The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. KNN algorithm uses in search applications where people looking for similar items. The KNN algorithm assumes similar things are near to each other. It suggests that if the new point added to the sample is similar with the neighbor points, that point will belong to the class of the neighbor points. K in the KNN algorithm denotes the number of nearest neighbors of the new point which needed to be predicted. To find the distance we use the **Euclidean-distance** formula to calculate the similarity/distance.Here, the ‘k’ in the k-nearest neighbor is the number of neighbors that our model takes into consideration to predict the outcome.

**About KNN:**

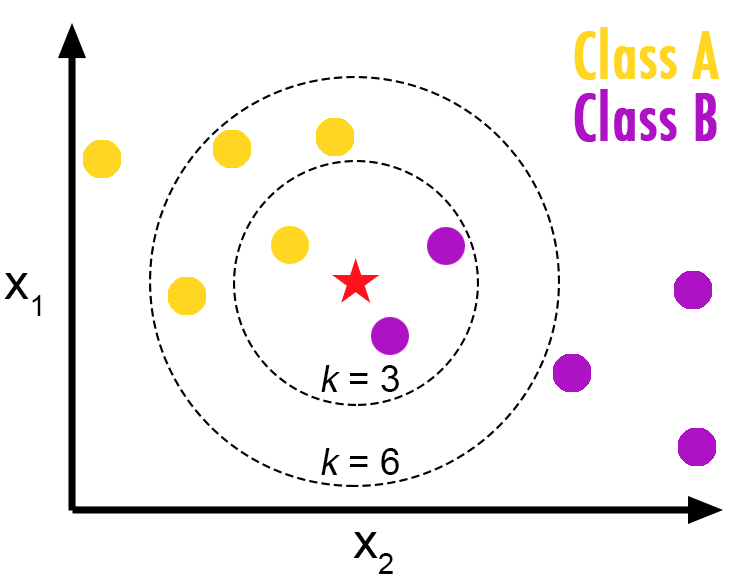
1. KNN tries to find similarities between predictors and values that are within the dataset.
2. KNN uses a non-parametric method as there is not a particular finding of parameters to a particular functional form.
3. It does not make any type of assumptions about the features and output of the dataset.
4. KNN is also called a lazy classifier as it memorizes the training data and not exactly learn and fix the weights.
5. KNN usually works by just trying to see to which class is the new feature near to and it just puts it to the class closest to that point. To predict the class of a given data point, KNN considers the classes of the 'K' nearest data points and chooses the class in which most of the 'K' nearest data points belong to as the predicted class.

**Working of KNN Algorithm:** Initially, we select a value for K in our KNN algorithm.

1. Now we go for a distance measure. Let’s find Eucleadean distance of k neighbours here.
2. Now we check all the neighbours to the new point we have given and see which is nearest to our point. We only check for k-nearest here.
3. Now we see to which class there is the highest number obtained. The max number is chosen and we assign our new point to that class.

**The ideal value of K in KNN:**

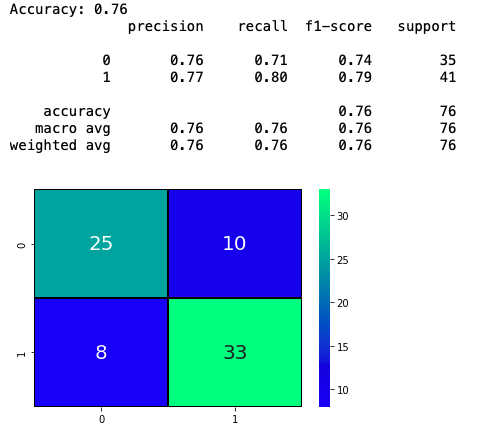
If we go for too small a value of k, there is a good chance we may have overfitting of data, that’s the algorithm may perform reasonably well of training but not well on testing data. And we also may encounter noise if we just use the small value of k. If we choose too large value of k, we may see underfitting. For example, we have data points of Class A and B. We want to predict what the star (test data point) is. If we consider a k value of 3 (3 nearest data points), we will obtain a prediction of Class B. Yet if we consider a k value of 6, we will obtain a prediction of Class A. From this following diagram, we get a sense of what the K-Nearest Neighbors algorithm is. It considers the 'K' Nearest Neighbors (data points) when it predicts the classification of the test point.



**Example:**

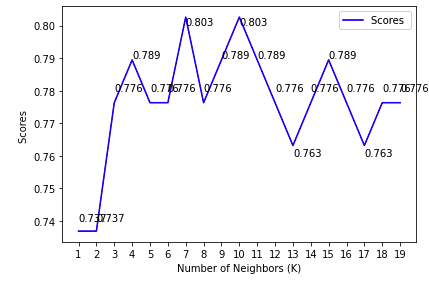
**Confusion Metrix and Accuracy Before the Tunning of K:**

We chose the value of K as a default parameter

****

# **What is the Right Value of Parameter K?**

K in KNN, is the number of nearest neighbors to examine. It is supposed to be specified by the user. So, how can we choose right value for K? The general solution is to reserve a part of your data for testing the accuracy of the model. Then choose k =1, use the training part for modeling, and calculate the accuracy of prediction using all samples in your test set. Repeat this process, increasing the k, and see which k is the best for your model. We can calculate the accuracy of KNN for different values of k. The selection of the proper value of k that is changing the distance measure for different applications may help improve the accuracy of the algorithm. From the following graph, we chose k=10 as the best value



**Confusion Metrix and Accuracy after Selecting the Best Value of K**

