



**towards**  
data science

500K Followers · About

This is your **last** free member-only story this month. [Sign up for Medium and get an extra one](#)

# Confusion Matrix for Your Multi-Class Machine Learning Model

A beginner's guide on how to calculate Precision, Recall, F1-score for a multi-class classification problem.



Joydwip Mohajon May 29 · 6 min read ★





Photo by [Scott Graham](#) on [Unsplash](#)

A confusion matrix is a tabular way of visualizing the performance of your prediction model. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly.

Anyone who is already familiar with the confusion matrix knows that most of the time it is explained for a binary classification problem. Well, this explanation is not one of them. Today we will see how does a confusion matrix work on multi-class machine learning models. However, we will start with a little background using a binary classification just to put things in perspective.

## Confusion Matrix for Binary Classification

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Confusion Matrix for Binary Classification

As you can see, a binary classification problem has only two classes to classify, preferably a positive and a negative class. Now let's look at the metrics of the Confusion Matrix.

**True Positive (TP):** It refers to the number of predictions where the classifier correctly predicts the positive class as positive.

**True Negative (TN):** It refers to the number of predictions where the classifier correctly predicts the negative class as negative.

**False Positive (FP):** It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.

**False Negative (FN):** It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

It's always better to use confusion matrix as your evaluation criteria for your machine learning model. It gives you a very simple, yet efficient performance measures for your model. Here are some of the most common performance measures you can use from the confusion matrix.

**Accuracy:** It gives you the overall accuracy of the model, meaning the fraction of the total samples that were correctly classified by the classifier. To calculate accuracy, use the following formula:  $(TP + TN) / (TP + TN + FP + FN)$ .

**Misclassification Rate:** It tells you what fraction of predictions were incorrect. It is also known as Classification Error. You can calculate it using  $(FP + FN) / (TP + TN + FP + FN)$  or  $(1 - \text{Accuracy})$ .

**Precision:** It tells you what fraction of predictions as a positive class were actually positive. To calculate precision, use the following formula:  $TP / (TP + FP)$ .

**Recall:** It tells you what fraction of all positive samples were correctly predicted as positive by the classifier. It is also known as True Positive Rate (TPR), Sensitivity, Probability of Detection. To calculate Recall, use the following formula:  $TP / (TP + FN)$ .

**Specificity:** It tells you what fraction of all negative samples are correctly predicted as negative by the classifier. It is also known as True Negative Rate (TNR). To calculate specificity, use the following formula:  $TN/(TN+FP)$ .

**F1-score:** It combines precision and recall into a single measure. Mathematically it's the harmonic mean of precision and recall. It can be calculated as follows:

$$F_1\text{--score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

Now, in a perfect world, we'd want a model that has a precision of 1 and a recall of 1. That means a F1-score of 1, i.e. a 100% accuracy which is often not the case for a machine learning model. So what we should try, is to get a higher precision with a higher recall value. Okay, now that we know about the performance measures for confusion matrix, Let's see how we can use that in a multi-class machine learning model.

. . .

### Confusion Matrix for Multi-Class Classification

For simplicity's sake, let's consider our multi-class classification problem to be a 3-class classification problem. Say, we have a dataset that has three class labels, namely *Apple*, *Orange* and *Mango*. The following is a possible confusion matrix for these classes.

		True Class		
		Apple	Orange	Mango
Class	Apple	7	8	9

Predicted	Orang	1	2	3
	Mango	3	2	1

Confusion Matrix for Multi-Class Classification

Unlike binary classification, there are no positive or negative classes here. At first, it might be a little difficult to find TP, TN, FP and FN since there are no positive or negative classes, but it's actually pretty easy. What we have to do here is to find TP, TN, FP and FN for each individual class. For example, if we take class Apple, then let's see what are the values of the metrics from the confusion matrix.

- $TP = 7$
- $TN = (2+3+2+1) = 8$
- $FP = (8+9) = 17$
- $FN = (1+3) = 4$

Since we have all the necessary metrics for class Apple from the confusion matrix, now we can calculate the performance measures for class Apple. For example, class Apple has

- $Precision = 7/(7+17) = 0.29$
- $Recall = 7/(7+4) = 0.64$
- $F1-score = 0.40$

Similarly, we can calculate the measures for the other classes. Here is a table that shows the values of each measure for each class.

Class	Precision	Recall	F1-score
-------	-----------	--------	----------

Apple	0.29	0.64	0.40
Orange	0.33	0.17	0.22
Mango	0.17	0.08	0.11

Precision, Recall and F1-score for Each Class

Now we can do more with these measures. We can combine the F1-score of each class to have a single measure for the whole model. There are a few ways to do that, let's look at them now.

. . .

## Micro F1

This is called micro-averaged F1-score. It is calculated by considering the total TP, total FP and total FN of the model. It does not consider each class individually, It calculates the metrics globally. So for our example,

- *Total TP = (7+2+1) = 10*
- *Total FP = (8+9)+(1+3)+(3+2) = 26*
- *Total FN = (1+3)+(8+2)+(9+3) = 26*

Hence,

- *Precision =  $10/(10+26) = 0.28$*
- *Recall =  $10/(10+26) = 0.28$*

Now we can use the regular formula for F1-score and get the Micro F1-score using the above precision and recall.

---

***Micro F1 = 0.28***

---

As you can see When we are calculating the metrics globally all the measures become equal. Also if you calculate accuracy you will see that,

# Precision = Recall = Micro F1 = Accuracy

## Macro F1

This is macro-averaged F1-score. It calculates metrics for each class individually and then takes unweighted mean of the measures. As we have seen from figure “*Precision, Recall and F1-score for Each Class*”,

- *Class Apple F1-score = 0.40*
- *Class Orange F1-score = 0.22*
- *Class Mango F1-score = 0.11*

Hence,

$$\text{Macro F1} = (0.40 + 0.22 + 0.11) / 3 = 0.24$$

## Weighted F1

The last one is weighted-averaged F1-score. Unlike Macro F1, it takes a weighted mean of the measures. The weights for each class are the total number of samples of that class. Since we had 11 Apples, 12 Oranges and 13 Mangoes,

$$\text{Weighted F1} = ((0.40 * 11) + (0.22 * 12) + (0.11 * 13)) / (11 + 12 + 13) = 0.24$$

• • •

Finally, let's look at a script to calculate these measures using Python's Scikit-learn.

```
1  #importing a 3-class dataset from sklearn's toy dataset
2  from sklearn.datasets import load_wine
3
4  dataset = load_wine()
5  X = dataset.data
6  y = dataset.target
7  from sklearn.model_selection import train_test_split
8  from sklearn.svm import SVC
9
10 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
11 svc = SVC(kernel='rbf', C=1).fit(X_train, y_train)
```



```

12 y_pred = svc.predict(X_test)
13
14 #importing confusion matrix
15 from sklearn.metrics import confusion_matrix
16 confusion = confusion_matrix(y_test, y_pred)
17 print('Confusion Matrix\n')
18 print(confusion)
19
20 #importing accuracy_score, precision_score, recall_score, f1_score
21 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
22 print('\nAccuracy: {:.2f}\n'.format(accuracy_score(y_test, y_pred)))
23
24 print('Micro Precision: {:.2f}'.format(precision_score(y_test, y_pred, average='micro')))
25 print('Micro Recall: {:.2f}'.format(recall_score(y_test, y_pred, average='micro')))
26 print('Micro F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred, average='micro')))
27
28 print('Macro Precision: {:.2f}'.format(precision_score(y_test, y_pred, average='macro')))
29 print('Macro Recall: {:.2f}'.format(recall_score(y_test, y_pred, average='macro')))
30 print('Macro F1-score: {:.2f}\n'.format(f1_score(y_test, y_pred, average='macro')))
31
32 print('Weighted Precision: {:.2f}'.format(precision_score(y_test, y_pred, average='weighted')))
33 print('Weighted Recall: {:.2f}'.format(recall_score(y_test, y_pred, average='weighted')))
34 print('Weighted F1-score: {:.2f}'.format(f1_score(y_test, y_pred, average='weighted')))
35
36 from sklearn.metrics import classification_report
37 print('\nClassification Report\n')
38 print(classification_report(y_test, y_pred, target_names=['Class 1', 'Class 2', 'Class 3']))

```

multi class confusion matrix with  by GitHub

[view raw](#)

Here is the output of the script.

CONFUSION MATRIX:

```
[[15  0  1]
 [ 0 17  4]
 [ 0  3  5]]
```

Accuracy: 0.82

Micro Precision: 0.82

Micro Recall: 0.82

Micro F1-score: 0.82

Macro Precision: 0.78

Macro Recall: 0.79

Macro F1-score: 0.78



```
Weighted Precision: 0.84
Weighted Recall: 0.82
Weighted F1-score: 0.83
```

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
Class 1	1.00	0.94	0.97	16
Class 2	0.85	0.81	0.83	21
Class 3	0.50	0.62	0.56	8
accuracy			0.82	45
macro avg	0.78	0.79	0.78	45
weighted avg	0.84	0.82	0.83	45

Credit: [Google Colab](#)

• • •

Hope you found what you were looking for. Thanks for reading.

## Sign up for The Daily Pick

By Towards Data Science

Hands-on real-world examples, research, tutorials, and cutting-edge techniques delivered Monday to Thursday. Make learning your daily ritual. [Take a look](#)

Your email

 Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

Thanks to Yenson Lau.

Data Science

Machine Learning

Confusion Matrix

Multiclass Classification

Python



[About](#) [Help](#) [Legal](#)

Get the Medium app

