K-Nearest Neighbours (KNN)

KNN is a supervised machine learning algorithm that can be used for both classification and regression tasks. It is a simple yet effective algorithm based on the principle that similar things are grouped together.

→ **How it works:**

1. Training phase → The algorithm stores all the training data points without any learning process.
2. When a new data point (query point) needs to be classified or predicted →

Prediction phase

(a) It calculates distance between all the training point and query point.

(b) Find the k-nearest neighbours to the query point.

(c) If the task is classification, assigns the query point to the most frequent class among the k neighbours.

(d) If the task is regression, calculate the average of the target values of the k neighbours and assign it to the query point.

* **Distance Metric** : The closeness between data points is measured using a distance metric. The most common metric used is **euclidean distance** which is given by the formula shown below and calculates straight line distance.

$$\boxed{d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}}$$

Suppose two points in space are (3,4) and (6,3) then distance between them using euclidean distance is—

$$d = \sqrt{(3-6)^2+(4-3)^2} = \sqrt{9+1} = \sqrt{10} \approx 3.16$$

Manhattan distance → The manhattan distance also known as (L1 distance or taxicab distance) between two points is the sum of the absolute differences of their coordinates. It is calculated using the formula—

$$\boxed{d_{\text{Manhattan}}(p,q) = \sum_{i=1}^{n} |p_i - q_i|}$$

where p & q are two points in an n-dimensional space, and p_i and q_i are the coordinates of these points in the i^{th} dimension.

The Manhattan distance measures how far two points are by moving along the gridlines (like a taxi navigating the streets of a city). The distance metric is more appropriate in situations where we can only move along the gridlines rather than in a straight line.

The Manhattan distance between two points (3,4)&(6,3) can be calculated as—

$$d = |6-3| + |3-4| = 3+1 = 4$$

→ It can be used in places where we require grid-based movement (e.g. city blocks, pixel grids in images). In case of high dimensional spaces and sparse data.

Minkowski Distance : Minkowski is a generalization of both Euclidean and Manhattan distances. It is calculated using the formula:

$$\boxed{d_{\text{Minkowski}}(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{1/p}}$$

where p is the parameter that defines the type of distance. When $p=1$, Minkowski distance becomes Manhattan distance and when $p=2$, Minkowski distance becomes Euclidean distance. When p increases, the distance metric becomes more sensitive to large differences ~~betw~~ in any single dimension.

→ If we suspect that some features might disproportionately influence the distance, we can choose a higher p to emphasize these differences.

The Minkowski distance between two points (3,4)&(6,3) taking $p=3$ & $p=4$, can be calculated as:—

For $p=3$, $d = (|6-3|^3 + |3-4|^3)^{1/3} = (3^3+1^3)^{1/3} = 28^{1/3}$

$d \approx 3.03$

For $p=4$, $d = (|6-3|^4 + |3-4|^4)^{1/4} = (3^4+1^4)^{1/4} = (82)^{1/4}$

$d \approx 3$

For $p=10$, $d = (|6-3|^{10} + |3-4|^{10})^{1/10} = (3^{10}+1^{10})^{1/10} = (59050)^{1/10}$

≈ 3

* **Choosing the correct value of k:** The choice of k in KNN significantly impacts the model's performance. A small k might lead to overfitting, while a large k can result in underfitting (ignoring local patterns).

(a) Using cross-validation can be a suitable ~~valu~~ method for choosing k.

(b) We can even do a hyperparameter search using different values of k to find the best k.

(c) We can consider square root of the dataset size as a starting point and start by choosing an odd value.

(d) For imbalanced datasets, a smaller k might be more appropriate to avoid the majority class dominating the predictions.

(e) If the data is noisy and contains many outliers, a larger k can help smooth out the noise and make the model more robust.

Pros and Cons of KNN algorithm:

→ Pros: (a) It is simple and easy to understand.

(b) No training phase is required (a lazy learning algorithm).

(c) It can handle high-dimensional data effectively.

→ Cons: (a) Can be computationally expensive for large datasets (because it needs to calculate distance to every data point), especially during prediction.

(b) It is sensitive to the choice of both 'k' and distance metric.

(c) It can be affected by presence of noisy data and outliers.

Applications:

(a) Recommendation Systems: Suggesting similar items based on user preferences.

(b) Image Recognition: For identifying and classifying objects in images

(c) Anamoly detection: Identifying unusual patterns that do not confirm to expected behaviour.