

Lazy Learners (or Learning from Your Neighbors)

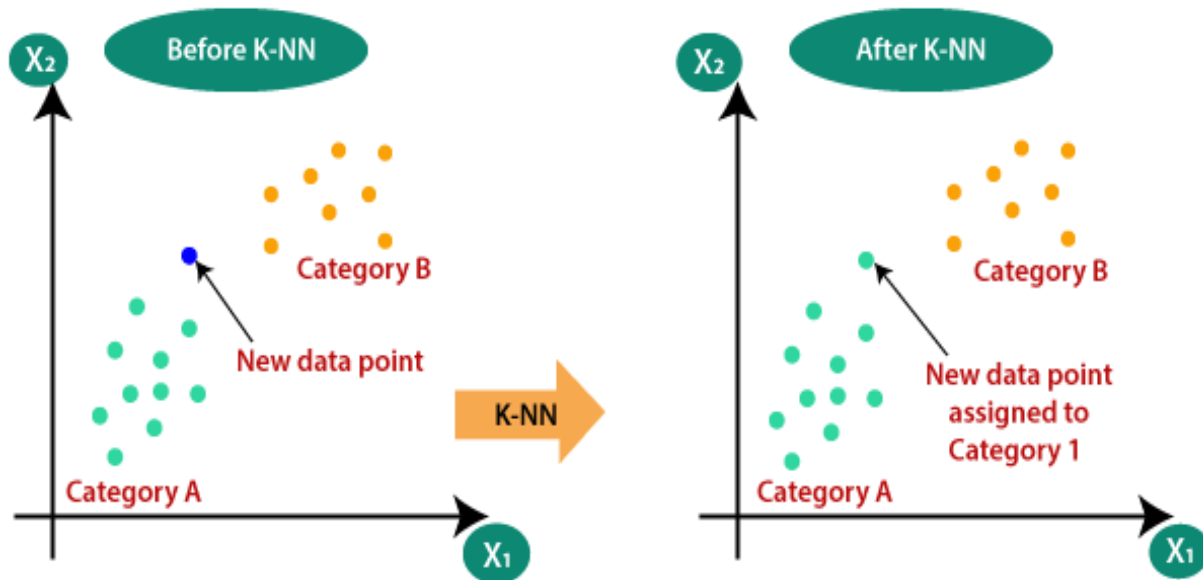
- ***Eager learners***, when given a set of training tuples, will construct a classification model before receiving new (e.g., test) tuples to classify.
- ***Lazy learners***, simply store training tuples (or perform minimal processing) and wait until a test tuple is encountered. Only when it sees the test tuple does it perform generalization to classify the tuple based on its similarity to the stored training tuples.
- Unlike eager learning methods, lazy learners do ***less work*** when a training tuple is presented and ***more work*** when making a classification or numeric prediction for the test tuple.
- The most popular **example** of Lazy learners is ***k-Nearest-Neighbor (k-NN) Classifiers***.

k -Nearest-Neighbor (k -NN)

Algorithm

- The K-Nearest Neighbors (**K-NN**) algorithm is a popular algorithm used mostly for solving classification problems.
- The K-NN algorithm stores all the available data (training data) and classifies a new data point (test data) based on the *similarity*.
- In other words, The K-NN algorithm compares a new data point to the data / 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).

Why do we need a K-NN Algorithm?



How k-Nearest-Neighbor (K-NN) Algorithm works?

The algorithm consists of the following steps.

Step -1: ***Choose the value K*** for the neighbors.

Step -2: ***Compute the distance*** between the new data point and all other existing data points. And arrange them in ascending order.

Step -3: ***Find the K nearest neighbors*** to the new data point based on the computed distances.

Step -4: ***Count the number of data points*** in each category among these k neighbors.

Step -5: ***Assign the new data point*** to that class / category for which the number of the neighbors is maximum.

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

There is ***no specific 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**.
- If the input data has ***more outliers or noise***, a higher value of **k** would be better.

How to Compute the Distance? (or) Distance Measures Used in KNN Algorithm?

- The K-NN algorithm enables us to classify or categorize the new data point by ***calculating the distance*** from other pre-existing data points.
- To calculate the distance we use any of the following are ***Distance Measures***.
 - ***Euclidean distance***
 - ***Manhattan distance***

Euclidean distance:

- Euclidean distance can be simply explained as the ***ordinary distance*** between two points.
- Mathematically it computes the ***root of squared differences*** between the coordinates between two objects.
- The ***Euclidean distance*** between two points or tuples, say, **$\mathbf{X} = (x_1, x_2, \dots, x_n)$** and **$\mathbf{Y} = (y_1, y_2, \dots, y_n)$** is

$$dist(X, Y) = dist(Y, X) = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2}$$

Manhattan distance:

- Manhattan Distance metric is generally used when we are interested in the total distance travelled by the object instead of the displacement.
- This distance is calculated by ***summing the absolute difference*** between the coordinates of the points in n-dimensions.
- The ***Manhattan distance*** between two points or tuples, say, $\mathbf{X} = (x_1, x_2, \dots, x_n)$ and $\mathbf{Y} = (y_1, y_2, \dots, y_n)$ is

$$\text{dist}(\mathbf{X}, \mathbf{Y}) = \text{dist}(\mathbf{Y}, \mathbf{X}) = \sum_{i=1}^n |X_i - Y_i|$$

Examples to calculate distances:

Let $X = (1, 2)$ and $Y = (3, 5)$ represent two data points.

- The ***Euclidean distance*** between the two data points is

$$\begin{aligned} \text{dist}(X, Y) &= \sqrt{(1 - 3)^2 + (2 - 5)^2} \\ &= \sqrt{(-2)^2 + (-3)^2} \\ &= \sqrt{4 + 9} \\ &= \sqrt{13} = 3.61 \end{aligned}$$

- The ***Manhattan distance*** between the two data points is

$$\begin{aligned} \text{dist}(X, Y) &= |1 - 3| + |2 - 5| \\ &= 2 + 3 = 5 \end{aligned}$$

Example: Consider the following training dataset,

Height (in CM)	Weight(in KG)	Class
189	87	Normal
149	61	Overweight
189	104	Overweight
195	81	Normal
155	51	Normal
191	79	Normal
190	95	Overweight
157	56	Normal
185	76	Normal
161	72	Overweight
164	75	Overweight
149	66	Overweight

And apply **K-NN algorithm** to find the ***class label*** for the following test tuple.

Height (in CM)	Weight(in KG)	Class
148	60	?

Solution:

- In the given dataset, we have ***two columns*** – *Height* (in CM) and *Weight* (in KG). Each row in the dataset has a class of either ***Normal*** or ***Overweight***.

Step-1:

- To start working with K-NN algorithm, first we need to choose the value for **K**.
- Let's assume the value of **K** is **5**.

Step-2:

- In this, we need to calculate the distance between ***new (test) tuple*** and all other tuples in the ***existing (training)*** dataset.
- To calculate the distance we choose ***Euclidean distance measure***.

- Now, we need to calculate the distance from this new tuple to all other data tuples.

Here's the formula:

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Where:

- X_2 = New entry's Height (148).
- X_1 = Existing entry's Height.
- Y_2 = New entry's Weight (60).
- Y_1 = Existing entry's Weight.

Distance#1

For the first row, **d1**:

Height (in CM)	Weight(in KG)	Class
189	87	Normal

$$\mathbf{d1} = \sqrt{(148 - 189)^2 + (60 - 87)^2}$$

$$= \sqrt{1681 + 729}$$

$$= \sqrt{2410}$$

$$= 49.1$$

Now, we know the distance from the new data entry/tuple. Let's update the table/dataset.

Height (in CM)	Weight(in KG)	Class	Distance
189	87	Normal	49.09

Distance#2

For the first row, **d2**:

Height (in CM)	Weight(in KG)	Class
149	61	Overweight

$$\begin{aligned}\mathbf{d2} &= \sqrt{(148 - 149)^2 + (60 - 61)^2} \\ &= \sqrt{2} \\ &= 1.41\end{aligned}$$

Now, we know the distance from the new data entry/tuple. Let's update the table/dataset.

Height (in CM)	Weight(in KG)	Class	Distance
149	61	Overweight	1.41

Distance#3

For the first row, **d3**:

Height (in CM)	Weight(in KG)	Class
189	104	Overweight

$$\begin{aligned} \mathbf{d3} &= \sqrt{(148 - 189)^2 + (60 - 104)^2} \\ &= \sqrt{1681 + 1936} \\ &= 60.14 \end{aligned}$$

Now, we know the distance from the new data entry/tuple. Let's update the table/dataset.

Height (in CM)	Weight(in KG)	Class	Distance
149	61	Overweight	60.14

- After ***calculating the distances*** from all other existing data tuples, the updated table or dataset is

Height(in CM)	Weight(in KG)	Class	Distance
189	87	Normal	49.09
149	61	Overweight	1.41
189	104	Overweight	60.14
195	81	Normal	51.48
155	51	Normal	11.40
191	79	Normal	47.01
190	95	Overweight	54.67
157	56	Normal	9.85
185	76	Normal	40.31
161	72	Overweight	17.69
164	75	Overweight	21.93
149	66	Overweight	6.08

- Let's *rearrange* the distances in *ascending order*:

Height(in CM)	Weight(in KG)	Class	Distance
149	61	Overweight	1.41
149	66	Overweight	6.08
157	56	Normal	9.85
155	51	Normal	11.40
161	72	Overweight	17.69
164	75	Overweight	21.93
185	76	Normal	40.31
191	79	Normal	47.01
189	87	Normal	49.09
195	81	Normal	51.48
190	95	Overweight	54.67
189	104	Overweight	60.14

Step -3:

- Since we chose **5** as the value of **K**, we'll only consider the first ***five*** rows.

Height(in CM)	Weight(in KG)	Class	Distance
149	61	Overweight	1.41
149	66	Overweight	6.08
157	56	Normal	9.85
155	51	Normal	11.40
161	72	Overweight	17.69

Step -4:

- In the above 5 nearest neighbors, **3** neighbors have ***Overweight*** as a class and **2** neighbours have ***Normal*** as a class.

Step -5:

- As you can see above, the class ***Overweight*** has maximum number of neighbors, Therefore, we'll classify the new entry/tuple as ***Overweight***.

The updated tuple is

Height (in CM)	Weight(in KG)	Class
148	60	Overweight

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.

Exercise:

Consider the following training dataset,

Brightness	Saturation	Class
40	20	Red
50	50	Blue
60	90	Blue
10	25	Red
70	70	Blue
60	10	Red
25	80	Blue

And apply **K-NN algorithm** to find the *class label* for the following test tuple.

Brightness	Saturation	Class
20	36	?