Confusion Matrix in Machine Learning

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix.  
A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.  
It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

**This article aims at:**

**1. What the confusion matrix is and why you need to use it.  
2. How to calculate a confusion matrix for a 2-class classification problem from scratch.  
3. How to create a confusion matrix in Python.**

**Confusion Matrix:**  
A confusion matrix is a summary of prediction results on a classification problem.  
The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.  
The confusion matrix shows the ways in which your classification model is confused when it makes predictions.  
It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.



Here,  
• Class 1 : Positive  
• Class 2 : Negative

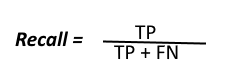
**Definition of the Terms:**  
• Positive (P) : Observation is positive (for example: is an apple).  
• Negative (N) : Observation is not positive (for example: is not an apple).  
• True Positive (TP) : Observation is positive, and is predicted to be positive.  
• False Negative (FN) : Observation is positive, but is predicted negative.  
• True Negative (TN) : Observation is negative, and is predicted to be negative.  
• False Positive (FP) : Observation is negative, but is predicted positive.

**Classification Rate/Accuracy:**  
Classification Rate or Accuracy is given by the relation:  


However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

**Recall or True Positive Rate: (When is actually YES, how often we predicted Yes)**

Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (small number of FN).

Recall is given by the relation:  


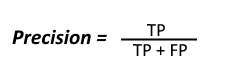
**False Positive Rate:** When it is actually YES, how often we predicted NO

**FPR=**TN/(TN+FP)

**Specificity or True Negative Rate:** When it is actually NO, how often we predicted NO

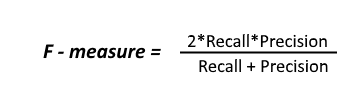
TNR= TN/(TN+FP)

**Precision: (When we predicted Yes, how often it is correct)**

To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labeled as positive is indeed positive (small number of FP).  
Precision is given by the relation:  


**High recall, low precision:**This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

**Low recall, high precision:**This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

**F-measure:**  
Since we have two measures (Precision and Recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.  
The F-Measure will always be nearer to the smaller value of Precision or Recall.  


**F-measure:**  
Fmeasure=(2\*Recall\*Precision)/(Recall+Presision)=(2\*0.95\*0.91)/(0.91+0.95)=0.92

# Python script for confusion matrix creation.

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]

predicted = [1, 0, 0, 1, 0, 0, 1, 1, 1, 0]

results = confusion\_matrix(actual, predicted)

print 'Confusion Matrix :'

print(results)

print 'Accuracy Score :',accuracy\_score(actual, predicted)

print 'Report : '

print classification\_report(actual, predicted)