Homework 2 (Group 5)

Applied Predictive Modeling

Maria A Ginorio

3/20/2022

Contents

1. Data

Download the classification output data set.

```r
class_data <- read_csv("classification-output-data.csv")
kable(skim(class_data))
```

skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0	numeric.p25	nu
numeric	pregnant	0	1	3.8618785	3.2365508	0.000000	1.0000000	
numeric	glucose	0	1	118.3038674	30.4840839	57.000000	99.0000000	11
numeric	diastolic	0	1	71.7016575	11.8029868	38.000000	64.0000000	7
numeric	skinfold	0	1	19.8011050	15.6923265	0.000000	0.0000000	2
numeric	insulin	0	1	63.7679558	88.7347563	0.000000	0.0000000	
numeric	bmi	0	1	31.5779006	6.6599348	19.400000	26.3000000	3
numeric	pedigree	0	1	0.4496409	0.2840056	0.085000	0.2570000	
numeric	age	0	1	33.3149171	11.1835816	21.000000	24.0000000	3
numeric	class	0	1	0.3149171	0.4657713	0.000000	0.0000000	
numeric	scored.class	0	1	0.1767956	0.3825539	0.000000	0.0000000	
numeric	scored.probability	0	1	0.3037256	0.2312346	0.023228	0.1170237	

2. Key Columns

The data set has three key columns we will use:

- class: the actual class for the observation
- scored.class: the predicted class for the observation (based on a threshold of 0.5)
- scored.probability: the predicted probability of success for the observation

Use the table() function to get the raw confusion matrix for this scored dataset. Make sure you understand the output.

In particular, do the rows represent the actual or predicted class? The columns?

The confusion Matrix is nothing but a table that represents the performance of a logistic regression model.

Rows	the actual class, the real value
Columns	predicted class

```
## # A tibble: 181 x 3
##    Actual Predicted Pred_prop
##     <dbl>     <dbl>     <dbl>
##  1      0         0    0.328
##  2      0         0    0.273
##  3      1         0    0.110
##  4      0         0    0.0560
##  5      0         0    0.100
##  6      0         0    0.0552
##  7      0         0    0.107
##  8      0         0    0.460
##  9      0         0    0.117
## 10      0         0    0.315
## # ... with 171 more rows
```

2.1 Confusion Matrix (1)

- There are two possible predicted classes: "yes" and "no".
- The classifier made a total of 181 predictions (e.g., 181 were being tested for the presence of that disease, in this case diabetes).
- Out of those 181 cases, the classifier predicted "yes" 32 times, and "no" 149 times.

In reality, 57 patients in the sample have the disease, and 124 patients do not.

```
class_data$Actual <- factor(class_data$class,
                        levels = c(1,0),
                        labels = c('Diabetes', 'No Diabetes'))

class_data$Predicted <- factor(class_data$scored.class,
                           levels = c(1,0),
                           labels = c('Diabetes', 'No Diabetes'))
```

```r
c_matrix <- with(class_data, table(Actual, Predicted))

c_matrix
```

```
##              Predicted
## Actual        Diabetes No Diabetes
##   Diabetes          27          30
##   No Diabetes        5         119
```

True Negatives	TN (0-0)	119 - Predicted NO, they DO NOT have the disease
True Positive	TP (1-1)	27 - Predicted YES, they DO have diabetes
False Negative	FN (1-0)	30- Predicted NO, but they DO HAVE diabetes
False Negative	FP (0 -1)	5 - Predicted YES, but they DO NOT have diabetes

3. Accuracy Function

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the accuracy of the predictions.

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

```
accuracy  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  Acc <- (TP + TN)/(TP + FP + TN + FN)
  return(Acc)

}
```

3.1 Accuracy Result

```
a <- accuracy(class_data, Actual = 'class', Predicted = 'scored.class')
a
```

```
## [1] 0.8232044
```

4. Classification Error Rate

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the classification error rate of the predictions.

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

```
c_e_r  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  Cer <- (FP + FN)/(TP + FP + TN + FN)
  return(Cer)

}
```

4.1 Classification Error Rate Result

```
cer <- c_e_r(class_data, Actual = 'class', Predicted = 'scored.class')
cer
```

```
## [1] 0.1767956
```

4.2 Error Rate Verification

Verify that you get an accuracy and an error rate that sums to one

```
a + cer
```

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

5. Precision

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the precision of the predictions.

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

```r
precision  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  Prec <- (TP)/(TP + FP)
  return(Prec)

}
```

```r
prec <- precision(class_data, Actual = 'class', Predicted = 'scored.class')
prec
```

```
## [1] 0.5263158
```

6. Sensitivity

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the sensitivity of the predictions. Sensitivity is also known as recall.

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

```
sensitivity  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  Sens <- (TP)/(TP + FP)
  return(Sens)

}
```

```
sens <- sensitivity(class_data, Actual = 'class', Predicted = 'scored.class')
sens
```

```
## [1] 0.5263158
```

7. Specificity

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the specificity of the predictions.

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

```
specificity  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  Spec <- (TP)/(TP + FP)
  return(Spec)

}
```

```
specif <- specificity(
  class_data, Actual = 'class',
  Predicted = 'scored.class')

specif
```

```
## [1] 0.5263158
```

8. F1 Score

Write a function that takes the data set as a data frame, with actual and predicted classifications identified, and returns the F1 score of the predictions

$$F1Score = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity}$$

```
f1_score  <- function(class_data, Actual, Predicted) {

  TN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 0)
  TP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 0)
  FN <- sum(class_data[Actual] == 0 & class_data[Predicted] == 1)
  FP <- sum(class_data[Actual] == 1 & class_data[Predicted] == 1)

  F1 <- (2 * prec * sens)/(prec + sens)
  return(F1)

}
```

```
f1_score <-f1_score(
  class_data, Actual = 'class',
  Predicted = 'scored.class')


f1_score
```

```
## [1] 0.5263158
```

9. F1 Score Bounds

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.

(Hint: If $0 < \quad < 1$ and $0 < \quad < 1$ then $\quad < \quad$.)

```r
f1_bounds <- function(p,s) {
  f1 <- (2 * p * s) / (p + s)
  return(f1)
}
```

```r
#lower
f1_bounds(0.01, 0.01)
```

```
## [1] 0.01
```

```r
#upper
f1_bounds(0.99, 0.99)
```

```
## [1] 0.99
```

```r
# ab < a
f1_bounds(0.01, 0.99)
```

```
## [1] 0.0198
```

```r
f1_bounds(0.99, 0.01)
```

```
## [1] 0.0198
```

```r
(0.01 * 0.99)
```

```
## [1] 0.0099
```

10. ROC Curve

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).

Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area under the curve (AUC).

Note that I recommend using a sequence of thresholds ranging from 0 to 1 at 0.01 intervals

10.1 Creating a ROC Curve

A Receiver Operator Characteristic (ROC) curve is a graphical plot used to show the diagnostic ability of binary classifiers. It was first used in signal detection theory but is now used in many other areas such as medicine, radiology, natural hazards and machine learning.

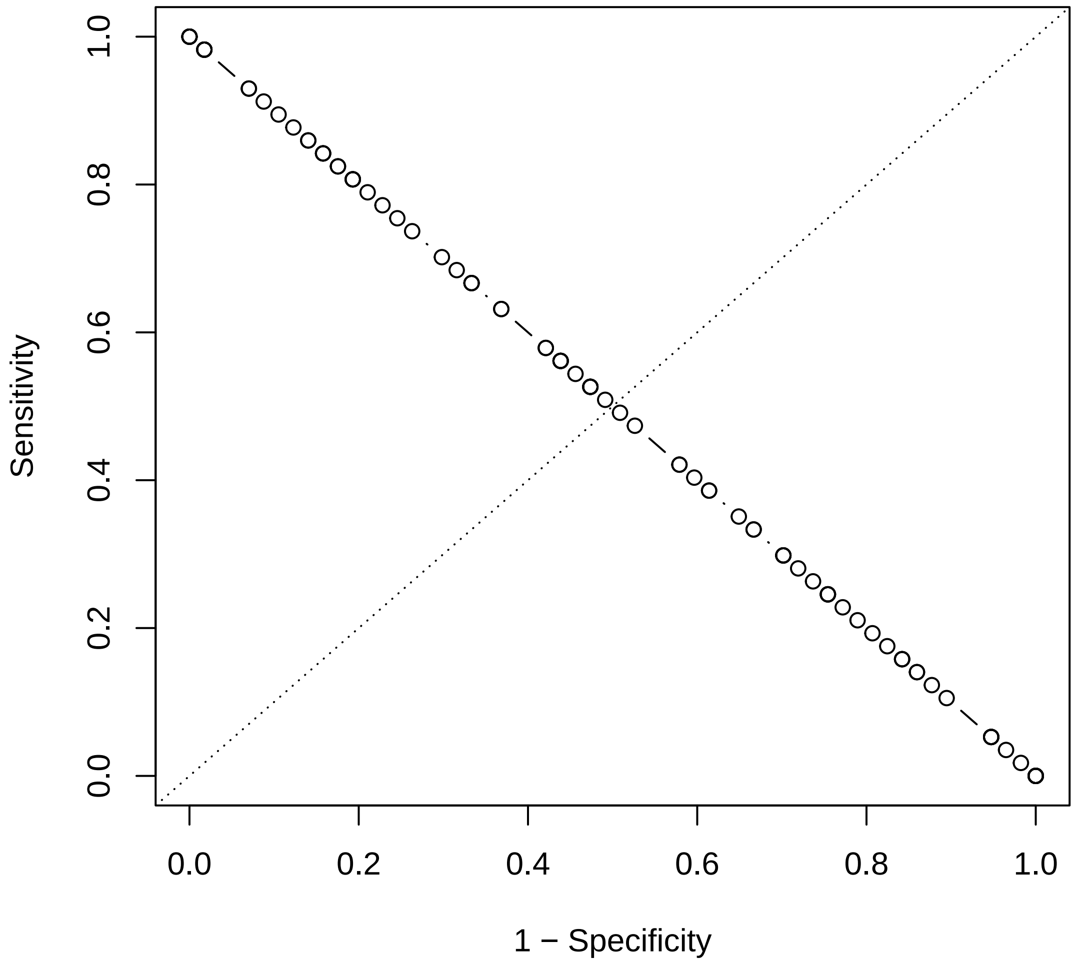
A ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR).

TPR - The true positive rate is the proportion of observations that were correctly predicted to be positive out of all positive observations (TP/(TP + FN)). - **Sensitivity**

FPR - Similarly, the false positive rate is the proportion of observations that are incorrectly predicted to be positive out of all negative observations (FP/(TN + FP)).

For example, in medical testing, the true positive rate is the rate in which people are correctly identified to test positive for the disease in question (Diabetes in our case)

```
thresholds <- seq(0,1,.01)

# FP - Specificity
x <- NULL
# TP - Sensitivity
y <- NULL


for (i in thresholds) {

  temp <- class_data
  temp$pred <- ifelse(temp$scored.probability >= i, 1, 0)

  # false positive rate FPT
  spec <- specificity(temp, Actual = 'class', Predicted = 'pred')
  x <- append(x, 1 - spec)

  # true positive rate TPR
  sensi <- sensitivity(temp, Actual = 'class', Predicted = 'pred')
  y <- append(y, sensi)

  rm(temp, sensi, spec)
}

plot(x, y, type = 'b', xlab = '1 - Specificity', ylab = 'Sensitivity')
abline(0,1, lty=3)
```



Sensitivity
1 − Specificity
0.0
0.2
0.4
0.6
0.8
1.0

11. Classification Output

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

```
c_output <- tibble(a, cer, prec, sens, specif, f1_score)

c_output
```

```
## # A tibble: 1 x 6
##       a   cer  prec  sens specif f1_score
##   <dbl> <dbl> <dbl> <dbl>  <dbl>    <dbl>
## 1 0.823 0.177 0.526 0.526  0.526    0.526
```

12. Caret Package

nvestigate the caret package. In particular, consider the functions confusionMatrix, sensitivity, and specificity. Apply the functions to the data set. How do the results compare with your own functions?

12.1 Confusion Matrix (Caret)

```
library(caret)

confusionMatrix(data = class_data$Predicted,
                reference = class_data$Actual,
                positive = 'Diabetes')
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    Diabetes No Diabetes
##   Diabetes          27           5
##   No Diabetes       30         119
##
##                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 : Diabetes
##
```

12.2 Comparison

Even though the confusion matrix seems to have the same values the the sensitivity and specificity are very different in values.

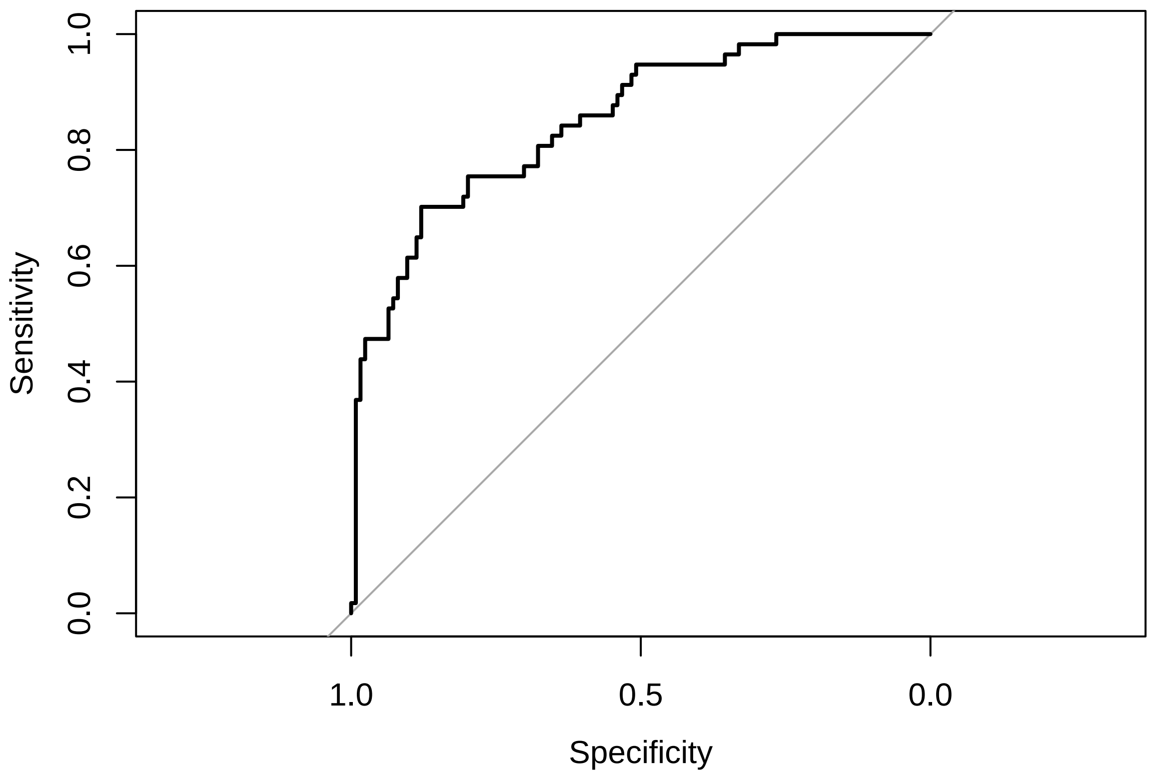
Created	Value	Caret	Value
Confusion Matrix	Same	Confusion Matrix	Same
Sensitivity	0.5263158	Sensitivity	0.4737
Specificity	0.5263158	Specificity	0.9597

13. pROC Package

Investigate the pROC package. Use it to generate an ROC curve for the data set.

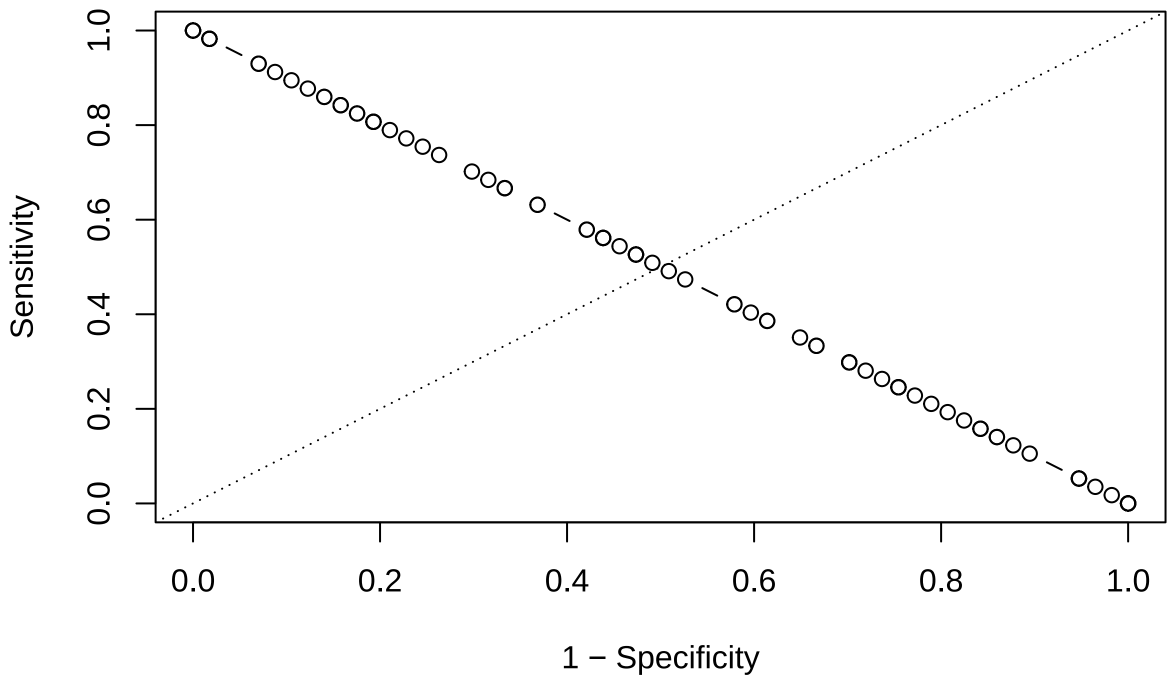
How do the results compare with your own functions?

```
library(pROC)

roc(class_data$class, class_data$scored.probability, plot = T, auc = T)
```




```
##
## Call:
## roc.default(response = class_data$class, predictor = class_data$scored.probability,     auc = T, plot =
##
## Data: class_data$scored.probability in 124 controls (class_data$class 0) < 57 cases (class_data$class 1
## Area under the curve: 0.8503
```

```
plot(x, y, type = 'b', xlab = '1 - Specificity', ylab = 'Sensitivity')
abline(0,1, lty=3)
```



Sensitivity
1 − Specificity
0.0
0.2
0.4
0.6
0.8
1.0