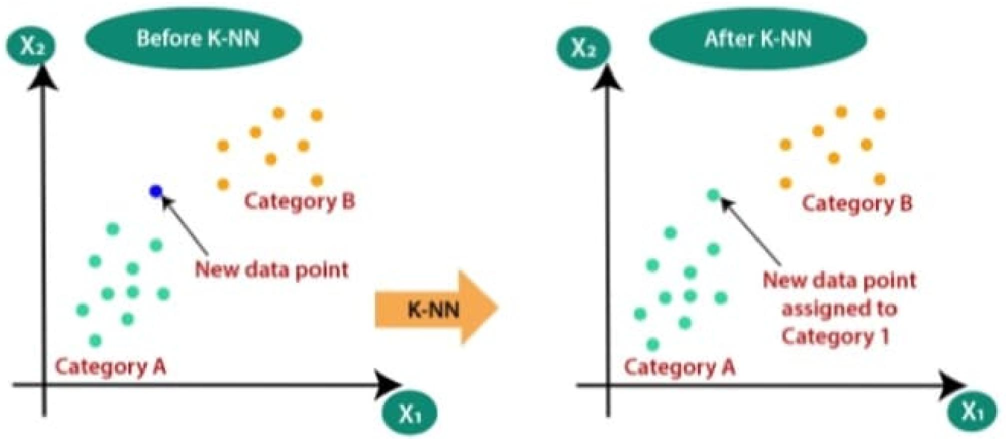
K-Nearest Neighbor(KNN) Algorithm for Machine Learning

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

(K-NN) K - Nearest Neighbors algorithm

	x_1 Height(cm)	y_1 Weight(kg)	Class
1	167	51	underweight
2	182	62	Normal
3	176	69	"
4	173	64	"
5	172	65	"
6	174	56	underweight
7	169	58	Normal
8	173	57	"
9	170	55	"
10	170	57	?

The Distance formula

$= \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

$x_2 = 170$

$y_2 = 57$

$\to d_1 (x_1=167, y_1=51)$

apply formula

$= \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

$= \sqrt{(170-167)^2+(57-51)^2}$

$= \sqrt{(3)^2+(6)^2}$

$= \sqrt{9+36}$

$= \sqrt{45}$

$= 6.7$

$d_2 - (x_1=182, y_1=62)$

$= \sqrt{(170-182)^2+(57-62)^2}$

$= \sqrt{(12)^2+(5)^2}$

$= \sqrt{144+25}$

$= \sqrt{169}$

$= 13$

$d_3 = (x_1 = 176 \quad y_1 = 69)$

$= \sqrt{(170-176)^2 + (57-69)^2}$

$= \sqrt{(6)^2 + (12)^2}$

$= \sqrt{36 + 144}$

$= \sqrt{178}$

$= 13.4$

$d_4 = (173, 64)$

$= \sqrt{(170-173)^2 + (57-64)^2}$

$= \sqrt{(3)^2 + (7)^2}$

$= \sqrt{9 + 49}$

$= \sqrt{58}$

$= 7.6$

$d_5 - (172, 65)$

$= \sqrt{(170-172)^2 + (57-65)^2}$

$= \sqrt{(2)^2 + (8)^2}$

$= \sqrt{4 + 64}$

$= \sqrt{68}$

$= 8.2$

$d_6 - (174, 56)$

$= \sqrt{(170-174)^2 + (57-56)^2}$

$= \sqrt{(4)^2 + (1)^2}$

$= \sqrt{16 + 1}$

$= \sqrt{17}$

$= 4.1$

$d_7 - (169, 58)$

$= \sqrt{(170-169)^2 + (57-58)^2}$

$= \sqrt{(1)^2 + (1)^2}$

$= \sqrt{2}$

$= 1.4$

$d_8 = (173, 57)$

$= \sqrt{(170-173)^2 + (57-57)^2}$

$= \sqrt{(3)^2 + (0)^2}$

$= \sqrt{9 + 0}$

$= 3$

$d_9 = (170, 55)$

$= \sqrt{(170-170)^2 + (57-55)^2}$

$= \sqrt{0 + (2)^2}$

$= \sqrt{4}$

$= 2$

Height(cm)	Weight(kg)	Class	Distance
167	51	Underweight	6.7
182	62	Normal	13
176	69	"	13.4
173	64	"	7.6
172	~~3~~65	"	8.2
174	56	Underweight	4.1
169	58	Normal	1.4
173	57	"	3
170	55	"	2
170	57	?	

Height(cm)	Weight(kg)	Class	Distance	Rank
169	58	Normal	1.4	1
170	55	"	2	2
173	57	"	3	3
174	56	underweight	4.1	4
167	51	"	6.7	5
173	64	Normal	7.6	6
172	65	"	8.2	7
182	62	"	13	8
176	69	"	13.4	9
170	57	?		

if R=1 Normal (To much neareast

so 170, 57 is belong to (169, 58)

→ Normal ~~cond~~ disease

(170, 57) is Normal)

M - 0
F - 1

Age	Gender	game
60	0	play
50	1	"
55	0	"
51	0	Not Play
45	0	"
40	1	"
38	1	Play
35	1	Play
30	0	?

$(x_2, y_2) = (30, 0) = ?$

Confusion Matrix in Machine Learning

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are **predicted values** and **actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

The above table has the following cases:

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

Calculations using Confusion Matrix:

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

- **Classification Accuracy:** It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

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

- **Misclassification rate:** It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

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

- **Precision:** It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

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

- **Recall:** It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

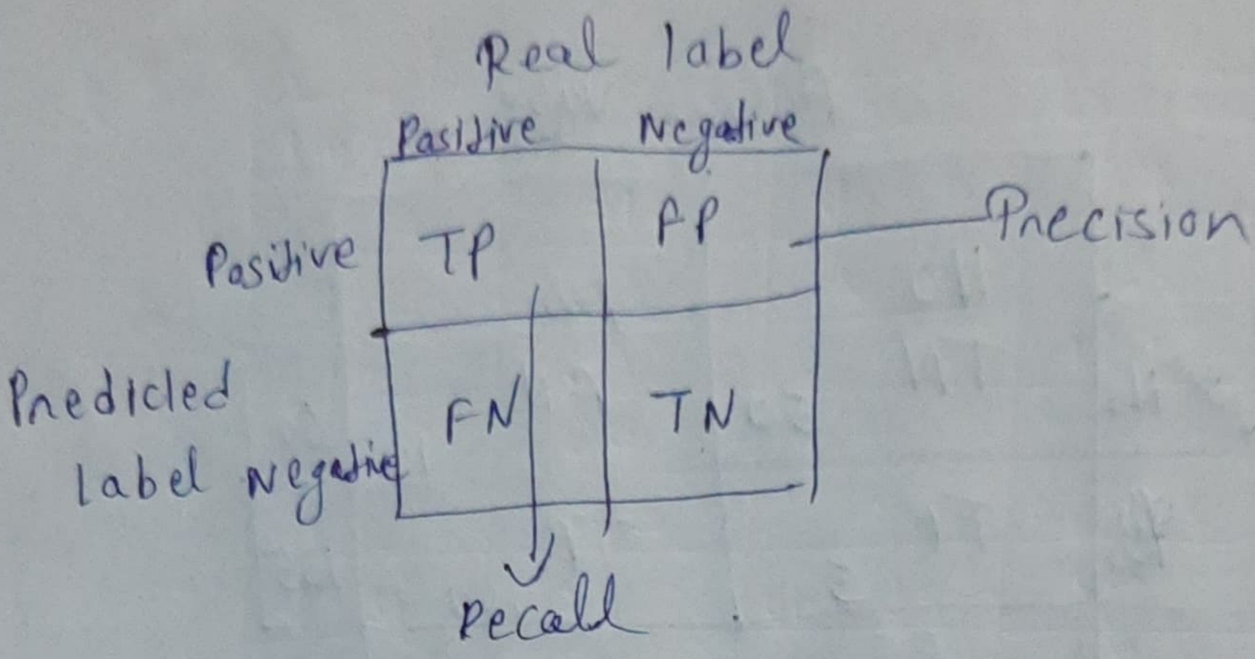
  Recall = $\frac{TP}{TP+FN}$

- **F-measure:** If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

  F-measure = $\frac{2*Recall*Precision}{Recall+Precision}$

Other important terms used in Confusion Matrix:

- **Null Error rate:** It defines how often our model would be incorrect if it always predicted the majority class. As per the accuracy paradox, it is said that "*the best classifier has a higher error rate than the null error rate.*"
- **ROC Curve:** The ROC is a graph displaying a classifier's performance for all possible thresholds. The graph is plotted between the true positive rate (on the Y-axis) and the false Positive rate (on the x-axis).



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

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

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

$$f_1\text{-mesure} = \frac{2TP}{2TP + FP + FN}$$

①

	Actual value		
Predicted value	No	Yes	
No	TN 50	FN 10	60
Yes	FP 5	TP 100	105
	55	110	

$N = 165$

$$\text{Precision} = \frac{100}{105}$$

$$\text{Recall} = \frac{100}{110}$$

$$\text{Accuracy} = \frac{150}{165}$$

$$F_1 = \frac{200}{215}$$

①

	Predicted value Yes	No
Actual value Yes	TP 1000	FP 350
No	FN 150	TN 500

$N = 2000$

Find → Accuracy, Recall, Precision, F_1