# DATA 621 - Homework 2

*Joshua Sturm*

*03/09/2018*

```
library(tidyverse)
library(knitr)
library(caret)
library(pROC)
```

## Task 1

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

### Load Data

```
cod.raw <- read.csv("classification-output-data.csv", stringsAsFactors = F)
```

## Task 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?

### Confusion Matrix

```
cod <- cod.raw %>%
  select(9:11)
table(Actual = cod$class, Predicted = cod$scored.class)
##       Predicted
## Actual   0   1
##      0 119   5
##      1  30  27
```

The rows are the actual values, while the columns are the predicted ones.

## Task 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.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

### Prediction Accuracy

```
pred_acc <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
  }

  tab <- table(dataset$class, dataset$scored.class)
  TP <- tab[4]
  TN <- tab[1]
  FP <- tab[3]
  FN <- tab[2]

  acc <- (TP + TN) / (TP + TN + FP + FN)

  return(acc)
}
```

Testing it on our dataset:

Prediction accuracy = 80.6629834254144%.

## Task 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.

$$Classification\ Error\ Rate = \frac{FP + FN}{TP + TN + FP + FN} \tag{2}$$

### Prediction Error Rate

```
pred_err <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
```

```
  }

  tab <- table(dataset$class, dataset$scored.class)
  TP <- tab[4]
  TN <- tab[1]
  FP <- tab[3]
  FN <- tab[2]

  err <- (FP + FN) / (TP + TN + FP + FN)

  return(err)
}
```

Testing it on our dataset:

Classification Error Rate = 19.3370165745856%.

Ensure that the prediction accuracy and the error rate sum up to 1:

Sum of prediction accuracy and classification error rate = 1.

# Task 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.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

## Precision

```
pred_prec <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
  }

  tab <- table(dataset$class, dataset$scored.class)
  TP <- tab[4]
  FP <- tab[3]

  prec <- (TP) / (TP + FP)

  return(prec)
}
```

Testing it on our dataset:

Prediction precision = 84.375%.

# Task 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.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

## Sensitivity

```
pred_sens <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
  }

  tab <- table(dataset$class, dataset$scored.class)
  TP <- tab[4]
  FN <- tab[2]

  sens <- (TP) / (TP + FN)

  return(sens)
}
```

Testing it on our dataset:

Prediction sensitivity = 47.3684210526316%.

# Task 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.

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

## Specificity

```
pred_spec <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
```

```
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
  }

  tab <- table(dataset$class, dataset$scored.class)
  TN <- tab[1]
  FP <- tab[3]

  spec <- (TN) / (TN + FP)

  return(spec)
}
```

Testing it on our dataset:

Prediction specificity = 95.9677419354839%.

# Task 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.

$$F1\ Score = \frac{2 \cdot Precision \cdot Sensitivity}{Precision + Sensitivity} \tag{6}$$

## F1 Score

```
pred_f1 <- function(dataset){
  # Check if entered dataset is a dataframe; if not, cooerce it to one
  if (!(is.data.frame(dataset))){
    dataset <- as.data.frame(dataset)
  }

  # Ensure dataframe is not empty
  if (is_empty(dataset)){
    stop("You've entered an empty dataset!")
  }

  tab <- table(dataset$class, dataset$scored.class)
  TP <- tab[4]
  TN <- tab[1]
  FP <- tab[3]
  FN <- tab[2]

  prec <- (TP) / (TP + FP)
  sens <- (TP) / (TP + FN)
```

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

  return(F1)
}
```

Testing it on our dataset:

F1 Score = 0.606741573033708.

# Task 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.)

## F1 Bounds

$F1 = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} = 2 \cdot \frac{(\frac{TP}{TP+FP}) \cdot (\frac{TP}{TP+FN})}{(\frac{TP}{TP+FP}) + (\frac{TP}{TP+FN})}$

For Precision, $\frac{TP}{TP+FP}$, the denominator will always be larger than the numerator, so the fraction must evaluate to some number $\{m \in \mathbb{Q} \mid 0 < m < 1\}$.

Similarly, for sensitivity, $\frac{TP}{TP+FN}$, the fraction will always to evaluate to some number $\{n \in \mathbb{Q} \mid 0 < n < 1\}$.

Multiplying two numbers $a$ and $b$, where $\{a, b \in \mathbb{Q} \mid 0 < a, b < 1\}$, will produce a number smaller than either $a$ or $b$. So, when calculating the F1 score, we will always be dividing a smaller number by a larger one, which will gives us a result between 0 and 1.

# Task 10

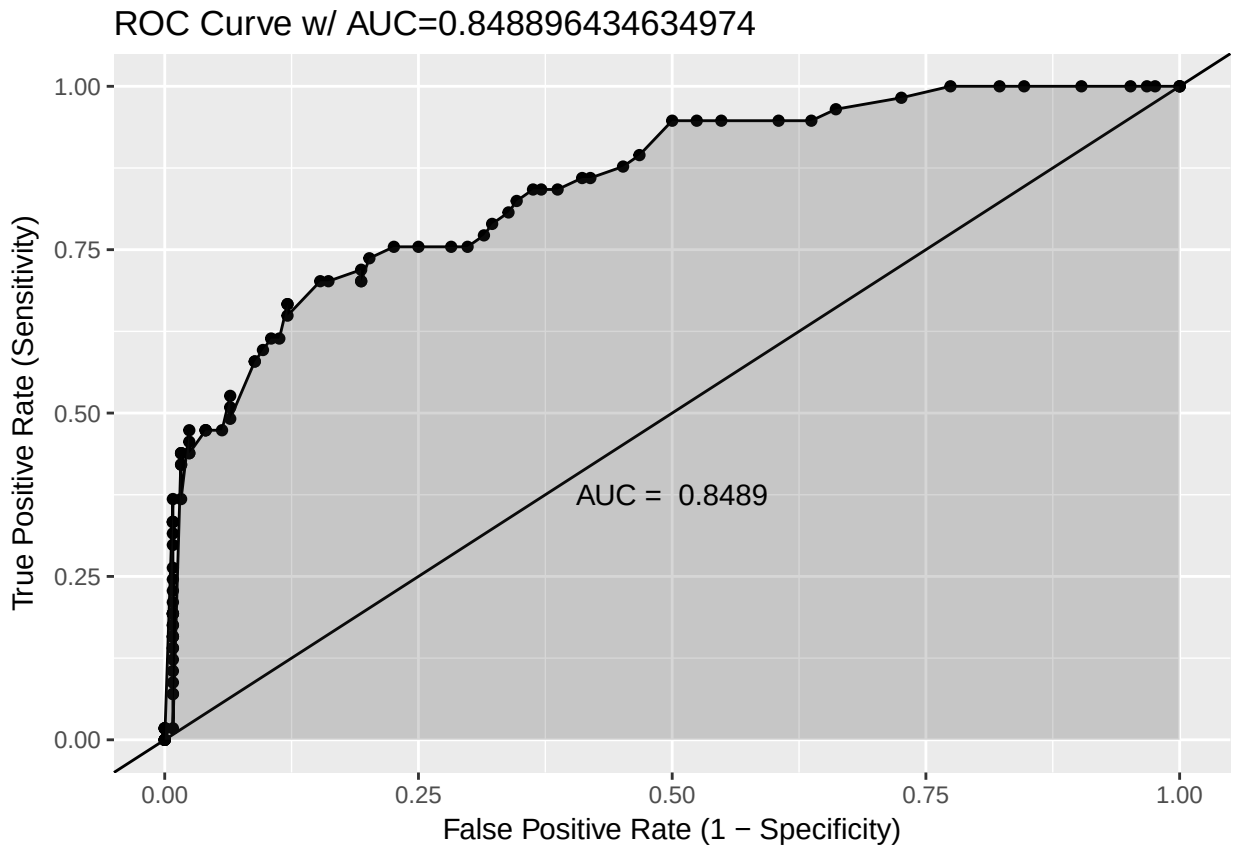
Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area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

## ROC Curve

```
roc_curve <- function(dataset){

  thrshld <- seq(from = 0, to = 1, by = 0.01)

  # Sum totals of positives and negatives in the true column
  P <- sum(dataset$class == 1)
  N <- sum(dataset$class == 0)

  # Ensure there's at least one of each classifier
  stopifnot(P > 0, N > 0)

  dataset <- arrange(dataset, desc(scored.probability))

  # Initialize TPR, FPR, and temporary threshold vectors
```

```
  tpr <- c()
  fpr <- c()
  tt <- integer(nrow(dataset))

  for (i in 1:length(thrshld)){

    tt <- ifelse(dataset$scored.probability > thrshld[i], 1, 0)

    # Calculate Sensitivity and Specificity
    TP <- sum(dataset$class == 1 & tt == 1)
    TN <- sum(dataset$class == 0 & tt == 0)
    FP <- sum(dataset$class == 0 & tt == 1)
    FN <- sum(dataset$class == 1 & tt == 0)

    sens <- (TP) / (P)
    spec <- (TN) / (N)

    tpr[i] <- sens
    fpr[i] <- 1 - spec
  }
  # Store results in a dataframe
  df <- data.frame(fpr, tpr)

  # Calculate the area under the ROC curve
  dfpr <- c(diff(df$fpr), 0)
  dtpr <- c(diff(df$tpr), 0)
  roc_auc <- abs(sum(tpr * dfpr) + sum(dtpr * dfpr)/2)

  # Store variable for roc curve plot
  roc_curve_plot <- ggplot(df, aes(x = fpr, y = tpr, ymin = 0, ymax = tpr, xmin = 0, xmax = 1)) +
    geom_abline(intercept = 0, slope = 1) +
    geom_point() +
    geom_line() +
    geom_ribbon(alpha = 0.2) +
    labs(title=paste0("ROC Curve w/ AUC=", roc_auc), x = "False Positive Rate (1 - Specificity)", y = "
    annotate("text", x = 0.5, y = 0.375, label = paste("AUC = ", round(roc_auc, 4)))

  roc_curve <- list(roc_curve_plot, roc_auc)
  return(roc_curve)
}
```

```
roc_curve(cod)
## [[1]]
```




```
##
## [[2]]
## [1] 0.8488964
```

# Task 11

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

### Function results

```
cat(sprintf("\n %s = %f \n", c("Accuracy", "Error Rate", "Precision", "Sensitivity", "Specificity", "F1
##
##  Accuracy = 0.806630
##
##  Error Rate = 0.193370
##
##  Precision = 0.843750
##
##  Sensitivity = 0.473684
##
##  Specificity = 0.959677
```

```
##
##  F1 Score = 0.606742
```

# Task 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?

## Caret package

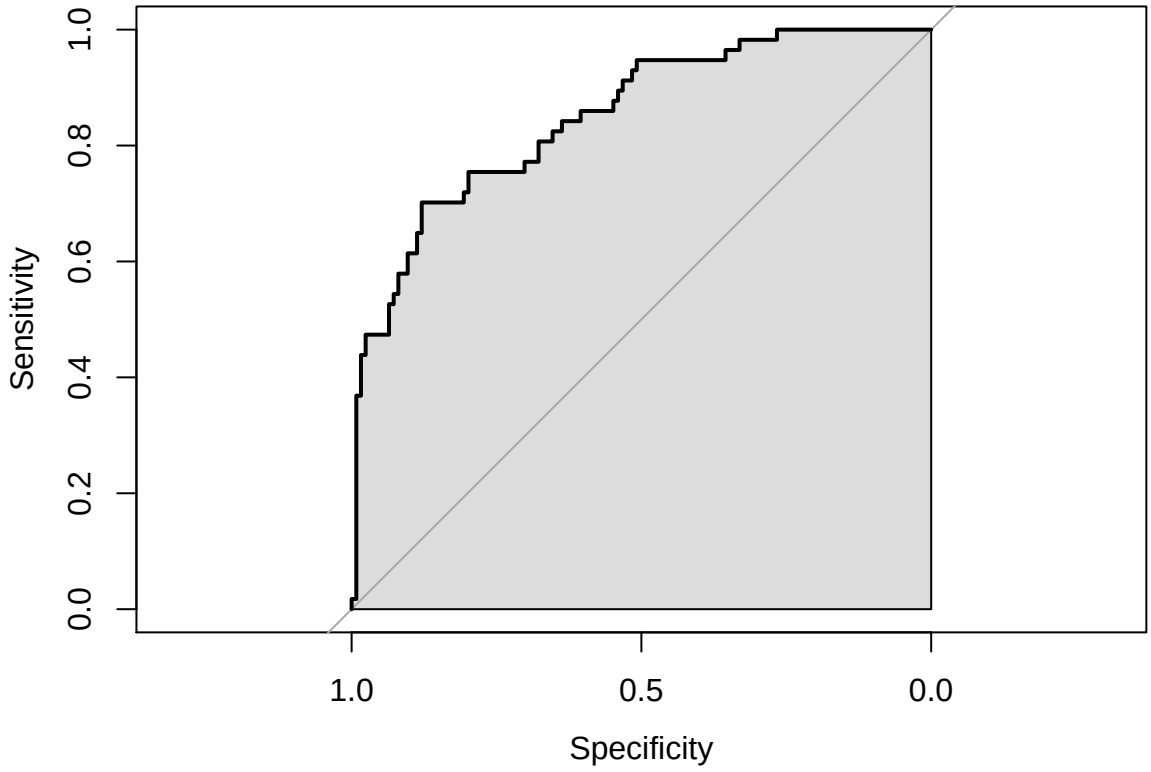
```
confmat <- confusionMatrix(cod$scored.class, cod$class, positive = "1")
confmat
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     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
##

# Compare accuracy
all.equal(confmat[["overall"]][["Accuracy"]], pred_acc(cod))
## [1] TRUE

# Compare sensitivity
all.equal(confmat[["byClass"]][["Sensitivity"]], pred_sens(cod))
## [1] TRUE

# Compare specificity
all.equal(confmat[["byClass"]][["Specificity"]], pred_spec(cod))
## [1] TRUE
```

## Task 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?

### pROC package

```
roc(cod$class ~ cod$scored.probability, plot=T, auc.polygon=TRUE)
```




```
##
## Call:
## roc.formula(formula = cod$class ~ cod$scored.probability, plot = T,     auc.polygon = TRUE)
##
## Data: cod$scored.probability in 124 controls (cod$class 0) < 57 cases (cod$class 1).
## Area under the curve: 0.8503
```

The pROC package calculated an AUC of 0.8503, compared to my value of 0.8489, which is within 1%.

## References


- https://www.r-bloggers.com/illustrated-guide-to-roc-and-auc/
- https://community.alteryx.com/t5/Data-Science-Blog/ROC-Curves-in-Python-and-R/ba-p/138430
- http://web.ydu.edu.tw/~alan9956/docu/refer/roc_introduction.pdf
- http://www.saedsayad.com/docs/ROC101.pdf

- https://www.linkedin.com/pulse/roc-curve-simple-terms-yanchao-liu/
- https://www.r-bloggers.com/on-calculating-auc/
- https://stat.ethz.ch/pipermail/r-help/2005-September/079872.html