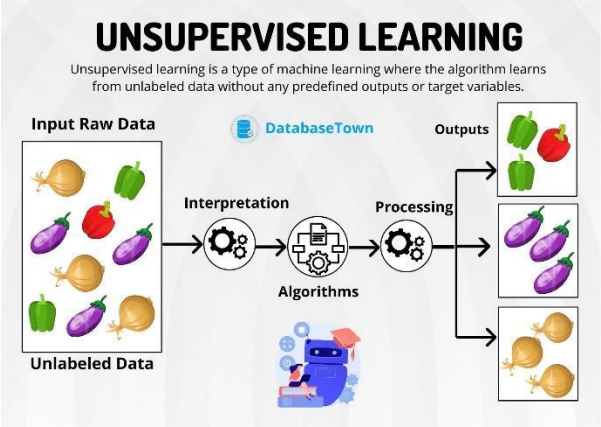
Chap 3: Unsupervised Learning

What is Unsupervised Learning?

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision.

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format**.



Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

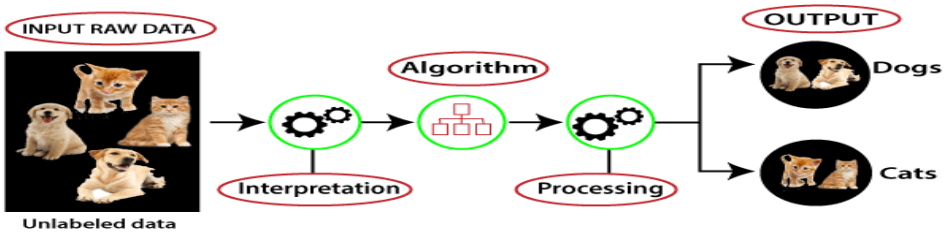
Why use Unsupervised Learning?

Below are some main reasons which describe the importance of Unsupervised Learning:

- Unsupervised learning is helpful for finding useful insights from the data.
- Unsupervised learning is much similar as a human learns to think by their own experiences, which makes it closer to the real AI.
- Unsupervised learning works on unlabeled and uncategorized data which make unsupervised learning more important.
- In real-world, we do not always have input data with the corresponding output so to solve such cases, we need unsupervised learning.

Working of Unsupervised Learning

Working of unsupervised learning can be understood by the below diagram:

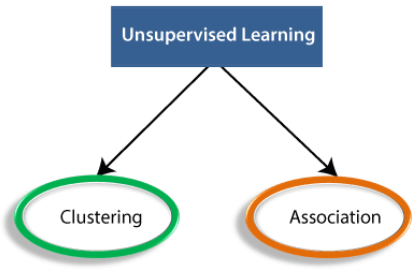


Here, we have taken an unlabeled input data, which means it is not categorized and corresponding outputs are also not given. Now, this unlabeled input data is fed to the machine learning model in order to train it. Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.

Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:



- **Clustering**: Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another

group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.

- **Association**: An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Unsupervised Learning algorithms:

Below is the list of some popular unsupervised learning algorithms:

- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchal clustering**
- **Anomaly detection**
- **Neural Networks**
- **Principle Component Analysis**
- **Independent Component Analysis**
- **Apriori algorithm**
- **Singular value decomposition**

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Differences between Supervised and Unsupervised Learning

Supervised Learning	**Unsupervised Learning**
Supervised Learning can be used for 2 different types of problems i.e. regression and classification	Unsupervised Learning can be used for 2 different types of problems i.e. clustering and association.
Input Data is provided to the model along with the output in the Supervised Learning.	Only input data is provided in Unsupervised Learning.
Output is predicted by the Supervised Learning.	Hidden patterns in the data can be found using the unsupervised learning model.

Labeled data is used to train supervised learning algorithms.	Unlabeled data is used to train unsupervised learning algorithms.
Accurate results are produced using a supervised learning model.	The accuracy of results produced are less in unsupervised learning models.
Training the model to predict output when a new data is provided is the objective of Supervised Learning.	Finding useful insights, hidden patterns from the unknown dataset is the objective of the unsupervised learning.
Supervised Learning includes various algorithms such as Bayesian Logic, Decision Tree, Logistic Regression, Linear Regression, Multi-class Classification, Support Vector Machine etc.	Unsupervised Learning includes various algorithms like KNN, Apriori Algorithm, and Clustering.
Some of the applications of Supervised Learning are Spam detection, handwriting detection, pattern recognition, speech recognition etc.	Some of the applications of Unsupervised Learning are detecting fraudulent transactions, data preprocessing etc.

Clustering in Machine Learning

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as ***"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."***

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique is commonly used for **statistical data analysis.**

Example: Let's understand the clustering technique with the real-world example of Mall: When we visit any shopping mall, we can observe that the things with similar usage are grouped together. Such as the t-shirts are grouped in one section, and trousers are at other sections, similarly, at vegetable sections, apples, bananas, Mangoes, etc., are grouped in separate sections, so that we can easily find out the things. The clustering technique also works in the same way. Other examples of clustering are grouping documents according to the topic.

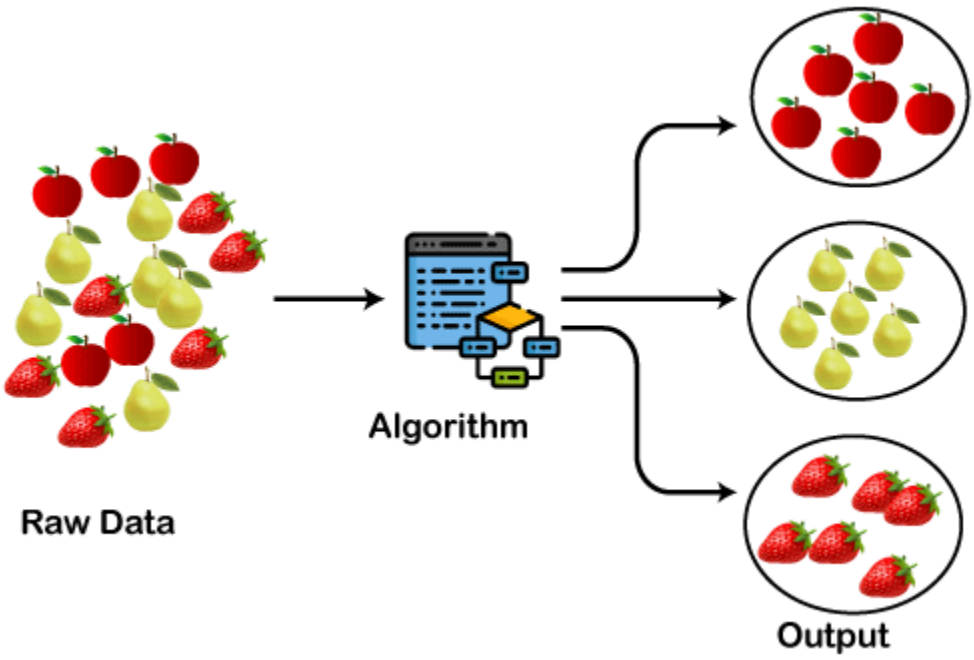
Applications: The clustering technique can be widely used in various tasks. Some most common uses of this technique are:

- Market Segmentation
- Statistical data analysis

- Social network analysis
- Image segmentation
- Anomaly detection, etc.

Apart from these general usages, it is used by the **Amazon** in its recommendation system to provide the recommendations as per the past search of products. **Netflix** also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types of Clustering Methods

The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

K-Means Clustering Algorithm:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

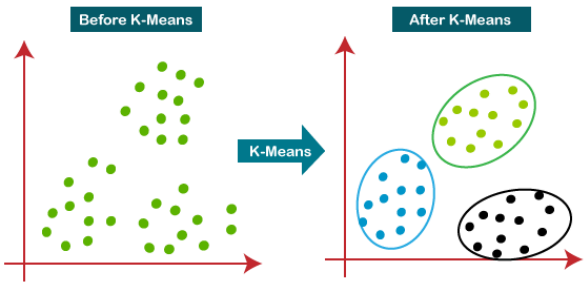
The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS= \sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{Pi \text{ in CLuster3}} distance(P_i\ C_3)^2$$

In the above formula of WCSS,

$\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

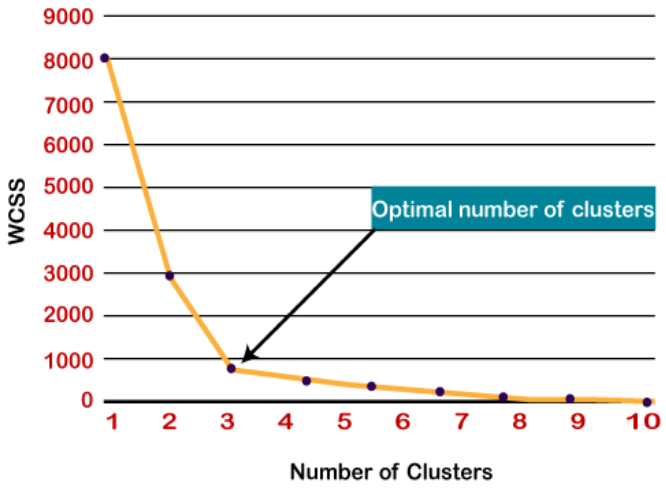
- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Identifying the Elbow Point: As we increase kkk, the WCSS typically decreases because we're creating more clusters, which tend to capture more data variations. However, there comes a point

where adding more clusters results in only a marginal decrease in WCSS. This is where we observe an "elbow" shape in the graph.

- **Before the elbow**: Increasing kkk significantly reduces WCSS, indicating that new clusters effectively capture more of the data's variability.
- **After the elbow**: Adding more clusters results in a minimal reduction in WCSS, suggesting that these extra clusters may not be necessary and could lead to overfitting.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Example:

Apply K(=2)-Means algorithm over the data (185, 72), (170, 56), (168, 60), (179,68), (182,72), (188,77) up to two iterations and show the clusters. Initially choose first two objects as initial centroids.

Solution:
Given, number of clusters to be created (K) = 2 say c1 and c2, number of iterations = 2 and The given data points can be represented in tabular form as:

Instance	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

also, first two objects as initial centroids:
Centroid for first cluster c1 = (185, 72)
Centroid for second cluster c2 = (170, 56)

Iteration 1: Now calculating similarity by using *Euclidean distance* measure as:

$d(c1, 3) = \sqrt{(185-168)^2 + (72-60)^2} = \sqrt{(17)^2 + (12)^2} = \sqrt{289 + 144} = \sqrt{433}$

$d(c2, 3) = \sqrt{(170-168)^2 + (56-60)^2} = \sqrt{(2)^2 + (-4)^2} = \sqrt{4 + 16} = \sqrt{20}$

Here, d(c2, 3) < d(c1, 3)

So, data point 3 belongs to c2.

$d(c1, 4) = \sqrt{(185-179)^2 + (72-68)^2} = \sqrt{(6)^2 + (4)^2} = \sqrt{36 + 16} = \sqrt{52}$

$d(c2, 4) = \sqrt{(170-179)^2 + (56-68)^2} = \sqrt{(-9)^2 + (-12)^2} = \sqrt{81 + 144} = \sqrt{225}$

Here, d(c1, 4) < d(c2, 4)

So, data point 4 belongs to c1.

$d(c1, 5) = \sqrt{(185-182)^2 + (72-72)^2} = \sqrt{(3)^2 + (0)^2} = \sqrt{9}$

$d(c2, 5) = \sqrt{(170-182)^2 + (56-72)^2} = \sqrt{(-12)^2 + (-16)^2} = \sqrt{144 + 256} = \sqrt{400}$

Here, d(c1, 5) < d(c2, 5)

So, data point 5 belongs to c1.

$d(c1, 6) = \sqrt{(185-188)^2 + (72-77)^2} = \sqrt{(-3)^2 + (-5)^2} = \sqrt{9 + 25} = \sqrt{34}$

$d(c2, 6) = \sqrt{(170-188)^2 + (56-77)^2} = \sqrt{(-18)^2 + (-21)^2} = \sqrt{324 + 441} = \sqrt{765}$

Here, d(c1, 6) < d(c2, 6)

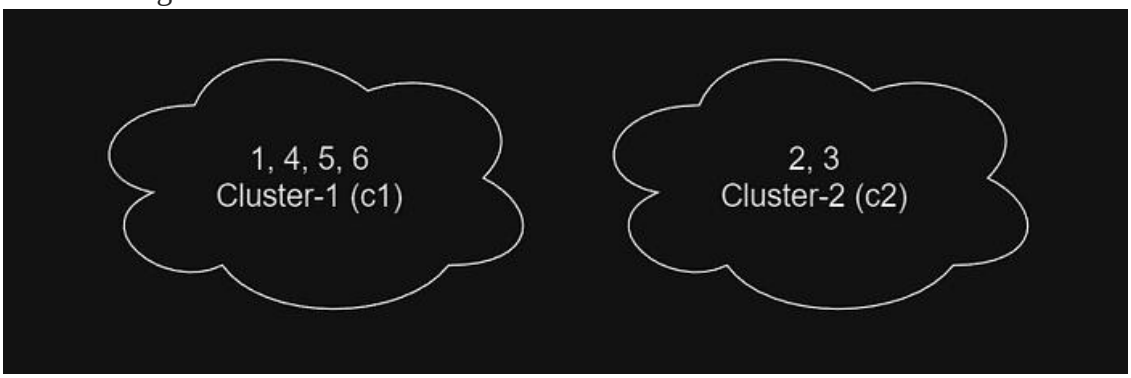
So, data point 6 belongs to c1.

Representing above information in tabular form:

Instance	X	Y	Distance(C1)	Distance(C2)	Cluster
1	185	72			c1
2	170	56			c2
3	168	60	√433	√20	c2
4	179	68	√52	√225	c1
5	182	72	√9	√400	c1
6	188	77	√34	√765	c1

Distance of each data points from cluster centroids

The resulting cluster after first iteration is:



Iteration 2: Now calculating centroid for each cluster: Calculating centroid as mean of data points

$$Centroid\ for\ first\ cluster\ c1 = \left(\frac{185+179+182+188}{4}, \frac{72+68+72+77}{4}\right) = (183.5, 72.25)$$

$$Centroid\ for\ second\ cluster\ c2 = \left(\frac{170+168}{2}, \frac{56+60}{2}\right) = (169, 58)$$

Now, again calculating similarity: Distance calculation between data points and centroids

$$d(c1, 1) = \sqrt{(183.5 - 185)^2 + (72.25 - 72)^2} = 1.5207$$
$$d(c2, 1) = \sqrt{(169 - 185)^2 + (58 - 72)^2} = 21.2603$$

Here, d (c1, 1) < d (c2, 1)
So, data point 1 belongs to c1.

$$d(c1, 2) = \sqrt{(183.5 - 170)^2 + (72.25 - 56)^2} = 21.1261$$
$$d(c2, 2) = \sqrt{(169 - 170)^2 + (58 - 56)^2} = 2.2361$$

Here, d (c2, 2) < d (c1, 2)
So, data point 2 belongs to c2.

$$d(c1, 3) = \sqrt{(183.5 - 168)^2 + (72.25 - 60)^2} = 19.7563$$
$$d(c2, 3) = \sqrt{(169 - 168)^2 + (58 - 60)^2} = 2.2361$$

Here, d (c2, 3) < d (c1, 3)
So, data point 3 belongs to c2.

$$d(c1, 4) = \sqrt{(183.5 - 179)^2 + (72.25 - 68)^2} = 6.1897$$
$$d(c2, 4) = \sqrt{(169 - 179)^2 + (58 - 68)^2} = 14.1421$$

Here, d (c1, 4) < d (c2, 4)
So, data point 4 belongs to c1.

$$d(c1, 5) = \sqrt{(183.5 - 182)^2 + (72.25 - 72)^2} = 1.5207$$
$$d(c2, 5) = \sqrt{(169 - 182)^2 + (58 - 72)^2} = 19.1050$$

Here, d (c1, 5) < d (c2, 5)
So, data point 5 belongs to c1.

$$d(c1, 6) = \sqrt{(183.5 - 188)^2 + (72.25 - 77)^2} = 6.5431$$
$$d(c2, 6) = \sqrt{(169 - 188)^2 + (58 - 77)^2} = 26.8701$$

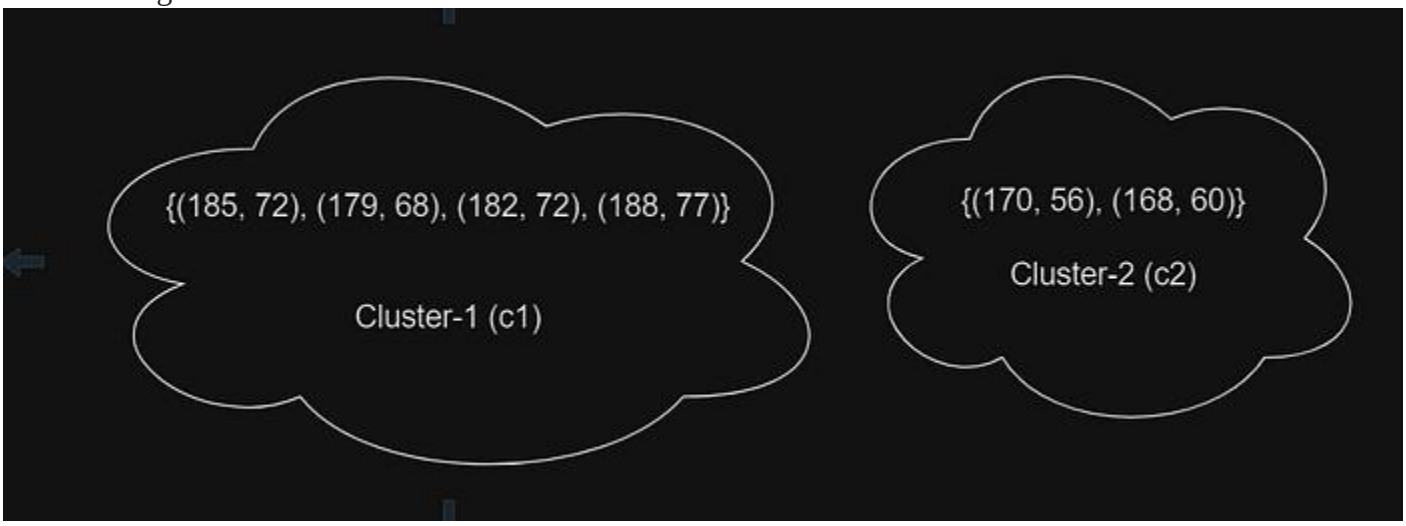
Here, d (c1, 6) < d (c2, 6)
So, data point 6 belongs to c1.

Representing above information in tabular form.

Instance	X	Y	Distance(C1)	Distance(C2)	Cluster
1	185	72	1.5207	21.2603	c1
2	170	56	21.1261	2.2361	c2
3	168	60	19.7563	2.2361	c2
4	179	68	6.1897	14.1421	c1
5	182	72	1.5207	19.105	c1
6	188	77	6.5431	26.8701	c1

Distance of each data points from cluster centroids

The resulting cluster after second iteration is:

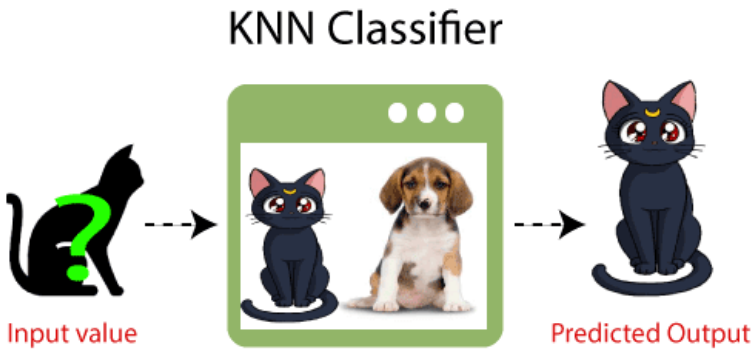


Data points cluster

K-Nearest Neighbor(KNN) Algorithm for Machine Learning

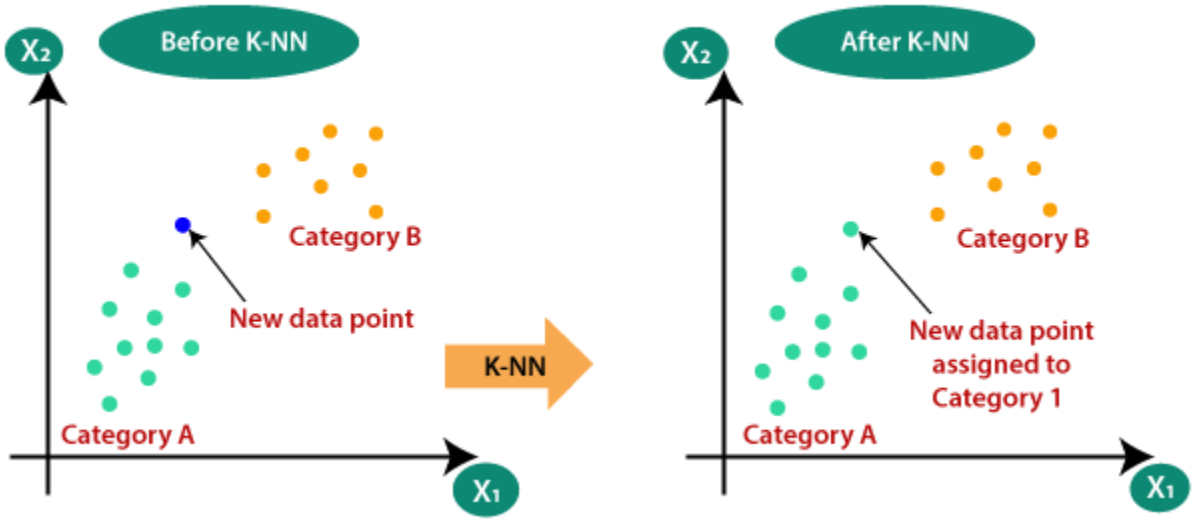
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

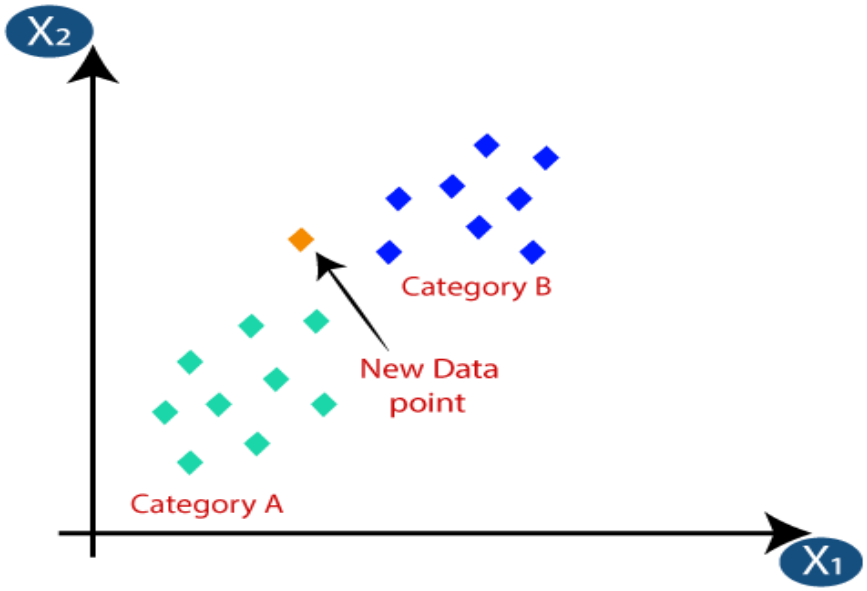


How does K-NN work?

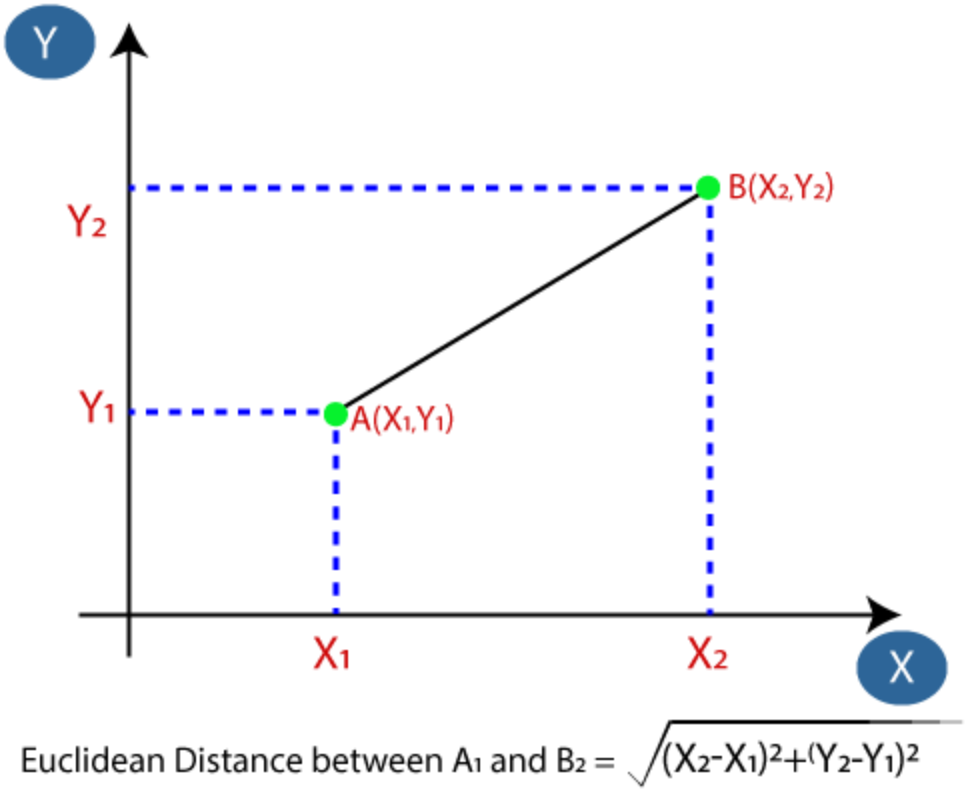
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

K-Nearest Neighbors Classifiers and Model Example With Data Set:

Refer url for same problem solved in detail: KNN Algorithm – K-Nearest Neighbors Classifiers and Model Example

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

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

Here's the formula: $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

Where:

- X_2 = New entry's brightness (20).
- X_1= Existing entry's brightness.
- Y_2 = New entry's saturation (35).
- Y_1 = Existing entry's saturation.

Let's do the calculation together. I'll calculate the first three.

Distance #1

For the first row, d1:

Brightness	Saturation	Class
40	20	Red

$d1 = \sqrt{(20 - 40)^2 + (35 - 20)^2}$
$= \sqrt{400 + 225}$
$= \sqrt{625}$
$= 25$

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 #2

For the second row, d2:

Brightness	Saturation	Class	Distance
50	50	Blue	?

$d2 = \sqrt{(20 - 50)^2 + (35 - 50)^2}$
$= \sqrt{900 + 225}$
$= \sqrt{1125}$
$= 33.54$

Here's the table with the updated distance:

Brightness	Saturation	Class	Distance
40	20	Red	25
50	50	Blue	33.54
60	90	Blue	?

Brightness	Saturation	Class	Distance
10	25	Red	?
70	70	Blue	?
60	10	Red	?
25	80	Blue	?

Distance #3

For the third row, d3:

Brightness	Saturation	Class	Distance
60	90	Blue	?

$d2 = \sqrt{(20 - 60)^2 + (35 - 90)^2}$
$= \sqrt{1600 + 3025}$
$= \sqrt{4625}$
$= 68.01$

Updated table:

Brightness	Saturation	Class	Distance
40	20	Red	25
50	50	Blue	33.54
60	90	Blue	68.01
10	25	Red	?
70	70	Blue	?
60	10	Red	?

Brightness	Saturation	Class	Distance
25	80	Blue	?

At this point, you should understand how the calculation works. Attempt to calculate the distance for the last four rows.

Here's what the table will look like after all the distances have been calculated:

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

Brightness	Saturation	Class	Distance
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

Brightness	Saturation	Class
60	10	Red
25	80	Blue
20	35	Red

Random Initialization Trap in K-Means

https://www.geeksforgeeks.org/ml-random-intialization-trap-in-k-means/

Mini Batch K-means clustering algorithm. // example explain

https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm /// short theory