-target classes. An AUC function takes predicted probabilities for the target class and the true outcomes (0, 1) for a test set and returns a score between 0.0 and 1.0.

An AUC of 0.0 results when model predictions yield no correct classifications; an AUC of 1 results when every classification is correct. An AUC of ~ 0.5 indicates that the model is unable to distinguish between target and non-target classes (Null Model).

Our AUC calculation (included below) is drawn from AUC - calculation based on https://bit.ly/3m7zW6k (https://bit.ly/3m7zW6k).

$AUC = \sum(TPR * FPR_{diff}) + \frac{TPR_{diff} * FPR}{2}$

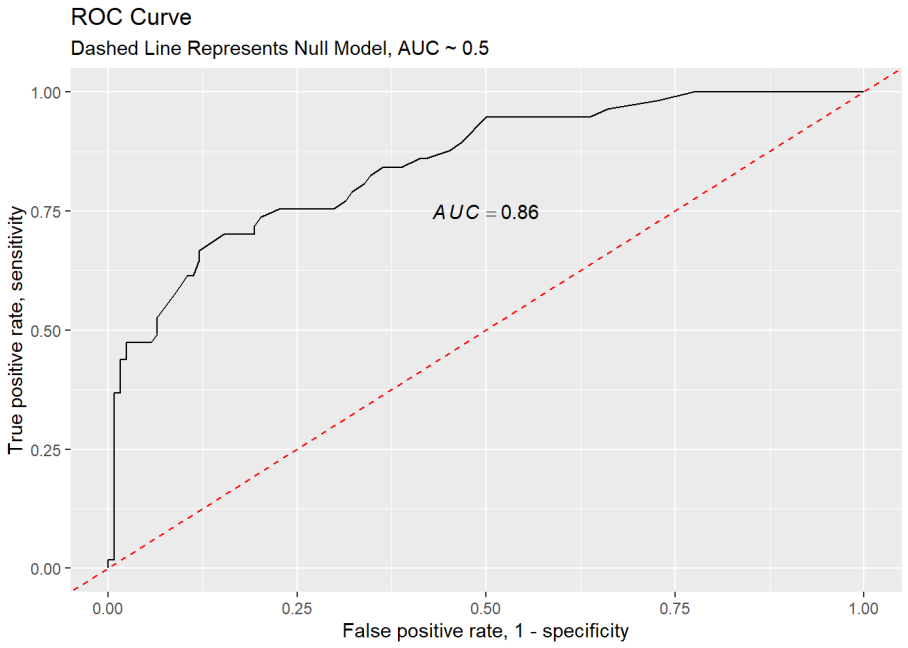
Where:

- TPR = true positive rate
- FPR = false positive rate
- TPR_{diff} = the difference between consecutive TPR observations
- FPR_{diff} = the difference between consecutive FPR observations

The AUC estimate is an approximation similar to a Riemann sum in integral calculus. The first entry in our equation is equivalent to a Riemann rectangle. The second entry extends this geometry with a triangle "cap" to compensate for curvature in the AUC function. Together they form a trapezoid. The area of each trapeziod included under the curve is summed to produce an overall area estimate.

```
# build function to create ROC plot and estimate AUC

roc_fn <- function(dframe, actual, predicted){
  res <- list()
  threshold <- seq(0, 1, 0.01)

  false_pos_rate <- 1 - vapply(threshold, class_metric, double(1), dframe = dframe, actual = actual, predicted = predicted,
 metric = "specificity")

  true_pos_rate <- vapply(threshold, class_metric, double(1), dframe = dframe, actual = actual, predicted = predicted, metri
c = "sensitivity")

  to_plot <- data.frame(threshold, false_pos_rate, true_pos_rate)

  plot_obj <- ggplot(data = to_plot, aes(x = false_pos_rate, y = true_pos_rate)) +
    geom_path() +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype="dashed" ) +
    labs(x="False positive rate, 1 - specificity", y="True positive rate, sensitivity") +
    labs(title="ROC Curve", subtitle="Dashed Line Represents Null Model, AUC ~ 0.5")

  FPRdiff <- abs(c(diff(false_pos_rate),0)) #Lag diff in false pos, pad end with 0 due to lag
  TPRdiff <- abs(c(diff(true_pos_rate),0)) #Lag diff in true pos
  auc <- round(sum(true_pos_rate * FPRdiff) + sum(TPRdiff * FPRdiff)/2, 2)

  res$plt <- plot_obj
  res$auc <- auc

  return(res)
}
```

```
# Print results for ROC and AUC

tmp <- roc_fn(df, "class", "scored.probability")
tmp$plt+annotate("text", x = .5, y = .75, label = "italic(AUC) == 0.86", parse=TRUE)
```




```
AUC<-tmp$auc
glue("Model AUC = {AUC}.")
```

```
## Model AUC = 0.86.
```

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

```
calculated_metrics=NULL

for(metric in c("accuracy", "sensitivity", "error", "specificity", "precision", "F1")){

    print(paste0(metric, " = ", round(class_metric(df, "class", "scored.probability", 0.5, metric), 4)))
    measured<-(class_metric(df, "class", "scored.probability", 0.5, metric))
    calculated_metrics[[metric]]<-measured
}
```

```
## [1] "accuracy = 0.8066"
## [1] "sensitivity = 0.4737"
## [1] "error = 0.1934"
## [1] "specificity = 0.9597"
## [1] "precision = 0.8438"
## [1] "F1 = 0.6067"
```

```
calculated_metrics<-calculated_metrics[-3] # we will use this list for comparison with Caret results in the next section
```

12. Investigate the Caret package.

Investigate the Caret package. In particular, consider the functions confusion matrix, sensitivity, and specificity. Apply the functions to the data set. How do the results compare with your own functions?

```
cfm_caret <- df

cfm_caret$class<-as.factor(cfm_caret$class)
cfm_caret$scored.class<-as.factor(cfm_caret$scored.class)

(conf<-confusionMatrix(cfm_caret$scored.class, cfm_caret$class, positive="1", mode = "everything")) #note x, y order matters here
```

```
## 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
##
```

```
#build comparison of metrics between Caret and our calculations

Caret<-c(conf[[3]]['Accuracy'], conf[[4]]['Sensitivity'], conf[[4]]['Specificity'], conf[[4]]['Precision'], conf[[4]]['F1'])

(Compare_metrics<-cbind(Caret, calculated_metrics))
```

```
##             Caret     calculated_metrics
## Accuracy    0.8066298 0.8066298
## Sensitivity 0.4736842 0.4736842
## Specificity 0.9596774 0.9596774
## Precision   0.84375   0.84375
## F1          0.6067416 0.6067416
```

Values for our calculated classification metrics are identical to those produced by the Caret package.

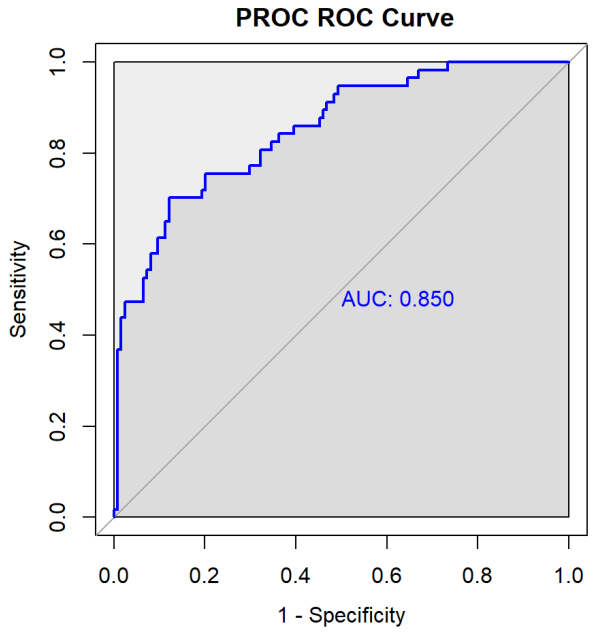
13. Investigate the pROC package.

Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```
# build ROC graph

par(pty="s")

proc_demo<-roc(response=df$class, predictor=df$scored.probability, plot=TRUE, legacy.axes=TRUE, auc.polygon=TRUE, col="blue"
, main=" PROC ROC Curve", max.auc.polygon=TRUE,  print.auc=TRUE)
```

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

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




```
# calculate AUC

auc(proc_demo)
```

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

The results from `pROC` are equivalent to those generated by our function, `roc_fn` after accounting for rounding error.

It's worth noting that the calculated AUC (.86) suggests that the model used to produce the probabilities included in our dataset is an effective classifier.