Homework2 Classification Metrics

Jonathan Hernandez

June 16, 2018

- Assignment for working on various classification metrics.

1. Download the dataset

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

2. Get raw confusion matrix and intrepret it

```
table(data$class, data$scored.class)
```

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

- The table (confusion matrix) is used to show the performance of a classification model on data. In this case it is the data based on the actual class vs predicted class.
- The rows show the number of actual class for the observation and the columns show the predicted class for the observation. Each entry in the table shows the different outcomes
- True Positives
- True Negatives
- False Positives
- False Negatives

Actual == 0 and Predicted == 0 is a True Negative

Actual == 1 and Predicted == 1 is a True Positive

Predicted == 0 and Actual == 1 is a False Negative

Predicted == 1 and Actual == 0 is a False Positive

3. Write a function to compute the Accuracy of the predictions.

```
accuracy <- function(df){
  tp <- nrow(df[df$class == 1 & df$scored.class == 1, ])
  tn <- nrow(df[df$class == 0 & df$scored.class == 0, ])

  acc <- (tp + tn) / nrow(df)
  return(acc)
}
accuracy(data)
```

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

4. Write a function to compute the classification error rate of the predictions.

```
classification_error_rate <- function(df){
  fp <- nrow(df[df$class == 0 & df$scored.class == 1, ])
  fn <- nrow(df[df$class == 1 & df$scored.class == 0, ])

  cer <- (fp + fn) / nrow(df)
  return(cer)
}
classification_error_rate(data)
```

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

- Show the accuracy and error rate add up to 1

```
accuracy(data) + classification_error_rate(data)
```

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

5. Write a function that computes the precision of the predictions.

```
precision <- function(df){
  tp <- nrow(df[df$class == 1 & df$scored.class == 1, ])
  fp <- nrow(df[df$class == 0 & df$scored.class == 1, ])

  prec <- tp / (tp + fp)
  return(prec)
}
precision(data)
```

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

6. Write a function that computes the sensitivity of the predictions.

```
sensitivity <- function(df){
  tp <- nrow(df[df$class == 1 & df$scored.class == 1, ])
  fn <- nrow(df[df$class == 1 & df$scored.class == 0, ])

  sen <- tp / (tp + fn)
  return(sen)
}
sensitivity(data)
```

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

7. Write a function that computes the specificity of the preductions.

```
specificity <- function(df){
  tn <- nrow(df[df$class == 0 & df$scored.class == 0, ])
  fp <- nrow(df[df$class == 0 & df$scored.class == 1, ])

  spec <- tn / (tn + fp)
  return(spec)
}
specificity(data)
```

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

8. Write a function to compute the F1 score of the predictions

```r
f1 <- function(df){
  f1score <- (2 * precision(df) * sensitivity(df)) /
    (precision(df) + specificity(df))
  return(f1score)
}
f1(data)
```

```
## [1] 0.443235
```

9. What are the bounds of 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 $ab < a$)

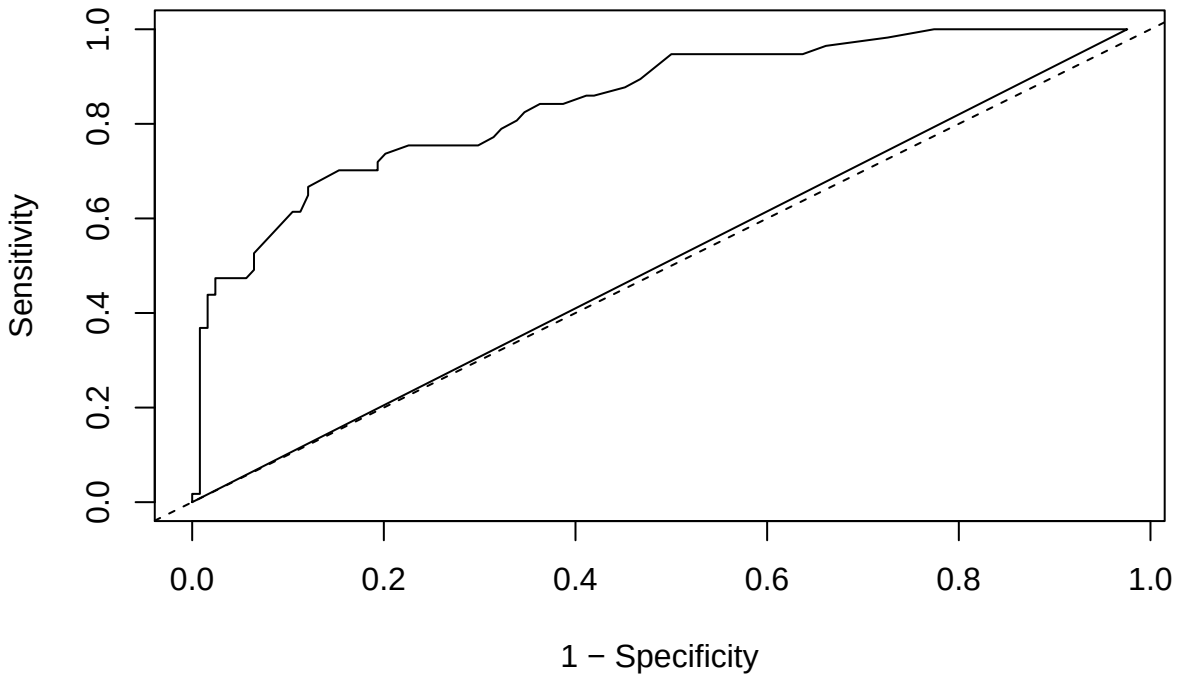
- Since we have limited the range of values a and b can take (assume a is the precision and b is the sensitivity), if we let a and b get close to 0 as possible but not touching it (taking the limit of a -> 0 and b -> 0), the F1 score will approach a very small value close to 0 but not quite. If we take the limit of a -> 1 and b -> 1, then the F1 score can be evaluated as $(2 * 1 * 1) / (1 + 1) = 1$ so we will always have F1 be bounded as $0 < F1 < 1$.

10. Write a function that generates an ROC curve from a data set with a true classification column (class in ourexample) 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.

```r
# first create a threshold sequence
ROC_curve <- function(df){
  threshold <- seq(0, by = 0.01, 1) # different cutoffs/thresholds
  # sensitivity and specificity vectors initialize to 0
  Sensitivity <- numeric(length(threshold))
  Specificity <- numeric(length(threshold))
  for (i in 1:length(threshold)){
    # are the scored probabilities smaller than the threshold
    thresh <- ifelse(df$scored.probability < threshold[i], "no", "yes")
    freq <- xtabs(~ class + thresh, data = df) # generate frequency/contingency table
    if (ncol(freq) < 2){ # if only yes or no; freq[,2] = 0 and can go out of bounds
      Specificity[i] <- 1
      Sensitivity[i] <- 0
    }
    else {
      # formulas for Sensitivity and Specificity based on each threshold
      Specificity[i] <- freq[1,1] / (freq[1,1] + freq[1,2])
      Sensitivity[i] <- freq[2,2] / (freq[2,1] + freq[2,2])
    }
  }
  # plot the ROC curve
  ROC_plot <- plot(1-Specificity, Sensitivity, type = "l")
  abline(0,1, lty=2)

# approximate the area under the curve (AUC)
```

```
  height <- Sensitivity
  width <- -diff(1-Specificity)
  area <- sum(height*width)
  return(list(ROC_plot, area))
}
# AUC
ROC_curve(data)[[2]]
```




```
## [1] 0.8297963
```

- AUC value is underneath the plot

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

- Accuracy

```
accuracy(data)
```

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

- Classification Error Rate

```
classification_error_rate(data)
```

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

- Precision

```
precision(data)
```

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

- Sensitivity

```
sensitivity(data)
```

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

- Specificity

```
specificity(data)
```

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

- F1 Score

```
f1(data)
```

```
## [1] 0.443235
```

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

- Install (if required) the caret package

```
if (!require(caret)) install.packages("caret", dependencies = TRUE)
```

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'caret'
## The following objects are masked _by_ '.GlobalEnv':
##
##     precision, sensitivity, specificity
```

```
library(caret)
conf_matrix <- table(data$scored.class, data$class)[2:1,2:1]
# confusion matrix reverse the order so the results will match to what I created
confusionMatrix(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##
##       1   0
##   1  27   5
##   0  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 : 1
##
```

- Sensitivity using the caret package:

```
caret::sensitivity(conf_matrix)
```

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

- Specificity using the caret package:

```
caret::specificity(conf_matrix)
```

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

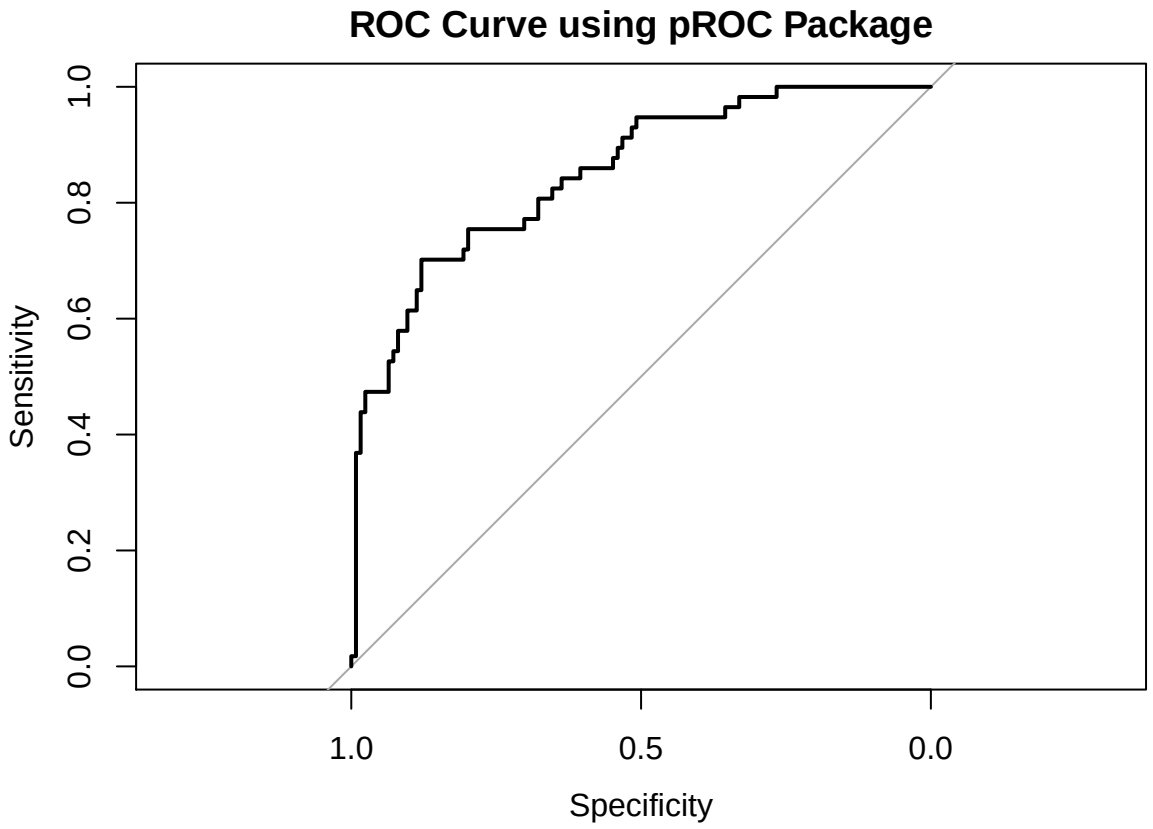
Looking at my functions and the ones provided in the caret package, the metrics are basically the same.

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?

```
if (!require(pROC)) install.packages("ROC", dependencies = TRUE)
```

```
## Loading required package: pROC
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(pROC)
plot.roc(data$class, data$scored.probability, main = "ROC Curve using pROC Package")
```




```
auc(roc(data$class, data$scored.probability))
```

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

- Results are nearly the same. Plot looks similar and AUC for using the pROC is slightly larger than my calculated AUC manually.