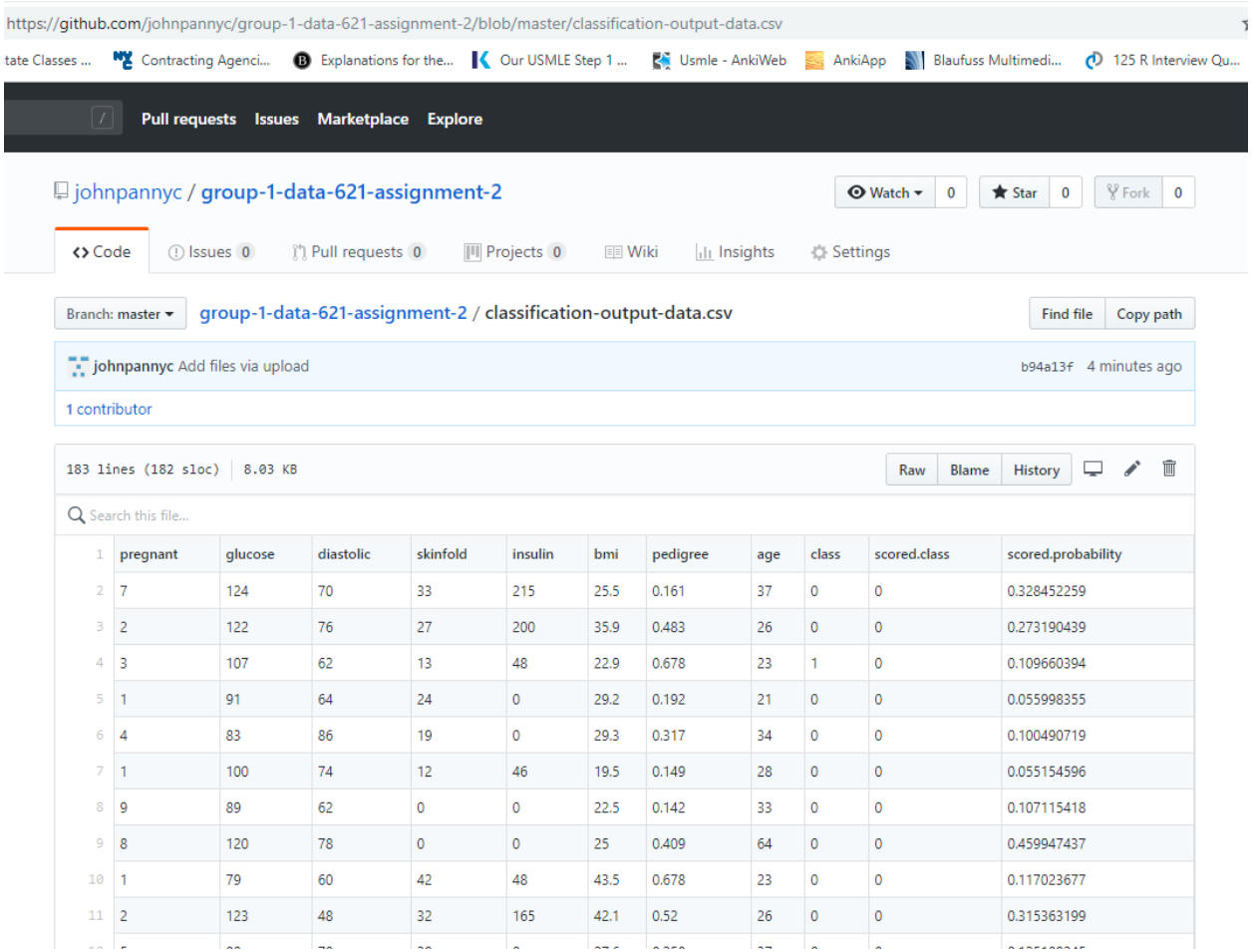
DATA 621 Assignment2

Joby John, Zachary Herold, Jun Pan

3-11-2019

Visualize Data from Github



https://github.com/johnpannyc/group-1-data-621-assignment-2/blob/master/classification-output-data.csv

tate Classes ... Contracting Agenci... Explanations for the... Our USMLE Step 1 ... Usmle - AnkiWeb AnkiApp Blaufuss Multimedi... 125 R Interview Qu...

Pull requests Issues Marketplace Explore

johnpannyc / group-1-data-621-assignment-2

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master group-1-data-621-assignment-2 / classification-output-data.csv Find file Copy path

johnpannyc Add files via upload b94a13f 4 minutes ago

1 contributor

183 lines (182 sloc) 8.03 KB Raw Blame History

Search this file...

	pregnant	glucose	diastolic	skinfold	insulin	bmi	pedigree	age	class	scored.class	scored.probability
2	7	124	70	33	215	25.5	0.161	37	0	0	0.328452259
3	2	122	76	27	200	35.9	0.483	26	0	0	0.273190439
4	3	107	62	13	48	22.9	0.678	23	1	0	0.109660394
5	1	91	64	24	0	29.2	0.192	21	0	0	0.055998355
6	4	83	86	19	0	29.3	0.317	34	0	0	0.100490719
7	1	100	74	12	46	19.5	0.149	28	0	0	0.055154596
8	9	89	62	0	0	22.5	0.142	33	0	0	0.107115418
9	8	120	78	0	0	25	0.409	64	0	0	0.459947437
10	1	79	60	42	48	43.5	0.678	23	0	0	0.117023677
11	2	123	48	32	165	42.1	0.52	26	0	0	0.315363199

Set working environment

```
---
title: "DATA 621 Assignment 2"
author: "Joby John, Zachary Herold, Jun Pan"
date: "March 11, 2019"
output: html_document
---

Set working environment
```{r, echo=FALSE, warning=FALSE, message=FALSE}
library(car)
library(caret)
library(corrplot)
library(data.table)
library(dplyr)
library(geoR)
library(ggplot2)
library(grid)
library(gridExtra)
library(knitr)
library(MASS)
library(naniar)
library(nortest)
library(psych)
library(testthat)
```

 'RandomFieldsUtils' will use OMP
```

Load data

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

1. DATA EXPLORATION

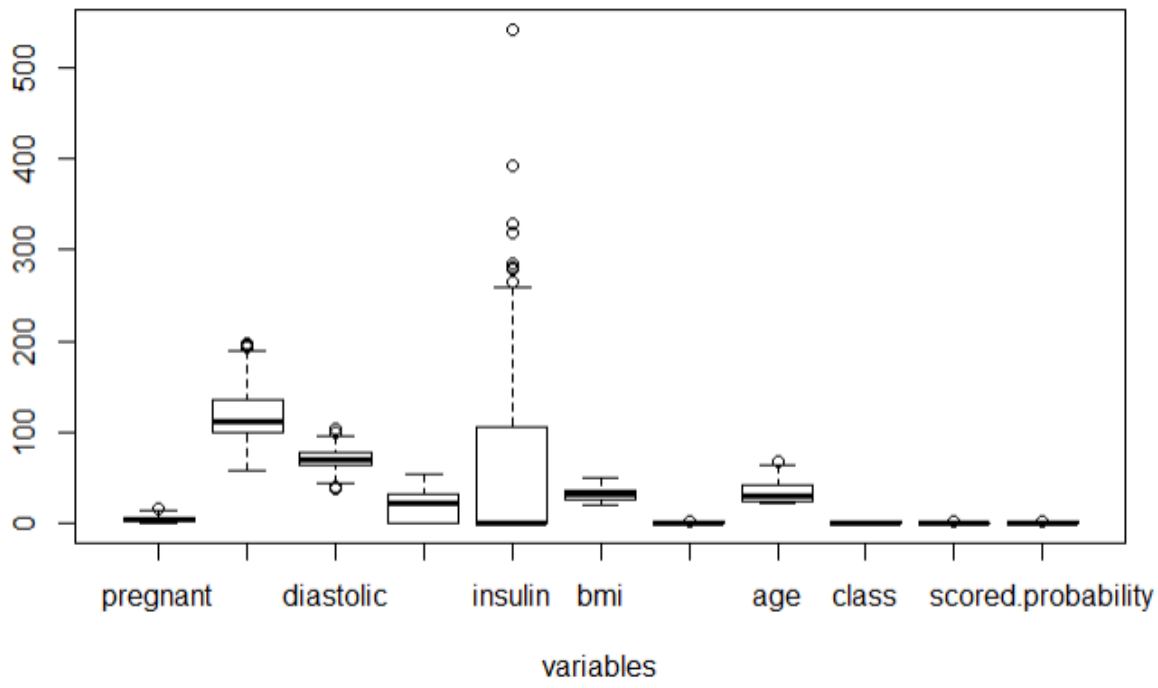
```{r}
glimpse(output)
```

```
Observations: 181
Variables: 11
$ pregnant           <int> 7, 2, 3, 1, 4, 1, 9, 8, 1, 2, 5, 5, 13, 0, 7, 12, 0, 3, 8, 5, 2, 3, 1, 1, 0, 3, 9, 0, 1, ...
$ glucose            <int> 124, 122, 107, 91, 83, 100, 89, 120, 79, 123, 88, 108, 76, 100, 194, 92, 173, 171, 196, 9...
$ diastolic          <int> 70, 76, 62, 64, 86, 74, 62, 78, 60, 48, 78, 72, 60, 70, 68, 62, 78, 72, 76, 74, 70, 62, 5...
$ skinfold           <int> 33, 27, 13, 24, 19, 12, 0, 0, 42, 32, 30, 43, 0, 26, 28, 7, 32, 33, 29, 27, 52, 0, 13, 0,...
$ insulin            <int> 215, 200, 48, 0, 0, 46, 0, 0, 48, 165, 0, 75, 0, 50, 0, 258, 265, 135, 280, 0, 57, 0, 50,...
$ bmi                <dbl> 25.5, 35.9, 22.9, 29.2, 29.3, 19.5, 22.5, 25.0, 43.5, 42.1, 27.6, 36.1, 32.8, 30.8, 35.9,...
$ pedigree           <dbl> 0.161, 0.483, 0.678, 0.192, 0.317, 0.149, 0.142, 0.409, 0.678, 0.520, 0.258, 0.263, 0.180...
$ age                <int> 37, 26, 23, 21, 34, 28, 33, 64, 23, 26, 37, 33, 41, 21, 41, 44, 58, 24, 57, 32, 25, 21, 2...
$ class              <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,...
$ scored.class       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,...
$ scored.probability <dbl> 0.32845226, 0.27319044, 0.10966039, 0.05599835, 0.10049072, 0.05515460, 0.10711542, 0.459...
```

```{r}
summary(output)
```

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

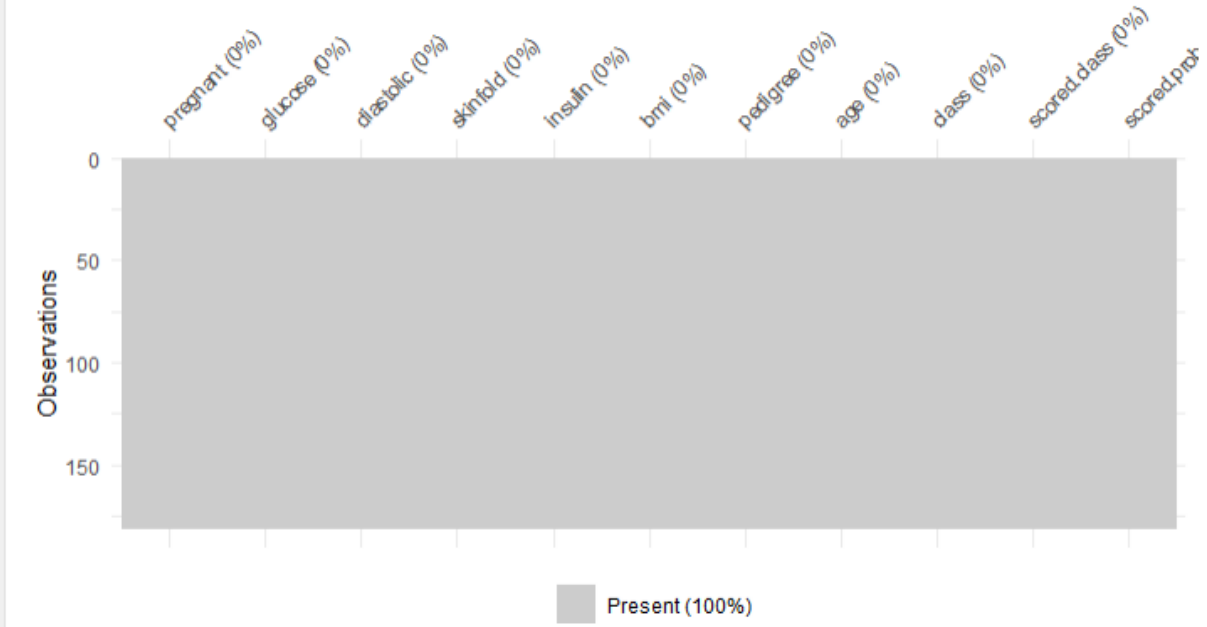
```r
boxplot(output,xlab="variables")
```



No missing data

Check missing data

```{r}
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)

```{r}
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:

119 true negative observations (TN)
5 false positive observations (FP)
30 false negative observations (FN)
27 true positive observations (TP)

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

```{r}
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

```{r}
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))
}
```

```{r}
Accuracy(output)
```

```
[1] 0.81
```

Classification_error_rate

```{r}
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))

}
```

```{r}
Classification_error_rate(output)
```

```
[1] 0.19
```

Precision

```r
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))

}
```

```r
Precision(output)
```

```
[1] 0.84
```

Sensitivity

```{r}
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))

}
```

```{r}
Sensitivity(output)
```

```
[1] 0.47
```

Specificity

```{r}
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))

}
```

```{r}
Specificity(output)
```

```
[1] 0.96
```

F1_Score

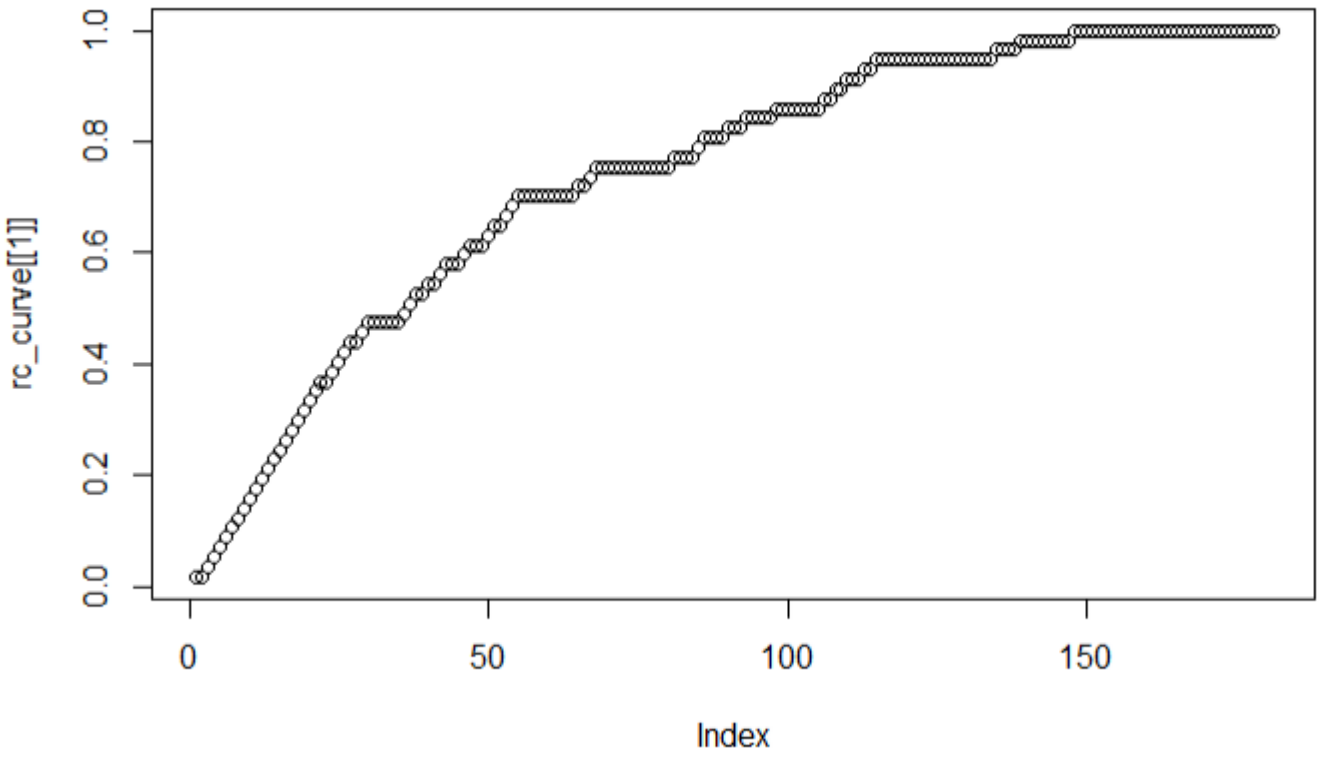
```{r}
F1_Score <- function(df)
{
  names     = c("class", "scored.class")
  cmatrix   = table(df[, names])
  precision = Precision(df)
  sensitivity = Sensitivity(df)
  f1_score =  (2 * precision * sensitivity) /(precision + sensitivity)

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

```{r}
F1_Score(output)
```

```
[1] 0.6
```

Manually Create ROC Curve

````
10.Manually create ROC curve
```{r}
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]]


```
````


rc_curve[[1]]
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
Index

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.

```{r}
Accuracy(output)
Classification_error_rate(output)
Precision(output)
Sensitivity(output)
Specificity(output)
F1_Score(output)
```

```
[1] 0.81
[1] 0.19
[1] 0.84
[1] 0.47
[1] 0.96
[1] 0.6
```

Q12 - Investigating the caret package

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

ls(pos = "package:caret")

?sensitivity
?confusionMatrix
?precision
```

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

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

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

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

```{r}
sens.caret <- round(sensitivity(cmatrix.t, positive = rownames(cmatrix)[2]),2)
sens.caret
identical(Sensitivity(data), sens.caret)

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

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

str(cMat.caret)

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

acc.caret <- round(cMat.caret$overall[1],2)
acc.caret
identical(Accuracy(data), acc.caret)  ## same value, but fail to match with identical function
```

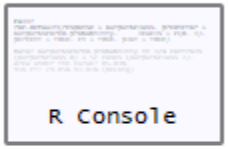
```
[1] 0.47
[1] TRUE
[1] 0.96
[1] TRUE
Confusion Matrix and Statistics

            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
```

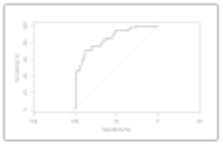
```
List of 6
 $ positive: chr "1"
 $ table   : 'table' int [1:2, 1:2] 119 5 30 27
  ..- attr(*, "dimnames")=List of 2
  .. ..$ scored.class: chr [1:2] "0" "1"
  .. ..$ class       : chr [1:2] "0" "1"
 $ overall : Named num [1:7] 0.807 0.492 0.741 0.861 0.685 ...
  ..- attr(*, "names")= chr [1:7] "Accuracy" "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 Value" ...
 $ mode    : chr "sens_spec"
 $ dots    : list()
 - attr(*, "class")= chr "confusionMatrix"
[1] 0.84
[1] TRUE
Accuracy
    0.81
[1] FALSE
```

pROC package for ROC curve

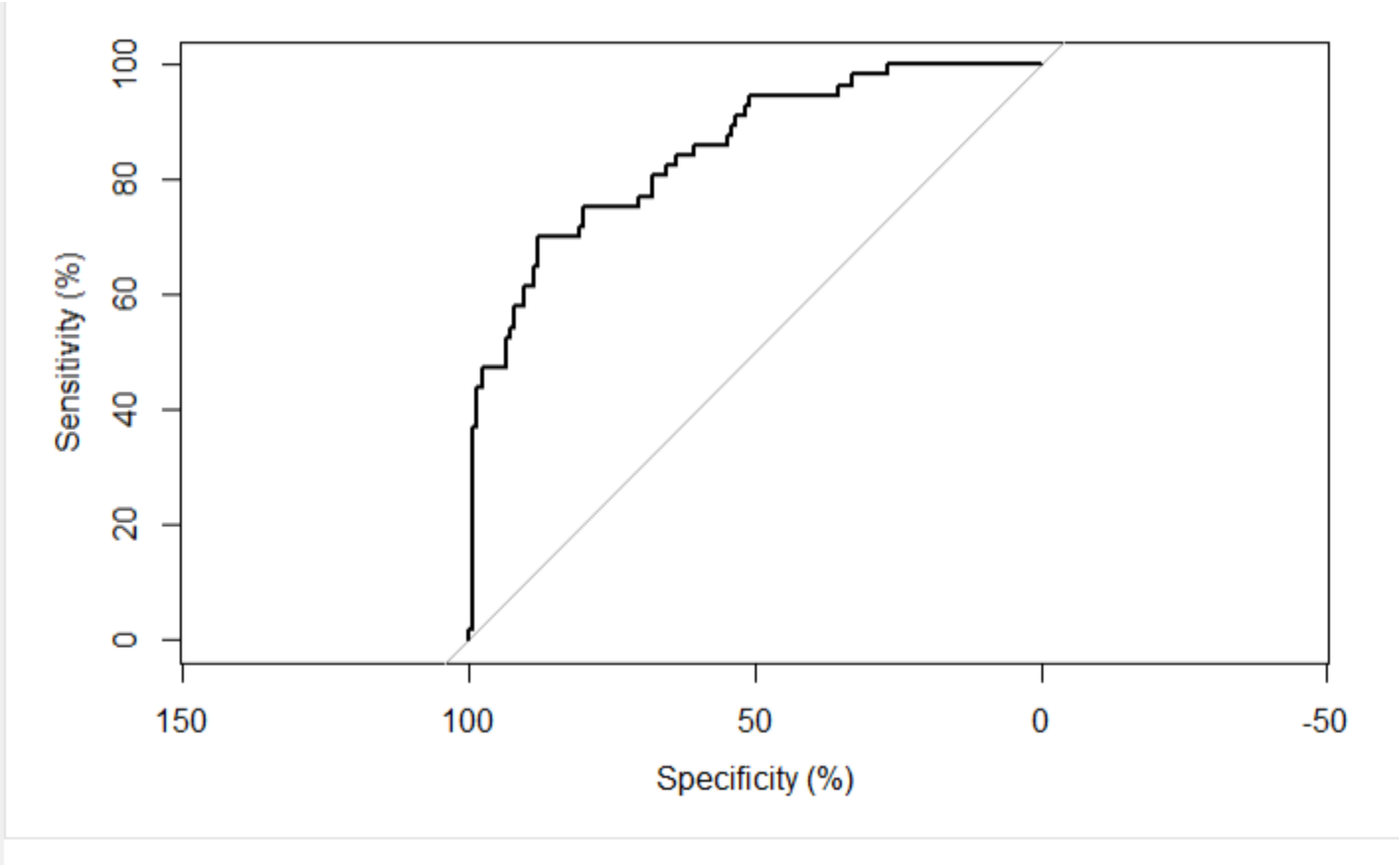
13. pROC Package
Let us try the pROC package.

```{r}
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)
```


Sensitivity (%)
0
20
40
60
80
100
150
100
50
0
-50
Specificity (%)