K-Nearest Neighbours

* This model can be used for both Classification and Regression, we will first understand the classification problem using KNN. It is an instance-based learning meaning it does not explicitly train the model using the training dataset but memorises it.
* Distance metric: KNN operates on a chosen distance metric i.e. it measures the distance between two points in a multidimensional feature space.
* For a given point in the multidimensional feature space (a data instance), this model identifies the K closest data points in the feature space based on the chosen distance metric. “K” is the hyperparameter that determines the number of neighbours to consider.
* Majority Voting: For classification tasks, once the K-nearest neighbours are identified, KNN predicts the class of the given instance as the same class as the majority of the neighbours.
* Averaging: For Regression tasks, it predicts the output value for the data point by averaging the value of its K-nearest Neighbours.
* Apart from “K”, other hyperparameters like choice of distance matrix and how to handle ties in voting are crucial in KNN and can impact the performance.

Distance Metric

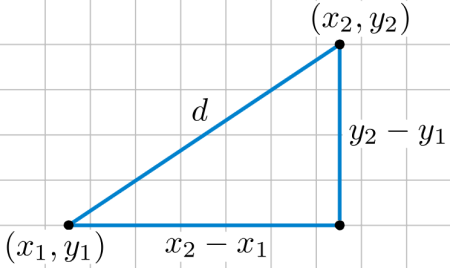
* Euclidean Distance: the Euclidean distance between two points in the Euclidean space is defined as the length of the line segment between two points.
* d =√[(x2 – x1)2 + (y2 – y1)2]

Where,

“d” is the Euclidean distance

(x1, y1) is the coordinate of the first point

(x2, y2) is the coordinate of the second point.



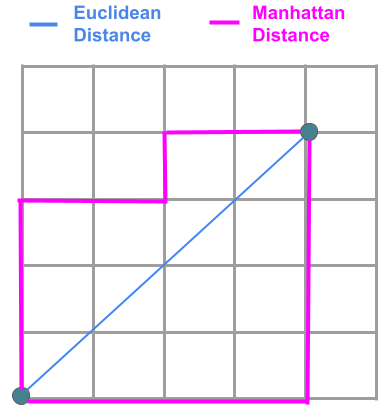
* Manhattan Distance: It measures the distance as the sum of the absolute differences between the coordinates along each dimension.
  + d = |x2 – x1| + |y2 – y1|

Where,

“d” is the Euclidean distance

(x1, y1) is the coordinate of the first point

(x2, y2) is the coordinate of the second point.

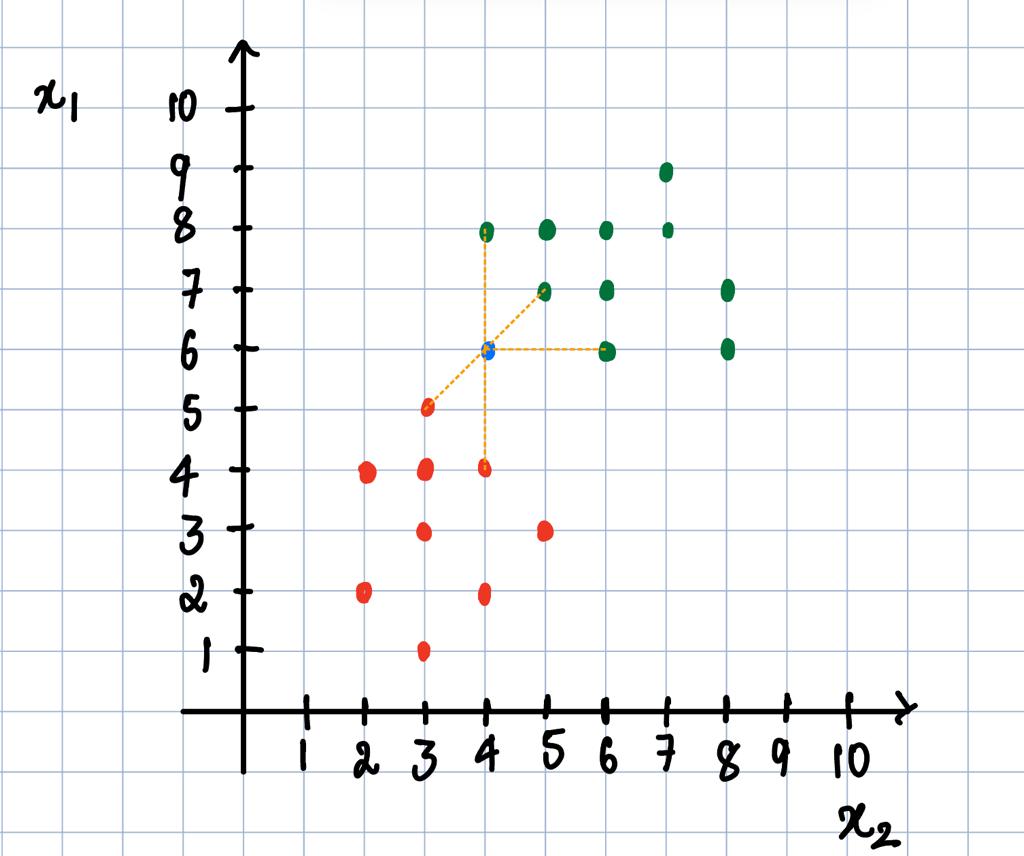


When to use which distance:

* Euclidien Distance:
  + Continuous Numerical Data: Euclidean distance is often preferred when dealing with continuous numerical data, such as:
    - Spatial Data: For instance, in geographical applications where latitude and longitude are used, Euclidean distance might be more appropriate as it measures straight-line distances.
    - Physical Sciences: Analysis involving measurements of physical quantities like length, weight, or temperature often benefits from the Euclidean distance as it represents direct spatial relationships.
* Manhattan Distance:
  + City Block Structure: Manhattan distance is suitable for situations where movements occur along grid-like structures or when features are not directly correlated but are influential independently. Use cases include:
    - Urban Planning: When analyzing city layouts, transportation, or infrastructure, Manhattan distance could represent more realistic path lengths.
    - Feature Engineering: When working with features that are not directly related or where the relationship between features is better represented by independent movements along axes.

To understand this better let’s go through an illustration,

* Let us consider a data set where we have only two features and them being x1 and x2.
* This dataset only consists of 2 classes i.e the green and the red class. The dataset is plotted on the graph as shown below.
* The blue point is the new point, here in this example we will be using the Euclidean distances from that point to the nearest K points, here in this case we are taking K = 5.



* As we can see from the figure, the blue point is closer to 3 green points and 2 red points and by majority voting we can say that the blue point belongs to the green class.
* Similarly if we had a continuous problem, instead of taking majority voting we would average the values of the 5 closest data points to the blue data point and the average value would be the predicted value.

Now that we have seen how the algorithm works, lets go over how the model is trained:

1. Data Storage:
   * During the "training" phase of KNN, the algorithm doesn't learn explicit model parameters.
   * Instead, KNN memorizes the entire training dataset, storing all the feature vectors and their corresponding class labels (in the case of classification) or target values (for regression).
2. Data Retrieval:

* When a new, unseen data point is to be classified or predicted using KNN, the algorithm uses the stored training data.

1. Prediction Phase:

* For each new data point, KNN:
  + Calculates the distance between the new data point and all the data points in the training set using a specified distance metric (e.g., Euclidean, Manhattan).
  + Identifies the K nearest neighbours to the new data point based on these distances.

1. Majority Voting (Classification) / Averaging (Regression):

* For classification tasks:
  + - KNN performs majority voting among the classes of the K nearest neighbours to determine the predicted class label for the new data point.
* For regression tasks:
  + - KNN calculates the average of the target values of the K nearest neighbours to predict the output value for the new data point.

**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 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.