DATA621 HW2

Misha Kollontai, Matthew Baker, Erinda Budo, Subhalaxmi Rout, Don Padmaperuma

9/28/2020

Libraries

```
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
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(reshape2)
```

Downloading the Data

```
df <- read.csv("https://raw.githubusercontent.com/mkollontai/DATA621_GroupWork/master/HW2/classificatio
df_hw2 <- df[,c('class', 'scored.class','scored.probability')]
```

We have pulled the data and simplified our dataframe into just the 3 columns in question:

- class
- scored.class
- scored.probability

Simple Confusion Matrix

```r
table(df_hw2$class, df_hw2$scored.class)
```

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

This confusion matrix shows us the number of entries of each class against what they were predicted to be. In this table the row indicates the actual class, whereas the column signifies the predicted value (scored.class). We can therefore deduce that of the 57 total cases of class '1', 27 were predicted to be class '1' correctly. The other 30 were mistakenly predicted to be of class '0'. For the actual class '0' cases, 5 were mistakenly predicted to be class '1', whereas 119 were correctly identified.

In order to calculate all of the following evaluations of the predictions made within this dataframe we need to identify the number of:

- True Positives - Correctly identified class '1' cases
- False Positives - Class '0' cases incorrectly identified as class '1'
- True Negatives - Correctly identified class '0' cases
- False Negatives - Class '1' cases incorrectly identified as class '0'

In order to avoid repeating code required to identify these outcomes, let's crate a function that calculates the number associated with each of these four outcomes.

```r
outcomes <- function(df, classCol, predCol){
  outs <- aggregate(df, by = list(df[,classCol], df[,predCol]), FUN=length) %>%
    select(1:3) %>%
    `colnames<-` (c('class','prediction','count'))
  return(outs)
  #Within this df, in the 'count' column:
  #the first row is our True Negative (TN),
  #the second row is False Negatives (FN),
  #third row is False Positives (FP) and
  #final row is the True Positives (TP).
}
```

Accuracy Calculation

To calculate accuracy we must use the formula

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

```r
acc <- function(df, classCol, predCol){
  outs <- outcomes(df, classCol, predCol)
  TN <- outs$count[1]
  FN <- outs$count[2]
  TP <- outs$count[3]
  FP <- outs$count[4]
  acc <- (TP+TN)/(TP+FP+TN+FN)
  return(acc)
}

acc(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.6850829
```

Classification Error Rate Calculation

To calculate classification error we must use the formula

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

```
cer <- function(df, classCol, predCol){
  outs <- outcomes(df, classCol, predCol)
  TN <- outs$count[1]
  FN <- outs$count[2]
  TP <- outs$count[3]
  FP <- outs$count[4]
  cer <- (FP+FN)/(TP+FP+TN+FN)
  return(cer)
}

cer(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.3149171
```

The calculations for these two measures are such that summing the two should yield 1:

```
acc(df_hw2, 'class', 'scored.class') + cer(df_hw2, 'class', 'scored.class')
```

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

Precision Calculation

To calculate precision we must use the formula

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

```
prec <- function(df, classCol, predCol){
  outs <- outcomes(df, classCol, predCol)
  TP <- outs$count[3]
  FP <- outs$count[4]
  prec <- (TP)/(TP+FP)
  return(prec)
}

prec(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.15625
```

Sensitivity Calculation

To calculate sensitivity we must use the formula

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

```r
sens <- function(df, classCol, predCol){
  outs <- outcomes(df, classCol, predCol)
  FN <- outs$count[2]
  TP <- outs$count[3]
  sens <- (TP)/(TP+FN)
  return(sens)
}

sens(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.1428571
```

Specificity Calculation

To calculate specficity we must use the formula

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

```r
spec <- function(df, classCol, predCol){
  outs <- outcomes(df, classCol, predCol)
  TN <- outs$count[1]
  FP <- outs$count[4]
  sens <- (TN)/(TN+FP)
  return(sens)
}

spec(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.8150685
```

F1 Calculation

To calculate F1 we must use the formula

$$F1 = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

```r
f1 <- function(df, classCol, predCol){
  prec <- prec(df, classCol, predCol)
  sens <- sens(df, classCol, predCol)
  f1 <- (2*prec*sens)/(prec+sens)
  return(f1)
}

f1(df_hw2, 'class', 'scored.class')
```

```
## [1] 0.1492537
```

Bounds of F1

To show the bounds of F1, let's simplify the euation to a more manageable form:

$$F1 = \frac{2ps}{p+s}$$

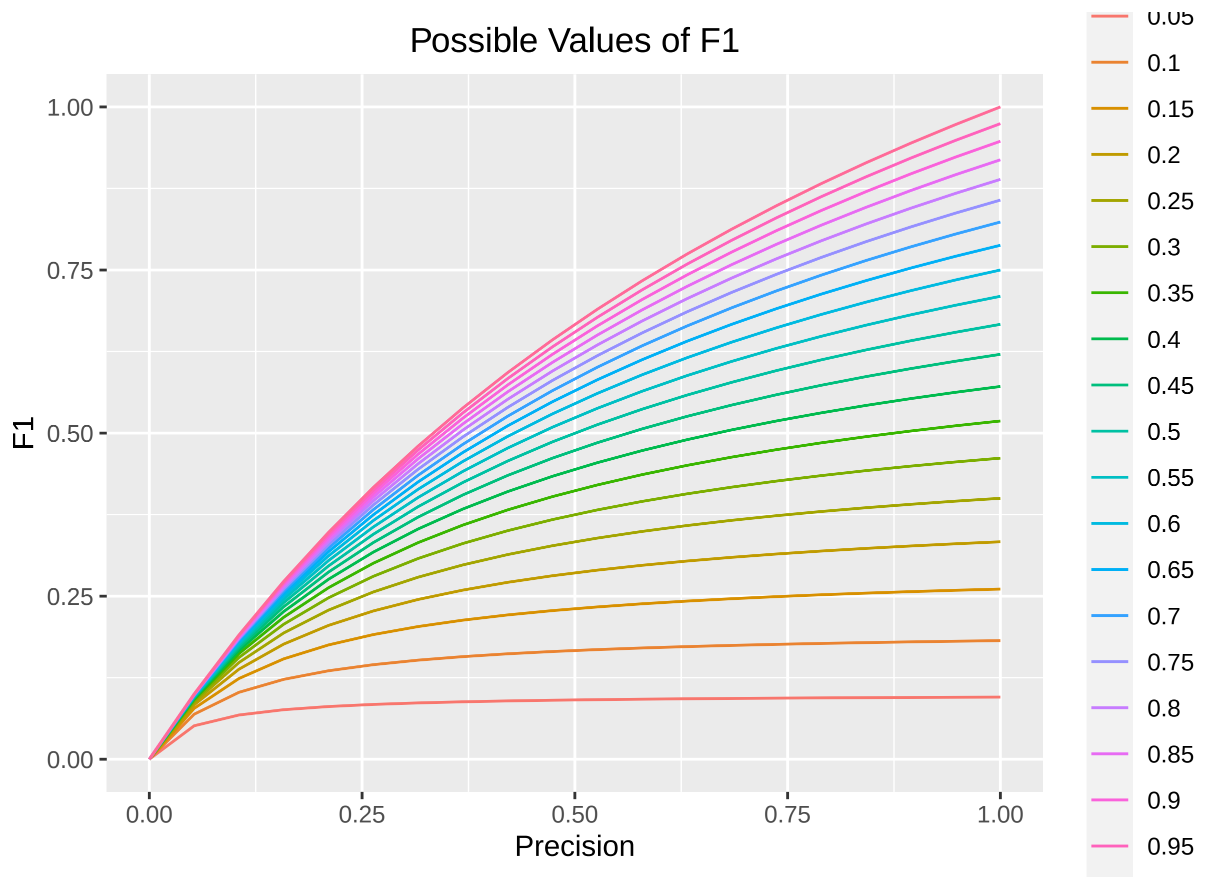
The following code generates 'l' evenly-spaced values between 0 and 1 for the Precision. It then uses a loop to generate the calculated F1 values through the same range of Sensitivity values corresponding to each of the Precision values. The plot below shows calculated F1 on the y-axis, with each line corresponding to the calculation associated with a particular Sensitivity value.

```r
#Partially used approach from (https://stackoverflow.com/questions/14704742/use-for-loop-to-plot-multip

l <- 20
p <- seq(0,1, length.out = l)

df <- NULL
for(i in 1:l){
  j <- i/l
  temp_df <- data.frame(x = p, y = (2*p*j)/(p+j), col = rep(j:j,each = l))
  df <- rbind(df, temp_df)
}


ggplot(df,aes(x=x,y=y,group=col,color=factor(col))) +
  geom_line() +
  ggtitle('Possible Values of F1') +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab('F1') +
  xlab('Precision') +
  ylim(0,1)
```



As we can clearly see, **the range of possible values for F1 is between 0 and 1**, as it was for both the Precision and the Sensitivity.