WEEK -08: K-Nearest Neighbour (K-NN) & ID-3 Decision Tree

Introduction

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.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

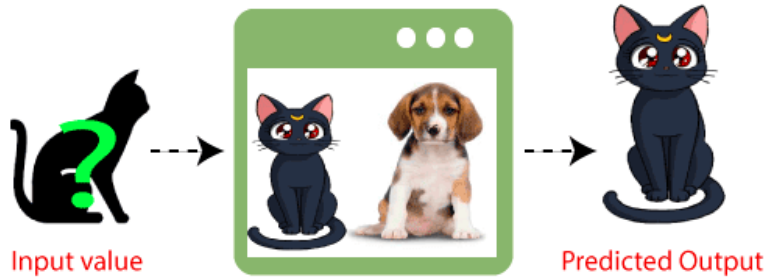
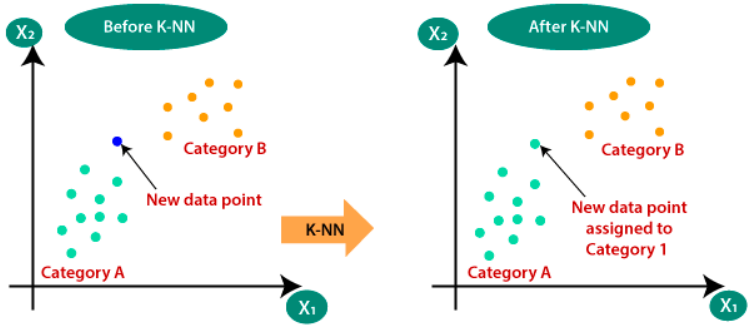


Figure 1 K-Nearest Neighbor Classifier

Need of a K-Nearest Neighbour Algorithm

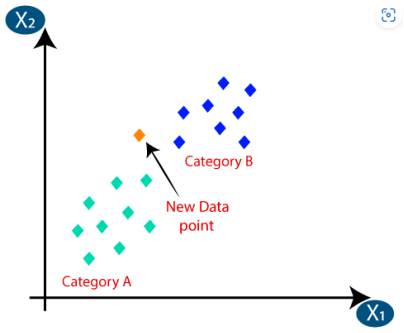
Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.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



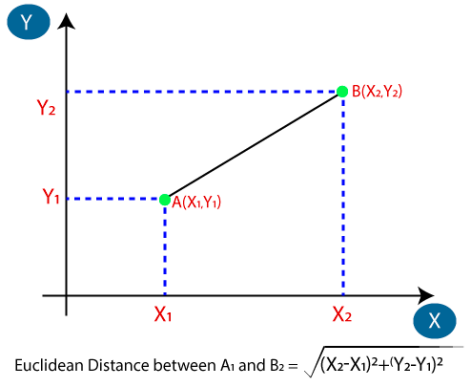
Working of a K-NN

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.

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. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



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 data point must belong to category A.

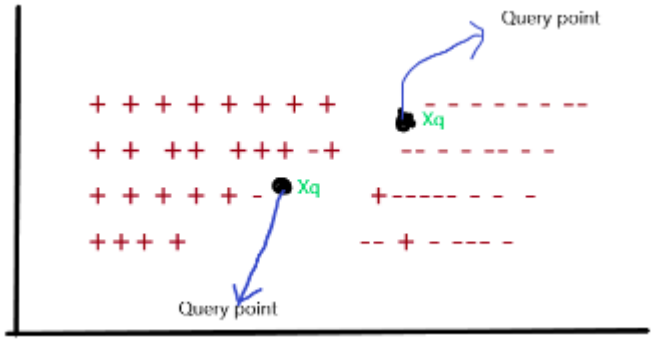
Selection of the value K in the K-NN Algorithm

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

- 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.
- 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.
- Large values for K are good, but it may find some difficulties.

Other Distance Measures used in K-NN

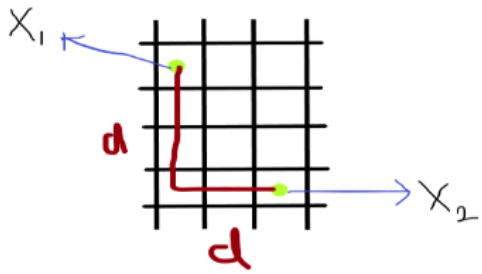
Nearest Neighbour:
let's take the simplest case of binary classification, suppose we have a group of +ve and -ve points in the dataset D such that the Xis belongs to the R-dim. data points and Yi are labels (+ve and -ve).



From the above image, you can conclude that there are several data points in 2 dim. Having the specific label, they are classified according to the +ve and -ve labels. If you noticed in the image there is one Query point referred to as **Xq** which has an unknown label. The surrounding points of **Xq** we considered as neighbours of Xq and the points which are close to the Xq are nearest neighbours. So how can we conclude that this point is nearest or not? It's by finding the distance b/w the points. So, here's the distance measures come existence.

We generally say that we will use distance to find the nearest neighbours of any query point Xq, but we still don't know how mathematically distance is measured between Xq and other nearest points? for further finding distance, we can't conclude that this is nearest or not.

In a theoretical manner, we can say that a distance measure is an objective score that summarizes the difference between two objects in a specific domain. There are several types of distance measures techniques but we only use some of them and they are listed below:
Manhatten Distance
This is the simplest way or technique to calculate the distance between two points, often called Taxicab distance or City Block distance, you can easily relate this with your daily life, If you start from somewhere and reached some destination so the Manhatten distance says that the distance between your starting point and the destination point. More mathematical we can say that It calculates the **absolute value** between two points. We calculate the distance exactly as the original path is we didn't take any diagonal or shortest path.
Let's take the geometric intuition for better understanding;

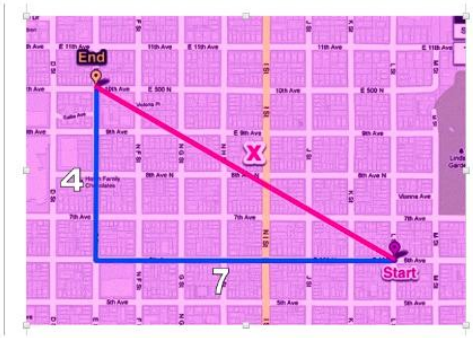


you can see in the image that the points **X1** and **X2** are 2-Dim vectors and having the coordinates same as before we discussed for euclidean distance, but here we can't calculate the distance as we calculate earlier apart from this we take the absolute value of the path from X1 to X2. In the image see that we take the path that actually covers from one point to another.

Manhatten Distance = sum for i to N sum || $X1_i$ – $X2_i$ ||
In mathematically we can write as :

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Note: Manhatten distance between two vectors or points is the L1 norm of two vector



As you see in the image the blue line represents the absolute path that the cab travel.
as Manhatten formula says: **distance = absolute sum ||xi-yi||**
distance = (7 + 4)
distance = 11
So, the absolute path the cab cover is 11.

Minkowski distance
Above we have discussed the L1 norm and L2 norm so this is the Lp norm of two vectors, More often we say that the Minkowski distance is the generalization or generalized form of the euclidean and Manhatten distance. Why do we call it to generalize? because we take both the distance technique and the new technique for finding the distance between vectors. It adds a parameter, called the "P", that allows different distance measures to be calculated.
Let's take it more mathematically, the equation of Minkowski distance is:

$$\|X_1 - X_2\| = \left(\sum_{i=1}^{d} |X_{1i} - X_{2i}|^p \right)^{1/p}$$

From the above equation you notice that the formula is the same as Euclidean distance but the change is that here we prefer the value of P, So if we take the P-value equals to 2 then it is euclidian distance and takes P-value equals to 1 then it is considered as Manhatten distance.
P = 1 => D = (∑i=1 to n(X1i−X2i)^1)$^{1/1}$
P = 2 => D = (∑i=1 to n(X1i−X2i)^2)$^{1/2}$
Note: **Minkowski distance between two vectors or points is the Lp norm of two vector.**

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^c \right)^{\frac{1}{c}}$$

Example:
So, here we will take the same example as we take in the euclidean distance measures.
X1(x1, y1) = X1(3, 4)
X2(x2, y2) = X2(4, 7)
and we take the value of P = 4
distance = ((x2-x1)^4 + (y2-y1)^4)$^{1/4}$
distance = ((4-3)^4 + (7-4)^4)$^{1/4}$
distance = ((1)^4 + (3)^4)$^{1/4}$
distance = (1+81)$^{1/4}$
distance = (82)$^{1/4}$
distance = approx(2)

Pseudo Code of KNN

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    a. Calculate the distance between test data and each row of training dataset. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other distance function or metrics that can be used are Manhattan distance, Minkowski distance, etc. If there are categorical variables, hamming distance can be used.
    b. Sort the calculated distances in ascending order based on distance values
    c. Get top k rows from the sorted array
    d. Get the most frequent class of these rows
    e. Return the predicted class

ID-3 Decision Tree

A decision tree is a structure that contains nodes (rectangular boxes) and edges(arrows) and is built from a dataset (table of columns representing features/attributes and rows corresponds to records). Each node is either used to make a decision (known as decision node) or represent an outcome (known as leaf node).

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step. Invented by Ross Quinlan, ID3 uses a top-down greedy approach to build a decision tree. In simple words, the top-down approach means that we start building the tree from the top and the greedy approach means that at each iteration we select the best feature at the present moment to create a node.

Metrics in ID3

ID3 uses **Information Gain** or just **Gain** to find the best feature. Information Gain calculates the reduction in the entropy and measures how well a given feature separates or classifies the target classes. The feature with the **highest Information Gain** is selected as the **best** one. **Entropy** is the measure of disorder and the Entropy of a dataset is the measure of disorder in the target feature of the dataset.

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

ID3 Steps

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset S into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Questions

Q-1. A Classification of Fruits

You are provided with a dataset of fruits. Each fruit is characterized by two features: weight (in grams) and sweetness level (on a scale of 1 to 10). You want to classify a new fruit as either an "Apple" or an "Orange" based on these features using the KNN algorithm.

Fruit ID	Weight (grams)	Sweetness Level	Label (Fruit Type)
1	180	7	Apple
2	200	6	Apple
3	150	4	Orange
4	170	5	Orange
5	160	6	Apple
6	140	3	Orange

Tasks:

1. Implement the KNN algorithm manually with k=3 to classify a new fruit with a weight of 165 grams and sweetness level of 5.5.
2. Calculate the Euclidean, Manhattan, and Minkowski distances between the new fruit and all the existing fruits in the dataset. Finally compare the calculated distances.
3. Based on the k-nearest neighbors, determine the label for the new fruit.
4. What is the effect of choosing different values of k (e.g., k=1, k=5) on the classification result?
5. Implement the above using function python program without using scikit learn library.
6. Plot the given samples, the Apple in Red color and the Orange in orange color. Also draw the decision boundary.

Q-1. B Classification of Fruits

Implement the Python code for Q-1. A using the ***scikit-learn*** library. Plot the given samples, using red for "Apple" and orange for "Orange." Also, plot the decision boundary. Calculate the distances using Euclidean, Manhattan, and Minkowski metrics, and compare the results.

Q-2. A Medical Diagnosis Decision

A dataset is provided to classify patients as "Healthy" or "Sick" based on their Age, Blood Pressure, and Cholesterol levels.

Patient ID	Age (years)	Blood Pressure (High/Low)	Cholesterol (High/Normal)	Diagnosis (Healthy/Sick)
1	30	High	High	Sick
2	45	Low	Normal	Healthy
3	50	High	High	Sick
4	35	Low	Normal	Healthy
5	60	High	High	Sick
6	55	Low	Normal	Healthy
7	40	High	High	Sick
8	25	Low	Normal	Healthy
9	65	High	High	Sick
10	45	Low	Normal	Healthy

Tasks:

1. Calculate the entropy for the target variable (Diagnosis).
2. Calculate the information gain for each feature (Age, Blood Pressure, Cholesterol).

3. Using the ID3 algorithm, decide which feature should be chosen as the root node for the decision tree.
4. Build the decision tree and explain the first few splits.
5. Predict whether a 50-year-old patient with low blood pressure and normal cholesterol is healthy or sick using the tree you built.
6. Implement the above using function python program without using scikit learn library.

Q-2. B Medical Diagnosis Decision

Implement the Python code for Q-2. A using the ***scikit-learn*** library. Using the ID3 algorithm, decide which feature should be chosen as the root node for the decision tree. Build the decision tree and explain the first few splits. Predict whether a 50-year-old patient with low blood pressure and normal cholesterol is healthy or sick using the tree you built.

Additional Questions

1. We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples as follows. Apply the K-nearest neighbour's (KNN) algorithm when K=2, 3 and 4 to classify an instance (3, 7) as good or bad.

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good
4	5	Bad
3	5	Good
4	6	Bad
8	7	Bad
7	9	Good
8	8	Bad

Implement the above using python code without using scikit learn library. Plot the given samples Bad in Red color and Good in green color. Also draw the decision boundary. Calculate the desistance using Euclidean, Manhattan, and Minkowski and compare.

2. Implement the Question number 1 with using scikit learn library. Plot the given samples Bad in Red color and Good in green color. Also plot the decision boundary. Calculate the desistance using Euclidean, Manhattan, and Minkowski and compare the results.

3.There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the Estimated Salary and Age we will consider for the independent variable and the Purchased variable is for the dependent variable. Below is the dataset:

1. Apply the K-NN algorithm when K=2, 3 and 4 to classify purchased or not. Calculate the desistance using Euclidean, Manhattan, and Minkowski and compare.
2. Test your developed K-NN without and with using Scikit Learn Library.
3. Plot the Yellow points are for Purchased(1) and Green Points for not Purchased(0) variable.
4. Show the graph has to classify users in the correct categories, as most of the users who didn't buy the SUV are in the red region, and users who bought the SUV are in the green region.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1