Measuring Performance in Classification Models

Rafal Decowski

Contents

Measuring Performance in Classification Models

Confusion Matrix

Using the built in table function, we can generate a raw confusiton matrix for the given dataset. The columns represent the predicted class and the rows the actual class.

```
# Determine cross reference on the classes
confusion_table = table(df[,'scored.class'], df[,'class'])
confusion_table
```

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

Define 4 possible outcomes

```
# Initialize 4 possible outcomes [TP, TN, FP, FN]
# True Positive, True Negative, False Positive, False Negative
confusion_table[1,1] = 'TN'
confusion_table[1,2] = 'FN'
confusion_table[2,1] = 'FP'
confusion_table[2,2] = 'TP'
confusion_table
```

```
##
##     0  1
##   0 TN FN
##   1 FP TP
```

Accuracy

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

```
get_accuracy <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]
  accuracy = round((TP + TN) / sum(TP,FP,TN,FN), 2)
  return(accuracy)
}
```

Classification Error Rate

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

```
get_classification_error_rate <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]
  classification_error_rate = round((FP + FN) / sum(TP,FP,TN,FN),2)
  return(classification_error_rate)
}
```

Precision

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

```
get_precision <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]
  precision = round(TP / (TP + FP), 2)
  return(precision)
}
```

Sensitivity

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

```
get_sensitivity <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]
  sensitivity = round(TP / (TP + FN), 2)
  return(sensitivity)
}
```

Specificity

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

```
get_specificity <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]
  specificity = round(TN / (TN + FP), 2)
  return(specificity)
}
```

F1

$$F1\ Score = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}$$

```
get_f1_score <- function(df, predicted, actual){
  confusion_table = table(df[,predicted], df[,actual])
  TP = confusion_table[2,2]
  TN = confusion_table[1,1]
  FN = confusion_table[1,2]
  FP = confusion_table[2,1]

  precision = round(TP / (TP + FP), 2)
  sensitivity = round(TP / (TP + FN), 2)
  f1_score = round((2 * precision * sensitivity) / (precision + sensitivity), 2)
  return(f1_score)
}
```

What are the bounds on the F1 score? - Knowing that both precision and sensitivity are bound between 0 and 1 as they are percentages of the calculated values, we can use their min and max values as a test to prove that f1 score also falls between 0 & 1.

Min: (2 x 0 x 0) / (0 + 0) - error div by 0

Mid: (2 x 0.5 x 0.5) / (0.5 + 0.5) 0.5 / 1 = 0.5

Max: (2 x 1 x 1) / (1 + 1) 2 / 2 = 1

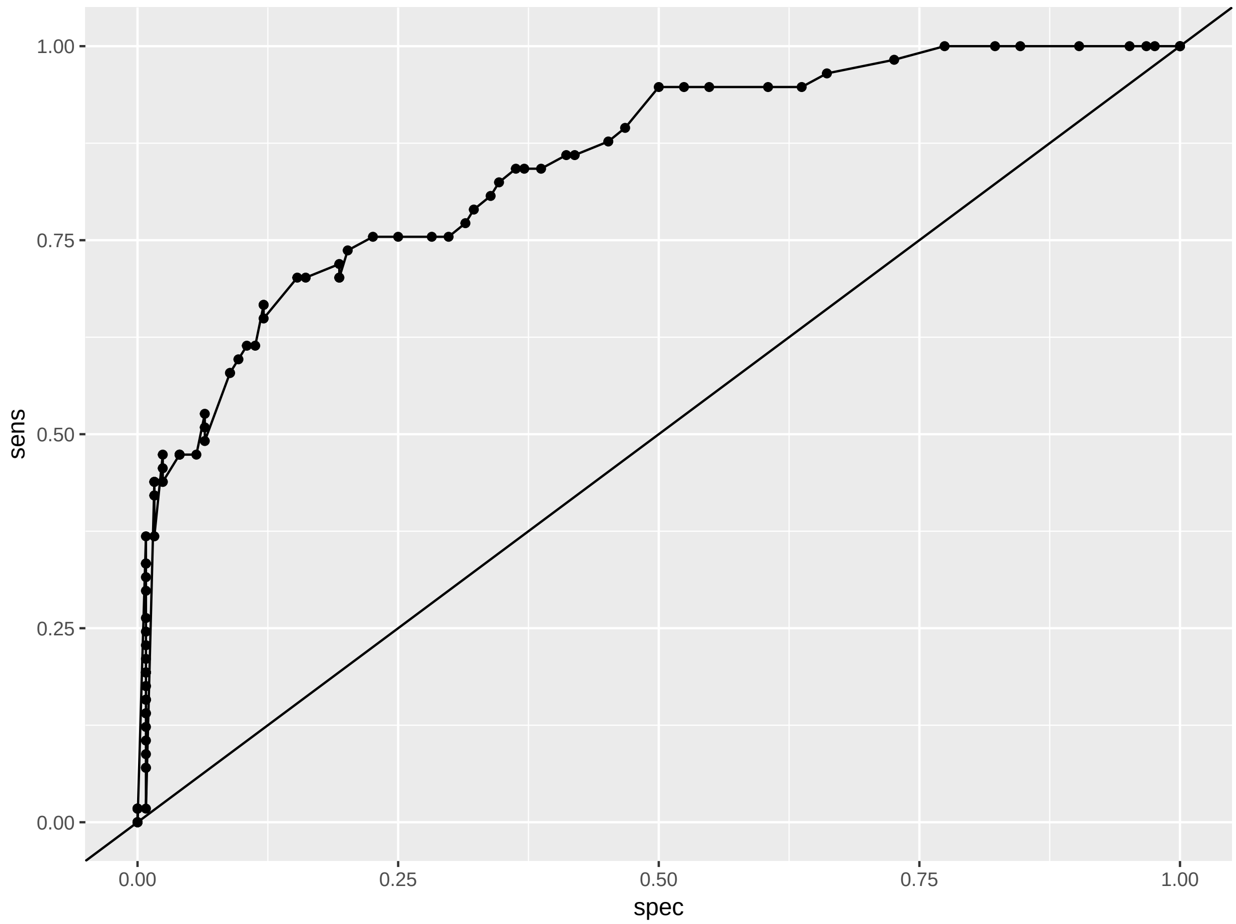
Programming test: Generate 100 random fractional values between 0 & 1 and test the following: If (0 < a < 1) and (0 < b < 1) then (ab < a)

```
a <- runif(100)
b <- runif(100)
for (i in range(1:100)){
  v <- a[i]*b[i]
  if ( v > a[i]){
    print(v)
  }
}
```

The loop will never print because ab is always smaller than a

Manual Calculation of Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC)

```
get_roc <- function(df){
  # Define threshold values between 0 and 1, incrementing by 0.01
  threshold <- seq(0,1,0.01)

  sens <- c()
  spec <- c()

  # For every threshold value, determine
  for (t in threshold){
    sens <- append(sens, sum((df$scored.probability >= t & df$class == 1)) / sum(df$class == 1))
    spec <- append(spec, sum((df$scored.probability >= t & df$class == 0)) / sum(df$class == 0))
  }
  # Push the resulted vectors to dataframe for plotting
  tmp_df <- data.frame(sens=sens, spec=spec)
  # Plot
  roc_plot <- ggplot(tmp_df, aes(x=spec, y=sens, group=1)) +
    geom_line() +
    geom_point() +
    geom_abline(intercept = 0, slope = 1)

  #Area Under the Curve (AUC)
  pos = df[df$class == 1, 11]
  neg = df[df$class == 0, 11]
  auc_value = mean(replicate(100000, sample(pos, size=1) > sample(neg, size=1)))

  return(list(plot=roc_plot, auc=auc_value))
}

rocauc <- get_roc(df)

rocauc$plot
```


1.00
0.75
0.50
0.25
0.00
sens
0.00
0.25
0.50
0.75
1.00
spec

Complete Evaluation

```
score = data.frame(accuracy=get_accuracy(df, 'scored.class', 'class'),
                   classification_error_rate=get_classification_error_rate(df, 'scored.class', 'class')
                   precision=get_precision(df, 'scored.class', 'class'),
                   sensitivity=get_sensitivity(df, 'scored.class', 'class'),
                   specificity=get_specificity(df, 'scored.class', 'class'),
                   f1_score=get_f1_score(df, 'scored.class', 'class'),
                   auc=unlist(rocauc[2]))
kable(score)
```

	accuracy	classification_error_rate	precision	sensitivity	specificity	f1_score	auc
auc	0.81	0.19	0.84	0.47	0.96	0.6	0.84891

Measurements using the caret package

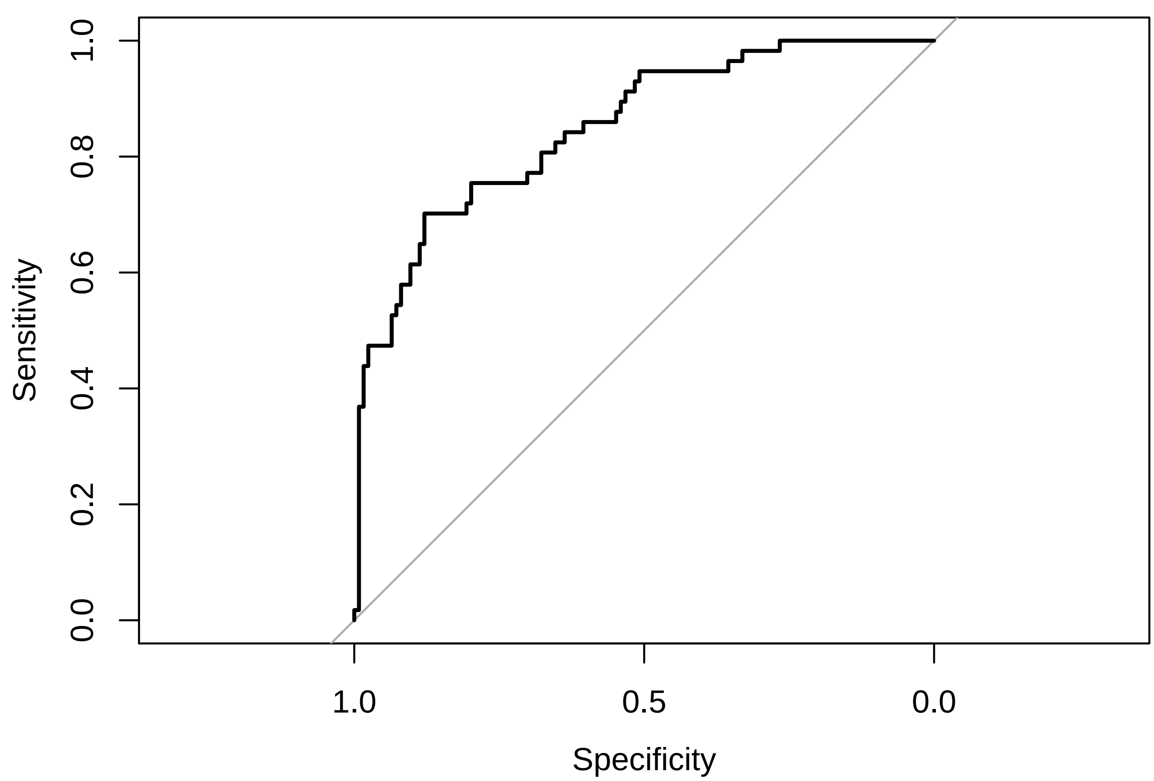
It appears that sensitivity and specificity are reversed from my results. Perhaps caret package requires the confusion matrix to be in a different order. The overall values match my manual calculations.

```r
library(caret)
confusionMatrix(df[,'scored.class'],df[,'class'])
```

```
## 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
##
```

ROC and AUC using the pROC package

```
library(pROC)
roc(df$class ~ df$scored.probability, df, plot=TRUE)
```




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