Course Details

Instructor Name: Dr. Nathan Bastian (Prof. B)
Course Name: Big Data Econometrics
Course Number: ADEC 7430

Overview

In this individual project, you will work through various classification metrics. You will be asked to create functions in R to carry out the various calculations. You will also investigate some functions in packages that will let you obtain the equivalent results. Finally, you will create graphical output that also can be used to evaluate the output of classification models, such as binary logistic regression.

Supplemental Material

- *Applied Predictive Modeling*, Ch. 11 (provided as a PDF file).
- *An Introduction to ROC Analysis* (provided as a PDF file).
- Web tutorials: http://www.saedsayad.com/model_evaluation_c.htm

Deliverables (60 Points)

- Upon following the instructions below, use your created R functions and the other packages to generate the classification metrics for the provided data set. A write-up of your solutions submitted in PDF format.

Instructions

Complete each of the following steps as instructed:

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

```
class.output <- read.csv("./classification-output-data.csv", header = T)
fix(class.output)
names(class.output)
str(class.output)
```

2. 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?

```
# Use the table() function to get the raw confusion matrix for this scored dataset.
confMat <- table(class.output$scored.class, class.output$class)
# The columns show us the actual classification, and the rows are the predicted classes.
```

confMat

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

```
# Write a function that returns the confusion matrix
genConfMat <- function(df, actual, predicted){
  confMat <- matrix(table(df[,actual], df[,predicted]), ncol=2, nrow=2)
  retDf <- data.frame(matrix(ncol=4, nrow=1))
  names(retDf) <- c("tn", "fp", "fn", "tp")
  retDf$tn <- confMat[1,1]
  retDf$fp <- confMat[1,2]
  retDf$fn <- confMat[2,1]
  retDf$tp <- confMat[2,2]
  return(retDf)
}
```

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.

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

```
accuracy <- function(df, actual, predicted){
  confDf <- genConfMat(df, actual, predicted)
  acc <- (confDf$tp + confDf$tn)/sum(confDf)
  return(acc)
}
```

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.

$$Classification\ Error\ Rate = \frac{FP + FN}{TP + FP + TN + FN}$$

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

```
error.rate <- function(df, actual, predicted){
  confDf <- genConfMat(df, actual, predicted)
  er <- (confDf$fp + confDf$fn)/sum(confDf)
  return(er)
}
```

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.

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

```
precision <- function(df, actual, predicted){
```

```
 confDf <- genConfMat(df, actual, predicted)
 pr <- confDf$tp/(confDf$tp + confDf$fp)
 return(pr)
}
```

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.

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

```
sensitivity <- function(df, actual, predicted){
 confDf <- genConfMat(df, actual, predicted)
 se <- confDf$tp/(confDf$tp + confDf$fn)
 return(se)
}
```

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.

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

```
specificity <- function(df, actual, predicted){
 confDf <- genConfMat(df, actual, predicted)
 sp <- confDf$tn/(confDf$tn + confDf$fp)
 return(sp)
}
```

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.

$$F1\ Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}$$

```
f.one <- function(df, actual, predicted){
 pr <- precision(df, actual, predicted)
 se <- sensitivity(df, actual, predicted)
 return((2 * pr * se)/(pr + se))
}
```

9. Let's consider the following question: 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 $ab < a$.)

Through algebraic manipulation, we can see that the F1 is score is equal to $\frac{2 \times TP}{2 \times TP + FP + FN}$. Thus, a perfect classification model would give $\frac{2 \times TP}{2 \times TP} = 1$, and one that is perfectly bad would give $\frac{0}{0 + FP + FN} = 0$.

10. Write a function that generates an ROC curve from a data set with a true classification column (i.e., class) and a probability column (i.e., scored.probability). Your function should return the plot of the ROC curve and the calculated area under the ROC curve (AUC). Note that I recommend using a sequence of thresholds ranging from 0 to 1 at 0.01 intervals.

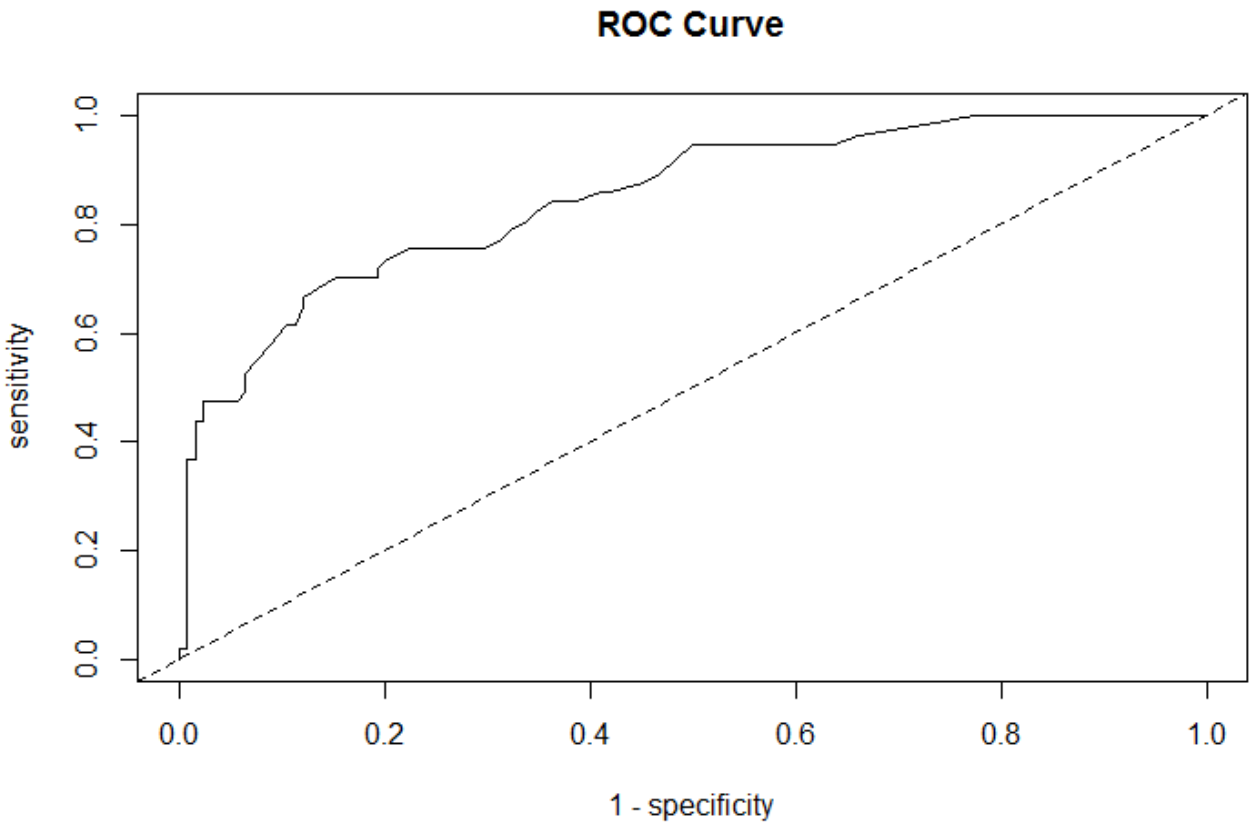
```
roc.curve <- function(predprob, class){
  cutpoints <- seq(0,1,by=0.01)
  sensitivity <- sapply(cutpoints,
                        function(result) mean(predprob>result & class)/mean(class))
  specificity <- sapply(cutpoints,
                        function(result) mean(predprob<=result & !class)/mean(!class))
  calc.auc <- function(predprob, class){
    N <- length(predprob)
    N_pos <- sum(class)
    df <- data.frame(out=class, prob=predprob)
    df <- df[order(-df$prob),]
    df$above <- (1:N) - cumsum(df$out)
    return(1-sum(df$above*df$out)/(N_pos *(N-N_pos)))
  }
  plot(1-specificity, sensitivity, type="l", main = "ROC Curve")
  abline(0,1,lty=2)
  return(list(auc=calc.auc(predprob, class)))
}
```

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

```
accuracy(class.output, 9, 10)        # accuracy
error.rate(class.output, 9, 10)      # classification error rate
sum(accuracy(class.output, 9, 10),
    error.rate(class.output, 9, 10))  # accuracy + error = 1
precision(class.output, 9, 10)       # precision
sensitivity(class.output, 9, 10)     # sensitivity
specificity(class.output, 9, 10)     # specificity
f.one(class.output, 9, 10)           # F1 score
roc.curve(class.output$scored.probability,
          class.output$class)        # ROC Curve and AUC
```

```
> accuracy(class.output, 9, 10)        # accuracy
[1] 0.8066298
> error.rate(class.output, 9, 10)      # classification error rate
[1] 0.1933702
> sum(accuracy(class.output, 9, 10),
+     error.rate(class.output, 9, 10))  # accuracy + error = 1
[1] 1
> precision(class.output, 9, 10)       # precision
```

```
[1] 0.84375
> sensitivity(class.output, 9, 10)      # sensitivity
[1] 0.4736842
> specificity(class.output, 9, 10)      # specificity
[1] 0.9596774
> f.one(class.output, 9, 10)           # F1 score
[1] 0.6067416
> roc.curve(class.output$scored.probability,
+           class.output$class)         # ROC Curve and AUC
$auc
[1] 0.8503113
```



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?

```
library(caret)
confusionMatrix(data = class.output$scored.class, reference = class.output$class, positive = '1')
sensitivity(data = as.factor(class.output$scored.class), reference = as.factor(class.output$class), positive = '1')
specificity(data = as.factor(class.output$scored.class), reference = as.factor(class.output$class), negative = '0')
```

```
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
```

```
> sensitivity(data = as.factor(class.output$scored.class), reference = as.factor(class.output$class), positive = '1')
[1] 0.4736842
> specificity(data = as.factor(class.output$scored.class), reference = as.factor(class.output$class), negative = '0')
[1] 0.9596774
```

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)
myRoc <- roc(response = class.output$class, predictor = class.output$scored.probability)
plot(myRoc)

> plot(myRoc)

Call:
roc.default(response = class.output$class, predictor = class.output$scored.probability)
```

Data: class.output$scored.probability in 124 controls (class.output$class 0) < 57 cases (class.output$class 1).
Area under the curve: 0.8503