DATA 621 Assignment 2

Joby John, Zachary Herold, Jun Pan

March 11, 2019

Set working environment

Load data

```
output <- read.csv("https://raw.githubusercontent.com/johnpannyc/group-1-data-621-assignment-2/m
aster/classification-output-data.csv")
```

1. DATA EXPLORATION

```
glimpse(output)
```

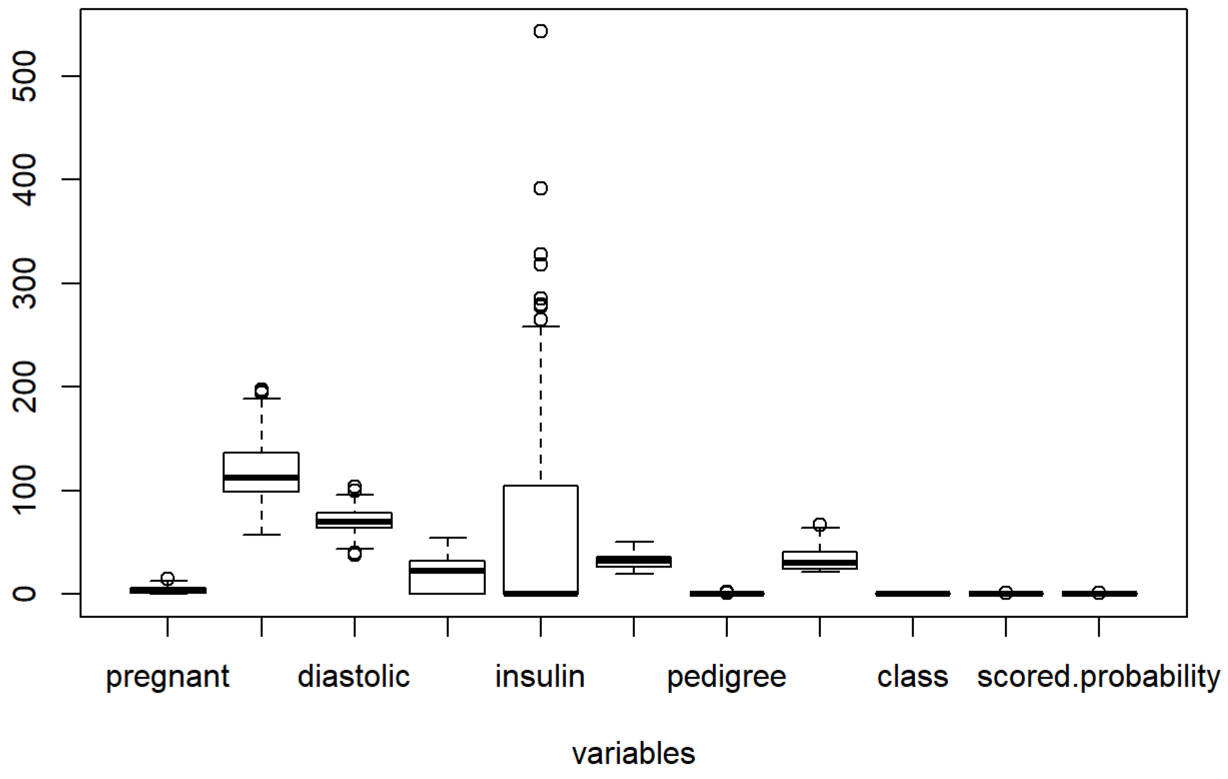
```
## Observations: 181
## Variables: 11
## $ pregnant          <int> 7, 2, 3, 1, 4, 1, 9, 8, 1, 2, 5, 5, 13, 0, ...
## $ glucose           <int> 124, 122, 107, 91, 83, 100, 89, 120, 79, 12...
## $ diastolic         <int> 70, 76, 62, 64, 86, 74, 62, 78, 60, 48, 78,...
## $ skinfold          <int> 33, 27, 13, 24, 19, 12, 0, 0, 42, 32, 30, 4...
## $ insulin           <int> 215, 200, 48, 0, 0, 46, 0, 0, 48, 165, 0, 7...
## $ bmi               <dbl> 25.5, 35.9, 22.9, 29.2, 29.3, 19.5, 22.5, 2...
## $ pedigree          <dbl> 0.161, 0.483, 0.678, 0.192, 0.317, 0.149, 0...
## $ age               <int> 37, 26, 23, 21, 34, 28, 33, 64, 23, 26, 37,...
## $ class             <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1...
## $ scored.class      <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1...
## $ scored.probability <dbl> 0.32845226, 0.27319044, 0.10966039, 0.05599...
```

classification-output-data.csv file contains 181 observations of 11 variables. Three variables will be considered for this report - class (actual class for the observation), scored.class (predicted class for the observation), and scored.probability (predicted probability of success for the observation).

```
summary(output)
```

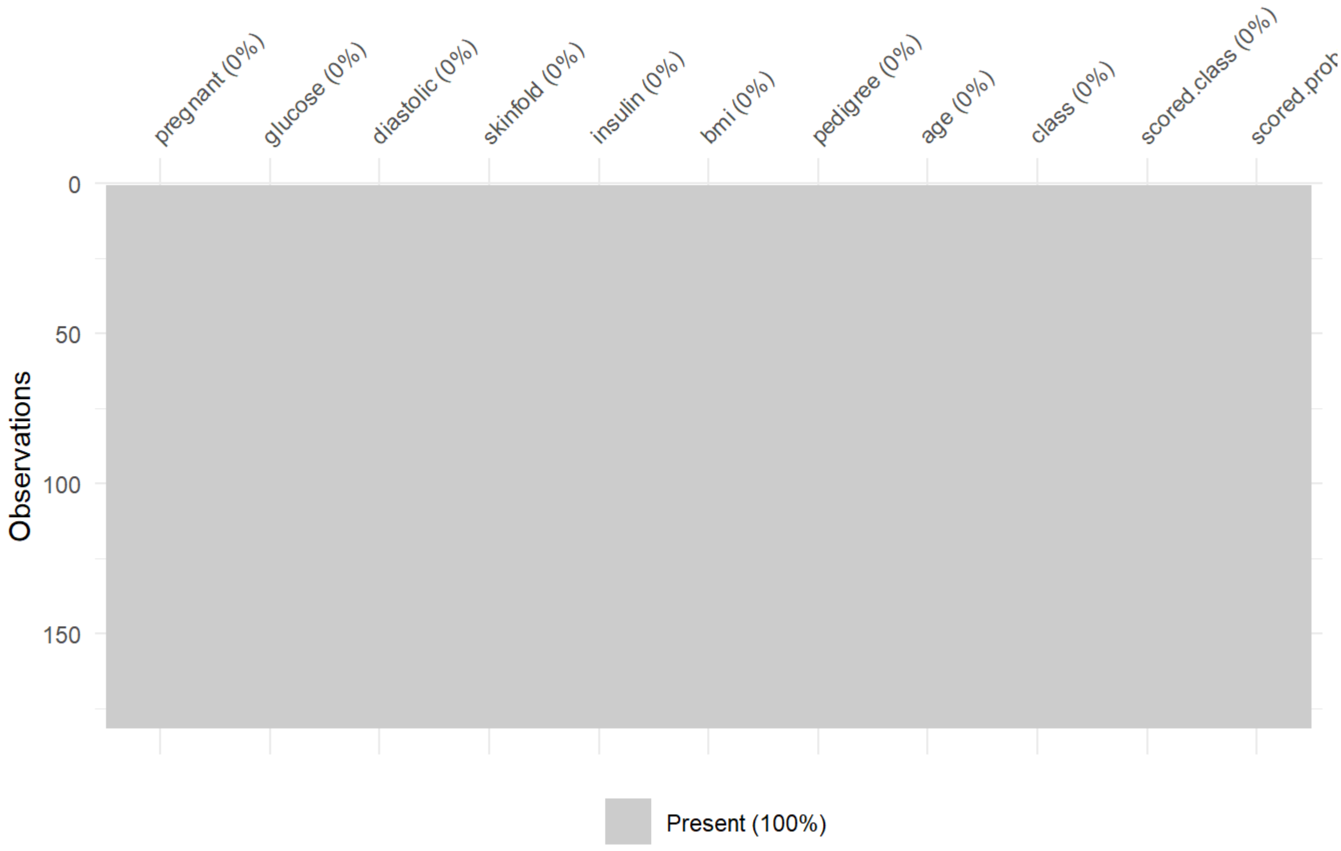
```
##     pregnant         glucose       diastolic       skinfold  
##  Min.   : 0.000   Min.   : 57.0   Min.   : 38.0   Min.   : 0.0  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 64.0   1st Qu.: 0.0  
##  Median : 3.000   Median :112.0   Median : 70.0   Median :22.0  
##  Mean   : 3.862   Mean   :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
```

```
boxplot(output,xlab="variables")
```



Check missing data

```
vis_miss(output)
```



No missing data in the file.

Use the table() function to get the raw confusion matrix for this scored dataset (method 1)

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

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

Looking at the matrix above, rows represent actual class values of 0 or 1. Columns represent predicted class values of 0 or 1. So in the top left corner 119 is the number of observations where the class was correctly predicted to be 0. The top right corner shows 5 observations where the class of 0 was incorrectly predicted as 1. Similarly, we have 30 observations of class 1 incorrectedly predicted as class 0 and 27 observations of class 1 correctly predicted.

Assuming that 0 is a negative class and 1 is a positive class we have:

TN = 119 FP = 5 FN = 30 TN = 27

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

```
data <- read.csv("https://raw.githubusercontent.com/johnpannyc/group-1-data-621-assignment-2/master/classification-output-data.csv")
cmatrix <-  table(data$class, data$scored.class)
cmatrix
```

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

```
Accuracy <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  accuracy = (cmatrix[2,2] + cmatrix[1,1]) / (cmatrix[2,2] + cmatrix[1,2] + cmatrix[1,1] + cmatrix[2,1])
  return(round(accuracy, 2))
}
```

```
Accuracy(output)
```

```
## [1] 0.81
```

```
Classification_error_rate <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  classification_error_rate = (cmatrix[1,2] + cmatrix[2,1]) / (cmatrix[2,2] + cmatrix[1,2] + cmatrix[1,1] + cmatrix[2,1])
  return(round(classification_error_rate, 2))

}
```

```
Classification_error_rate(output)
```

```
## [1] 0.19
```

```
Precision <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  precision = (cmatrix[2,2] / (cmatrix[2,2] + cmatrix[1,2]))
  return(round(precision, 2))

}
```

```
Precision(output)
```

```
## [1] 0.84
```

```
Sensitivity <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  sensitivity = cmatrix[2,2] / (cmatrix[2,2] + cmatrix[2,1])
  return(round(sensitivity, 2))

}
```

```
Sensitivity(output)
```

```
## [1] 0.47
```

```
Specificity <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  specificity =  cmatrix[1,1] / (cmatrix[1,1] + cmatrix[1,2])
  return(round(specificity, 2))

}
```

```
Specificity(output)
```

```
## [1] 0.96
```

```
F1_Score <- function(df)
{
  names    = c("class", "scored.class")
  cmatrix  = table(df[, names])
  precision = Precision(df)
  sensitivity = Sensitivity(df)
  f1_score =  (2 * precision * sensitivity) /(precision + sensitivity)

  return(round(f1_score, 2))
}
```

```
F1_Score(output)
```

```
## [1] 0.6
```

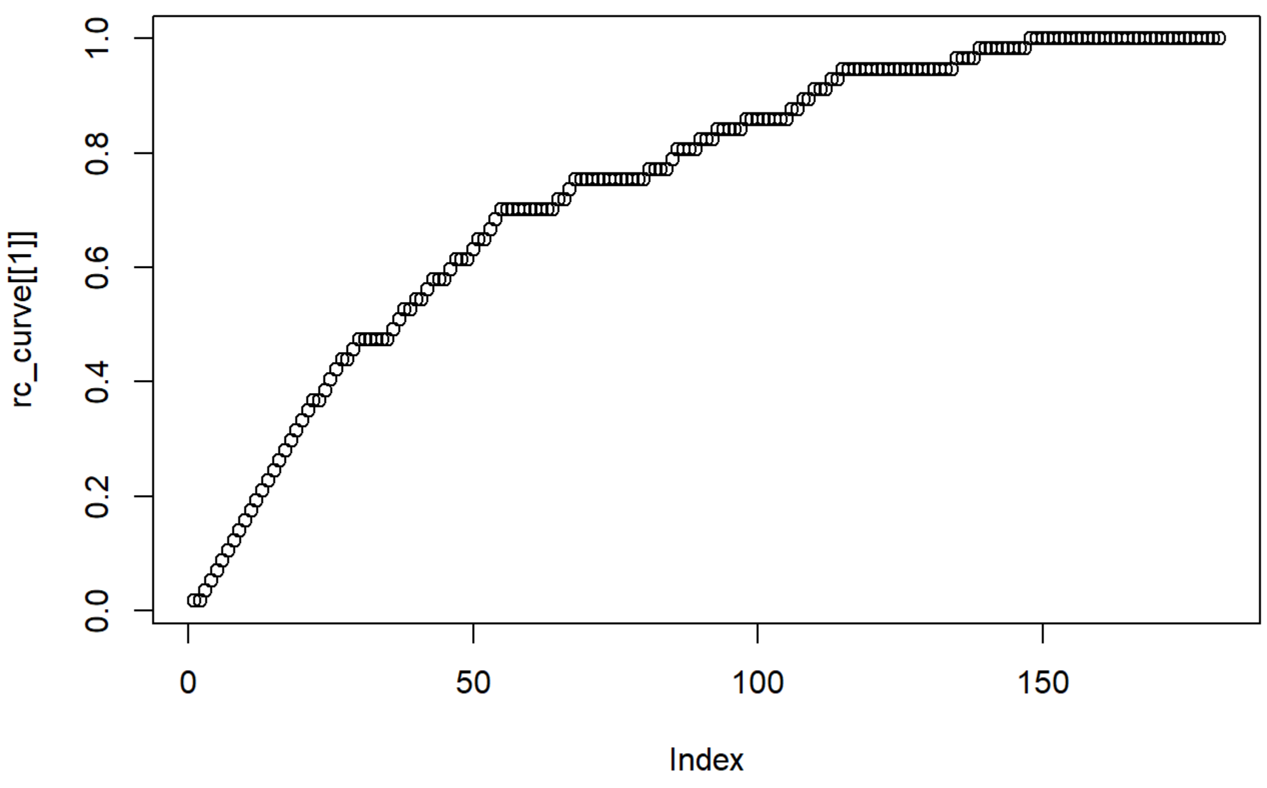
10.Manually create ROC curve

```
manual_roc <- function(labels, scores){
  labels <- labels[order(scores, decreasing=TRUE)]
  TPR=cumsum(labels)/sum(labels)
  FPR=cumsum(!labels)/sum(!labels)
  df<- data.frame(TPR,FPR)
  dFPR <- c(diff(FPR), 0)
  dTPR <- c(diff(TPR), 0)
  auc <-sum(TPR * dFPR) + sum(dTPR * dFPR)/2
  return(c(df,auc))
}



rc_curve <- manual_roc(output$class,output$scored.probability)

plot(rc_curve[[1]])
```



```
auc <- rc_curve[[2]]
```

Q11 - Using the functions to generate the classification metrics

All metrics were provided as they were calculated. As we will see below using built-in functions makes life easier.

```
Accuracy(output)
```

```
## [1] 0.81
```

```
Classification_error_rate(output)
```

```
## [1] 0.19
```

```
Precision(output)
```

```
## [1] 0.84
```

```
Sensitivity(output)
```

```
## [1] 0.47
```

```
Specificity(output)
```

```
## [1] 0.96
```

```
F1_Score(output)
```

```
## [1] 0.6
```

Q12 - Investigating the caret package

```
if (!"caret" %in% installed.packages()) install.packages(caret)
require(caret)

ls(pos = "package:caret")
```

```
##   [1] "anovaScores"           "avNNet"               
##   [3] "bag"                   "bagControl"           
##   [5] "bagEarth"              "bagEarthStats"        
##   [7] "bagFDA"                "best"                 
##   [9] "BoxCoxTrans"           "calibration"          
##  [11] "caretFuncs"            "caretGA"              
##  [13] "caretSA"               "caretSBF"             
##  [15] "caretTheme"            "cforestStats"         
##  [17] "checkConditionalX"     "checkInstall"         
##  [19] "checkResamples"        "class2ind"            
##  [21] "classDist"             "cluster"              
##  [23] "compare_models"        "confusionMatrix"      
##  [25] "confusionMatrix.train" "contr.dummy"          
##  [27] "contr.ltfr"            "createDataPartition"  
##  [29] "createFolds"           "createModel"          
##  [31] "createMultiFolds"      "createResample"       
##  [33] "createTimeSlices"      "ctreeBag"             
##  [35] "defaultSummary"        "dotPlot"              
##  [37] "downSample"            "dummyVars"            
##  [39] "expandParameters"      "expoTrans"            
##  [41] "extractPrediction"     "extractProb"          
##  [43] "F_meas"                "featurePlot"          
##  [45] "filterVarImp"          "findCorrelation"      
##  [47] "findLinearCombos"      "flatTable"            
##  [49] "gafs"                  "gafs.default"         
##  [51] "gafs_initial"          "gafs_lrSelection"     
##  [53] "gafs_raMutation"       "gafs_rwSelection"     
##  [55] "gafs_spCrossover"      "gafs_tourSelection"   
##  [57] "gafs_uCrossover"       "gafsControl"          
##  [59] "gamFormula"            "gamFuncs"             
##  [61] "gamScores"             "getModelInfo"         
##  [63] "getSamplingInfo"       "getTrainPerf"         
##  [65] "groupKFold"            "hasTerms"             
##  [67] "icr"                   "index2vec"            
##  [69] "ipredStats"            "knn3"                 
##  [71] "knn3Train"             "knnreg"               
##  [73] "knnregTrain"           "ldaBag"               
##  [75] "ldaFuncs"              "ldaSBF"               
##  [77] "learing_curve_dat"     "lift"                 
##  [79] "lmFuncs"               "lmSBF"                
##  [81] "LPH07_1"               "LPH07_2"              
##  [83] "lrFuncs"               "MAE"                  
##  [85] "maxDissim"             "MeanSD"               
##  [87] "minDiss"               "mnLogLoss"            
##  [89] "modelCor"              "modelLookup"          
##  [91] "multiClassSummary"     "nbBag"                
##  [93] "nbFuncs"               "nbSBF"                
##  [95] "nearZeroVar"           "negPredValue"         
##  [97] "nnetBag"               "nullModel"            
##  [99] "nzv"                   "oneSE"                
## [101] "outcome_conversion"    "panel.calibration"    
## [103] "panel.lift"            "panel.lift2"          
## [105] "panel.needle"          "pcaNNet"              
## [107] "pickSizeBest"          "pickSizeTolerance"
## [109] "pickVars"              "plot.gafs"
## [111] "plot.rfe"              "plot.train"
## [113] "plotClassProbs"        "plotObsVsPred"
## [115] "plsBag"                "plsda"
## [117] "posPredValue"          "postResample"
## [119] "precision"             "predict.bagEarth"
## [121] "predict.gafs"          "predict.train"
## [123] "predictionFunction"    "predictors"
## [125] "preProcess"            "print.train"
## [127] "probFunction"          "progress"
## [129] "prSummary"             "R2"
## [131] "recall"                "resampleHist"
## [133] "resamples"             "resampleSummary"
## [135] "resampleWrapper"       "rfe"
## [137] "rfeControl"            "rfeIter"
## [139] "rfFuncs"               "rfGA"
## [141] "rfSA"                  "rfSBF"
## [143] "rfStats"               "RMSE"
## [145] "safs"                  "safs_initial"
## [147] "safs_perturb"          "safs_prob"
## [149] "safsControl"           "sbf"
## [151] "sbfControl"            "sbfIter"
## [153] "sensitivity"           "SLC14_1"
## [155] "SLC14_2"               "sortImp"
## [157] "spatialSign"           "specificity"
## [159] "splsda"                "sumDiss"
## [161] "summary.bagEarth"      "svmBag"
## [163] "thresholder"           "tolerance"
## [165] "train"                 "trainControl"
## [167] "treebagFuncs"          "treebagGA"
## [169] "treebagSA"             "treebagSBF"
## [171] "twoClassSim"           "twoClassSummary"
## [173] "upSample"              "var_seq"
## [175] "varImp"                "well_numbered"
```

```
?sensitivity
```

```
## starting httpd help server ... done
```

```
?confusionMatrix
?precision
```

Transposing the table so that the actual referenced value (i.e., truth, “class”) is in columns, and the predicted measurement system (i.e. “scored.class”) is in rows

```
df <- data[c("class","scored.class")]
cmatrix.t <- t(table(df))
cmatrix.t
```

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

```
str(cmatrix.t)
```

```
##  'table' int [1:2, 1:2] 119 5 30 27
##  - attr(*, "dimnames")=List of 2
##   ..$ scored.class: chr [1:2] "0" "1"
##   ..$ class       : chr [1:2] "0" "1"
```

Comparing the home-made functions and the caret package ones

```
sens.caret <- round(sensitivity(cmatrix.t, positive = rownames(cmatrix)[2]),2)
sens.caret
```

```
## [1] 0.47
```

```
identical(Sensitivity(data), sens.caret)
```

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

```
spec.caret <- round(specificity(cmatrix.t, negative = rownames(cmatrix)[1]),2)
spec.caret
```

```
## [1] 0.96
```

```
identical(Specificity(data), spec.caret)
```

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

```
cMat.caret <- confusionMatrix(cmatrix.t, positive = "1")
cMat.caret
```

```
## Confusion Matrix and Statistics
## 
##             class
## scored.class   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               
## 
```

```
str(cMat.caret)
```

```
## List of 6
##  $ positive: chr "1"
##  $ table   : 'table' int [1:2, 1:2] 119 5 30 27
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ scored.class: chr [1:2] "0" "1"
##   .. ..$ class       : chr [1:2] "0" "1"
##  $ overall : Named num [1:7] 0.807 0.492 0.741 0.861 0.685 ...
##   ..- attr(*, "names")= chr [1:7] "Accuracy" "Kappa" "AccuracyLower" "AccuracyUpper" ...
##  $ byClass : Named num [1:11] 0.474 0.96 0.844 0.799 0.844 ...
##   ..- attr(*, "names")= chr [1:11] "Sensitivity" "Specificity" "Pos Pred Value" "Neg Pred Val
ue" ...
##  $ mode    : chr "sens_spec"
##  $ dots    : list()
##  - attr(*, "class")= chr "confusionMatrix"
```

```
prec.caret <- round(precision(cmatrix.t, relevant = "1"),2)
prec.caret
```

```
## [1] 0.84
```

```
identical(Precision(data), prec.caret)
```

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

```
acc.caret <- round(cMat.caret$overall[1],2)
acc.caret
```

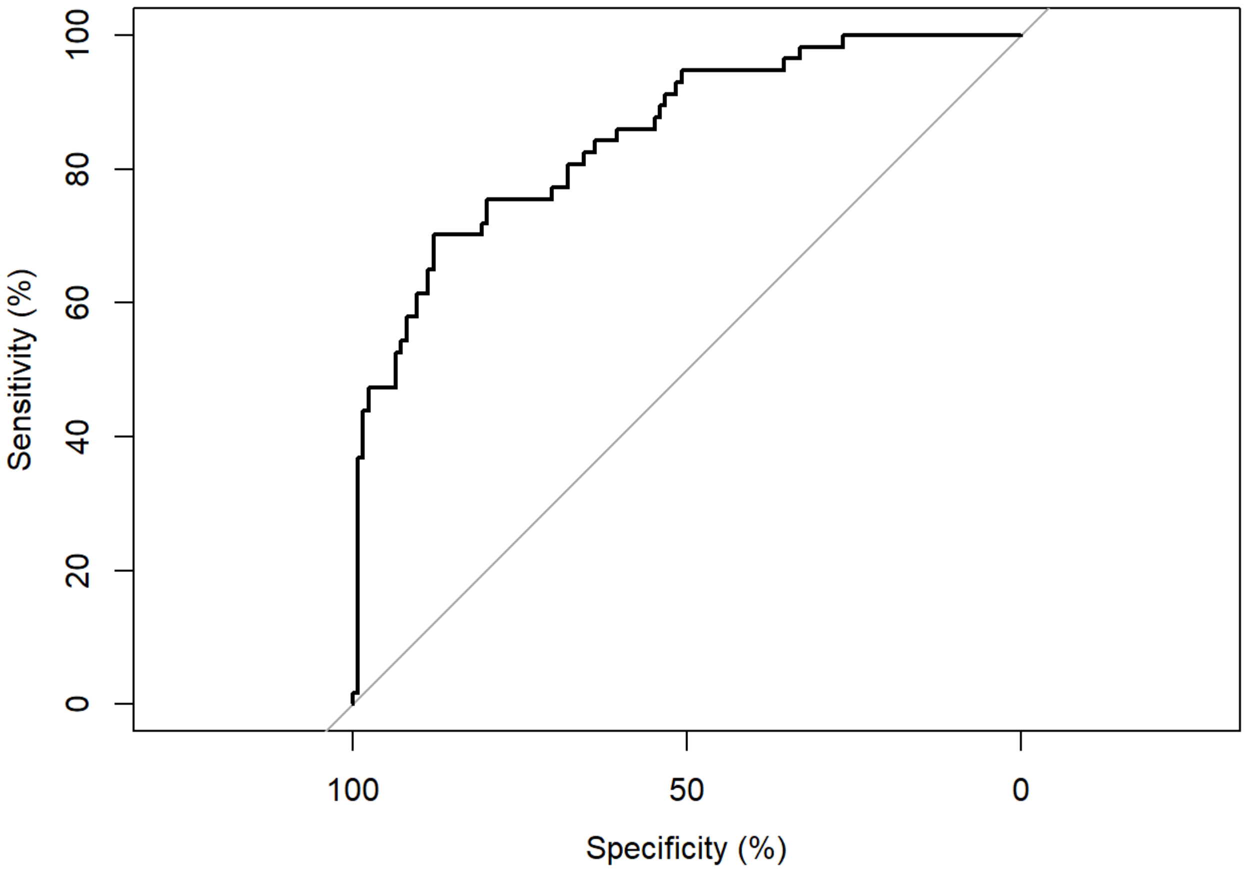
```
## Accuracy
##     0.81
```

```
identical(Accuracy(data), acc.caret)  ## same value, but fail to match with identical function
```

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

13. pROC Package Let us try the pROC package.

```
roc(output$class, output$scored.probability, levels=c(0,1), percent=TRUE, plot=TRUE, ci=TRUE)
```



```
##
## Call:
## roc.default(response = output$class, predictor = output$scored.probability,     levels = c(0,
1), percent = TRUE, ci = TRUE, plot = TRUE)
##
## Data: output$scored.probability in 124 controls (output$class 0) < 57 cases (output$class 1).
## Area under the curve: 85.03%
## 95% CI: 79.05%-91.01% (DeLong)
```