K-Nearest Neighbor(KNN) Algorithm

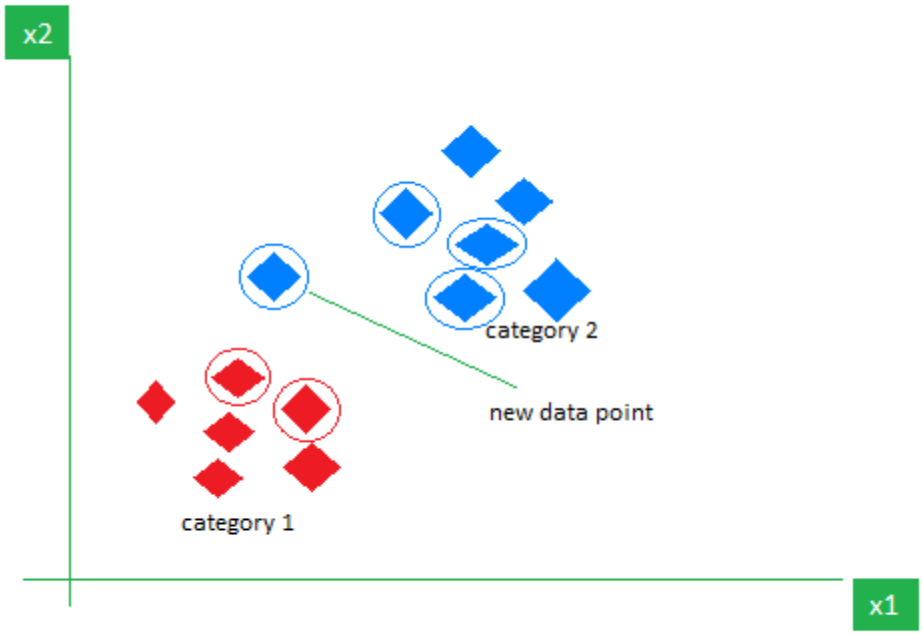
The **K-Nearest Neighbors (KNN) algorithm** is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The article explores the fundamentals, workings, and implementation of the KNN algorithm.

What is the K-Nearest Neighbors Algorithm?

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



KNN Algorithm working visualization

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

Intuition Behind KNN Algorithm

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to. This means a point close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green', and the second point (5.5, 4.5) should be classified as 'Red'.

Why do we need a KNN algorithm?

(K-NN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and

regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

distance(x,Xi)=∑j=1d(xj–Xij)2]distance(*x*,*Xi*)=∑*j*=1*d*(*xj*–*Xij*)2]

Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

d(x,y)=∑i=1n|xi−yi|*d*(*x*,*y*)=∑*i*=1*n*|*xi*−*yi*|

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

d(x,y)=(∑i=1n(xi−yi)p)1p*d*(*x*,*y*)=(∑*i*=1*n*(*xi*−*yi*)*p*)*p*1

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be Boolean as well as string values.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

Algorithm for K-NN

DistanceToNN=sort(distance from 1st example, distance from kth example)

value i=1 to number of training records:

Dist=distance(test example, ith example)

if (Dist<any example in DistanceToNN):

Remove the example from DistanceToNN and value.

Put new example in DistanceToNN and value in sorted order.

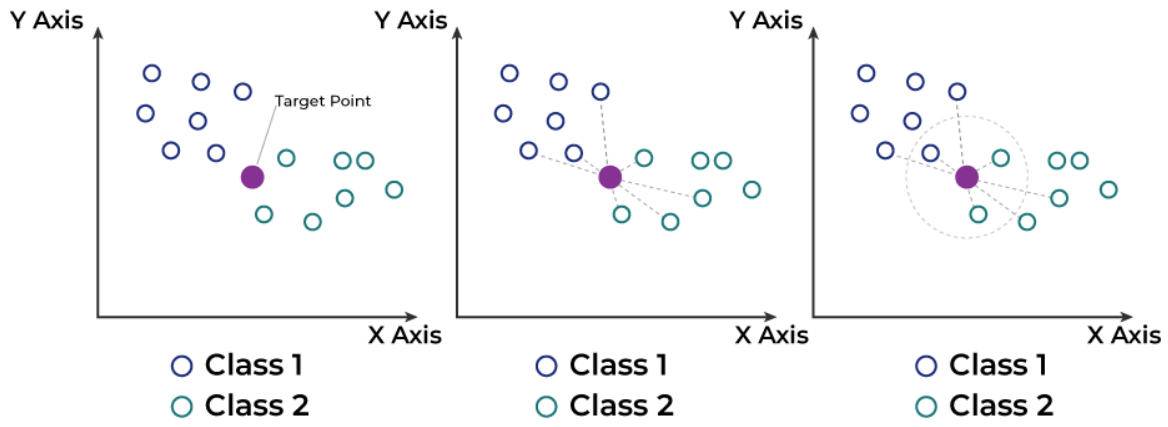
Return average of value

Fit using K-NN is more reasonable than 1-NN, K-NN affects very less from noise if dataset is large.

In K-NN algorithm, We can see jump in prediction values due to unit change in input. The reason for this due to change in neighbors. To handles this situation, We can use weighting of neighbors in algorithm. If the distance from neighbor is high, we want less effect from that neighbor. If distance is low, that neighbor should be more effective than others.

Workings of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K

- K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

- To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

- The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

- In the classification problem, the class labels of K-nearest neighbors are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
- In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector Xi*Xi* and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point Xi*Xi* in X using a distance metric, such as Euclidean distance:distance(x,Xi)=∑j=1d(xj–Xij)2]distance(*x*,*Xi*)=∑*j*=1*d*(*xj*–*Xij*)2]

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x.

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

Applications of the KNN Algorithm

- **Data Preprocessing** – While dealing with any Machine Learning problem we first perform the EDA part in which if we find that the data contains missing values then there are multiple imputation methods are available as well. One of such method is KNN Imputer which is quite effective ad generally used for sophisticated imputation methodologies.
- **Pattern Recognition** – KNN algorithms work very well if you have trained a KNN algorithm using the MNIST dataset and then performed the evaluation process then you must have come across the fact that the accuracy is too high.
- **Recommendation Engines** – The main task which is performed by a KNN algorithm is to assign a new query point to a pre-existed group that has been created using a huge corpus of datasets. This is exactly what is required in the recommender systems to assign each user to a particular group and then provide them recommendations based on that group's preferences.