Classfication Metrics

Shyam BV

March 10, 2018

Contents

1 Classification Metrics : Create various functions and perform classification metrics

---

```
#packages <- c("dplyr", "caret","ggplot","pROC")

#installed_and_loaded <- function(pkg){
#  new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
#  if (length(new.pkg)) install.packages(new.pkg, dependencies = TRUE)
#  sapply(pkg, require, character.only = TRUE, quietly = TRUE, warn.conflicts = #FALSE)
#}

#excute function and display the loaded packages
#data.frame(installed_and_loaded(packages))


library("dplyr")
library("caret")
library("ggplot2")
library("pROC")
```

1.1 Data Loading

The data set has three key columns we will use: 1. class: the actual class for the observation 2. scored.class: the predicted class for the observation (based on a threshold of 0.5) 3. scored.probability: the predicted probability of success for the observation

```
df = read.csv('./data/classification-output-data.csv') %>% select(c('class','scored.class','scored.proba
```

Below is the sample of our dataset. Also the table shows confusion matrix of all the rows. It shows how many many number records or classes that were correctly and wrongly predicted by the model.

Actual response classes `class` are in columns and predicted response classes `scored.class` are in row format.

True positive classes - 119 True negative classes - 27 False Positive classes - 30 False Negative classes - 5

```
print(head(df))
```

```
##   class scored.class scored.probability
## 1     0            0         0.32845226
## 2     0            0         0.27319044
## 3     1            0         0.10966039
## 4     0            0         0.05599835
## 5     0            0         0.10049072
## 6     0            0         0.05515460
```

```
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

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

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

1.2 Accuracy

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

It is the overall accuracy of the model.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

```
accuracy = function(df,positive){

  # Get unique class
all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

#TP
tp = dim(df[df$class== df$scored.class & df$class== positive,])[1]
#TN
tn = dim(df[df$class== df$scored.class & df$class== negative,])[1]

fp = dim(df[df$class != df$scored.class & df$class== negative & df$scored.class== positive ,])[1]
fn = dim(df[df$class!= df$scored.class & df$class== positive& df$scored.class== negative,])[1]

#print(paste(positive,':',tp,' ',fp))
#print(paste(negative,':',fn,' ',tn))

return((tp+tn)/(tp+tn+fp+fn))

}
```

```
print("Accuracy:")
```

```
## [1] "Accuracy:"
```

```
accuracy(df,0)
```

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

1.3 Classification Error Rate

Classification error rate is overall error of the model. It is a ratio of all the misclassified classes and the overall prediction of the model.

$$classificationerrorRate = (FP + FN)/(TP + TN + FP + FN)$$

```
classification_error_Rate = function(df,positive){

all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

tp = dim(df[df$class== df$scored.class & df$class== positive,])[1]
tn = dim(df[df$class== df$scored.class & df$class== negative,])[1]
```

```
fp = dim(df[df$class != df$scored.class & df$class== negative & df$scored.class== positive ,])[1]
fn = dim(df[df$class!= df$scored.class & df$class== positive& df$scored.class== negative,])[1]

#print(paste(positive,':',tp,' ',fp))
#print(paste(negative,':',fn,' ',tn))

return((fp+fn)/(tp+tn+fp+fn))

}

classification_error_Rate(df,0)
```

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

```
print("Classification Error Rate:")
```

```
## [1] "Classification Error Rate:"
```

```
classification_error_Rate(df,0)
```

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

1.3.1 Validation

For a model, accuracy and error rate should add up to 100%. These two measures are complement to each other.

$$Accuracy + ClassificationErrorRate = 1$$

```
verify = function(accuracy, classfication_error){
  return(accuracy+ classfication_error)
}

verify(0.8066298,0.1933702)
```

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

1.4 Precision or PPV

Precision is a metric which shows how good is the model. It shows the ratio of true positives and the predicted positives.

$$precision = (TP)/(TP + FP)$$

```
precision =  function(df,positive){

all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

tp = dim(df[df$class== df$scored.class & df$class== positive,])[1]
tn = dim(df[df$class== df$scored.class & df$class== negative,])[1]
```

```
fp = dim(df[df$class != df$scored.class & df$class== negative & df$scored.class== positive ,])[1]
fn = dim(df[df$class!= df$scored.class & df$class== positive& df$scored.class== negative,])[1]

return((tp)/(tp+fp))

}
```

```
print("Precession:")
```

```
## [1] "Precession:"
```

```
precision(df,0)
```

```
## [1] 0.7986577
```

1.5 Recall or Sensitivity

Recall or Sensitivity or True positive rate is ratio of True positives of model to the actual positives. It is an important metric which measures the propotion of posivites which are correctly identified.

$$Sensitivity = (TP)/(TP + FN)$$

```
senstivitiy =  function(df,positive,original,predicted){

all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

tp = dim(df[df[original]== df[predicted] & df[original]== positive,])[1]
tn = dim(df[df[original]== df[predicted] & df[original]== negative,])[1]

fp = dim(df[df[original] != df[predicted] & df[original]== negative & df[predicted]== positive ,])[1]
fn = dim(df[df[original]!= df[predicted] & df[original]== positive& df[predicted]== negative,])[1]

return((tp)/(tp+fn))

}
```

```
print("Sensitivity:")
```

```
## [1] "Sensitivity:"
```

```
senstivitiy(df,0,'class','scored.class')
```

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

1.6 Specificity

Specificity or True negative rate is ratio of True negatives of model to the actual negatives It is an important metric which measures the propotion of negatives which are correctly identified.

$$Specificity = (TN)/(TN + FP)$$

```
specificity = function(df,positive,original,predicted){

all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

tp = dim(df[df[original]== df[predicted] & df[original]== positive,])[1]
tn = dim(df[df[original]== df[predicted] & df[original]== negative,])[1]

fp = dim(df[df[original] != df[predicted] & df[original]== negative & df[predicted]== positive ,])[1]
fn = dim(df[df[original]!= df[predicted] & df[original]== positive& df[predicted]== negative,])[1]

return((tn)/(fp+tn))

}

specificity(df,0,'class','scored.class')
```

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

```
print("Specificity:")
```

```
## [1] "Specificity:"
```

```
specificity(df,0,'class','scored.class')
```

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

1.7 F1-score

F1-score is the harmonic mean of Sensitivity and specificity. It is an single number which mentiones about the performance of model. Bounds of F1-score is between 0 and 1.

$$F1 - score = (2 * Precision * Specificity)/(Precision + Specificity)$$

```
f1_score =  function(df,positive){

all_class = unique(df$class)
positive =c(positive)
negative = all_class[-match(positive, all_class)][1]

score_precision = precision(df,positive)
score_senstivitiy = senstivitiy(df,positive,'class','scored.class')

return((2*score_precision*score_senstivitiy)/(score_precision+score_senstivitiy))

}
```

```
print("F1-score:")
```

```
## [1] "F1-score:"
```

```
f1_score(df,0)
```

```
## [1] 0.8717949
```

```
#F1 score validation

f1_score_check =  function(precision, sensitivity){

  return((2*precision*sensitivity)/(precision+sensitivity))

}

# Precision is 0 and Sensitivity is 1
f1_score_check(0,1)
```

```
## [1] 0
```

```
# Precision is 1 and Sensitivity is 0
f1_score_check(1,0)
```

```
## [1] 0
```

```
# Precision is 1 and Sensitivity is 1
f1_score_check(1,1)
```

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

```
# Precision is .50 and Sensitivity is .90
f1_score_check(.5,.9)
```

```
## [1] 0.6428571
```

```
# Precision is .80 and Sensitivity is .50
f1_score_check(.8,.5)
```

```
## [1] 0.6153846
```

1.8 ROC curve & AUC

`ROC curve` & `AUC` are important evaluation metrics which identifies the performance of the model.

1.8.1 ROC Curve

ROC curve(Receiver Operating Characterstic) is a plot which shows the model classification at various threshold probability. At each threshold, it calculates the Sensitivity and False Positive Rate(1-Specificity) for the whole instance. If the ROC curve is stepwise, then the performance of the model is good.

Here the ROC curve is generated manually and the thresholds are ranging from 0 to 1 at 0.01 intervals.

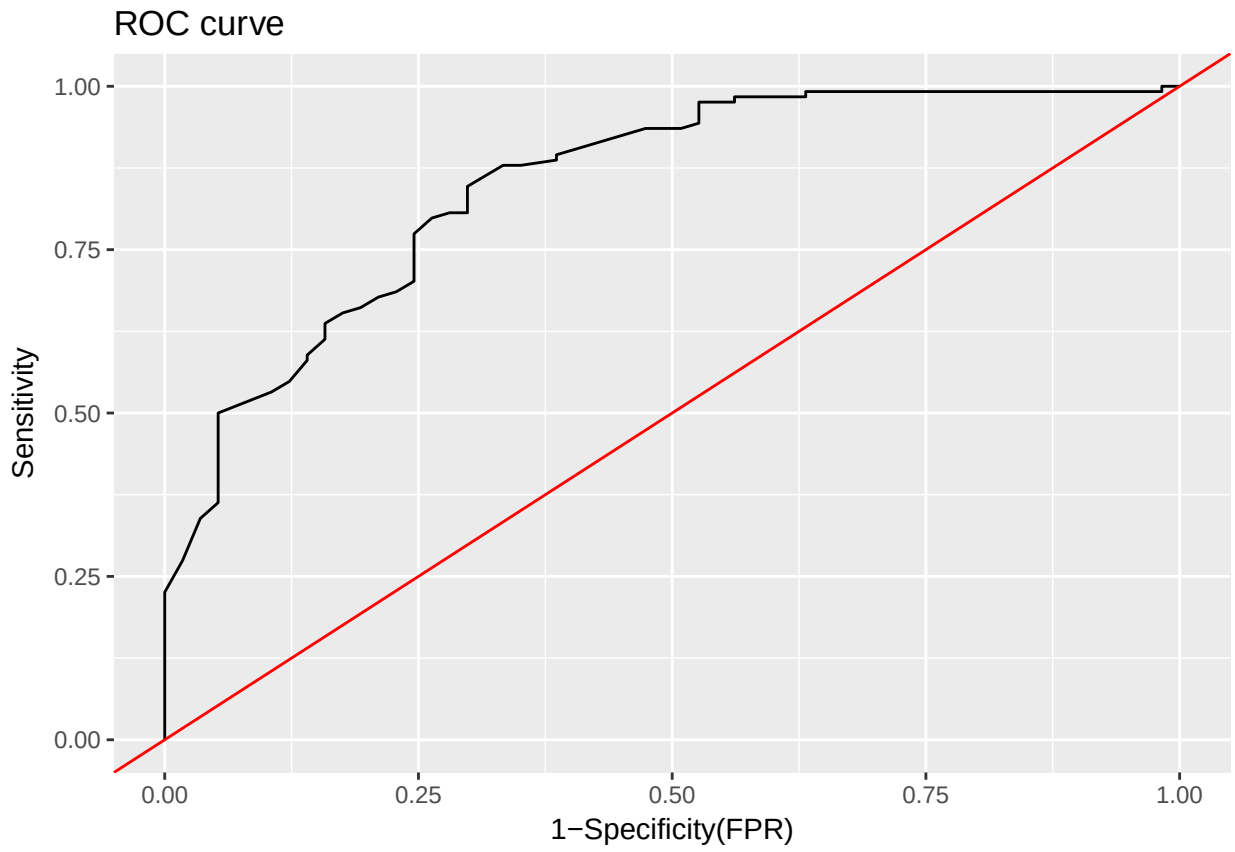
```
roc_curve = function(df,positive, threshold){

# Predict the class based on the threshold
  df$predicted_class =  if_else(df$scored.probability > threshold,1, 0, missing = NULL)

#TPR
tpr = senstivitiy(df,0,'class','predicted_class')

#FPR
fpr = 1- specificity(df,0,'class','predicted_class')

return(c(fpr,tpr))
```

```
}


roc_df = data.frame(matrix(ncol=2,nrow=0))

for(i in seq(0,1,0.01)) {
roc_df <- rbind(roc_df,roc_curve(df,0,i))

}
colnames(roc_df)  <- c('fpr','tpr')
```

```
head(roc_df)
```

```
##   fpr        tpr
## 1   0 0.00000000
## 2   0 0.00000000
## 3   0 0.00000000
## 4   0 0.02419355
## 5   0 0.03225806
## 6   0 0.04838710
```

```
ggplot(roc_df,aes(fpr,tpr)) + geom_line() + geom_abline(color='red') +xlab('1-Specificity(FPR)') + ylab
```



Diagonal line is the random lane. If the points fall on it, then the model is same as random model. If the curve is below reference lane, then the model is bad.

To create this curve, we need to follow below steps

1. Findout the Probability
2. Generate threshold and set it
3. Classify the classes according to threshold
4. Calculate Sensitivity, specificity
5. Loop from step2 until all thresholds are met.

1.8.2 AUC Curve

Area under the curve is an metric which calcuates the overall area under the ROC curve. This is a good metric to mention about the overall performance of model.

```
simple_auc <- function(TPR, FPR){
# Area under the curve
  dFPR <- c(diff(FPR), 0)
  dTPR <- c(diff(TPR), 0)

    sum(TPR * dFPR) + sum(dTPR * dFPR)/2
}
```

```
print('Area under the curve:')
```

```
## [1] "Area under the curve:"
```

```
with(roc_df, simple_auc(tpr, fpr))
```

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

1.9 Compare function

Lets run all the manually created evaluation metric functions to show the overall report.

```
# Accuracy
print('Accuracy:')
```

```
## [1] "Accuracy:"
```

```
accuracy(df,0)
```

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

```
#Classfication Error rate
print('Classification Error Rate:')
```

```
## [1] "Classification Error Rate:"
```

```
classification_error_Rate(df,0)
```

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

```
print('Precision:')
```

```
## [1] "Precision:"
```

```
precision(df,0)
```

```
## [1] 0.7986577
```

```
print('Sensitivity:')
```

```
## [1] "Sensitivity:"
```

```
senstivitiy(df,0,'class','scored.class')
```

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

```
print('Specificity:')
```

```
## [1] "Specificity:"
```

```
specificity(df,0,'class','scored.class')
```

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

```
print('F1-score:')
```

```
## [1] "F1-score:"
```

```
f1_score(df,0)
```

```
## [1] 0.8717949
```

```
print('Area under the curve:')
```

```
## [1] "Area under the curve:"
```

```
with(roc_df, simple_auc(tpr, fpr))
```

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

Below is the metrics which are calculated from the predefined functions from `caret` package.

```
confusionMatrix(data = df$scored.class,
                reference = df$class,
                positive = "0")
```

```
## 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
##              Prevalence : 0.6851
##          Detection Rate : 0.6575
##    Detection Prevalence : 0.8232
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 0
```

##

By comparing the results, it shows that all the metrics which are calculated manually matches with the predefined functions.

1.10 ROC curve using pROC

Below is the ROC curve from the package `pROC`.

```
rocCurve <- roc(response = df$class,
                predictor = df$scored.probability,
                  levels = c(1,0))
```

```
print('ROC Curve:')
```

```
## [1] "ROC Curve:"
```

```
auc(rocCurve)
```

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

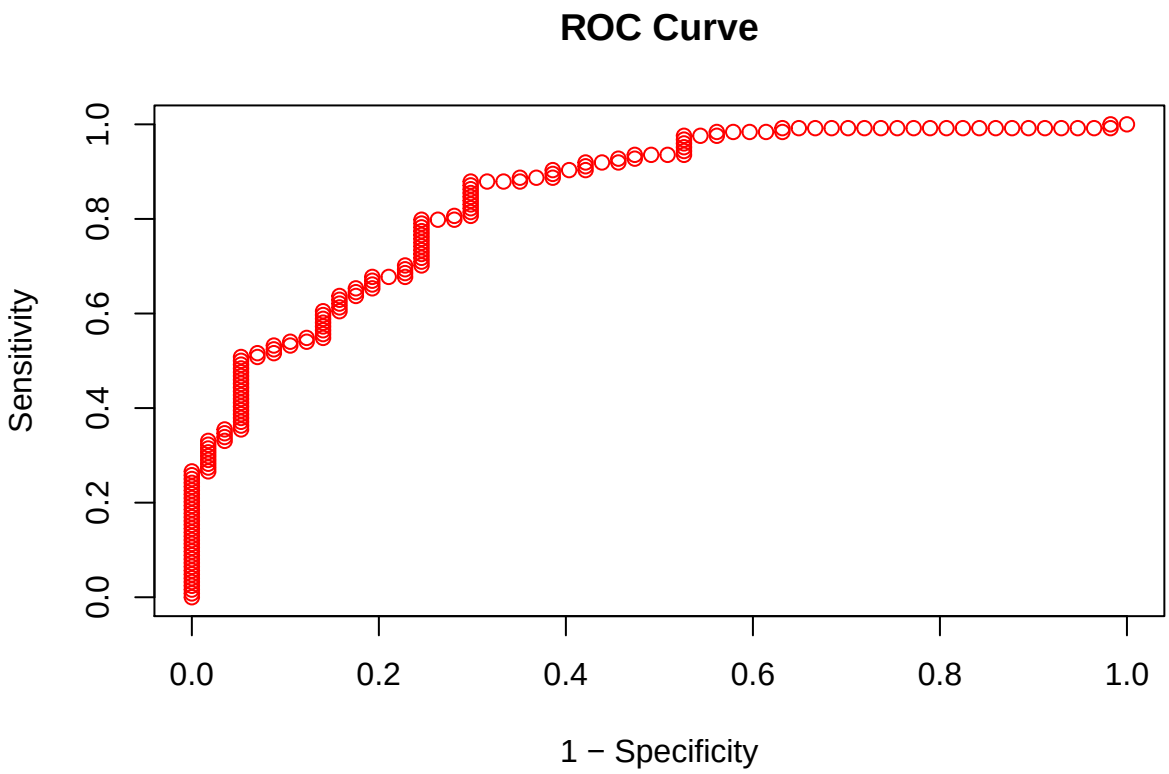
```
print('Confidence interval of AUC curve:')
```

```
## [1] "Confidence interval of AUC curve:"
```

```
ci.auc(rocCurve)
```

```
## 95% CI: 0.7905-0.9101 (DeLong)
```

```
plot(y = rocCurve$sensitivities, ylab = "Sensitivity", x = 1 - rocCurve$specificities, xlab = "1 - Speci
main = "ROC Curve", col = "red")
```


ROC Curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1 − Specificity