**Chapter 2**

**Introduction**

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. In Credit ratings, financial institutes will predict the credit rating of customers. In loan disbursement, banking institutes will predict whether the loan is safe or risky. In political science, classifying potential voters in two classes will vote or won’t vote. KNN algorithm used for both classification and regression problems. KNN algorithm based on feature similarity approach.

KNN is a non-parametric and indolent learning algorithm. Non-parametric betokens there is no postulation for underlying data distribution. In other words, the model structure determined from the dataset. This will be very subsidiary in practice where most of the genuine world datasets do not follow mathematical theoretical postulations. Slothful algorithm designates it does not require any training data points for model generation. All training data utilized in the testing phase. This makes training more expeditious and testing phase more gradual and costlier. Costly testing phase betokens time and recollection. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more recollection for storing training data.

**Existing System**

In the existing system, K is the number of nearest neighbors (KNN). The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.

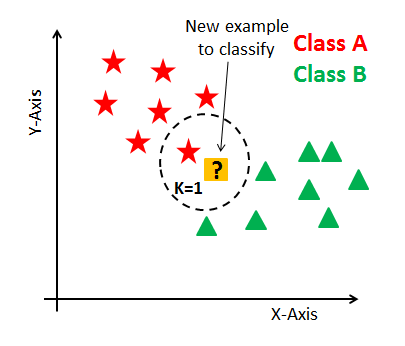


Fig: Initial Data

Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Murkowski distance. KNN has the following basic steps:

* Calculate distance
* Find closest neighbors
* Vote for labels

KNN performs better with a lower number of features than an astronomically immense number of features. You can verbally express that when the number of features increases than it requires more data. Increment in dimension withal leads to the quandary of overfitting. To eschew overfitting, the needed data will require to grow exponentially as you increment the number of dimensions. This quandary of higher dimension is kenned as the Curse of Dimensionality.

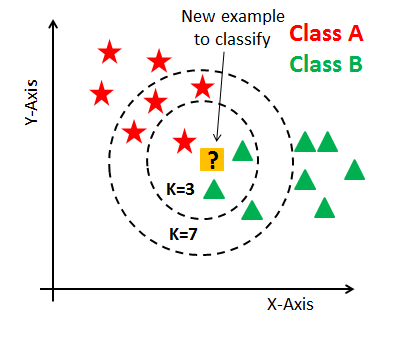
To deal with the quandary of the imprecation of dimensionality, you require to perform principal component analysis afore applying any machine learning algorithm, or you can additionally use feature cull approach. Research has shown that in sizably voluminous dimension Euclidean distance is not utilizable anymore. Consequently, you can prefer other measures such as cosine homogeneous attribute, which get decidedly less affected by high dimension.



At this point, the question arises that How to choose the optimal number of neighbors? And what are its effects on the classifier? The number of neighbors(K) in KNN is a hyper-parameter that you need choose at the time of model building. You can think of K as a controlling variable for the prediction model.

Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has its own requirements. In the case of a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. Research has also shown that a small amount of neighbors is most flexible fit which will have low bias but high variance and a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias.

Generally, Data scientists choose as an odd number if the number of classes is even. You can also check by generating the model on different values of k and check their performance. You can also try Elbow method here.



**Defining dataset**

Let's first create your own dataset. Here you need two kinds of attributes or columns in your data: Feature and label. The reason for two type of column is "supervised nature of KNN algorithm".

**Encoding data columns**

Various machine learning algorithms require numerical input data, so you need to represent categorical columns in a numerical column.



**Generating Model**

Let's build KNN classifier model.

First, import the K Neighbors Classifier module and create KNN classifier object by passing argument number of neighbors Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

Loading data …. Exploring data…. Splitting data …. Generating model for K=4 (need to add screen shot or add the data sets)