Data 621 HW 2 - Confusion Matrix

Monu Chacko

3/15/2020

```
require("plyr")
require("knitr")
require("psych")
require("knitr")
require("ggplot2")
require("pracma")
require("caret")
require("pROC")
```

1. Download the classification output data set(attached in Blackboard to the assignment).

```
df <- read.csv(url('https://raw.githubusercontent.com/monuchacko/cuny_msds/master/data_621/Homework2/cla

kable(head(df))
```

pregnant	glucose	diastolic	skinfold	insulin	bmi	pedigree	age	class	scored.class	scored.probability
7	124	70	33	215	25.5	0.161	37	0	0	0.3284523
2	122	76	27	200	35.9	0.483	26	0	0	0.2731904
3	107	62	13	48	22.9	0.678	23	1	0	0.1096604
1	91	64	24	0	29.2	0.192	21	0	0	0.0559984
4	83	86	19	0	29.3	0.317	34	0	0	0.1004907
1	100	74	12	46	19.5	0.149	28	0	0	0.0551546

2. The dataset 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?

```
kable(table(df$class,df$scored.class))
```

	0	1
0	119	5
1	30	27

Here 0 represents non event and 1 represents event.

3. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the accuracy of the predictions.

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

```
confusion_matrix <- function(df){
  data.frame(TP=nrow(df[df$class==1 & df$scored.class==1,]),
             TN=nrow(df[df$class==0 & df$scored.class==0,]),
             FP=nrow(df[df$class==0 & df$scored.class==1,]),
             FN=nrow(df[df$class==1 & df$scored.class==0,])
  )
}
kable(confusion_matrix(df))
```

TP	TN	FP	FN
27	119	5	30

```
accuracy<-function(df){
  f <- confusion_matrix(df)
  (f$tp+f$tn)/(f$tp+f$fp+f$tn+f$fn)
}
accuracy(df)
```

```
## numeric(0)
```

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

true negatives (TN): We predicted no, and they don't have the disease.

false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

4. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the classification error rate of the predictions.

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

```
classification_error<-function(df){
  f <- confusion_matrix(df)
  (f$fp+f$fn)/(f$tp+f$fp+f$tn+f$fn)
}
classification_error(df)
```

```
## numeric(0)
```

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

```
sum <- classification_error(df) + accuracy(df)
sum
```

```
## numeric(0)
```

5. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the precision of the predictions.

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

```
precision<-function(df){
  f <- confusion_matrix(df)
  (f$tp)/(f$tp+f$fp)
}
precision(df)
```

```
## numeric(0)
```

6. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the sensitivityof the predictions.Sensitivity is also known as recall.

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

```
sensitivity<-function(df){
  f <- confusion_matrix(df)
  (f$tp)/(f$tp+f$fn)
}
sensitivity(df)
```

```
## numeric(0)
```

7. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the specificity of the predictions

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

```
specificity<-function(df){
  f <- confusion_matrix(df)
  (f$tn)/(f$tn+f$fp)
}
specificity(df)
```

```
## numeric(0)
```

8. Write a function that takes the dataset as a dataframe, with actual and predicted classifications identified, and returns the F1 score of the predictions.

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

```
f1_score<-function(df){
  p<- precision(df)
  s<- sensitivity(df)
  2*p*s/(p+s)
}
f1_score(df)
```

```
## numeric(0)
```

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. (Hint: If 0 < a <1 and 0 < b <1 then ab < a.)

```r
# assume p is prescision and s is sensitivity.
p <- runif(100, min = 0, max = 1)
s <- runif(100, min = 0, max = 1)
f <- (2*p*s)/(p+s)
summary(f)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006338 0.155156 0.420184 0.403983 0.577932 0.908512
```

10. Write a function that generates an ROC curve from a dataset 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). Notethat I recommend using a sequence of thresholds ranging from 0 to 1 at 0.01 intervals.

```r
ROC <- function(df)
{
  data1 = df
  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-specificity(data1))
    Y <- append(Y,sensitivity(data1))
    }
  df1 <- data.frame(X=X,Y=Y)
  df1 <- na.omit(df1)
  g <- ggplot(df1,aes(X,Y)) + geom_line() + ggtitle('Custom ROC Curve') +
    xlab('Specificity') + ylab('Sensitivity')
  height = (df1$Y[-1]+df1$Y[-length(df1$Y)])/2
  width = -diff(df1$X)
  area = round(sum(height*width),4)
  return(list(Plot =g,AUC = area))
}
```

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

```r
Name <- c('Accuracy','Classification Error Rate', 'Precision', 'Sensitivity','Specificity', 'F1 Score')
Value <- round(c(accuracy(df), classification_error(df), precision(df), sensitivity(df), specificity(df)
df1 <- as.data.frame(cbind(Name, Value))
kable(df1)
```

Name
Accuracy
Classification Error Rate
Precision

Name
Sensitivity
Specificity
F1 Score

12. Investigate 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?

```
d_tab <- table(df$class,df$scored.class)
confusionMatrix(d_tab, reference = df$class)
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 119   5
##   1  30  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.8232
##     P-Value [Acc > NIR] : 0.7559
##
##                   Kappa : 0.4916
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.7987
##             Specificity : 0.8438
##          Pos Pred Value : 0.9597
##          Neg Pred Value : 0.4737
##              Prevalence : 0.8232
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.6851
##       Balanced Accuracy : 0.8212
##
##        'Positive' Class : 0
##
```

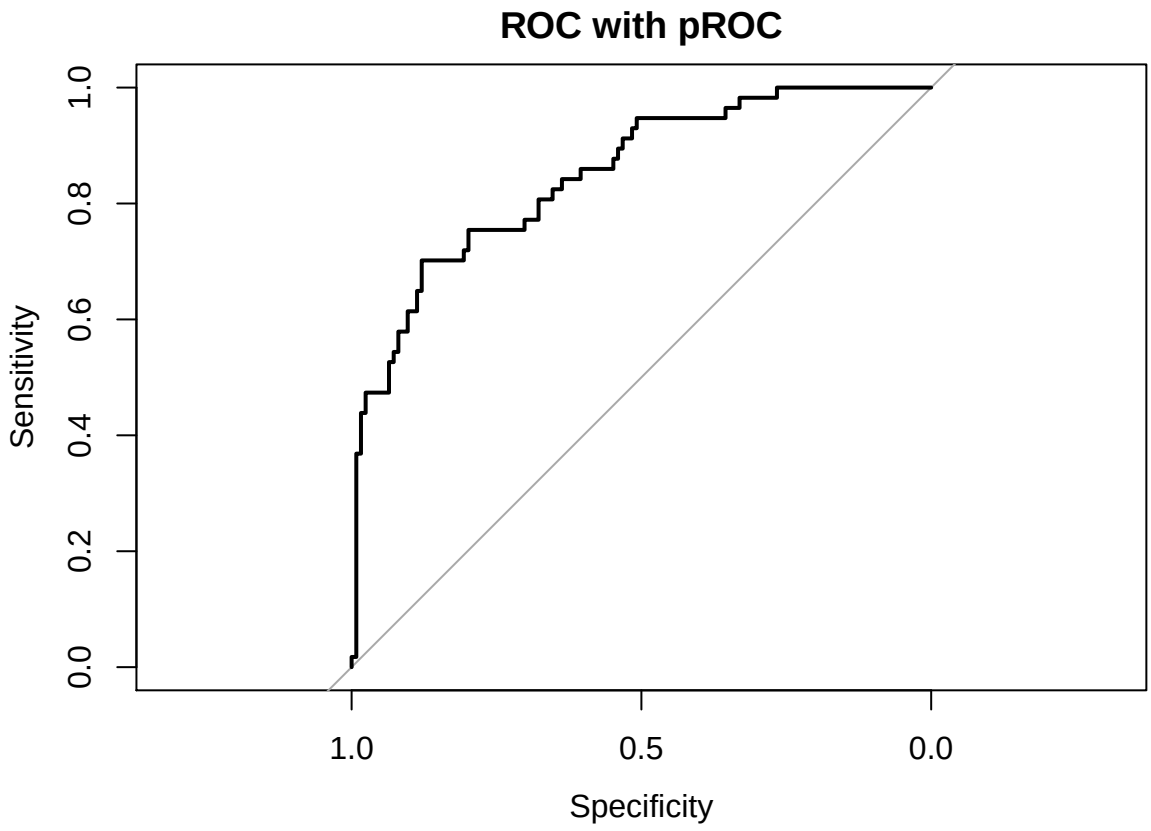
13. Investigate the pROC package. Use it to generate an ROC curve for the data set. How do the results compare with your own functions?

```
d_roc <- roc(df$class,df$scored.probability)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
plot(d_roc, main = "ROC with pROC")
```




```r
ci(d_roc)
```

```
## 95% CI: 0.7905-0.9101 (DeLong)
```

Useful Resource: https://www.datacamp.com/community/tutorials/confusion-matrix-calculation-r