Overview

DATA 621 – Business Analytics and Data Mining

Homework 2

Ramnivas Singh

2022-03-20

Overview

In this homework assignment, you will work through various classification metrics. You will be asked to create functions in R to carry out the various calculations. You will also investigate some functions in packages that will let you obtain the equivalent results. Finally, you will create graphical output that also can be used to evaluate the output of classification models, such as binary logistic regression.

Supplemental Material

- Applied Predictive Modeling, Ch. 11 (provided as a PDF file).
- Web tutorials: http://www.saedsayad.com/model_evaluation_c.htm (http://www.saedsayad.com/model_evaluation_c.htm)

Deliverables (100 Points)

- Upon following the instructions below, use your created R functions and the other packages to generate the classification metrics for the provided data set. A write-up of your solutions submitted in PDF format.

```
library(knitr)
library(caret)
library(tidyr)
library(dplyr)
library(ggplot2)
library(DT)
library(data.table)
library(kableExtra)
library(corrplot)
library(ggcorrplot)
library(pROC)
```

Instructions

Complete each of the following steps as instructed:

1. Download the classification data set

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

Solution :

```
df <- read.csv("https://raw.githubusercontent.com/rnivas2028/MSDS/Data621/HW2/classification-output-data.csv")
DT::datatable(df, options = list(pagelength=5))
```

Show 10 entries Search:

	pregnant	glucose	diastolic	skinfold	insulin	bmi	pedigree	age	class	scored.class	scored.probability
1	7	124	70	33	215	25.5	0.161	37	0	0	0.328452259
2	2	122	76	27	200	35.9	0.483	26	0	0	0.273190439
3	3	107	62	13	48	22.9	0.678	23	1	0	0.109660394
4	1	91	64	24	0	29.2	0.192	21	0	0	0.055998355
5	4	83	86	19	0	29.3	0.317	34	0	0	0.100490719
6	1	100	74	12	46	19.5	0.149	28	0	0	0.055154596

	pregnant	glucose	diastolic	skinfold	insulin	bmi	pedigree	age	class	scored.class	scored.probability
7	9	89	62	0	0	22.5	0.142	33	0	0	0.107115418
8	8	120	78	0	0	25	0.409	64	0	0	0.459947437
9	1	79	60	42	48	43.5	0.678	23	0	0	0.117023677
10	2	123	48	32	165	42.1	0.52	26	0	0	0.315363199

Showing 1 to 10 of 181 entries Previous 1 2 3 4 5 … 19 Next

Defined a class variable to use later in this assignment

```
df_class <- dplyr::select(df, scored.class, class)
table(df_class)
```

```
##             class
## scored.class   0   1
##            0 119  30
##            1   5  27
```

2. Confusion Matrix & Interpretation

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?

Solution :

```
#conf_mat_table<-table(df$class, df$scored.class)
conf_mat_table <- df %>%
  select(class, scored.class, scored.probability)
DT::datatable(conf_mat_table, options = list(pagelength=5))
```

Show 10 entries Search:

	class	scored.class	scored.probability
1	0	0	0.328452259
2	0	0	0.273190439
3	1	0	0.109660394
4	0	0	0.055998355
5	0	0	0.100490719
6	0	0	0.055154596
7	0	0	0.107115418
8	0	0	0.459947437
9	0	0	0.117023677
10	0	0	0.315363199

Showing 1 to 10 of 181 entries Previous 1 2 3 4 5 … 19 Next

Interpreting the output of this Confusion Matrix: The rows represent Actual Values, and the columns represent Predicted Values. Let's rename the rows and columns to to make it clearer.

```
conf_mat_table %>%
select(class, scored.class) %>%
mutate(class = recode(class,
                      '0' = 'Actual Negative',
                      '1' = 'Actual Positive'),
       scored.class = recode(scored.class,
                             '0' = 'Predicted Negative',
                             '1' = 'Predicted Positive')) %>%
table()
```

```
##                 scored.class
## class            Predicted Negative Predicted Positive
##   Actual Negative                119                  5
##   Actual Positive                 30                 27
```

The field class (the rows) represent the actual class, and the field scored.class (the columns) represent the predicted class

3. Accuracy Predictions

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

Solution :

A function accuracy is created to represent the formula of Accuracy. Accuracy means the closeness of the measurements to a specific value. Description of variables in the function: TP: True Positive, TN: True Negative

```
accuracy <- function(actual, predicted)
{
  conf_mat_table <- as.matrix(table(predicted, actual))
  print(conf_mat_table)
  TN <- conf_mat_table[1,1]
  FN <- conf_mat_table[1,2]
  FP <- conf_mat_table[2,1]
  TP <- conf_mat_table[2,2]
  return ((TP + TN) / (TN + FN + TP + FP))
}
```

Lets call function accuracy

```
accuracy_val<-accuracy(df$class, df$scored.class)
```

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

```
print(accuracy_val)
```

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

4. Classification 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.

Solution :

A function class_err_rate is created to returns the classification error rate of the predictions.

```
class_err_rate <- function(actual, predicted)
{
  conf_mat_table <- as.matrix(table(predicted, actual))
  TN <- conf_mat_table[1,1]
  FN <- conf_mat_table[1,2]
  FP <- conf_mat_table[2,1]
  TP <- conf_mat_table[2,2]
  return ((FP + FN) / (TN + FN + TP + FP))
}
```

Lets call function class_err_rate

```
class_err_rt_val<-class_err_rate(df$class, df$scored.class)
print(class_err_rt_val)
```

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

Hypotheitcal Test

Verify that you get an accuracy and an error rate that sums to one.

```
accuracy(df$class, df$scored.class) + class_err_rate(df$class, df$scored.class)
```

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

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

5. Precision of the predictions

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

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

Note: False positive rate of the classifier is otherwise called Type-1 Error.

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

Solution :

```
precision <- function(actual, predicted){
conf_mat_table <- as.matrix.data.frame(table(predicted, actual))
TN <- conf_mat_table[1,1]
FN <- conf_mat_table[1,2]
FP <- conf_mat_table[2,1]
TP <- conf_mat_table[2,2]
return (TP / (TP + FP))
}
```

Lets call function precision

```
precision_val<-precision(df$class, df$scored.class)
print(precision_val)
```

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

6. Sensitivity of the predictions

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 sensitivity.

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

Solution :

```
sensitivity <- function(actual, predicted){
  conf_mat_table <- as.matrix.data.frame(table(predicted, actual))
  TN <- conf_mat_table[1,1]
  FN <- conf_mat_table[1,2]
  FP <- conf_mat_table[2,1]
  TP <- conf_mat_table[2,2]
  return (TP / (TP + FN))
}
```

Lets call function sensitivity

```
sensitivity_val<-sensitivity(df$class, df$scored.class)
print(sensitivity_val)
```

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

7. Specificity of the predictions

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

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

Solution :

```
specficity <- function(actual, predicted){
  conf_mat_table <- as.matrix.data.frame(table(predicted, actual))
  TN <- conf_mat_table[1,1]
  FN <- conf_mat_table[1,2]
  FP <- conf_mat_table[2,1]
  TP <- conf_mat_table[2,2]
  return (TN / (TN + FP))
}
```

Lets call function specficity

```
specficity_val<-specficity(df$class, df$scored.class)
print(specficity_val)
```

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

8. F1 score of the predictions

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.

$F1Score = \frac{2*precision*Sensitivity}{precision+Sensitivity}$

Solution :

```
  f1score <- function(precision_val, sensitivity_val){
  return (2* precision_val * sensitivity_val/ (precision_val + sensitivity_val))
}
```

Lets call function f1score

```
f1score_val<-f1score(precision_val, sensitivity_val)
print(f1score_val)
```

```
## [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.

Solution :

F1 score is calculated based on precision and sensitivity. Bounds for Precision and sensitivity. are between 0 and 1. For any values of precision and sensitivity. between their bounds, F1 score will fall in the range of 0 and 1.

10. ROC Curve

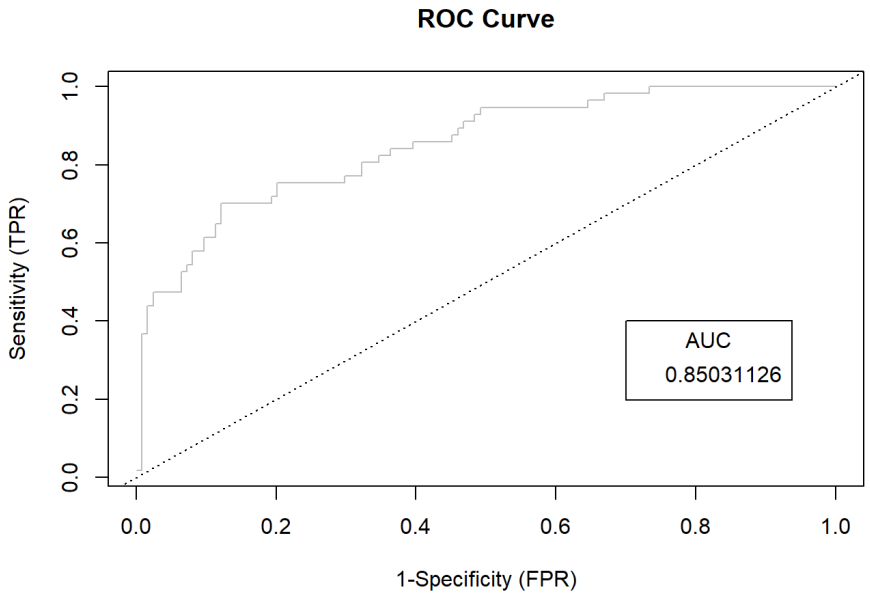
Write a function that generates an ROC curve from a data set with a true classification column (class in our example) and a probability column (scored.probability in our example). Your function should return a list that includes the plot of the ROC curve and a vector that contains the calculated area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

Solution :

```
roc_curv <- function(x,y){
  x <- x[order(y, decreasing=TRUE)]
  TP = cumsum(x)/sum(x)
  FP = cumsum(!x)/sum(!x)
  df <- data.frame(TP, FP)
  diffFP <- c(diff(FP), 0)
  diffTP <- c(diff(TP), 0)
  auc <- sum(TP * diffFP) +
    sum(diffTP * diffFP)/2
  return(c(df=df, auc = auc))
}
```

Lets call function roc_curv

```
roc_data <- roc_curv( df$class,  df$scored.probability)
plot(roc_data[[2]],roc_data[[1]], type = 'l',
     main = "ROC Curve",
     xlab="1-Specificity (FPR)",
     ylab = "Sensitivity (TPR)", col="grey")
abline(0,1, lty=3)
legend(0.7,0.4, round(roc_data$auc,8), title = 'AUC')
```



11. Classification metrics

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

Solution :

```
result_df <- c(accuracy_val, class_err_rt_val, f1score_val, precision_val, sensitivity_val, specficity_val)
names(result_df) <- c("Accuracy", "Error", "F1", "precision", "Sensitivity", "Specficity")
result_df
```

```
##    Accuracy       Error          F1   precision Sensitivity  Specficity
##   0.8066298   0.1933702   0.6067416   0.8437500   0.4736842   0.9596774
```

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?

Solution :

```
df_class$scored.class <- as.factor(df_class$scored.class)
df_class$class <- as.factor(df_class$class)
confusionMatrix(df_class$scored.class, df_class$class, mode = 'everything')
```

```
## 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 : 0.9597
##             Specificity : 0.4737
##          Pos Pred Value : 0.7987
##          Neg Pred Value : 0.8438
##               Precision : 0.7987
##                  Recall : 0.9597
##                      F1 : 0.8718
##              Prevalence : 0.6851
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.8232
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
##
```

```
sensitivity(df_class$scored.class, df_class$class)
```

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

```
specificity(df_class$scored.class, df_class$class)
```

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

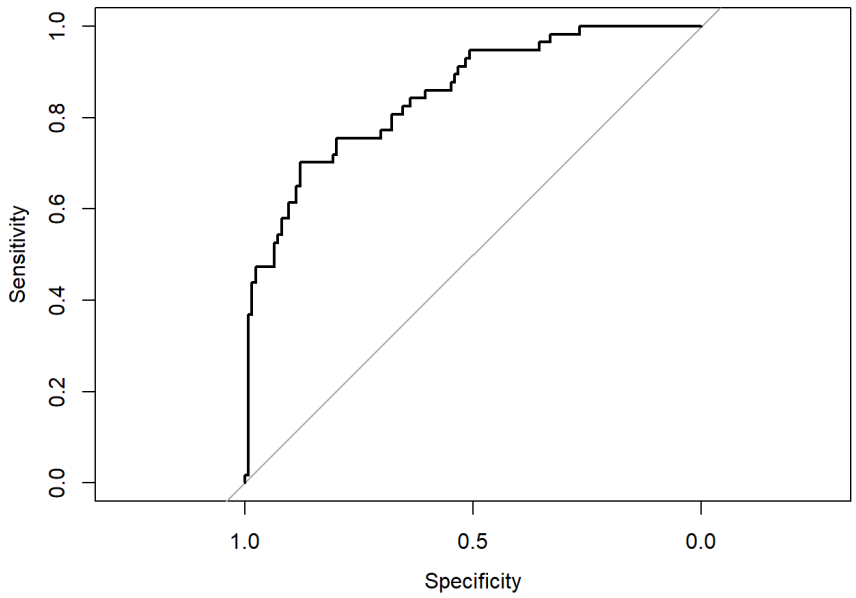
The values from the built in function are almost the similar to the hand created ones.

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

Solution :

```
plot(roc(df$class, df$scored.probability),
     colorize=TRUE, print.cutoffs.at=seq(0.1,by=0.1))
```



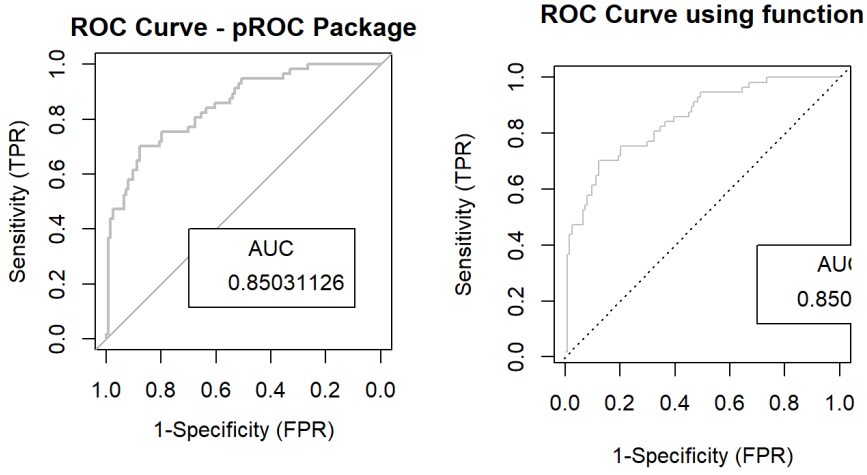

```
auc(roc(df$class, df$scored.probability))
```

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

Both the built in function and hand-made functions are almost similar to each other

```
par(mfrow = c(1, 2), pty = 's')
plot(roc(df$class, df$scored.probability),
     main = 'ROC Curve - pROC Package',
     xlab="1-Specificity (FPR)",
     ylab = "Sensitivity (TPR)", col="grey")
#abline(0,1, lty=3)
legend(0.7,0.4, round(roc_data$auc,8), title = 'AUC')

plot(roc_data[[2]],roc_data[[1]], type = 'l',
     main = "ROC Curve using function",
     xlab="1-Specificity (FPR)",
     ylab = "Sensitivity (TPR)", col="grey")
abline(0,1, lty=3)
legend(0.7,0.4, round(roc_data$auc,8), title = 'AUC')
```



References

Bibliography

Applied Predictive Modeling

Resource Links

• Web tutorials: http://www.saedsayad.com/model_evaluation_c.htm (http://www.saedsayad.com/model_evaluation_c.htm)

• Caret Package Homepage: http://topepo.github.io/caret/index.html (http://topepo.github.io/caret/index.html)

• Caret Package on CRAN: http://cran.r-project.org/web/packages/caret/ (http://cran.r-project.org/web/packages/caret/)

• A Short Introduction to the caret Package: http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf (http://cran.r-project.org/web/packages/caret/vignettes/caret.pdf)