Homework #2: Classification Metrics

Data 621 Business Analytics and Data Mining

Aadi Kalloo, Nathan Lim, Asher Meyers, Daniel Smilowitz, Logan Thomson

Due June 26, 2016

1. Download the classification output data set

The supplied data for this assignment contains a sample (181 cases) of the `pima` dataset. There are eight predicotrs and a classification column describing whether or not the case showed signs of of diabetes (coded "1"), or not (coded "0"). Two aditional columns, `scored.class` and `scored.probability` have been added from the results of a classification model, which attempts to predict the class membership of the cases in the sample.

Using this data, we will calculate statistics that help to measure the effectiveness of the classification model using a confusion matrix.

2. Use the table() function to get the raw confusion matrix for this scored dataset.

	0	1
0	119	5
1	30	27

In this confusion matrix, the rows represent the actual class for the observation and the columns represent the predicted class for the observation. The values in the table represent the number of observations that fall into each combination of predicted and true values.

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.

The accuracy of the predicted classifications is: 0.807

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.

The classification error rate is: 0.193

Verify by summing the rates:
The sum of the accuracy and error rate is: 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.

The precision of the predictions is: 0.844

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.

The sensitivity of the predictions is: 0.96

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.

The specificity of the predictions is: 0.474

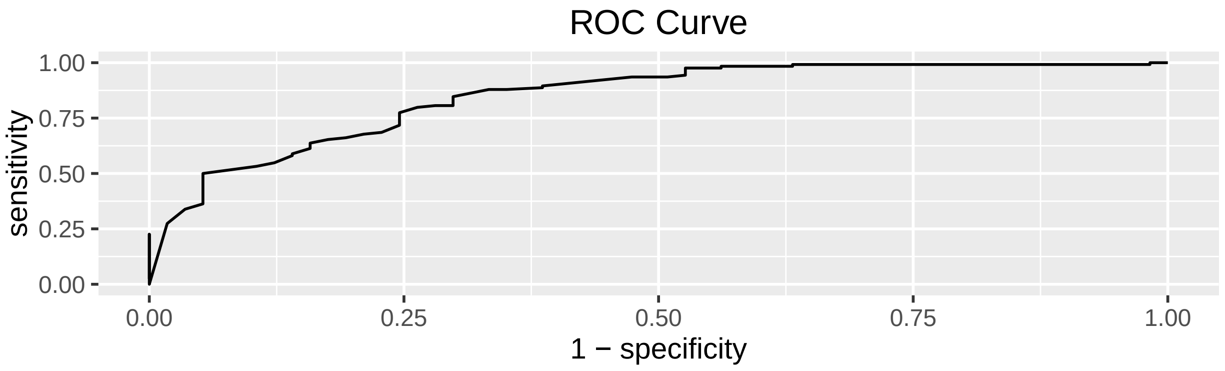
8. Write a function that takes the data set as a dataframe, with actual and predicted classifications identified, and returns the F1 score of the predictions.

The F1 score is: 0.898

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.

The variables used by the F1 score are Precision and Sensitivity. Precision is defined as True Positives divided by All Actual Positives. Precision will have a lower bound of 0 if True Positives is 0, and an upper bound at 1 if True Positives is equal to All Actual Positives. Sensitivity is defined as True Positives divided by All Predicted Positives. Sensitivity will have a lower bound of 0 if True positives is 0, and an upper bound at 1 if True Positives is equal to All Predicted Positives. If either Precision or Sensitivity is 0, this will give the F1 score a numerator of 0. If both Precision *and* Sensitivity is 0, the F1 Score will be invalid as it will have a denominator of 0. If both Precision *and* Sensitivity is 1, the numerator and denominator of the F1 score is equal to 2, giving an F1 Score of 1. In this way, the F1 score is bounded by the upper and lower limits of Precision and Sensitivity and the F1 score will always fall between 0 and 1.

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.



The area under the curve is: 0.749

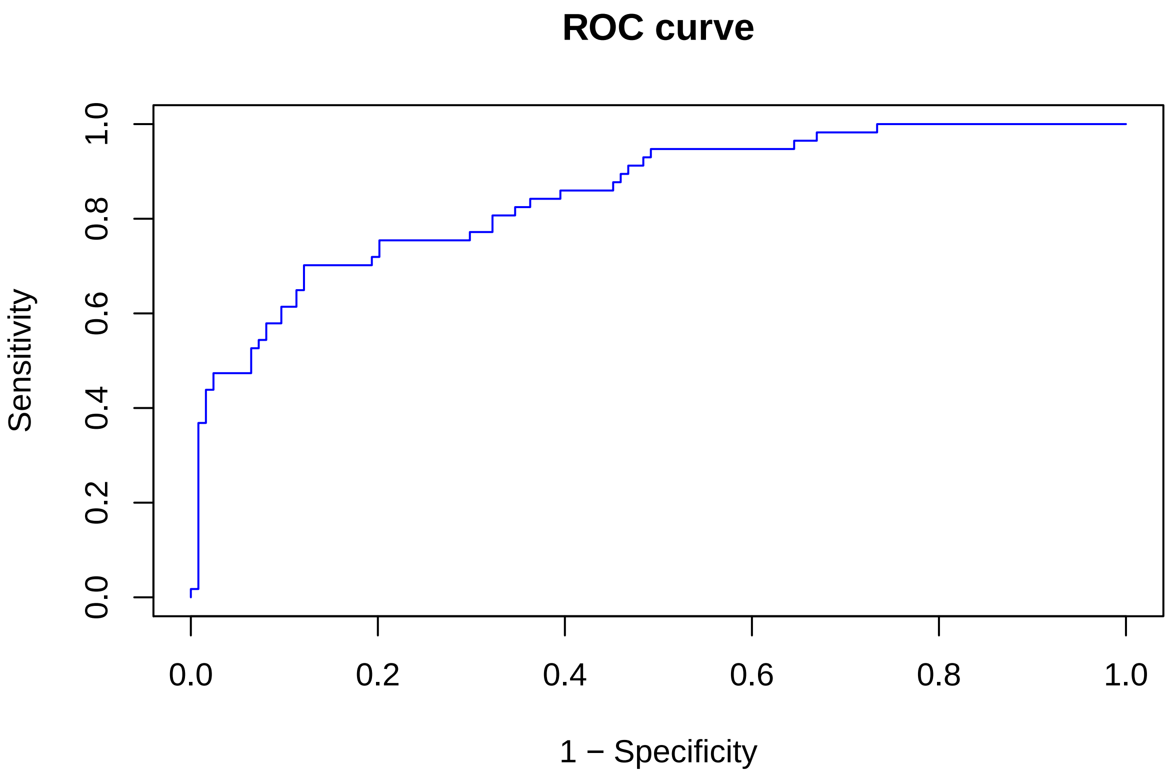
12. Investigate the caret package. In particular, consider the functions confusionMatrix, sensitivity, and specificity. Apply the functions to the data set. How do the results compare with your own functions?

Caret Confusion Matrix:

	0	1
0	119	30
1	5	27

The caret package sensitivity is: 0.96. Our sensitivity result is: 0.96.
The caret package specificity is: 0.474. Our specificity result is: 0.474.
The caret package seems to match well with our results.

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?



The pROC package seems to match well with our results – the area under the curve for the pROC::roc function is 0.739, while our value for the area under the curve is 0.749. We believe that this difference of 1% falls within natural variance given the difference in the two methods.

Appendix A – R Code

```
# 1. Load data
library(dplyr)
library(pander)
library(ggplot2)
library(caret)
library(e1071)
library(pROC)
options(digits = 3)
df <- read.csv(url("https://raw.githubusercontent.com/dsmilo/DATA621/master/HW2/Data/classification-outp

# 2. Build confusion matrix
confusion_matrix = table(select(df, class, scored.class))
predicted = "scored.class"
pander(confusion_matrix)

# 3. Accuracy function
accuracy_function <- function(df, scored.class) {
    confusion <- table(select(df, class, scored.class))
    accuracy <- (confusion[1, 1] + confusion[2, 2])/sum(confusion)
    return(accuracy)
}

# 4. Classification error rate function
CER_function <- function(df, predicted) {
    confusion <- table(select(df, class, get(predicted)))
    classification_error_rate <- (confusion[1, 2] + confusion[2,
        1])/sum(confusion)
    return(classification_error_rate)
}

# 5. Precision function
precision_function <- function(df, predicted) {
    confusion <- table(select(df, class, get(predicted)))
    precision <- confusion[2, 2]/(confusion[1, 2] + confusion[2,
        2])
    return(precision)
}

# 6. Sensitivity function
sensitivity_function <- function(df, predicted) {
    confusion <- table(select(df, class, get(predicted)))
    if (dim(confusion)[2] == 1) {
        if (colnames(confusion) == 0) {
            sensitivity <- 0
        } else {
            sensitivity <- 1
        }
    } else {
        sensitivity <- confusion[1, 1]/(confusion[1, 1] + confusion[1,
            2])
    }
    return(sensitivity)
}

# 7. Specificity function
specificity_function <- function(df, predicted) {
    confusion <- table(select(df, class, get(predicted)))
    if (dim(confusion)[2] == 1) {
        if (colnames(confusion) == 0) {
            specificity <- 1
        } else {
            specificity <- 0
        }
    } else {
        specificity <- confusion[2, 2]/(confusion[2, 1] + confusion[2,
            2])
    }
    return(specificity)
}

# 8. F1score function
f1score_function <- function(df, predicted) {
    precision <- precision_function(df, predicted)
    sensitivity <- sensitivity_function(df, predicted)
    f1score <- (2 * precision * sensitivity)/(precision + sensitivity)
    return(f1score)
}

# 10. ROC function
roc_function <- function(df) {

    sensitivity_list <- c()
    specificity_list <- c()

    for (i in seq(0, 1, 0.01)) {

        df$new.scored <- ifelse(df$scored.probability < i, 0,
            1)

        sensitivity_list[i * 100 + 1] <- sensitivity_function(df,
            "new.scored")

        specificity_list[i * 100 + 1] <- specificity_function(df,
            "new.scored")

    }

    roc_df = data.frame(threshold = seq(0, 1, 0.01), sensitivity = sensitivity_list,
        specificity = specificity_list)

    data = data.frame(y = sensitivity_list, x = rep(0.01, length(roc_df[,
        1])))
    data$area = data$y * data$x
    # data = complete.cases(data)
```

```r
    area = sum(data$area, na.rm = TRUE)
    # print(area) print(data)
    return(list(roc_df, area))

}

roc.results <- roc_function(df)
results1 = roc.results[[1]]

ggplot(results1, aes(x = 1 - specificity, y = sensitivity)) +
    geom_line() + ggtitle("ROC Curve")

# 12. Using Caret package
caret_confusion = confusionMatrix(df$scored.class, df$class,
    positive = "1")
pander(caret_confusion$table)

# 13. ROC curve
roc_curve <- roc(class ~ scored.probability, data = df)

plot(y = roc_curve$sensitivities, x = 1 - roc_curve$specificities,
    col = "blue", main = "ROC curve", type = "l", ylab = "Sensitivity",
    xlab = "1 - Specificity")

area_roc = sum(roc_curve$sensitivities * (1/length(roc_curve$sensitivities)))
```