# Understanding the Confusion Matrix (II)

Banso D. Wisdom May 5 · 8 min read

#python #machinelearning #deeplearning #ai

In the first part of this article, I talked about the confusion matrix in general, the 2-class confusion matrix, how to calculate accuracy, precision and other metrics using it and also how to generate a confusion matrix in python.

In this article, we'll be looking at the multi-class confusion matrix.

## What is the multi-class confusion matrix?

As the name implies, it is a confusion matrix that deals with multiple classes (i.e. more than 2 classes). Just like the 2-class confusion matrix, it describes the performance of a multi-class classification model.

For the purpose of this article, we'll be assuming that our multi-class classification model is one that classifies images of dogs into the following breeds: **Greyhound**, **Mastiff** and **Samoyed**.

A confusion matrix for this classifier can be visualized as such:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

In this visualization, we have two sections which have been outlined. We have the **predicted** classifications section which contains three subsections for each of the classes we want to classify into and the **actual** classifications section which has three subsections for each of the classes.

Having visualized this confusion matrix, we can use this visualization to calculate the following metrics:

- True Positives.
- True Negatives.
- False Positives.
- False Negatives.
- Accuracy.
- Precision.
- True Positive Rate is also known as **Sensitivity** or **Recall**.
- True Negative Rate is also known as **Specificity**.

Before we calculate these metrics, let's define the variables in the visualization:

**PGG**

This variable represents the number of predictions where images of a **Greyhound** were correctly classified [as a **Greyhound**]. This is also the **True Positive** for the **Greyhound** class.

**PMG**

This variable represents the number of predictions where images of a **Greyhound** were incorrectly classified as a **Mastiff**.

**PSG**

This variable represents the number of predictions where images of a **Greyhound** were incorrectly classified as a **Samoyed**.

**PGM**

This variable represents the number of predictions where images of a **Mastiff** were incorrectly classified as a **Greyhound**.

**PMM**

This variable represents the number of predictions where images of a **Mastiff** were correctly classified [as a **Mastiff**]. This is also the **True Positive** for the **Mastiff** class.

**PSM**

This variable represents the number of predictions where images of a **Mastiff** were incorrectly classified as a **Samoyed**.

**PGS**

This variable represents the number of predictions where images of a **Samoyed** were incorrectly classified as a **Greyhound**.

**PMS**

This variable represents the number of predictions where images of a **Samoyed** were incorrectly classified as a **Mastiff**.

**PSS**

This variable represents the number of predictions where images of a **Samoyed** were correctly classified [as a **Samoyed**]. This is also the **True Positive** for the **Samoyed** class.

Now that we have defined these variables, we can now calculate the aforementioned metrics.

## True Positives

The definition for the **True Positive** is the same as in the 2-class confusion matrix. However, here we calculate the **True Positives** for each class in the confusion matrix unlike the general or absolute **True Positives** in the 2-class confusion matrix.

The **True Positives** is the number of predictions where data labelled to belong to a particular class was correctly classified as the said class. E.g Number of predictions where images of a Samoyed was correctly classified as a Samoyed.

From the definition of the matrix variables, we have already identified the **True Positives** for each of the classes:

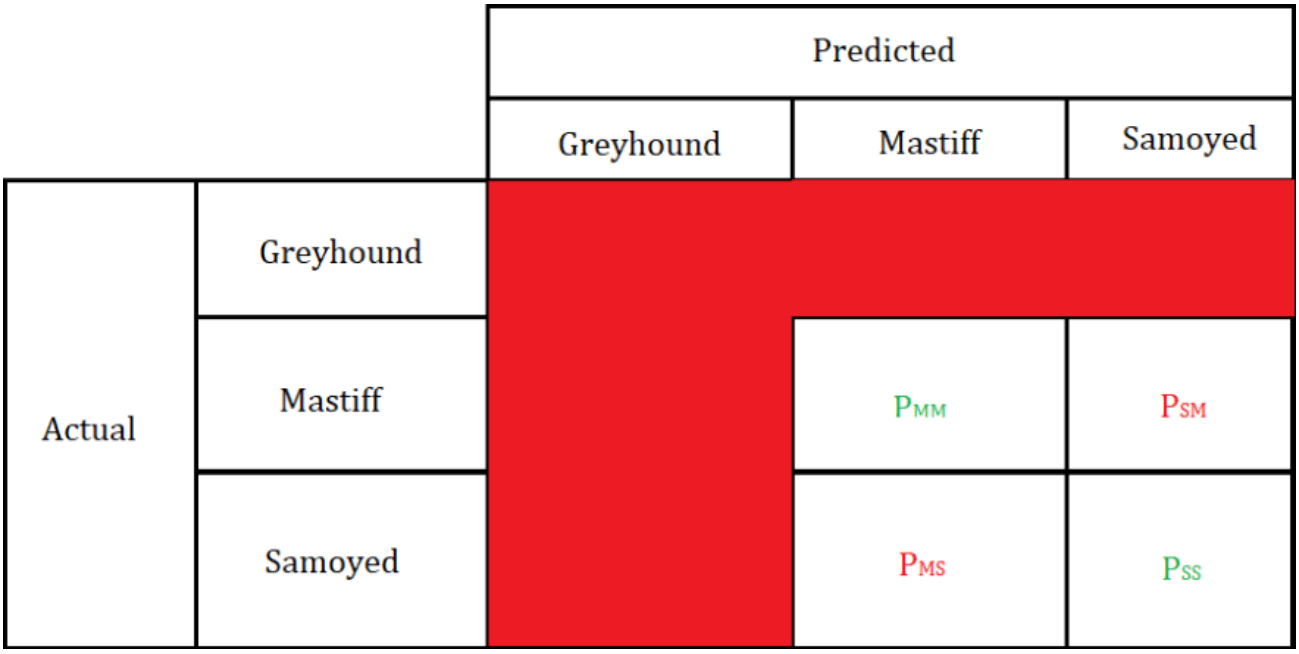
> **True Positives** for the **Greyhound** class is the variable **PGG** in the confusion matrix.
>
> **True Positives** for the **Mastiff** class is the variable **PMM** in the confusion matrix.
>
> **True Positives** for the **Samoyed** class is the variable **PSS** in the confusion matrix.

## True Negatives

The definition of the **True Negative** is the same as in the 2-class confusion matrix. Here we calculate the **True Negatives** for each class in the confusion matrix unlike the general or absolute **True Negatives** in the 2-class confusion matrix.

The **True Negatives** for a particular class is calculated by taking the sum of the values in every row and column except the row and column of the class we're trying to find the **True Negatives** for.

For example, calculating the **True Negatives** for the **Greyhound** class:



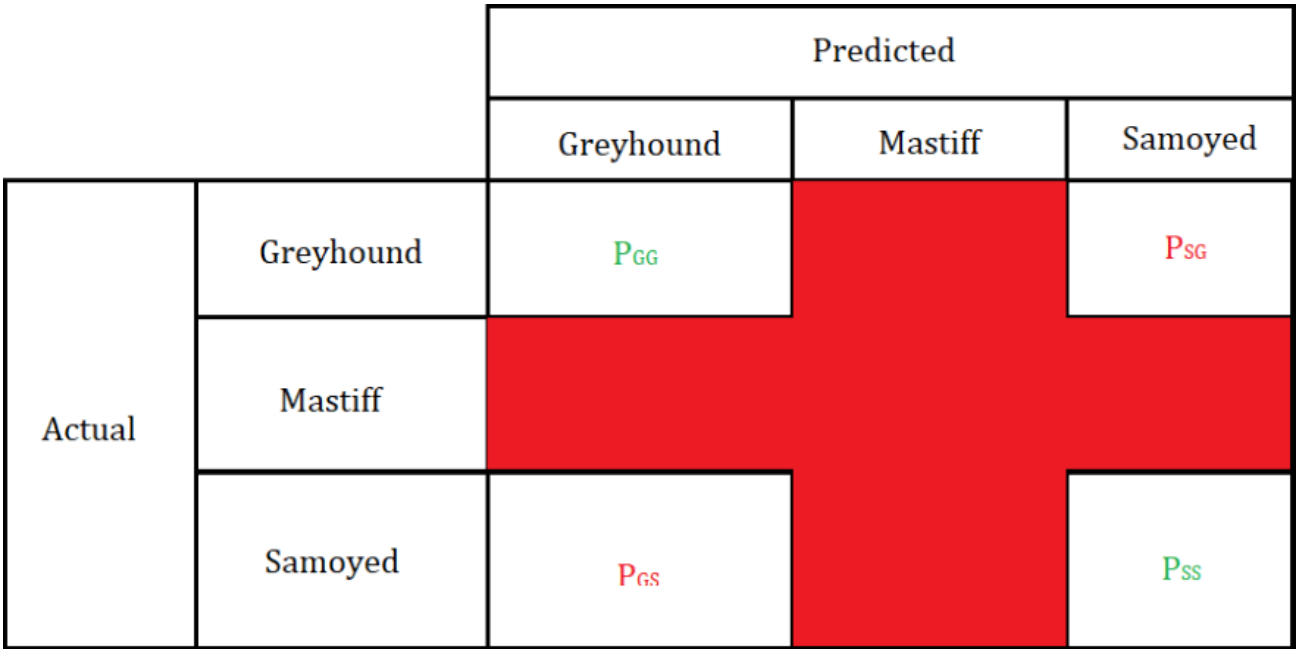

We omit the row and columns belonging to the **Greyhound** class and sum the variables that are left, which are the rows and columns of the other classes (**Mastiff** and **Samoyed**).

Therefore the **True Negatives** for the **Greyhound** class is:

**TN = PMM + PSM + PMS + PSS**

Similarly, we can calculate the **True Negatives** for the other classes.

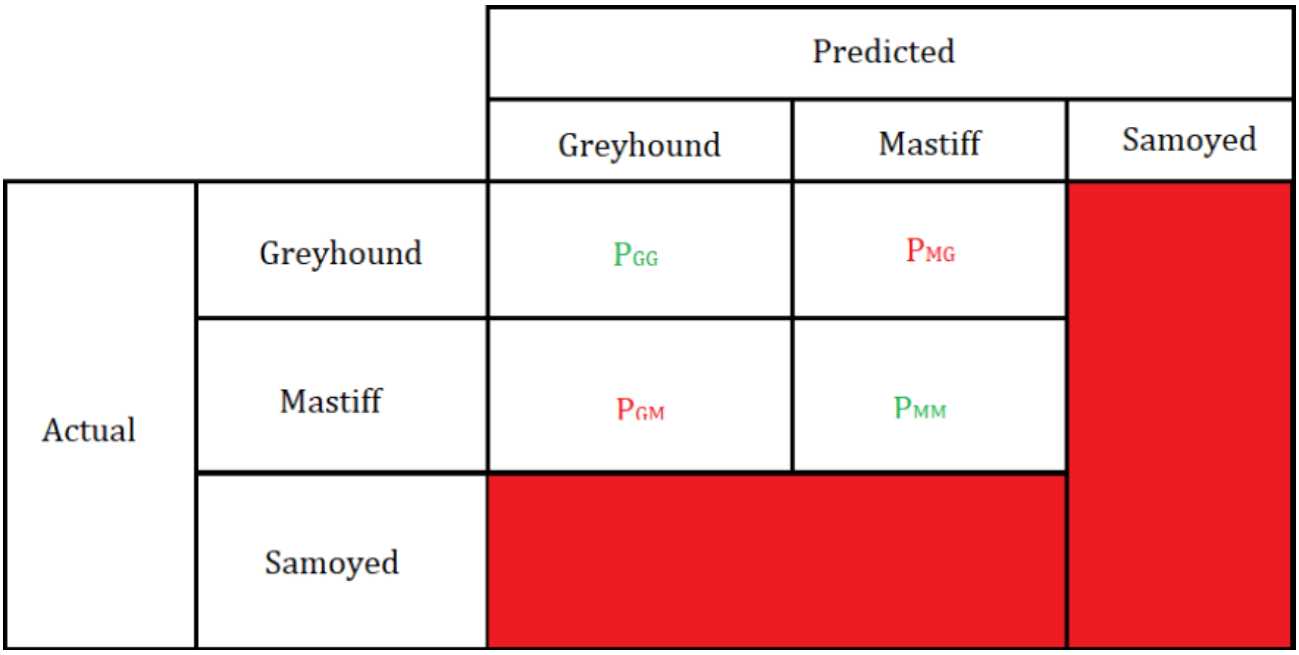
Calculating the **True Negatives** for the **Mastiff** class:




The **True Negatives** for the **Mastiff** class is:

**TN = PGG + PSG + PGS + PSS**

Calculating the **True Negatives** for the **Samoyed** class:




The **True Negatives** for the **Samoyed** class is:

**TN = PGG + PMG + PGM + PMM**

## False Positives

The definition of the **False Positive** is the same as in the 2-class confusion matrix. Here we calculate the **False Positives** for each class in the confusion matrix unlike the general or absolute **False Positives** in the 2-class confusion matrix.

The **False Positives** for a particular class can be calculated by taking the sum of all the values in the column corresponding to that class except the **True Positives** value.

For example, calculating the **False Positives** for the **Greyhound** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

We sum all the values in the highlighted area, which is the column corresponding to the **Greyhound** class with the

exception of the variable **PGG** which we had earlier identified to be the **True Positives** for the **Greyhound** class.

Therefore the **False Positives** for **Greyhound** class is:

**FP = PGM + PGS**

Similarly, we can calculate the **False Positives** for the other classes.

Calculating the **False Positives** for the **Mastiff** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

The **False Positives** for the **Mastiff** class is:

**FP = PMG + PMS**

Calculating the **False Positives** for the **Samoyed** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

The **False Positives** for the **Samoyed** class is:

**FP = PSG + PSM**

## False Negatives

The definition of the **False Negative** is the same as in the 2-class confusion matrix. Here we calculate the **False Negatives** for each class in the confusion matrix unlike the general or absolute **False Positives** in the 2-class confusion matrix.

The **False Negatives** for a particular class can be calculated by taking the sum of all the values in the row corresponding to that class except the **True Positives** value.

For example, calculating the **False Negatives** for the **Greyhound** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

We sum all the values in the highlighted area, which is the row corresponding to the **Greyhound** class with the exception of the variable **PGG** which we had earlier identified to be the **True Positives** for the **Greyhound** class.

Therefore the **False Negatives** for **Greyhound** class is:

**FN = PMG + PSG**

Similarly, we can calculate the **False Negatives** for the other classes.

Calculating the **False Negatives** for the **Mastiff** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

The **False Negatives** for the **Mastiff** class is:

**FN = PGM + PSM**

Calculating the **False Negatives** for the **Samoyed** class:

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | $P_{GG}$ | $P_{MG}$ | $P_{SG}$ |
| | Mastiff | $P_{GM}$ | $P_{MM}$ | $P_{SM}$ |
| | Samoyed | $P_{GS}$ | $P_{MS}$ | $P_{SS}$ |

The **False Negatives** for the **Samoyed** class is:

**FN = PGS + PMS**

## Accuracy

**Accuracy** is calculated as the ratio of the number of correct classifications to the total number of classifications. From our confusion matrix, the correct classifications are the **True Positives** for each class and the total number of classifications is the sum of every value in the confusion matrix, including the **True Positives**.

Therefore, the accuracy is:

**AC = (PGG + PMM + PSS) / (PGG + PMG + PSG + PGM + PMM + PSM + PGS + PMS + PSS)**

## Precision

**Precision** is a multi-class confusion matrix is the measure of the accuracy relative to the prediction of a specific class. It is calculated as the ratio of the **True Positives** of the class in question to the sum of its **True Positives** and **False Positives**.

For example, calculating the Precision of the **Greyhound** class:

**Precision (G) = TP / (TP + FP)**
**= PGG / (PGG + (PGM + PGS))**

Similarly, we can calculate the Precision of other classes.

Calculating the Precision of the **Mastiff** class:

**Precision (M) = TP / (TP + FP)**
**= PMM / (PMM + (PMG + PMS))**

Calculating the Precision of the **Samoyed** class:

**Precision (S) = TP / (TP + FP)**
**= PSS / (PSS + (PSG + PSM))**

## True Positive Rate

The **True Positive Rate** (also known as **Recall** or **Sensitivity**) is calculated as the ratio of the **True Positives** of a specific class to the sum of its **True Positives** and **False Negatives**.

For example, calculating the **True Positive Rate** of the **Greyhound** class:

**TPR (G) = TP / (TP + FN)**
**= PGG / (PGG + (PMG + PSG))**

Similarly, we can calculate the True Positive Rate of other classes.

Calculating the True Positive Rate of the **Mastiff** class:

**TPR (M) = TP / (TP + FN)**
**= PMM / (PMM + (PGM + PSM))**

Calculating the True Positive Rate of the **Samoyed** class:

**TPR (S) = TP / (TP + FN)**
**= PSS / (PSS + (PGS + PMS))**

## True Negative Rate

The **True Negative Rate** (also known as **Specificity**) is calculated as the ratio of the **True Negatives** of a specific class to the sum of its **True Negatives** and **False Positives**.

For example, calculating the **True Negative Rate** of the **Greyhound** class:

**TNR (G) = TN / (TN + FP)**
**= (PMM + PSM + PMS + PSS) / ((PMM + PSM + PMS + PSS) + (PGM + PGS))**

Similarly, we can calculate the True Negative Rate of other classes.

Calculating the True Negative Rate of the **Mastiff** class:

**TNR (M) = TN / (TN + FP)**
**= (PGG + PSG + PGS + PSS) / ((PGG + PSG + PGS + PSS) + (PMG + PMS))**

Calculating the True Negative Rate of the **Samoyed** class:

**TNR (S) = TN / (TN + FP)**
**= (PGG + PMG + PGM + PMM) / ((PGG + PMG + PGM + PMM) + (PSG + PSM))**

Phewww, that was a lot. 😅😅

## Now, a quick example

Suppose we have the image below as the visualized confusion matrix for our classifier, we can use the information in the visualization and the metrics defined above to evaluate its performance.

| | | Predicted | | |
|---|---|---|---|---|
| | | Greyhound | Mastiff | Samoyed |
| Actual | Greyhound | 250 | 25 | 18 |
| | Mastiff | 21 | 320 | 24 |
| | Samoyed | 22 | 12 | 180 |

From this confusion matrix, we can identify that:

> - The variable **PGG** and the **True Positives** of the **Greyhound** class is **250**.
> - The variable **PMM** and the **True Positives** of the **Mastiff** class is **320**.

> - The variable **PSS** and the **True Positives** of the **Samoyed** class is **180**.
> - The variable **PMG** is **25**.
> - The variable **PSG** is **18**.
> - The variable **PGM** is **21**.
> - The variable **PSM** is **24**.
> - The variable **PGS** is **22**.
> - The variable **PMS** is **12**.

Using this information that we've "extracted", we can calculate the metrics mentioned earlier and thus evaluate the performance of the classifier.

#### True Negatives

> - **TN** (**Greyhound**) = **PMM + PSM + PMS + PSS** = 320 + 24 + 24 + 180 = **548**
> - **TN** (**Mastiff**) = **PGG + PSG + PGS + PSS** = 250 + 18 + 22 + 180 = **470**
> - **TN** (**Samoyed**) = **PGG + PMG + PGM + PMM** = 250 + 25 + 21 + 320 = **616**

#### False Positives

> - **FP** (**Greyhound**) = **PGM + PGS** = 21 + 22 = **43**
> - **FP** (**Mastiff**) = **PMG + PMS** = 25 + 12 = **37**
> - **FP** (**Samoyed**) = **PSG + PSM** = 18 + 24 = **42**

### False Negatives

> - **FN** (**Greyhound**) = **PMG + PSG** = 25 + 18 = **43**
> - **FN** (**Mastiff**) = **PGM + PSM** = 21 + 24 = **45**
> - **FN** (**Samoyed**) = **PGS + PMS** = 22 + 12 = **34**

### True Positive Rate / Recall / Sensitivity

> - **TPR** (**Greyhound**) = **TP** / (**TP + FN**) = 250 / (250 + 43) = 250 / 293 = 0.8532423208 = **0.85**
> - **TPR** (**Mastiff**) = **TP** / (**TP + FN**) = 320 / (320 + 45) = 320 / 365 = 0.8767123288 = **0.88**
> - **TPR** (**Samoyed**) = **TP** / (**TP + FN**) = 180 / (180 + 34) = 180 / 214 = 0.8411214953 = **0.81**

### True Negative Rate / Specificity

> - **TNR (Greyhound)** = **TN** / (**TN + FP**) = 548 / (548 + 43) = 548 / 591 = 0.9272419628 = **0.93**
> - **TNR (Mastiff)** = **TN** / (**TN + FP**) = 470 / (470 + 37) = 470 / 507 = 0.9270216963 = **0.93**
> - **TNR (Samoyed)** = **TN** / (**TN + FP**) = 616 / (616 + 42) = 616 / 658 = 0.9361702128 = **0.94**

### Precision

> - **Precision** (**Greyhound**) = **TP** / (**TP + FP**) = 250 / (250 + 43) = 250 / 293 = 0.8532423208 = **0.85**

Therefore, the classifier has a precision of **0.85**, which is **85%**, in classifying images of Greyhounds.

> - **Precision** (**Mastiff**) = **TP** / (**TP + FP**) = 320 + (320 + 37)
> = 320 / 357 = 0.8963585434 = **0.90**

Therefore, the classifier has a precision of **0.90**, which is **90%**, in classifying images of Mastiffs.

> - **Precision** (**Samoyed**) = **TP** / (**TP + FP**) = 180 / (180 + 42)
> = 180 / 222 = 0.8108108108 = **0.81**

Therefore, the classifier has a precision of **0.81**, which is **81%**, in classifying images of Samoyeds.

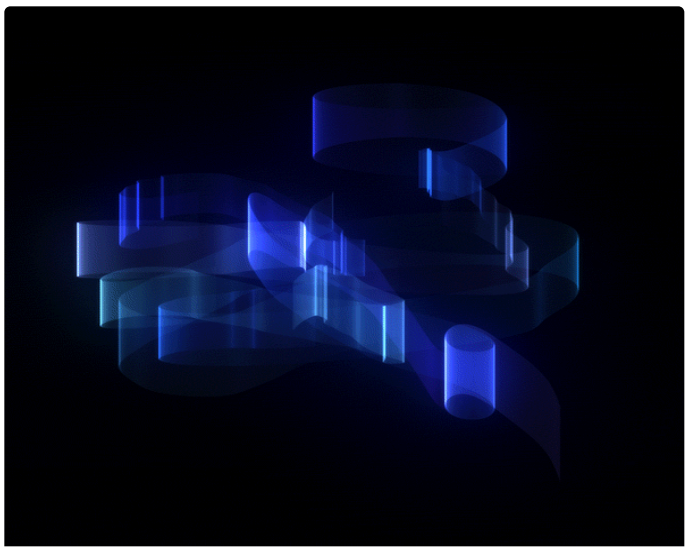
### Accuracy

**AC** = (**PGG** + **PMM** + **PSS**) / (**PGG + PMG + PSG + PGM + PMM + PSM + PGS + PMS + PSS**)
= (250 + 320 + 180) / (250 + 25 + 18 + 21 + 320 + 24 + 22 + 12 + 180)
= 750 / 872
= 0.8600917431
= **0.86**

Therefore, the classifier has a total accuracy of **0.86** which is **86%**

# How to generate a multi-class confusion matrix in Python

A multi-class confusion matrix is created in the same way as a 2-class confusion matrix. See the first part of this article.





Banso D. Wisdom + FOLLOW
software ng-neer
@overridedeveloper overridedeveloper Overridedeveloper

Classic DEV Post from Feb 2

## Progress Not Perfection


Ilona Codes

Most people want to make things perfect. Sometimes we evaluate the complexity of an upcoming goal or a problem. So, the fear to not complete it perfectly or "wrong" (Yeah, who are judges? 🤔) stops us even from trying.

133 7

Another Post You Might Like

## On Becoming a Technical Writer


Ada Nduka Oyom

One of my many role models once said, the reason as to which she started out ...

113 10

Another Post You Might Like

## Play With the React Router


Sai gowtham

how routing works in react router

106 4

## Making own wordbook by Natural language processing

8pockets - Nov 22

## Segment Tree

Artur Curinga - Nov 22

## Python to React

SMAmmar - Nov 23

## Neural Network, the universal solver?

EdRome - Nov 20