DATA_621_HW1

Chi Pong, Euclid Zhang, Jie Zou, Joseph Connolly, LeTicia Cancel

2/19/2022

```
library("dplyr")
library("ggplot2")
```

1.Loading Data

```
classification_df <- read.csv("https://raw.githubusercontent.com/ezaccountz/DATA_621/main/HW2/classifica
```

2.confusion matrix

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?

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

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

The tows represent the predicted class
The columns represent the actual class

3.accuracy

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

```
cal_accurary <- function (df, actual_var_name, pred_var_name) {
  accurary <- sum(df[actual_var_name] == df[pred_var_name])/nrow(df)
  return (accurary)
}
```

```
cal_accurary(classification_df,"class","scored.class")
```

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

4.error rate

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.

```
cal_error <- function (df, actual_var_name, pred_var_name) {
  error <- sum(df[actual_var_name] != df[pred_var_name])/nrow(df)
  return (error)
}
```

```
cal_error(classification_df,"class","scored.class")
```

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

5.precision

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

```
cal_precision <- function (df, actual_var_name, pred_var_name) {
  n_tp <- sum(df[actual_var_name] == 1 & df[pred_var_name] == 1)
  n_fp <- sum(df[actual_var_name] == 0 & df[pred_var_name] == 1)

  precision <- n_tp / (n_tp + n_fp)
  return (precision)
}
```

```
cal_precision(classification_df,"class","scored.class")
```

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

6.sensitivity

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.

```
cal_sensitivity <- function (df, actual_var_name, pred_var_name) {
  n_tp <- sum(df[actual_var_name] == 1 & df[pred_var_name] == 1)
  n_fn <- sum(df[actual_var_name] == 1 & df[pred_var_name] == 0)

  sensitivity <- n_tp / (n_tp + n_fn)
  return (sensitivity)
}
```

```r
cal_sensitivity(classification_df,"class","scored.class")
```

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

7.specificity

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

```r
cal_specificity <- function (df, actual_var_name, pred_var_name) {
  n_tn <- sum(df[actual_var_name] == 0 & df[pred_var_name] == 0)
  n_fp <- sum(df[actual_var_name] == 0 & df[pred_var_name] == 1)

  specificity <- n_tn / (n_tn + n_fp)
  return (specificity)
}
```

```r
cal_specificity(classification_df,"class","scored.class")
```

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

8.F1 score

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.

```r
cal_f1 <- function (df, actual_var_name, pred_var_name) {
  precision <- cal_precision(classification_df,"class","scored.class")
  sensitivity <- cal_sensitivity(classification_df,"class","scored.class")
  f1 <- 2*precision*sensitivity/(precision+sensitivity)
  return (f1)
}
```

```r
cal_f1(classification_df,"class","scored.class")
```

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

9.bounds on the F1 score

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.)

$$\begin{aligned}F1 &= \frac{2 * Precision * Sensitivity}{Precision + Sensitivity} \\ &= 2 * \frac{\frac{TP}{TP+FP} * \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \\ &= 2 * \frac{\frac{TP^2}{(TP+FP)(TP+FN)}}{\frac{TP^2+TP*FN+TP^2+TP*FP}{(TP+FP)(TP+FN)}} \\ &= 2 * \frac{TP^2}{TP^2 + TP * FN + TP^2 + TP * FP} \\ &= 2 * \frac{TP}{2TP + FN + FP} \\ &= \frac{2TP}{2TP + FN + FP}\end{aligned}$$

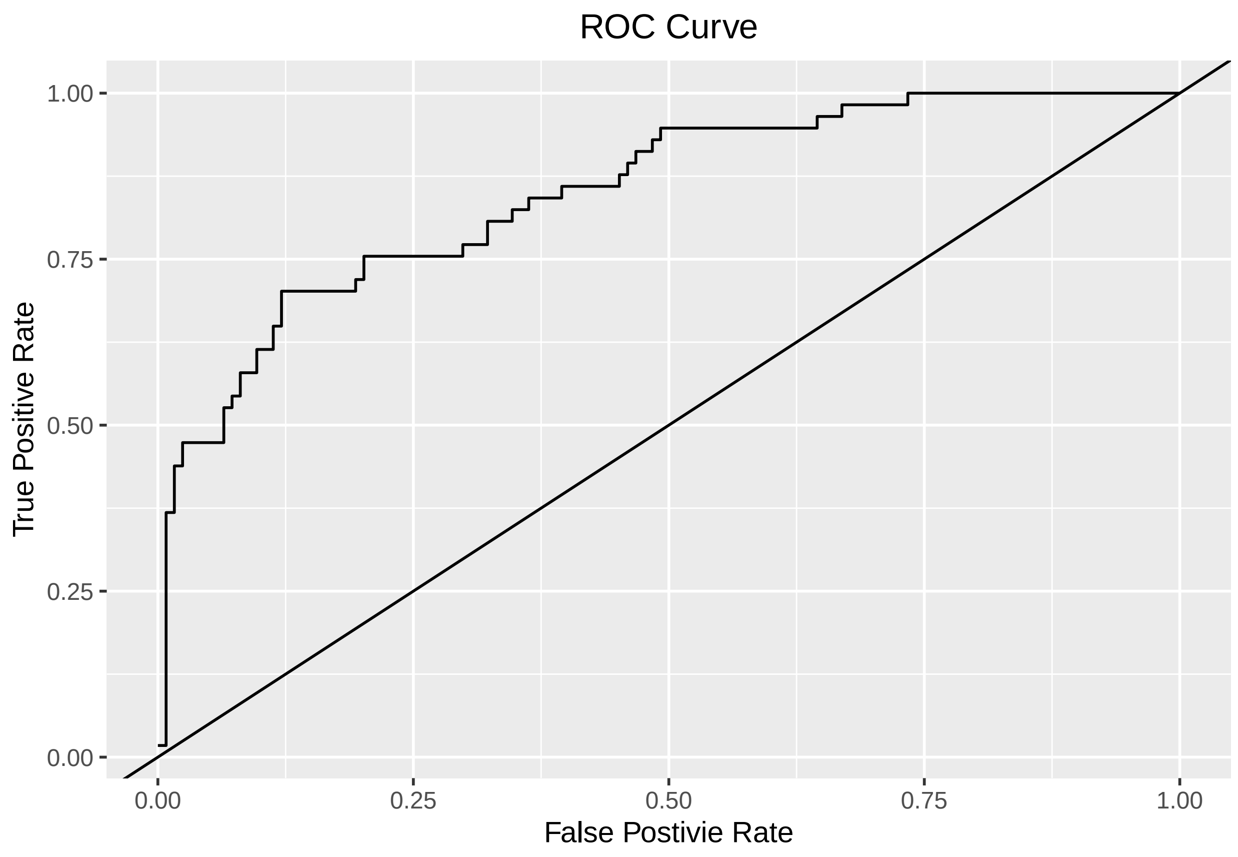
Since $0 <= TP <= 1$, $0 <= FN <= 1$ and $0 <= FP <= 1$, $0 <= 2TP <= 2TP + FN + FP$. So $F1 = \frac{2TP}{2TP+FN+FP}$ is always between 0 and 1

10.ROC curve

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.

```r
ROC <- function(actual_class, prob){

  pred_df <- data.frame(actual_class = actual_class, prob = prob)
  pred_df <- pred_df[order(pred_df$prob, decreasing = TRUE),]

  tpr <- cumsum(pred_df$actual_class) / sum(pred_df$actual_class)
  fpr <- cumsum(!pred_df$actual_class) / sum(!pred_df$actual_class)

  tpr_fpr <- data.frame(tpr = tpr, fpr = fpr)
  rect_table <- tpr_fpr %>%
  group_by(tpr) %>%
  summarise(
    fpr_min = min(fpr),
    fpr_max = max(fpr)
  ) %>%
  arrange(tpr)

  AUC <- sum(rect_table$tpr * (rect_table$fpr_max-rect_table$fpr_min))

  roc <- ggplot(tpr_fpr, aes(x=fpr, y=tpr)) +
    geom_line() +
    geom_abline() +
    labs(title = "ROC Curve", x = "False Postivie Rate",y = "True Positive Rate") +
    theme(plot.title = element_text(hjust = 0.5))

  return(list(roc = roc, AUC = AUC))
}
```

```
ROC(classification_df$class,classification_df$scored.probability)$roc
```



```
print(paste0("The AUC is: ", ROC(classification_df$class,classification_df$scored.probability)$AUC))
```

```
## [1] "The AUC is: 0.850311262026033"
```

11.utilization of the created functions

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

```
data.frame(metric = c("accurary","error rate",
                      "precision","sensitivity","specificity","F1"),
           score = c(cal_accurary(classification_df,"class","scored.class"),
                     cal_error(classification_df,"class","scored.class"),
                     cal_precision(classification_df,"class","scored.class"),
                     cal_sensitivity(classification_df,"class","scored.class"),
                     cal_specificity(classification_df,"class","scored.class"),
                     cal_f1(classification_df,"class","scored.class")
            ))
```

```
##        metric     score
## 1    accurary 0.8066298
## 2  error rate 0.1933702
## 3   precision 0.8437500
## 4 sensitivity 0.4736842
## 5 specificity 0.9596774
## 6          F1 0.6067416
```

12.caret package

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)
```

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

```
classification_df2 <- data.frame(actual_class = classification_df$class,
                                 predicted_class = classification_df$scored.class)
classification_df2$actual_class = as.factor(classification_df2$actual_class)
classification_df2$predicted_class = as.factor(classification_df2$predicted_class)

confusionMatrix(classification_df2$predicted_class,classification_df2$actual_class,
                positive = "1")
```

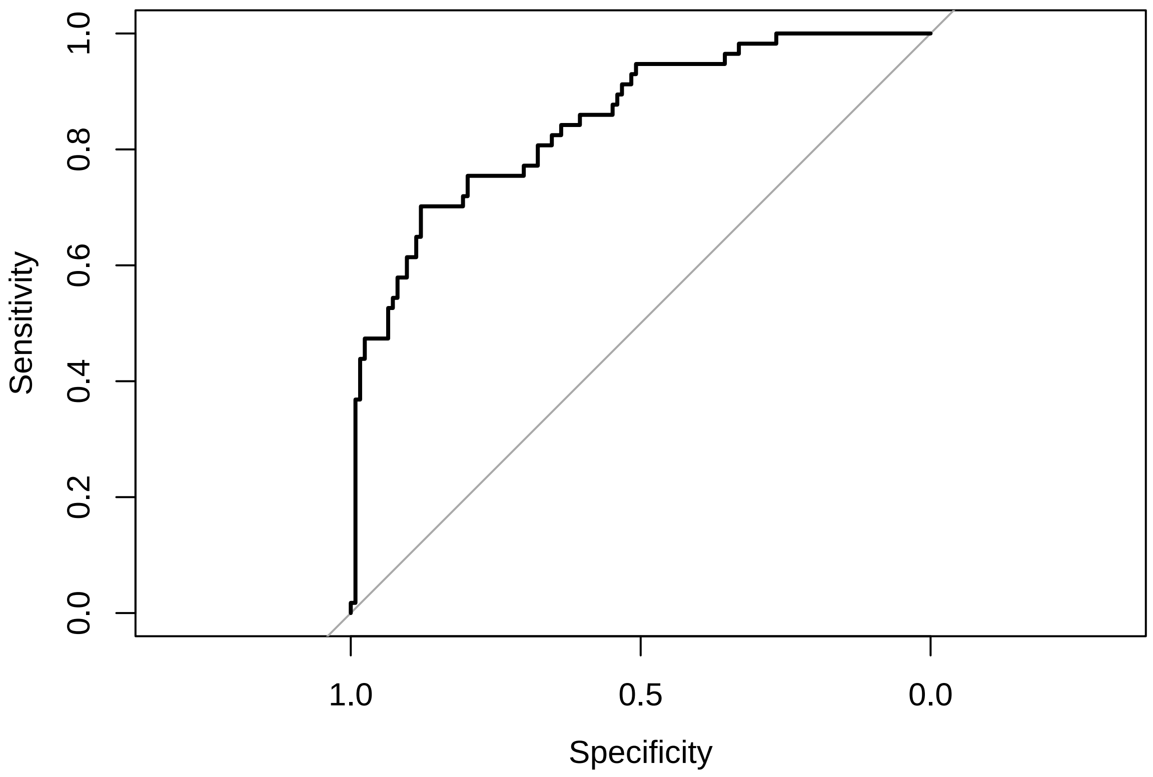
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
##
##                   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
##
```

The results are the same as the results produced by our own functions.

13.pROC package

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

```r
library(pROC)
rocCurve <- roc(classification_df$class, classification_df$scored.probability)
plot(rocCurve)
```



The ROC curve is exactly the same as we plotted using our own function

```r
print(paste0("The AUC is: ", auc(rocCurve)))
```

```
## [1] "The AUC is: 0.850311262026033"
```

The AUC is exactly the same as we calculated using our own function