Eric_Hirsch_621_Assignmrnt_2

Eric Hirsch

3/20/2022

Contents


```r
#tinytex::install_tinytex()
```

```r
library(caret)
library(pROC)
library(tidyverse)
```

Read the data

```r
dfB <- read.csv("D:\\RStudio\\CUNY_621\\Assignment 2\\classification-output-data.csv")
dfB <- dfB %>%
  dplyr::select(class, scored.class, scored.probability)
```

A. Review the raw confusion matrix -

```r
dfB1 <- dfB %>%
  dplyr::select(-scored.probability)

table(dfB1)
```

```
##      scored.class
## class   0   1
##     0 119   5
##     1  30  27
```

```
dfTable <- as.data.frame(table(dfB1))
```

The raw confusion matrix shows us the actual values as rows and the scored values as columns. The upper left-lower right diagonal contains correct predictions, while the opposite diagonal contains incorrect predictions.

B. Calculate Prediciton Metrics

We write a function to return accuracy (correct predictions over all predictions)

```
TableToDf <- function (df)
{
  df1 = as.data.frame(table(df))
  li <- vector(mode = "list", length = 4)
  names(li) <- c("TP", "TN", "FP", "FN")

  li$TP <- df1$Freq[4]
  li$TN <- df1$Freq[1]
  li$FP <- df1$Freq[3]
  li$FN <- df1$Freq[2]

  return(li)

}

x <- TableToDf(dfB1)
```

1. Accuracy Function

```
Accuracy <- function(df) {

li <- TableToDf(df)
Accuracy <- (li$TP + li$TN)/(li$TP + li$TN + li$FP + li$FN)

return (Accuracy)

}

print(Accuracy(dfB1))
```

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

2. Classification Error Rate Function

Classification Error Rate (incorrect predictions over all predicitons:

```
ClassificationErrorRate <- function(df) {

li <- TableToDf(df)
Error <- (li$FP + li$FN)/(li$TP + li$TN + li$FP + li$FN)

return (Error)


}
print(ClassificationErrorRate(dfB1))
```

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

Accuracy + error rate = 1

```
print(as.numeric(ClassificationErrorRate(dfB1)) + as.numeric(Accuracy(dfB1)))
```

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

3. Precision Function (True positives/All who tested positive):

```
Precision <- function(df) {

li <- TableToDf(df)
Precision <- (li$TP)/(li$TP + li$FP)

return(Precision)

}
print(Precision(dfB1))
```

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

4. Sensitivity/Recall Function (True positives/All who are positive):

```
Sensitivity <- function(df) {

li <- TableToDf(df)


  Sensitivity <- (li$TP)/(li$TP + li$FN)



return(Sensitivity)

}
print(Sensitivity(dfB1))
```

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

5. Specificity Function (True negatives/All who are negative):

```
Specificity <- function(df) {

li <- TableToDf(df)
Specificity <- (li$TN)/(li$TN + li$FP)

return(Specificity)

}
print(Specificity(dfB1))
```

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

6. F1 score Function:

```
F1 <- function(df) {


p <- Precision(dfB1)
s <- Sensitivity(dfB1)

F1 <- (2*p*s)/(p+s)

return(F1)

}

print(F1(dfB1))
```

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

C. Bounds on the F1 score

Precision (P) and Sensitivity (S) themselves are always <= 1 because their denominators are a simple sum which include the numerators. Therefore, $P*S < P$ and $P*S < S$. Therefore, $2PS < P + S$.

D. ROC curve Function

```
ROC <- function(df) {

list1 <- list()

dfNew <- data.frame(class = numeric(),
 scored.class = numeric())

dfFinal <- data.frame(Specificity = numeric(),
```

```
 Sensitivity = numeric(), Area = numeric(), Width=numeric())

spec_prev = 0
sens_prev = 0

for (i in 1:100) {
  for (j in 1:length(df$class))
  {

  dfNew[j, 1] = df$class[j]
  dfNew[j, 2]  <- ifelse(df$scored.probability[j] > i/100, 1, 0)
  }

  sens <- Sensitivity(dfNew)
  spec <- 1 - Specificity(dfNew)

  width = spec_prev - spec
  ave_sens = (sens + sens_prev)/sens

  dfFinal[i,1] <- spec
  dfFinal[i,2] <- sens
  dfFinal[i,3] <- sens*width
  dfFinal[i,4] <- width

  spec_prev = spec
  sens_prev = sens
}

dfFinal <- na.omit(dfFinal)

AUC <- sum(dfFinal$Area)/sum(dfFinal$Width)
print (paste("AUC:", AUC))

g <- ggplot(dfFinal, aes(x=Specificity, y=Sensitivity)) +
             geom_line()

x <- list(g, dfFinal$Area)

return (x)
}
```

E. Provide all metrics

```
## [1] "Accuracy:  0.806629834254144"
```

```
## [1] "Classification Error:  0.193370165745856"
```

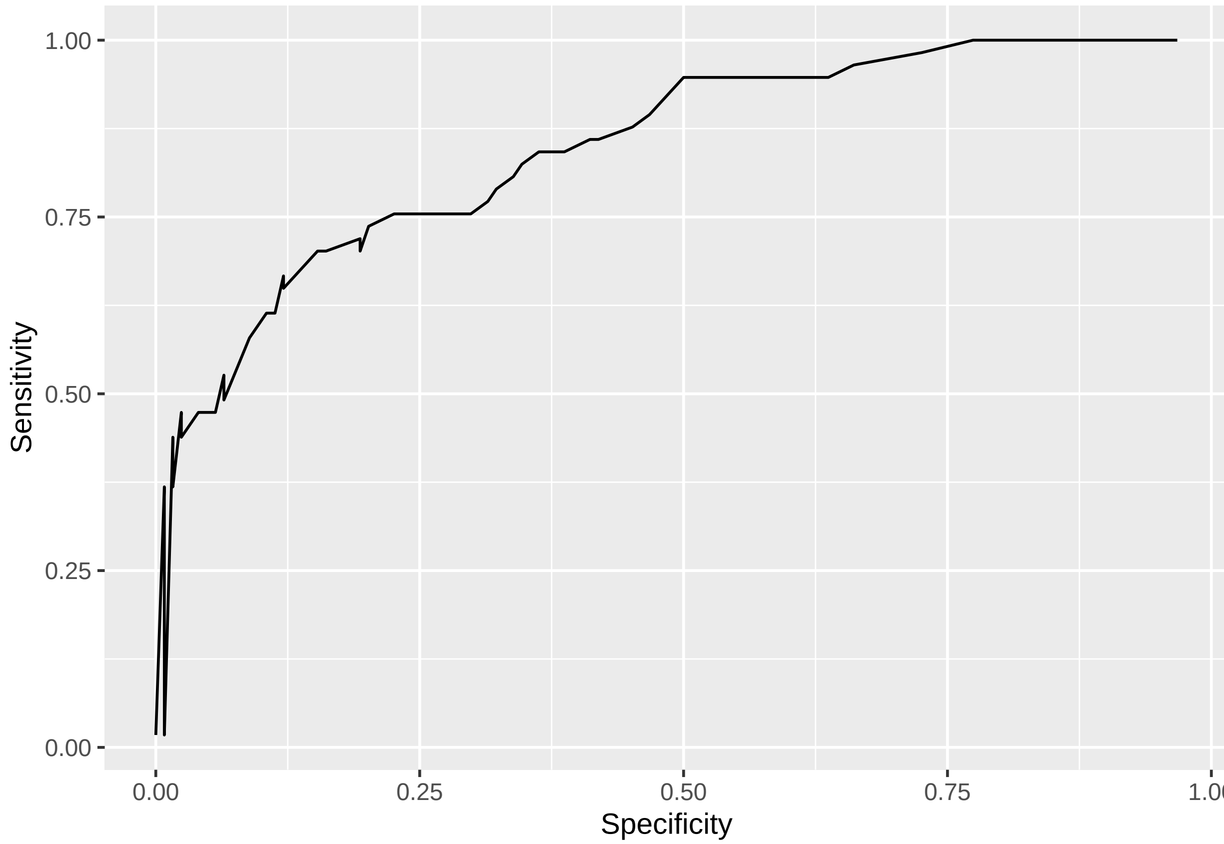
```
## [1] "Precision:  0.84375"
```

```
## [1] "Sensitivity:  0.473684210526316"
```

```
## [1] "Specificity: 0.959677419354839"
```

```
## [1] "F1: 0.606741573033708"
```

```
## [1] "AUC: 0.839930404523706"
```



F. The caret package provides statistics that match ours.

```
dfB1$class <- as.factor(dfB1$class)
dfB1$scored.class <- as.factor(dfB1$scored.class)

a <- confusionMatrix(data = dfB1$scored.class, reference = dfB1$class, positive="1")

q <- a$byClass["Sensitivity"]

Caret_Metrics <- list()
Caret_Metrics[1] <- a$overall["Accuracy"]
Caret_Metrics[2] <- 1 - as.numeric(Caret_Metrics[[1]])
Caret_Metrics[3] <- a$byClass["Precision"]
Caret_Metrics[4] <- a$byClass["Sensitivity"]
Caret_Metrics[5] <- a$byClass["Specificity"]
Caret_Metrics[6] <- a$byClass["F1"]

Metric <- c("Accuracy", "ClassificationErrorRate", "Precision", "Sensitivity", "Specificity", "F1")
```

```r
dfX <- as.data.frame(cbind(Metric, My_Metrics, Caret_Metrics))
knitr::kable(dfX)
```

Metric	My_Metrics	Caret_Metrics
Accuracy	0.8066298	0.8066298
ClassificationErrorRate	0.1933702	0.1933702
Precision	0.84375	0.84375
Sensitivity	0.4736842	0.4736842
Specificity	0.9596774	0.9596774
F1	0.6067416	0.6067416

```r
print(a)
```

```
## 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
##
```

G. The pROC ROC curve

The pROC package can be used to create an ROC curve. This curve matches our own. The AUC scores are slightly different (.84 vs .85)

```r
roc(class ~ scored.probability, dfB)
```

```
##
```

```
## Call:
## roc.formula(formula = class ~ scored.probability, data = dfB)
##
## Data: scored.probability in 124 controls (class 0) < 57 cases (class 1).
## Area under the curve: 0.8503
```

```
roc1 <- roc(dfB$class,
            dfB$scored.probability, plot=TRUE)
```