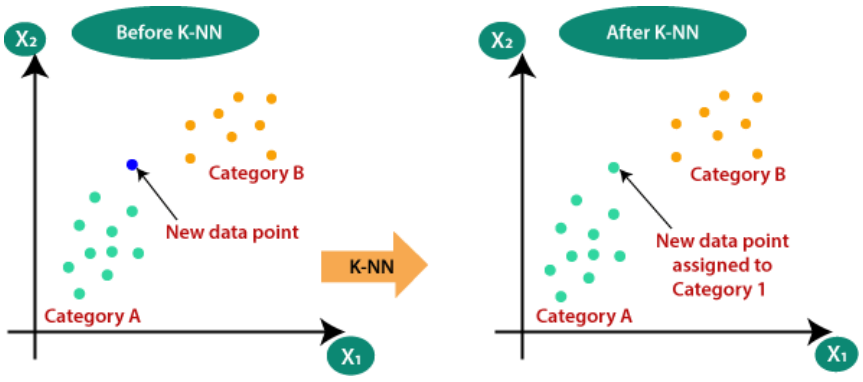
K – Nearest Neighbour (KNN) (not effected by outliers)

- 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 data and available data and put the new data 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.
- KNN is Suitable for noisy data and smaller data

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:

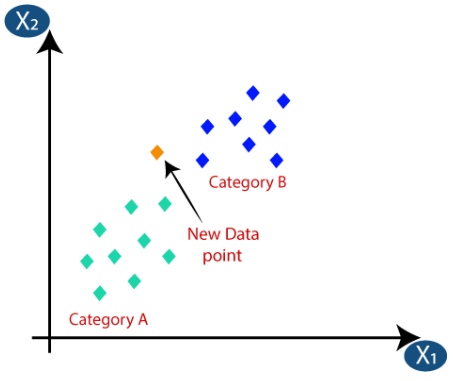
- **Step-1:** Select the number K of the neighbors

  Below are some points to remember while selecting the value of K in the K-NN algorithm

  I. There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
  II. A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
  III. Large values for K are good, but it may find some difficulties.

- **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.

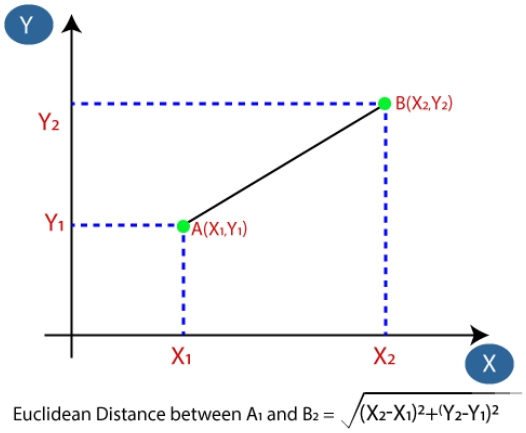
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.

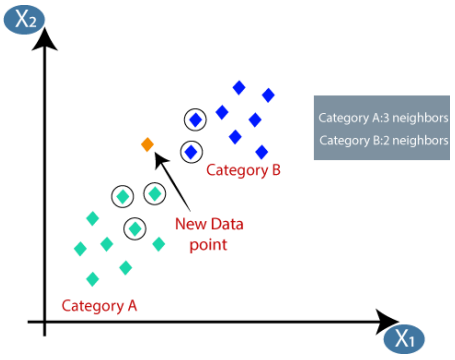
- Next, we will calculate the **Euclidean distance** between the data points.

Euclidean Distance represents the shortest distance between two



points.

- By calculating the Euclidean distance, we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new datapoint must belong to category A.

Tuning

1. Feature Selection
2. Dedicated approach → Hyperparameter Tuning
    - Change the value ok K or the metric
    - We can apply for loop and find optimum value of k from 1 to its square root also the value of k should not be even its should be an odd number
    - Metric is by default **minkowski** we can change it to

Euclidean,Manhattan or Hamming