ASSIGNMENT-16

**1.What are the key hyperparameters in KNN?**

In the **K-Nearest Neighbors (KNN)** algorithm, several key hyperparameters influence the model's performance. Here are the most important ones:

**1. n\_neighbors (Number of Neighbors)**

* **Description**: The number of nearest neighbors to consider when making a prediction.
* **Effect**:
  + A smaller value of n\_neighbors makes the model more sensitive to noise in the data, as each data point is influenced by fewer neighbors.
  + A larger value smooths the decision boundary, which may improve generalization but can also result in underfitting if n\_neighbors is too large.
* **Typical values**: Common values are 3, 5, 7, etc., but this is often chosen through cross-validation.

**2. weights**

* **Description**: Determines how much weight each neighbor has when making a prediction.
* **Values**:
  + 'uniform': All neighbors are weighted equally.
  + 'distance': Neighbors closer to the query point have a greater influence on the prediction.
  + **Custom function**: You can provide your own weight function, but this is less common.
* **Effect**: Using 'distance' can make the model more robust to outliers because it gives closer neighbors more influence.

**3. metric (Distance Metric)**

* **Description**: The distance metric used to measure the similarity between data points.
* **Common values**:
  + 'euclidean' (default): The straight-line distance between points (most common).
  + 'manhattan': The sum of the absolute differences between the coordinates.
  + 'minkowski': Generalized distance metric, where you can specify a power parameter p to control the distance behavior (Euclidean for p=2, Manhattan for p=1).
  + 'chebyshev': Maximum absolute difference in any coordinate.
  + 'cosine', 'hamming', 'jaccard': Used for specific types of data like text or categorical variables.
* **Effect**: Different metrics are suited to different types of data. For example, Euclidean is typically used for continuous numeric data, while cosine distance is often used for text data.

**4. algorithm**

* **Description**: The algorithm used to compute the nearest neighbors.
* **Values**:
  + 'auto': The algorithm will automatically choose the most efficient option based on the dataset.
  + 'ball\_tree': Efficient for large, high-dimensional datasets. It uses a binary tree structure.
  + 'kd\_tree': Efficient for low-dimensional datasets, uses a k-dimensional tree.
  + 'brute': Computes distances between all points, which is inefficient for large datasets but can work for smaller datasets.
* **Effect**: This affects the speed of the KNN algorithm, particularly for large datasets. The 'auto' option is often sufficient, but you may want to manually choose the algorithm based on the dataset size and structure.

**5. leaf\_size**

* **Description**: The size of the leaf nodes in the tree-based algorithms (ball\_tree and kd\_tree).
* **Effect**: Smaller values lead to more detailed trees but slower training and prediction. Larger values can speed up computation but may reduce accuracy.
* **Typical values**: Common values range from 10 to 50, depending on the dataset size.

**6. p (Power Parameter for Minkowski Distance)**

* **Description**: The power parameter p is used in the **Minkowski distance metric**.
* **Effect**:
  + When p=1, the metric becomes **Manhattan distance**.
  + When p=2, the metric becomes **Euclidean distance**.
  + For other values of p, it defines a generalized distance metric.
* **Typical values**: Usually p=1 (Manhattan) or p=2 (Euclidean).

**7. n\_jobs**

* **Description**: The number of CPU cores to use for computing the nearest neighbors.
* **Effect**: If you set n\_jobs=-1, the algorithm will use all available cores, speeding up computation, especially on larger datasets.
* **Typical values**: -1 (use all cores), 1 (use one core), or any specific number of cores.

**8. metric\_params**

* **Description**: A dictionary of additional parameters for the distance metric (e.g., for the **cosine** or **hamming** distance).
* **Effect**: Allows you to pass additional arguments to the metric function, which can help optimize the calculation depending on the data type or distance metric used.

**2. What distance metrics can be used in KNN?**

In **K-Nearest Neighbors (KNN)**, the choice of distance metric is crucial because it defines how similarity between data points is measured. Different metrics are suited for different types of data and problem contexts. Here are some common distance metrics used in KNN:

**1. Euclidean Distance (L2 Norm)**

* **Formula**:

d(p,q)=∑i=1n(pi−qi)2d(p, q)

* **Description**: Measures the straight-line (or "as-the-crow-flies") distance between two points in a Euclidean space.
* **Use case**: It is the most widely used distance metric and works well for continuous, numeric features where the magnitude and direction of features are important.
* **Example**: Used when comparing geographical locations, pixel values in images, or other measurements where the "closeness" is defined by straight-line distance.

**2. Manhattan Distance (L1 Norm)**

* **Formula**:

d(p,q)=∑i=1n∣pi−qi∣d(p, q)

* **Description**: Measures the sum of the absolute differences between the coordinates of two points. This is like walking along grid lines (similar to how a taxi would drive in a city grid, hence the name "taxicab distance").
* **Use case**: Works well for problems where features may have different scales or when the space is more grid-like, such as city block distances or when features represent counts.
* **Example**: Used in some problems where you care about absolute differences between coordinates (e.g., comparing vectors with sparse data, such as in text classification using word counts).

**3. Minkowski Distance**

* **Formula**:

d(p,q)=(∑i=1n∣pi−qi∣p)1pd(p, q)

* **Description**: A generalization of both Euclidean and Manhattan distance. When p=2, it becomes Euclidean distance; when p=1, it becomes Manhattan distance.
* **Use case**: Provides flexibility to define a distance metric between points by adjusting the parameter p.
* **Example**: Typically used when you need a flexible metric that can adapt to different datasets by choosing a different value for p.
* **Effect**:
  + p = 1 → Manhattan distance
  + p = 2 → Euclidean distance
  + Other values for p yield different distance measures.

**4. Chebyshev Distance**

* **Formula**:

d(p,q)=max⁡i∣pi−qi∣d(p, q) = \max\_{i} |p\_i - q\_i|d(p,q)=imax​∣pi​−qi​∣

* **Description**: Measures the maximum absolute difference between the coordinates of two points. This metric considers the greatest difference across all dimensions.
* **Use case**: Useful when you care more about the largest deviation in any feature, rather than the sum or Euclidean distance.
* **Example**: It can be used in certain machine learning problems where you want to focus on the worst-case (maximum) difference in feature values.

**5. Cosine Similarity (Cosine Distance)**

* **Formula**:

cosine\_similarity(p,q)=p⋅q∥p∥∥q∥\text{cosine\\_similarity}(p, q) = \frac{p \cdot q}{\|p\| \|q\|}cosine\_similarity(p,q)=∥p∥∥q∥p⋅q​

* **Description**: Measures the cosine of the angle between two vectors. It is a measure of similarity (not distance), so the result ranges from 0 (no similarity) to 1 (perfect similarity).
* **Use case**: Commonly used in text mining and natural language processing (NLP) for measuring document similarity, as it ignores the magnitude of vectors and only cares about the direction.
* **Example**: Used for comparing text documents where the features are represented by word counts or TF-IDF values.