HW2_Rnash

Raphael Nash

3/2/2018

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
```

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

```
data_df <- read.csv("classification-output-data.csv")
```

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?

```
data_df <- data_df [ , c("class", "scored.class", "scored.probability" )]

as.data.frame(table(Actual=data_df$class, Precidted=data_df$scored.class))
```

```
##   Actual Precidted Freq
## 1      0         0  119
## 2      1         0   30
## 3      0         1    5
## 4      1         1   27
```

Row1 True Negatives Row2 False Negatives Row3 False Positives Row 4 True positives

Cell 1,1 is True Negatices Cell 1,2 is False Positives Cell 2,1 is False Negatives Cell 2,2 is True Positives

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.

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

```
calculate_accuracy  <- function(classification){
  confusion <- as.data.frame(table(Actual=classification$class, Precidted=classification$scored.class))
   return((confusion$Freq[1] + confusion$Freq[4])/sum(confusion$Freq))
```

```
}


accuracy <- calculate_accuracy (data_df)
accuracy
```

```
## [1] 0.8066298
```

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.

$$classification_error = /fracFP + FNTP + FP + TN + FN$$

```
calculate_class_error  <- function(classification) {
   confusion<-  as.data.frame(table(Actual=classification$class, Precidted=classification$scored.class))
 return((confusion$Freq[2] + confusion$Freq[3])/sum(confusion$Freq))
}


class_error <- calculate_class_error (data_df )
class_error
```

```
## [1] 0.1933702
```

```
class_error + accuracy
```

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

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.

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

```
calculate_precision  <- function(classification){
    confusion <-  as.data.frame(table(Actual=classification$class, Precidted=classification$scored.class
 return(confusion$Freq[4]/(confusion$Freq[4]+confusion$Freq[3]))
}


precision <- calculate_precision (data_df )
precision
```

```
## [1] 0.84375
```

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.

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

```
calculate_sensitivity  <- function(classification){
  confusion <- as.data.frame(table(Actual=classification$class, Precidted=classification$scored.class))
 return(confusion$Freq[4]/(confusion$Freq[4] + confusion$Freq[2]))
}
```

```
sensitivity <- calculate_sensitivity (data_df)
sensitivity
```

```
## [1] 0.4736842
```

```
sens <- function(t)
{
  confusion <- as.data.frame (table(Actual=data_df[,c("class")], Predicted=data_df[,c("scored.class")])
return(confusion$Freq[1]/(confusion$Freq[1]+confusion$Freq[3]))
}
sens(data_df)
```

```
## [1] 0.9596774
```

Cell 1,1 is True Negatives Cell 1,2 is False Positives Cell 2,1 is False Negatives Cell 2,2 is True Positives

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.

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

```
calculate_specificity <- function(classification){

   confusion <-  as.data.frame(table(Actual=classification$class, Precidted=classification$scored.class
 return(confusion$Freq[1]/(confusion$Freq[1]+confusion$Freq[3]))
}


#specificity <- calculate_specificity (data_df , "class", "scored.class")
specificity <- calculate_specificity(data_df)
specificity
```

```
## [1] 0.9596774
```

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.

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

```
calculate_f1  <- function(classification){
  sensitivity <- calculate_sensitivity (classification  )
  precision <- calculate_precision (classification  )
  f1 <- ( 2 * precision * sensitivity ) / (precision + sensitivity )
  f1
}


f1 <- calculate_f1 (data_df )
f1
```

```
## [1] 0.6067416
```

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.

If you re-write the f1 score as $\frac{ps+ps}{p+s}$ you have a standard fraction. Since the top of the fraction is always greater than the bottom it is always less

If percision and sensitivity are less then 1, but greater than 0 then the numberator of the F1 equation will always be less then the denomonator, due to multipling the Precision and Sensitivity (as decimals) vs adding decimals. When you multple decimals you get a smaller number when you add decimals you get a larger number.

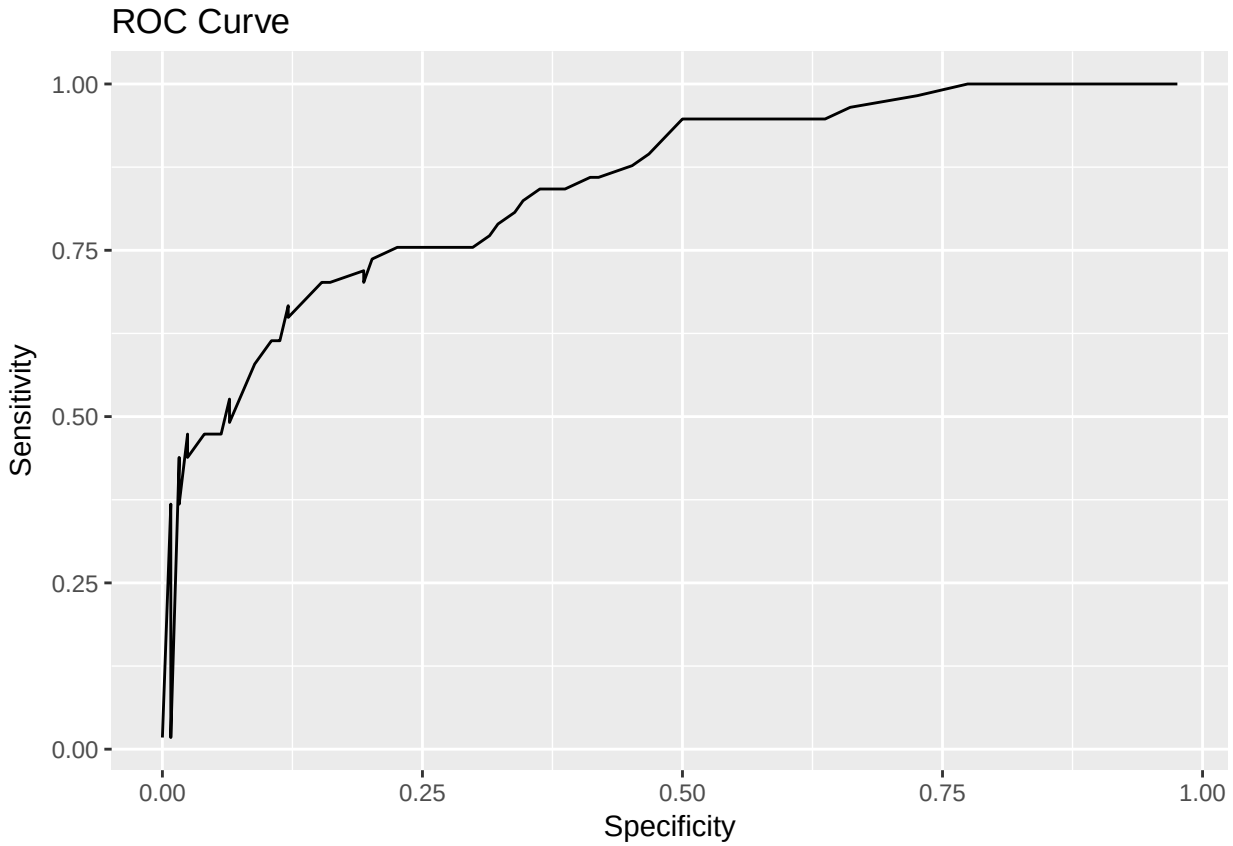
Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area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

```
#ROC Curve
generate_roc <- function(classification)
{

  thresholds <- seq(0,1,0.01)
  Y <- c()
  X <- c()
  for (threshod in thresholds) {
    classification$scored.class <- ifelse(classification$scored.probability > threshod,1,0)
    X <- append(X,1-calculate_specificity(classification))
    Y <- append(Y,calculate_sensitivity(classification))
    }
  roc <- data.frame(X=X,Y=Y)
  roc <- na.omit(roc)
  g <- ggplot(roc,aes(X,Y)) + geom_line() + ggtitle('ROC Curve') +
    xlab('Specificity') + ylab('Sensitivity')
  height = (roc$Y[-1]+roc$Y[-length(roc$Y)])/2
  width = -diff(roc$X)
  area = sum(height*width)
  return(list(Plot =g,AUC = area))
}

roc_curve <- generate_roc(data_df)
```

```
roc_curve$AUC
```

```
## [1] 0.8247029
```

```
roc_curve$Plot
```



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?

```
confusionMatrix(data_df$scored.class, data_df$class, positive = "1")
```

```
## 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
##
```

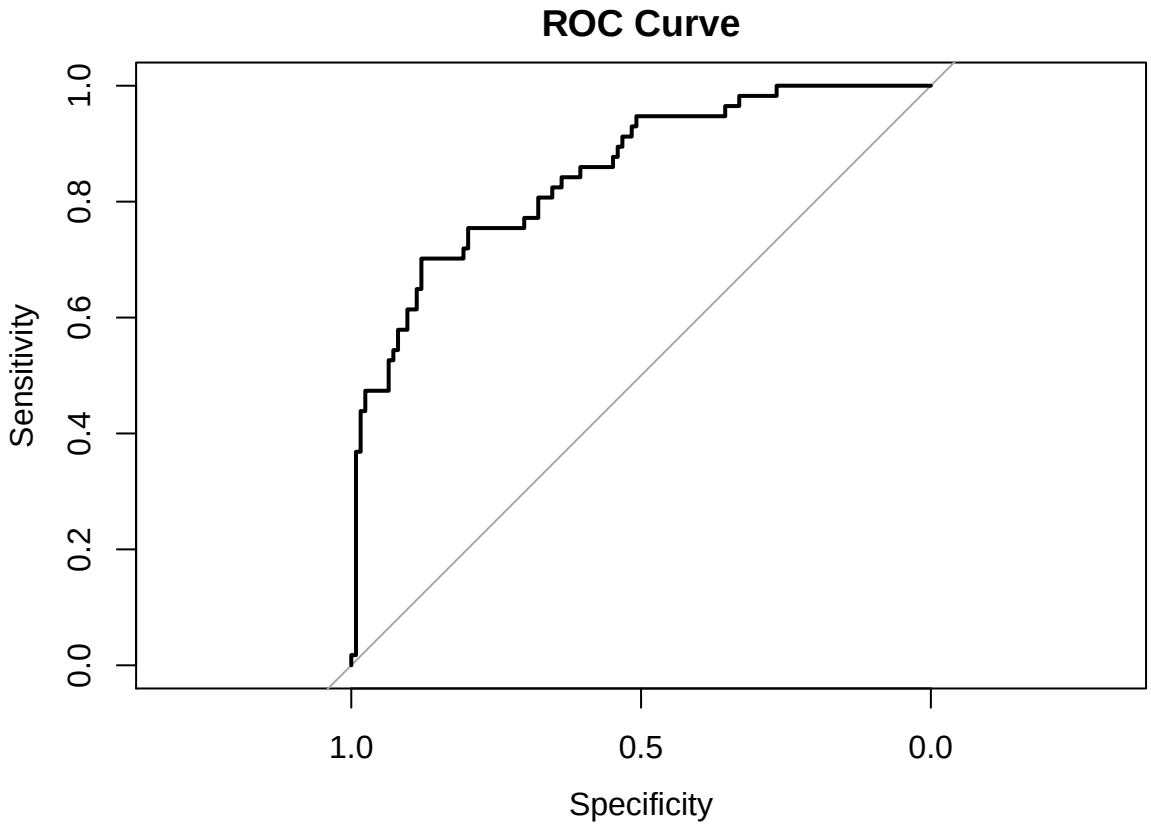
```
table(Actual=data_df$class, Precidted=data_df$scored.class)
```

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

Matrix is the same

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

```
plot(roc(as.factor(data_df$class) ~ data_df$scored.probability),main='ROC Curve')
```




```
auc(roc(as.factor(data_df$class) ~ data_df$scored.probability))
```

```
## Area under the curve: 0.8503
```

```
roc_curve$AUC
```

```
## [1] 0.8247029
```

ROC Curve is a lot smoother, but it is essentially the same. AUC is essential the same the pROC package, probably more accurate.