Week 2

Daniel Brooks (daniel.brooks@spsmail.cuny.edu), Daniel Fanelli (daniel.fanelli@spsmail.cuny.edu), Christopher Fenton (christopher.fenton@spsmail.cuny.edu), James Hamski (james.hamski@spsmail.cuny.edu), Youqing Xiang (youqing.xiang@spsmail.cuny.edu)

6/26/2016

0.0.1 1. Download/read the classification output data set

```
data <- read.csv('classification-output-data.csv')
summary(data)
```

```
##     pregnant         glucose        diastolic        skinfold
##  Min.   : 0.000   Min.   : 57.0   Min.   : 38.0   Min.   : 0.0
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.0   1st Qu.: 0.0
##  Median : 3.000   Median :112.0   Median : 70.0   Median :22.0
##  Mean   : 3.862   Mean   :118.3   Mean   : 71.7   Mean   :19.8
##  3rd Qu.: 6.000   3rd Qu.:136.0   3rd Qu.: 78.0   3rd Qu.:32.0
##  Max.   :15.000   Max.   :197.0   Max.   :104.0   Max.   :54.0
##     insulin            bmi           pedigree           age
##  Min.   :  0.00   Min.   :19.40   Min.   :0.0850   Min.   :21.00
##  1st Qu.:  0.00   1st Qu.:26.30   1st Qu.:0.2570   1st Qu.:24.00
##  Median :  0.00   Median :31.60   Median :0.3910   Median :30.00
##  Mean   : 63.77   Mean   :31.58   Mean   :0.4496   Mean   :33.31
##  3rd Qu.:105.00   3rd Qu.:36.00   3rd Qu.:0.5800   3rd Qu.:41.00
##  Max.   :543.00   Max.   :50.00   Max.   :2.2880   Max.   :67.00
##      class         scored.class    scored.probability
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.02323
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.11702
##  Median :0.0000   Median :0.0000   Median :0.23999
##  Mean   :0.3149   Mean   :0.1768   Mean   :0.30373
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.43093
##  Max.   :1.0000   Max.   :1.0000   Max.   :0.94633
```

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

```
cf <- table(data[,9:10])
cf
```

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

Explain:

- column (scored.class): the predicted class

- row (class): the actual class
- class = 0 and scored.class = 0: there are 119 observations which are predicted correctly with class 0
- class = 0 and scored.class = 1: there are 5 observations which are class 0 but are predicted with class 1
- class = 1 and scored.class = 0: there are 30 obervations which are class 1 but are predicted with class 0
- class = 1 and scored.class = 1: there ae 27 obervations which are correctly predicted with class 1.

0.0.3 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.

```
my_accuracy <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  accuracy <- (cf$Freq[1] + cf$Freq[4])/sum(cf$Freq)
  return(accuracy)
}
```

0.0.4 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.

```
my_error <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  error <- (cf$Freq[2] + cf$Freq[3])/sum(cf$Freq)
  return(error)
}
```

0.0.5 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.

```
my_precision <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  precision <- cf$Freq[4]/(cf$Freq[4] + cf$Freq[3])
  return(precision)
}
```

0.0.6 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.

```
my_sensitivity <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  sensitivity <- cf$Freq[4]/(cf$Freq[4] + cf$Freq[2])
  return(sensitivity)
}
```

0.0.7 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.

```
my_specificity <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  specificity <- cf$Freq[1]/(cf$Freq[1] + cf$Freq[3])
  return(specificity)
}
```

0.0.8 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.

```
my_f1s <- function(data) {
  cf <- table(data[,9:10])
  cf <- as.data.frame(cf)
  f1s <- 2*cf$Freq[4]/(2*cf$Freq[4] + cf$Freq[2] + cf$Freq[3])
  return(f1s)
}
```

0.0.9 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.

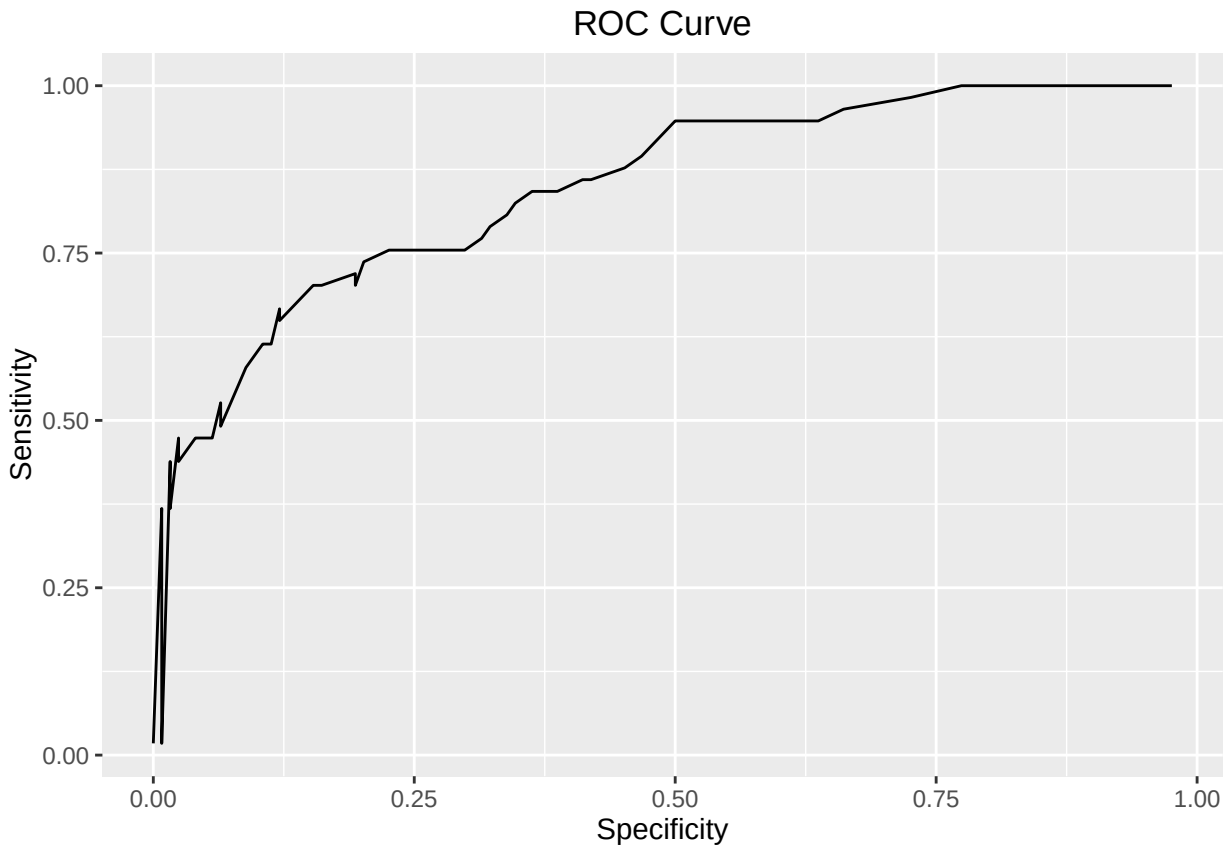
- Answer: after transformation, F1 score $= \frac{2TP}{2TP+FN+FP}$. If FN and FP are very small (close to 0), F1 score is close to 1; if FN or FP is very large (close to 1) and TP is very small, F1 score is close to 0. So, the F1 score will always be between 0 and 1.

0.0.10 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).

```
library(ggplot2)
my_fun <- function(data) {
  data1 = data
  thresholds <- seq(0,1,0.01)
  Y <- c()
  X <- c()
  for (threshod in thresholds) {
    data1$scored.class <- ifelse(data1$scored.probability > threshod,1,0)
    X <- append(X,1-my_specificity(data1))
    Y <- append(Y,my_sensitivity(data1))
    }
  df <- data.frame(X=X,Y=Y)
  df <- na.omit(df)
  g <- ggplot(df,aes(X,Y)) + geom_line() + ggtitle('ROC Curve') +
    xlab('Specificity') + ylab('Sensitivity')
  height = (df$Y[-1]+df$Y[-length(df$Y)])/2
  width = -diff(df$X)
```

```
  AUC = sum(height*width)
  return(list(AUC=AUC,g=g))
}

result = my_fun(data)
result$g
```




```
result$AUC
```

```
## [1] 0.8247029
```

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

```
my_accuracy(data)
```

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

```
my_error(data)
```

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

```
my_precision(data)
```

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

```
my_sensitivity(data)
```

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

```
my_specificity(data)
```

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

```
my_f1s(data)
```

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

0.0.12 12. nvestigate the caret package. In particular, consider the functions confusionMatrix, sensitivity, and specificity. Apply the functions to the data set.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

```
## Loading required package: lattice
```

```
confusionMatrix(data$scored.class, data$class, positive = "1")
```

```
## 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
##
```

I got the same accuracy, sensitivity and specificity.

0.0.13 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?

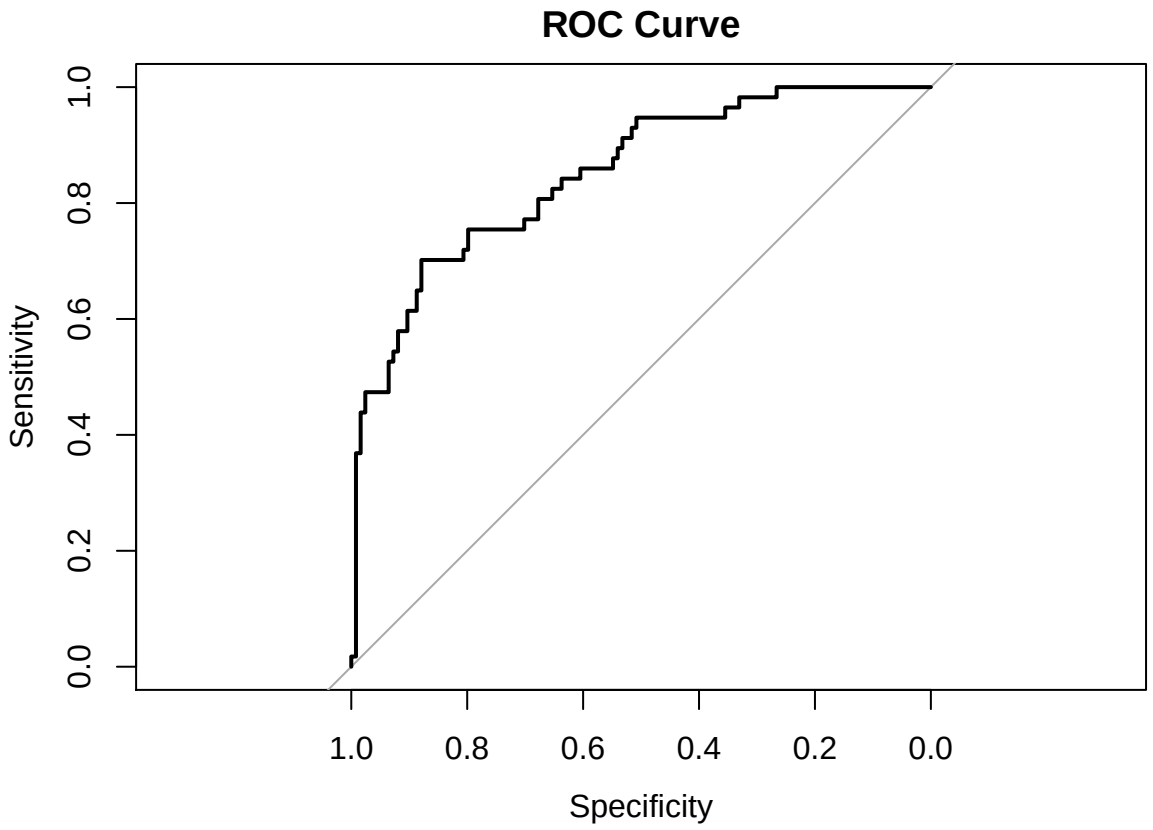
```
library(pROC)
```

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

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
rc <- roc(as.factor(data$class) ~ data$scored.probability)
plot(rc,main='ROC Curve')
```



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

```
rc$auc
```

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

- I got the similar curve and the area under the curve.