K-Nearest Neighbor

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a popular and simple algorithm used for classification tasks in machine learning. It relies on the proximity of data points in the feature space to classify new instances.

K-Nearest Neighbors (KNN)

The basic idea behind KNN classification is to assign a class label to a new, unlabeled instance based on the majority vote of its k nearest neighbors in the feature space. In other words, if we have a labeled dataset, KNN determines the class of an unseen instance by examining the classes of its k nearest neighbors.

K-Nearest Neighbors (KNN)

When a new instance is presented for classification, KNN calculates the distance between the new instance and all instances in the training dataset using a distance metric, such as Euclidean distance.The k nearest neighbors of the new instance are then determined based on the shortest distances

K-Nearest Neighbors (KNN)

To calculate the distance between two instances in the K-Nearest Neighbors (KNN) algorithm, you can use a distance metric such as Euclidean distance : Euclidean Distance:

- Euclidean distance is the most widely used distance metric in KNN. It measures the straight-line distance between two points in a multi-dimensional feature space.
- The Euclidean distance between two instances, A and B, with n features can be calculated using the following formula:
  distance(A, B) = √((x1 - x2)^2 + (y1 - y2)^2 + ... + (xn - yn)^2)
- In this formula, (x1, x2, ..., xn) and (y1, y2, ..., yn) represent the feature values of instances A and B, respectively.

Example:

Instance	Age	BMI	Weight Category
1	25	20.5	Underweight
2	35	22.1	Underweight
3	28	24.8	Normal
4	40	28.3	Overweight
5	50	31.6	Obese

consider a simplified example with a dataset containing two features (Age and BMI) BMI = weight / (height)^2
and a class label indicating the weight category (underweight, overweight, obese, or normal). We'll use Euclidean distance to calculate the distance between instances.

Example Cont…

we have a new instance with Age = 32 and BMI = 26.
We want to determine its weight category using KNN with k=3.
Calculate Euclidean distance:

For each instance in the dataset, calculate the Euclidean distance from the new instance using the following formula:
distance = √((Age_new - Age_instance)^2 + (BMI_new - BMI_instance)^2)

Example Cont…

Let's calculate the distances for the new instance:

- Distance to Instance 1: sqrt((32 - 25)^2 + (26 - 20.5)^2) = 8.19
- Distance to Instance 2: sqrt((32 - 35)^2 + (26 - 22.1)^2) = 5.05
- Distance to Instance 3: sqrt((32 - 28)^2 + (26 - 24.8)^2) = 4.13
- Distance to Instance 4: sqrt((32 - 40)^2 + (26 - 28.3)^2) = 13.89
- Distance to Instance 5: sqrt((32 - 50)^2 + (26 - 31.6)^2) = 23.38

Example Cont…

Select the k nearest neighbors:

Sort the distances in ascending order and select the k instances with the shortest distances. In this case, let's consider k=3.

The five nearest neighbors are: Instance 3, Instance 2 and Instance 1.

Determine the majority class:

Among the k nearest neighbors, count the number of instances in each weight category.

In our case, the weight categories of the three nearest neighbors are:

Instance 3: Normal

Instance 2: Underweight

Instance 1: Underweight

Example Cont…

The majority class among the k nearest neighbors is “Underweight"

Predict the weight category:
Since the majority class is " Underweight " among the three nearest neighbors, we predict that the new instance with Age=32 and BMI=26 belongs to the " Underweight " weight category.

Advantages of KNN:

- Simplicity: KNN is a straightforward algorithm that is easy to understand and implement. It does not require complex mathematical or statistical models.
- No assumption about data distribution: KNN is a non-parametric algorithm, which means it does not make any assumptions about the underlying data distribution. It can handle data with arbitrary distributions.

Advantages of KNN:

- Interpretable: The predictions made by KNN can be easily interpretable. The class label or value is determined based on the majority vote or average of the nearest neighbors, which can be easily understood and explained.

- Effective with small training datasets: KNN can perform well when the training dataset is small. It can capture local patterns in the data and make predictions based on nearby instances.

Disadvantages of KNN:

- Computational complexity: The main drawback of KNN is its computational complexity. As the size of the training dataset grows, the algorithm needs to calculate distances between the new instance and all existing instances, which can be time-consuming and memory-intensive.

- If the data is noisy or contains irrelevant information, it can indeed impact the performance of the KNN algorithm. KNN relies on calculating distances between data points to make decisions, and if the data is dirty or contains noise, it can lead to increased errors in point classification.