

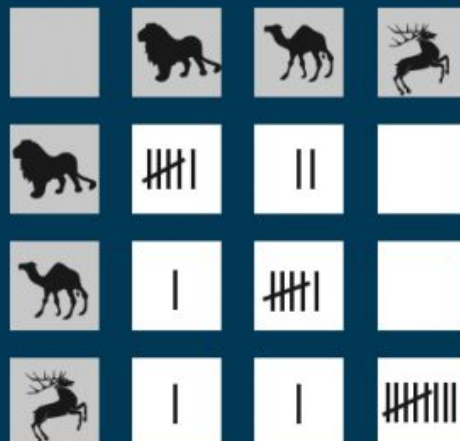
# Everything you Should Know about Confusion Matrix for Machine Learning

[BEGINNER](#)[CLASSIFICATION](#)[MACHINE LEARNING](#)[PYTHON](#)[TECHNIQUE](#)

## Confusion Matrix – Not So Confusing!

Have you been in a situation where you expected your machine learning model to perform really well but it sputtered out a poor accuracy? You've done all the hard work – so where did the classification model go wrong? How can you correct this?

There are plenty of ways to gauge the performance of your classification model but none have stood the test of time like the confusion matrix. It helps us evaluate how our model performed, where it went wrong and offers us guidance to correct our path.



In this article, we will explore how a Confusion matrix gives a holistic view of the performance of your model. And unlike its name, you will realize that a Confusion matrix is a pretty simple yet powerful concept. So let's unravel the mystery around the confusion matrix!



*Learning the ropes in the machine learning field? These courses will get you on your way:*

- [Python for Data Science \(free course!\)](#)
- [Machine Learning Starter Program](#)

## Here's what we'll cover:

- What is a Confusion Matrix?
  - True Positive
  - True Negative
  - False Positive – Type 1 Error
  - False Negative – Type 2 Error
- Why need a Confusion matrix?
- Precision vs Recall
- F1-score
- Confusion matrix in Scikit-learn
- Confusion matrix for multi-class Classification

## What is a Confusion Matrix?

The million dollar question – what, after all, is a confusion matrix?

A Confusion matrix is an  $N \times N$  matrix used for evaluating the performance of a classification model, where  $N$  is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a  $2 \times 2$  matrix as shown below with 4 values:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Let's decipher the matrix:

- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

But wait – what's TP, FP, FN and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

## Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

### True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

### True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

### False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the **Type 1 error**

### False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the **Type 2 error**

Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier on it and get the below confusion matrix:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	560	60
	NEGATIVE	50	330

The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 560; meaning 560 positive class data points were correctly classified by the model
- True Negative (TN) = 330; meaning 330 negative class data points were correctly classified by the model
- False Positive (FP) = 60; meaning 60 negative class data points were incorrectly classified as belonging to the positive class by the model
- False Negative (FN) = 50; meaning 50 positive class data points were incorrectly classified as belonging to the negative class by the model

This turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

*Remember the Type 1 and Type 2 errors. Interviewers love to ask the difference between these two!*

## Why Do We Need a Confusion Matrix?

Before we answer this question, let's think about a hypothetical classification problem.

Let's say you want to predict how many people are infected with a contagious virus in times before they show the symptoms, and isolate them from the healthy population (ringing any bells, yet? 😊). The two values for our target variable would be: Sick and Not Sick.

Now, you must be wondering – why do we need a confusion matrix when we have our all-weather friend – Accuracy? Well, let's see where accuracy falters.

Our dataset is an example of an [imbalanced dataset](#). There are 947 data points for the negative class and 3 data points for the positive class. This is how we'll calculate the accuracy:

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

Let's see how our model performed:

ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FP
8	0	1	FN
9	0	0	TN
10	1	0	FP
:	:	:	:
1000	0	0	FN

The total outcome values are:

TP = 30, TN = 930, FP = 30, FN = 10

So, the accuracy for our model turns out to be:

$$Accuracy = \frac{30 + 930}{30 + 30 + 930 + 10} = 0.96$$

96%! Not bad!

But it is giving the wrong idea about the result. Think about it.

Our model is saying “I can predict sick people 96% of the time”. However, it is doing the opposite. It is predicting the people who will not get sick with 96% accuracy while the sick are spreading the virus!

Do you think this is a correct metric for our model given the seriousness of the issue? Shouldn’t we be measuring how many positive cases we can predict correctly to arrest the spread of the contagious virus? Or maybe, out of the correctly predicted cases, how many are positive cases to check the reliability of our model?

This is where we come across the dual concept of Precision and Recall.

## Precision vs. Recall

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

Here’s how to calculate Precision:

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

This would determine whether our model is reliable or not.

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

And here's how we can calculate Recall:

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

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP (30)	FP (30)
	NEGATIVE	FN (10)	TN (930)

Sick people correctly predicted as sick by the model

Healthy people incorrectly predicted as sick by the model

Sick people incorrectly predicted as not sick by the model

Healthy people correctly predicted as not sick by the model

We can easily calculate Precision and Recall for our model by plugging in the values into the above questions:

$$Precision = \frac{30}{30 + 30} = 0.5$$

$$Recall = \frac{30}{30 + 10} = 0.75$$

50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by our model. Awesome!

Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.

Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.

Recall is a useful metric in cases where False Negative trumps False Positive.

Recall is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!

In our example, Recall would be a better metric because we don't want to accidentally discharge an infected person and let them mix with the healthy population thereby spreading the contagious virus. Now you can understand why accuracy was a bad metric for our model.

But there will be cases where there is no clear distinction between whether Precision is more important or Recall. What should we do in those cases? We combine them!

# F1-Score

In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value:

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

**F1-score is a harmonic mean of Precision and Recall**, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

But there is a catch here. The interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall? So, we use it in combination with other evaluation metrics which gives us a complete picture of the result.

## Confusion Matrix using scikit-learn in Python

You know the theory – now let's put it into practice. Let's code a confusion matrix with the [Scikit-learn \(sklearn\) library](#) in Python.

```
Confusion matrix :  
[[2 2]  
 [1 5]]  
Outcome values :  
2 2 1 5  
Classification report :  
              precision    recall  f1-score   support  
  
      1       0.67       0.50       0.57         4  
      0       0.71       0.83       0.77         6  
  
   micro avg       0.70       0.70       0.70        10  
   macro avg       0.69       0.67       0.67        10  
weighted avg       0.70       0.70       0.69        10
```

Sklearn has two great functions: **confusion\_matrix()** and **classification\_report()**.

- Sklearn [confusion\\_matrix\(\)](#) returns the values of the Confusion matrix. The output is, however, slightly different from what we have studied so far. It takes the rows as Actual values and the columns as Predicted values. The rest of the concept remains the same.
- Sklearn [classification\\_report\(\)](#) outputs precision, recall and f1-score for each target class. In addition to this, it also has some extra values: **micro avg**, **macro avg**, and **weighted avg**

**Mirco average** is the precision/recall/f1-score calculated for all the classes.

$$\text{Micro avg Precision} = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2}$$

**Macro average** is the average of precision/recall/f1-score.

$$\text{Macro avg Precision} = \frac{P1 + P2}{2}$$

**Weighted average** is just the weighted average of precision/recall/f1-score.

## Confusion Matrix for Multi-Class Classification

How would a confusion matrix work for a multi-class classification problem? Well, don't scratch your head! We will have a look at that here.

Let's draw a confusion matrix for a multiclass problem where we have to predict whether a person loves Facebook, Instagram or Snapchat. The confusion matrix would be a 3 x 3 matrix like this:

	Facebook	Instagram	Snapchat
Facebook	Cell <sub>11</sub>	Cell <sub>12</sub>	Cell <sub>13</sub>
Instagram	Cell <sub>21</sub>	Cell <sub>22</sub>	Cell <sub>23</sub>
Snapchat	Cell <sub>31</sub>	Cell <sub>32</sub>	Cell <sub>33</sub>

The true positive, true negative, false positive and false negative for each class would be calculated by adding the cell values as follows:

Facebook	Instagram	Snapchat
TP = Cell <sub>11</sub>	TP = Cell <sub>22</sub>	TP = Cell <sub>33</sub>
FP = Cell <sub>12</sub> + Cell <sub>13</sub>	FP = Cell <sub>21</sub> + Cell <sub>23</sub>	FP = Cell <sub>31</sub> + Cell <sub>32</sub>
TN = Cell <sub>23</sub> + Cell <sub>32</sub> + Cell <sub>33</sub> + Cell <sub>11</sub>	TN = Cell <sub>11</sub> + Cell <sub>13</sub> + Cell <sub>21</sub> + Cell <sub>22</sub>	TN = Cell <sub>11</sub> + Cell <sub>12</sub> + Cell <sub>21</sub> + Cell <sub>23</sub>
FN = Cell <sub>21</sub> + Cell <sub>31</sub>	FN = Cell <sub>12</sub> + Cell <sub>32</sub>	FN = Cell <sub>13</sub> + Cell <sub>23</sub>

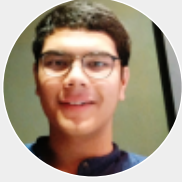
That's it! You are ready to decipher any N x N confusion matrix!

## End Notes

And suddenly the Confusion matrix is not so confusing any more! This article should give you a solid base on how to intepret and use a confusion matrix for classification algorithms in machine learning.

We will soon come out with an article on the AUC-ROC curve and continue our discussion there. Until next time, don't lose hope in your classification model, you just might be using the wrong evaluation metric!





## **Aniruddha Bhandari**

I am on a journey to becoming a data scientist. I love to unravel trends in data, visualize it and predict the future with ML algorithms! But the most satisfying part of this journey is sharing my learnings, from the challenges that I face, with the community to make the world a better place!