CONFUSION 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. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

I wanted to create a **"quick reference guide" for confusion matrix terminology** because I couldn't find an existing resource that suited my requirements: compact in presentation, using numbers instead of arbitrary variables, and explained both in terms of formulas and sentences.

Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes):

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

➢ What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

➢ **Let's now define the most basic terms, which are whole numbers (not rates):**

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Demystifying 'Confusion Matrix' Confusion

Step 1: Importing libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
```

Step 2: Loading dataset

```
df = pd.read_csv('C://Users/MALVIKA/Desktop/18mcl2/Project_datascience/cyber_Crime1.csv')
print(df)
```

Step 3 : Defining features and target variable

```
#defining features and target variable
y = df['Below 18 Years']  #target variable we want to predict
X = df.drop(columns = ['Below 18 Years'])  #set of required features, in this case all
```

Step 4 : Splitting the data into train and test set

```
#splitting the data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=50)
```

Step 5 : Predicting using Logistic Regression for Binary classification

```
#Predicting using Logistic Regression for Binary classification
LR = LogisticRegression()
LR.fit(X_train,y_train) #fitting the model
y_pred = LR.predict(X_test) #prediction
```

Step 6 : Evaluation of Model - Con Matrix Plot

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    # """
    # This function prints and plots the confusion matrix.
    # Normalization can be applied by setting `normalize=True`.
    # """
     if normalize:
         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
         print("Normalized confusion matrix")
     else:
         print('Confusion matrix, without normalization')

     print(cm)

     plt.imshow(cm, interpolation='nearest', cmap=cmap)
     plt.title(title)
     plt.colorbar()
     tick_marks = np.arange(len(classes))
     plt.xticks(tick_marks, classes, rotation=45)
     plt.yticks(tick_marks, classes)

     fmt = '.2f' if normalize else 'd'
     thresh = cm.max() / 2.
     for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
         plt.text(j, i, format(cm[i, j], fmt),
                  horizontalalignment="center",
                  color="white" if cm[i, j] > thresh else "black")

     plt.ylabel('True label')
     plt.xlabel('Predicted label')
     plt.tight_layout()
```

Step 7 : Compute confusion matrix

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
np.set_printoptions(precision=2)
```

Step 8 : Plot non-normalized confusion matrix

```
# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['Forged','Authorized'],
                      title='Confusion matrix, without normalization')
```

Step 9 : extracting true_positives, false_positives, true_negatives, false_negatives

```
#extracting true_positives, false_positives, true_negatives, false_negatives
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

Step 10 : `Accuracy`

```
#Accuracy (%)
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy {:0.2f}%".format(Accuracy))
```

Step 11 : `Precision`

```
#Precision
Precision = tp/(tp+fp)
print("Precision {:0.2f}".format(Precision))
```

Step 12 : `Recall`

```
#Recall
Recall = tp/(tp+fn)
print("Recall {:0.2f}".format(Recall))
```

Step 13 : Score

```
#F1 Score
f1 = (2*Precision*Recall)/(Precision + Recall)
print("F1 Score {:0.2f}".format(f1))
```

Step 14 : F1 Score

```
#F1 Score
f1 = (2*Precision*Recall)/(Precision + Recall)
print("F1 Score {:0.2f}".format(f1))
```

Step 15 : F-beta score calculation

```
#Fbeta score
def fbeta(precision, recall, beta):
    return ((1+pow(beta,2))*precision*recall)/(pow(beta,2)*precision + recall)

f2 = fbeta(Precision, Recall, 2)
f0_5 = fbeta(Precision, Recall, 0.5)

print("F2 {:0.2f}".format(f2))
print("\nF0.5 {:0.2f}".format(f0_5))
```

Step 16 : Specificity

```
#Specificity
Specificity = tn/(tn+fp)
print("Specificity {:0.2f}".format(Specificity))
```

OUTPUT :

```
   YEAR  Below 18 Years  ...    Total  class
0  2008              10  ...        0      0
1  2008               0  ...        0      0
2  2008               0  ...        0      0
3  2008              20  ...        0      0
4  2008               0  ...        0      0

[5 rows x 8 columns]
```

```
0    63
1    46
Name: Below 18 Years, dtype: int64
Confusion matrix, without normalization
[[15  0]
 [12  1]]
True Negatives:  15
False Positives:  0
False Negatives:  12
True Positives:  1
Accuracy 57.14%
Precision 1.00
Recall 0.08
F1 Score 0.14
F2 0.09

F0.5 0.29
Specificity 1.00
```