Artificial Intelligence

Supervised Learning

Challenges in Training Neural Network

Vanishing Gradient

Network is unable to propagate useful gradient information from the output end of the model back to the layers near the input end of the model which eventually leaves the weights of the initial or lower layers nearly unchanged This means that the updates to the weights in early layers are so small that these layers fail to learn meaningful features from the data.

Exploding Gradient

In contrast, During Backpropagation if the gradients keep on getting larger and larger as the backpropagation algorithm progresses. This, in turn, causes very large weight updates and causes the gradient descent to diverge.

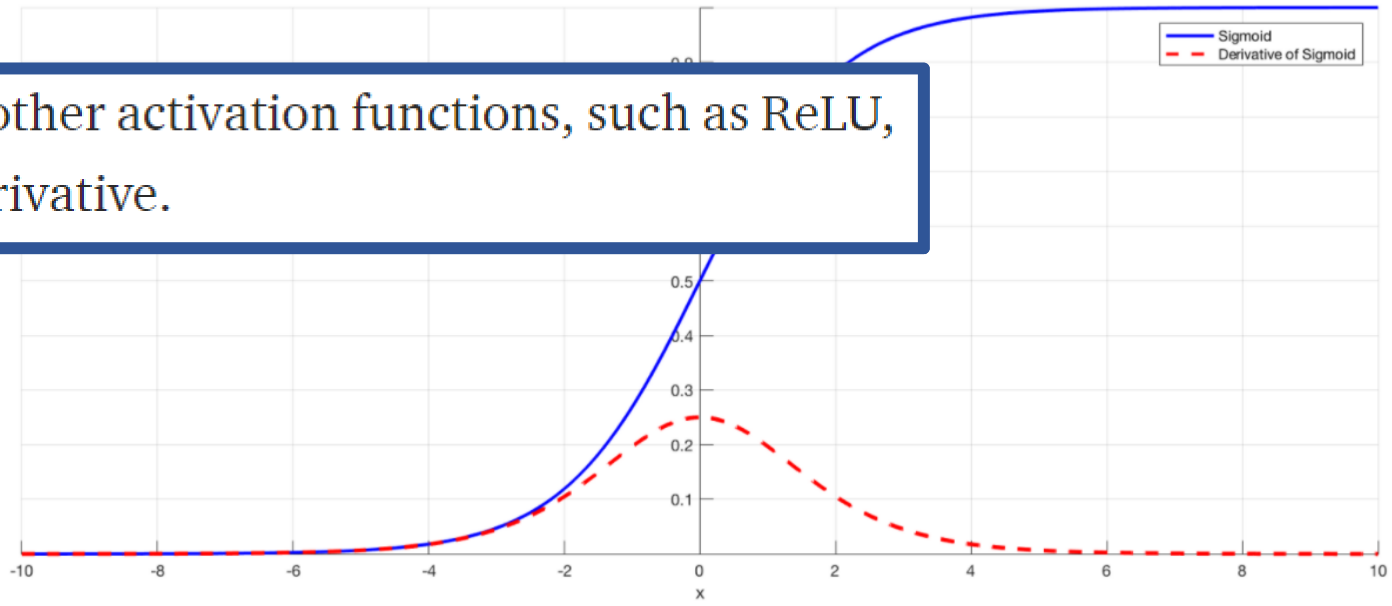
- Solution:
  - Alternate Weight Initialization Methods
  - Alternate Activation Function
  - Gradient clipping

https://www.analyticsvidhya.com/blog/2021/05/how-to-initialize-weights-in-neural-networks/#:~:text=Step%2D1%3A%20Initialization%20of%20Neural,to%20linear%20combination%20(A).

Vanishing Gradient

- The gradients of the loss function approaches zero, making the network hard to train.

The simplest solution is to use other activation functions, such as ReLU, which doesn’t cause a small derivative.



How to Identify Vanishing or Exploding Gradient?

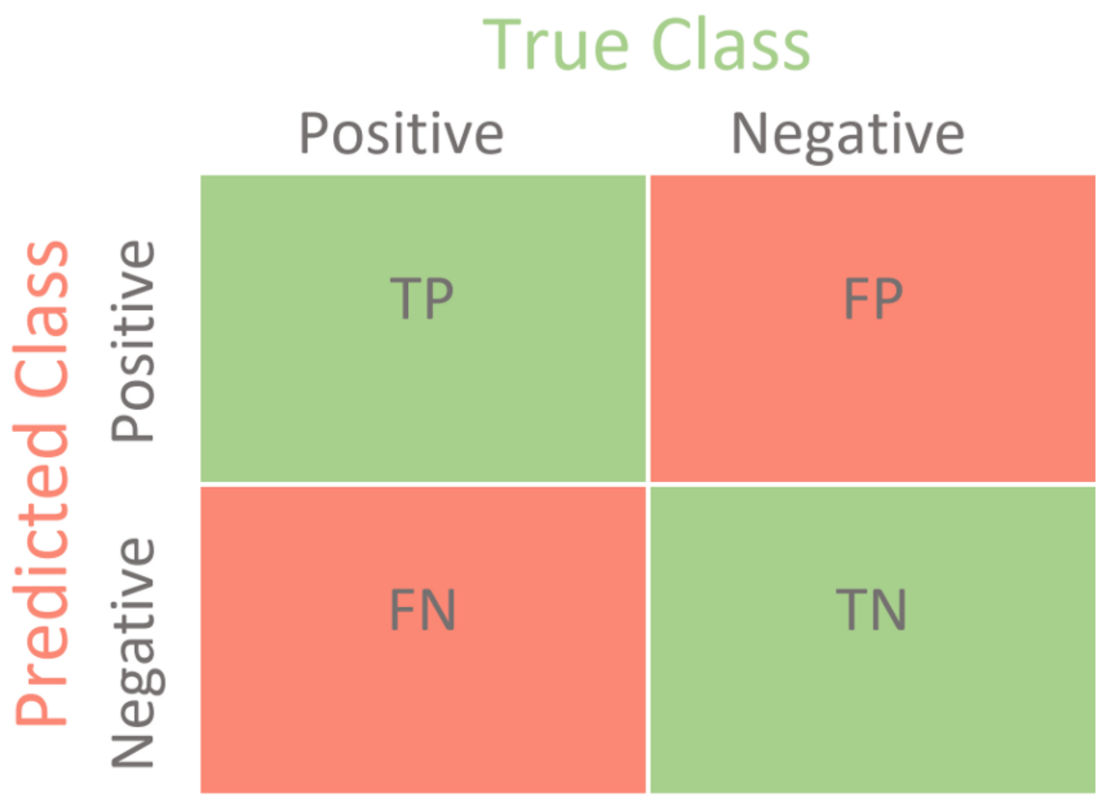
- Vanishing
  - The parameters near output layers changes significantly whereas change will become to zero near input layers.
  - Updated weights may become 0 during training.
  - Network learns very slowly, or training stops just after a few iterations.

- Exploding
  - There is an exponential growth in the model parameters.
  - Updated weights may become NaN during training. If the loss becomes too large to be represented as a floating-point number.
  - Their will be a rapid or sudden increase in learning.

Evaluation: Confusion Matrix

- It is an N x 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.

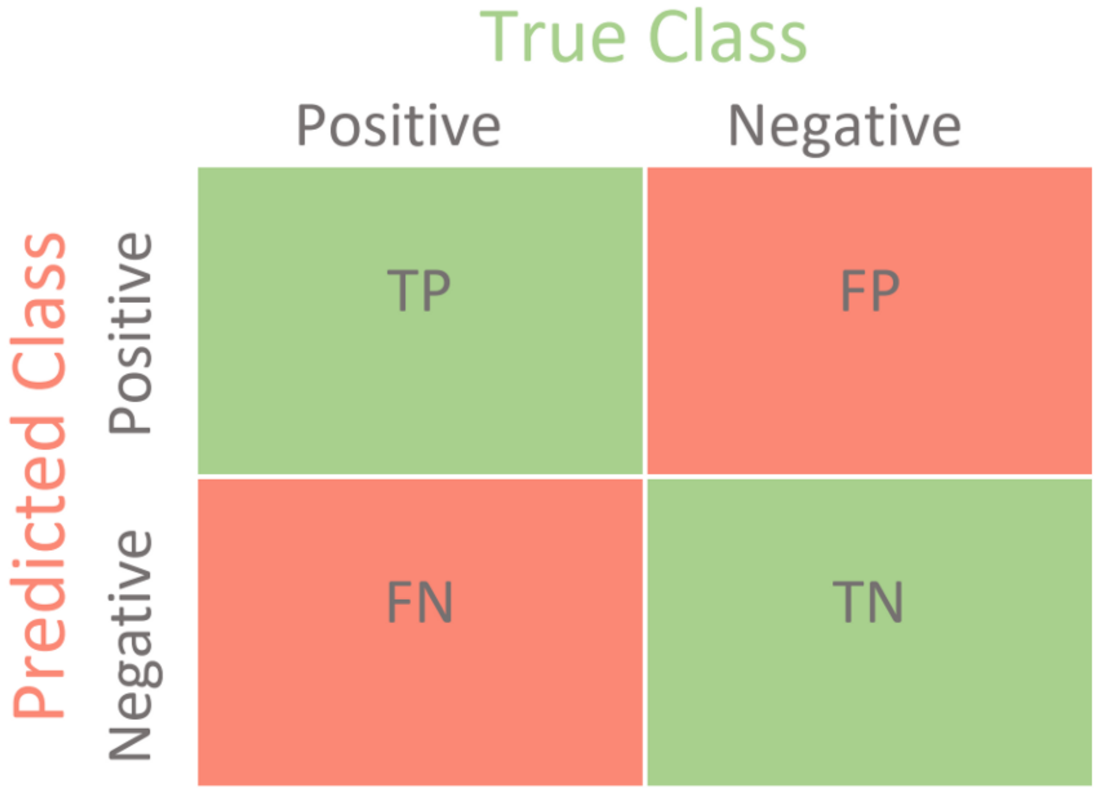
For a binary classification problem, we would have a 2 x 2 matrix as shown in figure with 4 values:



•The target/true 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

Evaluation: Confusion Matrix

- TP – True Positive
  - Number of correct positive predictions that are actually positive
- FP – False Positive
  - Number of incorrect positive predictions that are actually negative
- TN – True Negative
  - Number of correct negative predictions that are actually negative
- FN – False Negative
  - Number of incorrect negative predictions that are actually positive



Example

- Predict how many people are infected with a contagious virus in times before they show the symptoms, and isolate them from the healthy population;
  - The two values for our target will be: Sick and Not Sick
  - Dataset: 1000 data samples:
  - Train Binary Classifier: Neural Network
  - Network Prediction:

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:

Predicted \ Actual		
	TP=30	**FP=30**
	FN=10	TN=930

Evaluation Metric

- Accuracy:

It can be defined as how much data is correctly classified

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

$$\text{Misclassification Error} = \frac{FP \quad + FN}{TP+FN+FP+TN}$$

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

Misclassification Error = 1 - accuracy

$$\text{Accuracy} = 0.96 \Rightarrow 96\%$$

Evaluation Metric: Accuracy

- Network Prediction: 96% Accuracy
- It tells: “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! Because of TN

For Imbalanced Dataset: Classification Accuracy is not a correct measure

Evaluation Metric

- Precision:
  - It tells how many of the correctly predicted cases actually turned out to be positive. OR
  - It can be defined as how much returned data is correct

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

- Recall:
  - It tells how many of the actual positive cases we were able to predict correctly with our model OR
  - It can be defined as how much correct data is returned

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

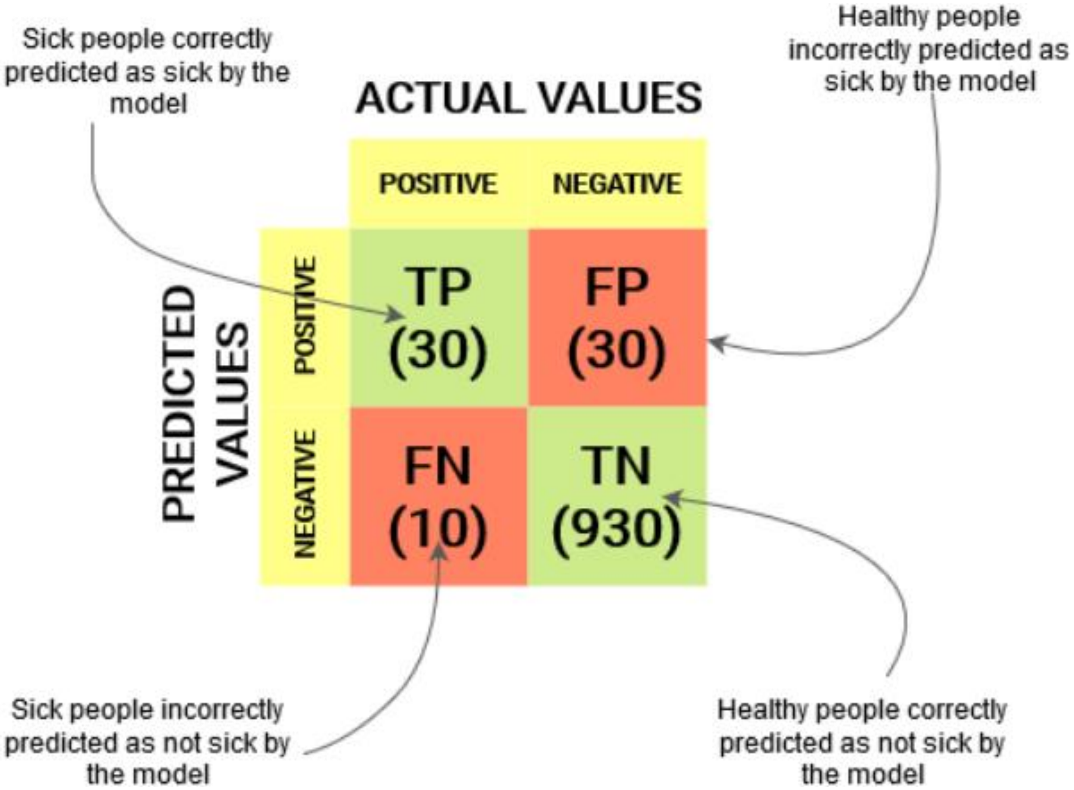
Evaluation Metric

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

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

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

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



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

Analysis: Precision vs Recall

- 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 if we raise a false alarm, but the actual positive cases should not go undetected!

- In previous 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.

Evaluation Metric:

- 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 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.

$$F1 - \text{Score} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 0.6$$

Confusion Matrix: Multiclass

- IRIS Dataset
  - The dataset has 3 flowers as outputs or classes, Versicolor, Virginia, Setosa.

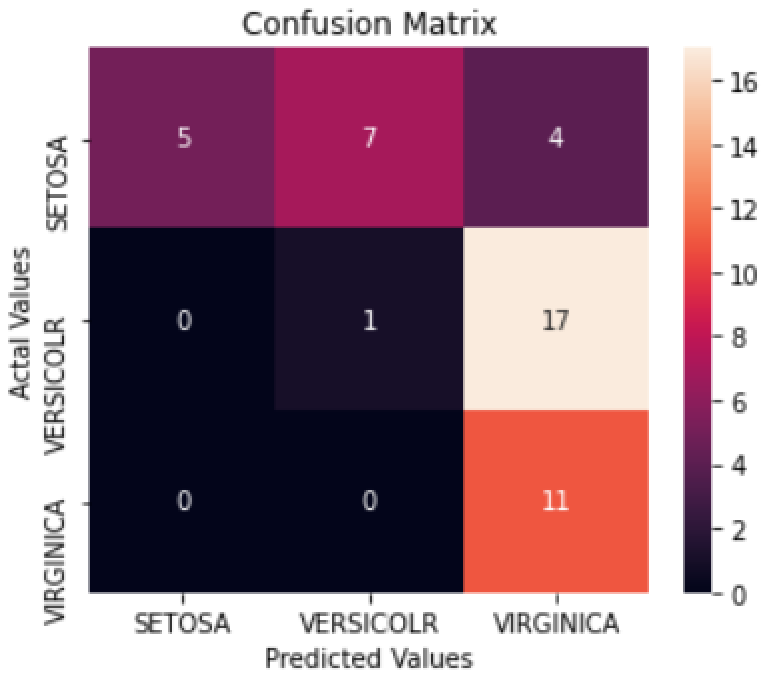
Confusion Matrix: Multiclass

- IRIS Dataset
  - The dataset has 3 flowers as outputs or classes, Versicolor, Virginia, Setosa.

  - How to calculate TP, FP, TN, FN

Confusion Matrix: Multiclass

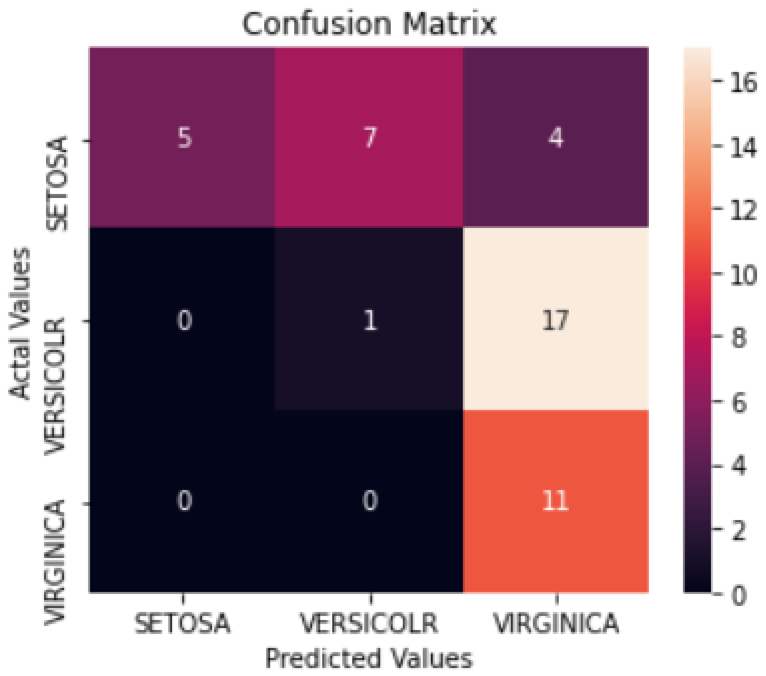
- **How to calculate FN, FP, TN, TP :**
- FN: The False-negative value for a class will be the sum of values of corresponding rows except for the TP value.



For SETOSA:
FN= 7 + 4

Confusion Matrix: Multiclass

- **How to calculate FN, FP, TN, TP :**
- FP: The False-positive value for a class will be the sum of values of the corresponding column except for the TP value.



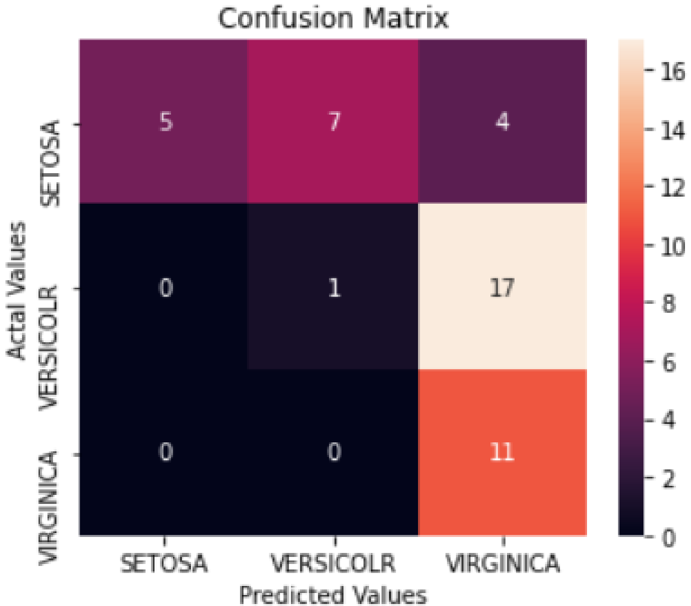
For SETOSA:

FN= 7 + 4 (R1C2+ R1C3)

FP= 0 + 0 (R2C1 + R3C1)

Confusion Matrix: Multiclass

- **How to calculate FN, FP, TN, TP :**
- TN: The True Negative value for a class will be the sum of values of all columns and rows except the values of that class that we are calculating the values for.



For SETOSA:

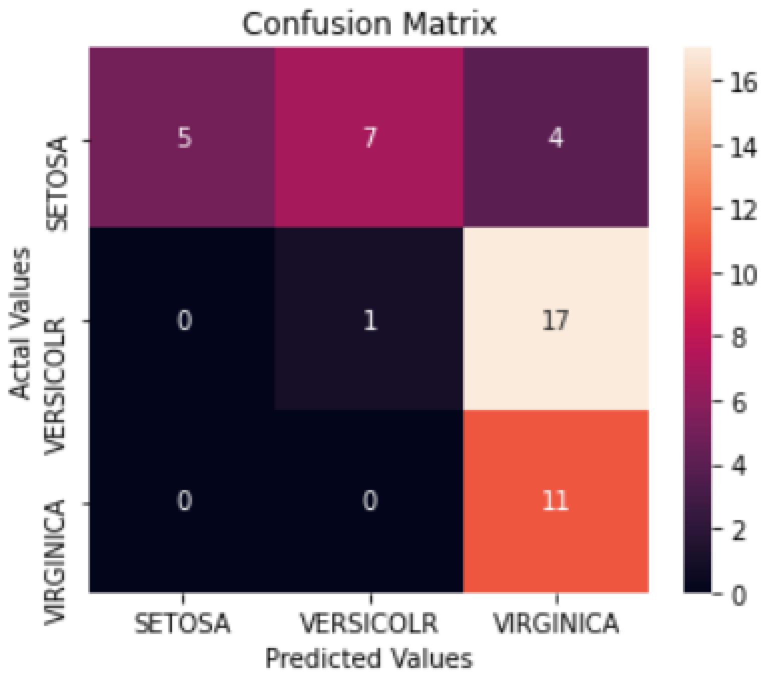
FN= 7 + 4 (R1C2+ R1C3)

FP= 0 + 0 (R2C1 + R3C1)

TN= 1 + 17 + 0 + 11 (R2C2 + R2C3 + R3C2 + R3C3

Confusion Matrix: Multiclass

- **How to calculate FN, FP, TN, TP :**
- TP: The True positive value is where the actual value and predicted value are the same.



For SETOSA:
FN= 7 + 4 (R1C2+ R1C3)
FP= 0 + 0 (R2C1 + R3C1)
TN= 1 + 17 + 0 + 11 (R2C2 + R2C3 + R3C2 + R3C3

TP= 5 (R1C1)

Confusion Matrix: Multiclass

- **How to calculate FN, FP, TN, TP :**
- FN: The False-negative value for a class will be the sum of values of corresponding rows except for the TP value.
- FP: The False-positive value for a class will be the sum of values of the corresponding column except for the TP value.
- TN: The True Negative value for a class will be the sum of values of all columns and rows except the values of that class that we are calculating the values for.
- TP: The True positive value is where the actual value and predicted value are the same.

Multiclass Classification: Confusion Matrix

- MNIST Dataset: labels[0-9]

```
[[5825    1   37    9   12   20   31   13   12   22]
 [   0 6657   53   23   14   34   25   46  101   17]
 [   1   19 5627   44   12   13    6   52   12    3]
 [   4   21   50 5827    0   67    2   24  111   59]
 [   6   12   48    6 5480   33    8   40   33   61]
 [  10    7    9   79    1 5095   38    2   13    8]
 [  26    1   30   14   43   45 5789    4   20    2]
 [   1    4   29   34    6    1    0 5872    3   26]
 [  38   12   63   51   10   51   19   11 5447   24]
 [  12    8   12   44  264   62    0  201   99 5727]]
```

Macro-average Precision

		gold labels urgent	gold labels normal	gold labels spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
system output	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
system output	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u$ = $\frac{8}{8+5+3}$	**recall**$_n$ = $\frac{60}{10+60+30}$	**recall**$_s$ = $\frac{200}{1+50+200}$	

Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

The **weighted-averaged** F1 score is calculated by taking the mean of all per-class F1 scores

Micro-average Precision

		gold labels			
		urgent	normal	spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u = \frac{8}{8+5+3}$	**recall**$_n = \frac{60}{10+60+30}$	**recall**$_s = \frac{200}{1+50+200}$	

Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

Micro-avg precision= TP1+TP2+TP3/ TP1+TP2+TP3+FP1+FP2+FP3

TP1=8, TP2= 60, TP3=200

FP1= 10+1

FP2= 50+5

FP3= 30 + 3

Micro-avg Precision= 268/268+99=0.73

- A micro-average is dominated by the more frequent class (in this case spam)
  - as the counts are pooled
- The macro-average better reflects the statistics of the smaller classes,
  - is more appropriate when performance on all the classes is equally important.

Evaluation

- In general, if you are working with an imbalanced dataset where all classes are equally important, using the **macro** average would be a good choice as it treats all classes equally.

- If you have an imbalanced dataset but want to assign greater contribution to classes with more examples in the dataset, then the weighted average is preferred.
  - This is because, in weighted averaging, the contribution of each class to the F1 average is weighted by its size.