**K-Nearest Neighbors**

The K-NN algorithm compares a new data entry to the values in a given data set (with different classes or categories).

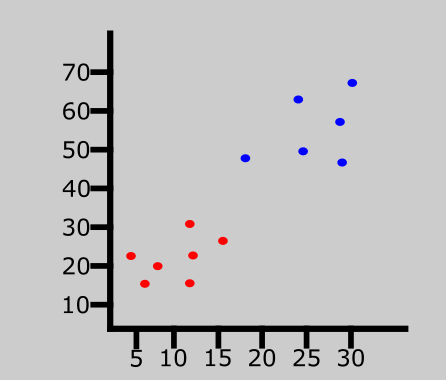
Based on its closeness or similarities in a given range (**K**) of neighbors, the algorithm assigns the new data to a class or category in the data set (training data).

**Steps:**

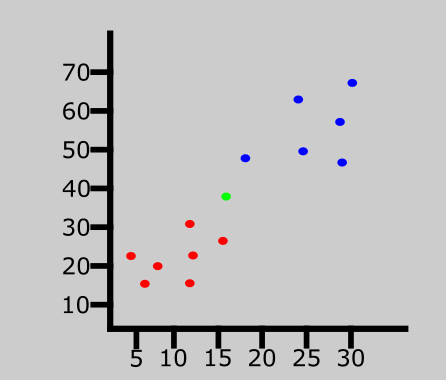
1. Assign a value to **K**.
2. Calculate the distance between the new data entry and all other existing data entries. Arrange them in ascending order.
3. Find the **K** nearest neighbors to the new entry based on the calculated distances.
4. Assign the new data entry to the majority class in the nearest neighbors.

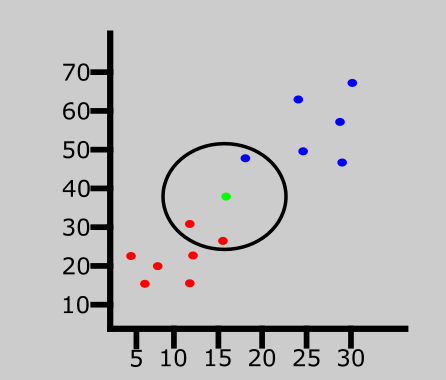
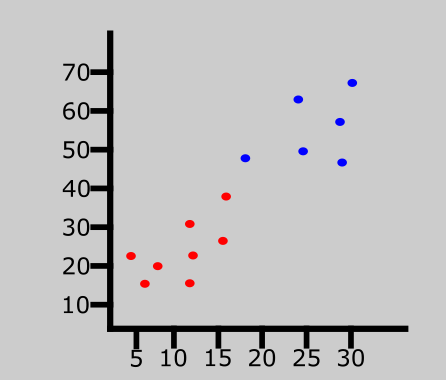
**Example:**

* Consider the diagram below:



* The graph above represents a data set consisting of two classes — red and blue.



* A new data entry has been introduced to the data set. This is represented by the green point in the graph above.
* We'll then assign a value to K which denotes the number of neighbors to consider before classifying the new data entry. Let's assume the value of K is 3.
* Since the value of K is 3, the algorithm will only consider the 3 nearest neighbors to the green point (new entry). This is represented in the graph above.
* Out of the 3 nearest neighbors in the diagram above, the majority class is red so the new entry will be assigned to that class.
* The last data entry has been classified as red.

**Example With Data Set**

We'll dive a bit deeper, and learn how to calculate the distance between a new entry and other existing values using the Euclidean distance formula.

The table represents our data set. We have two columns — Brightness and Saturation. Each row in the table has a class of either Red or Blue.

|  |  |  |
| --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |

Before we introduce a new data entry, let's assume the value of **K**is 5.

**How to Calculate Euclidean Distance in the K-Nearest Neighbors Algorithm:**

Here's the new data entry:

|  |  |  |
| --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS |
| 20 | 35 | ? |

We have a new entry but it doesn't have a class yet. To know its class, we have to calculate the distance from the new entry to other entries in the data set using the Euclidean distance formula.

Euclidean distance formula = √(X₂-X₁)²+(Y₂-Y₁)²

Where:

* X₂ = New entry's brightness (20).
* X₁= Existing entry's brightness.
* Y₂ = New entry's saturation (35).
* Y₁ = Existing entry's saturation.

**Distance #1**

For the first row, d1:

|  |  |  |
| --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS |
| 40 | 20 | Red |

d1 = √(20 - 40)² + (35 - 20)²

= √400 + 225

= √625

= 25

We now know the distance from the new data entry to the first entry in the table. Let's update the table.

|  |  |  |  |
| --- | --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | ? |
| 60 | 90 | Blue | ? |
| 10 | 25 | Red | ? |
| 70 | 70 | Blue | ? |
| 60 | 10 | Red | ? |
| 25 | 80 | Blue | ? |

**Distance #1**

For the second row, d2:

|  |  |  |  |
| --- | --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
| 50 | 50 | Blue | ? |

d2 = √(20 - 50)² + (35 - 50)²

= √900 + 225

= √1125

= 33.54

**Updated table:**

|  |  |  |  |
| --- | --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | 10 |
| 70 | 70 | Blue | 61.03 |
| 60 | 10 | Red | 47.17 |
| 25 | 80 | Blue | 45 |

Let's rearrange the distances in ascending order:

|  |  |  |  |
| --- | --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |
| 70 | 70 | Blue | 61.03 |
| 60 | 90 | Blue | 68.01 |

Since we chose 5 as the value of K, we'll only consider the first five rows. That is:

|  |  |  |  |
| --- | --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS | DISTANCE |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |

As you can see above, the majority class within the 5 nearest neighbors to the new entry is Red. Therefore, we'll classify the new entry as Red.

Here's the updated table:

|  |  |  |
| --- | --- | --- |
| BRIGHTNESS | SATURATION | CLASS |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |
| 20 | 35 | Red |

**How to Choose the Value of K in the K-NN Algorithm**

There is no particular way of choosing the value K, but here are some common conventions to keep in mind:

* Choosing a very low value will most likely lead to inaccurate predictions.
* The commonly used value of K is 5.
* Always use an odd number as the value of K.

**Advantages of K-NN Algorithm**

* It is simple to implement.
* No training is required before classification.

**Disadvantages of K-NN Algorithm**

* Can be cost-intensive when working with a large data set.
* A lot of memory is required for processing large data sets.
* Choosing the right value of K can be tricky.