# Machine Learning

**(IV CSE – I SEM.)**

**A.Y.: 2022 – 2023**

UNIT-IV

**Unsupervised Learning**

Introduction to clustering, K-means clustering, K-Mode Clustering, Distance based clustering, Clustering around mediods, Silhouettes, Hierarchical Clustering.

## Unsupervised Machine Learning:

### Introduction to clustering

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 learningnew 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.

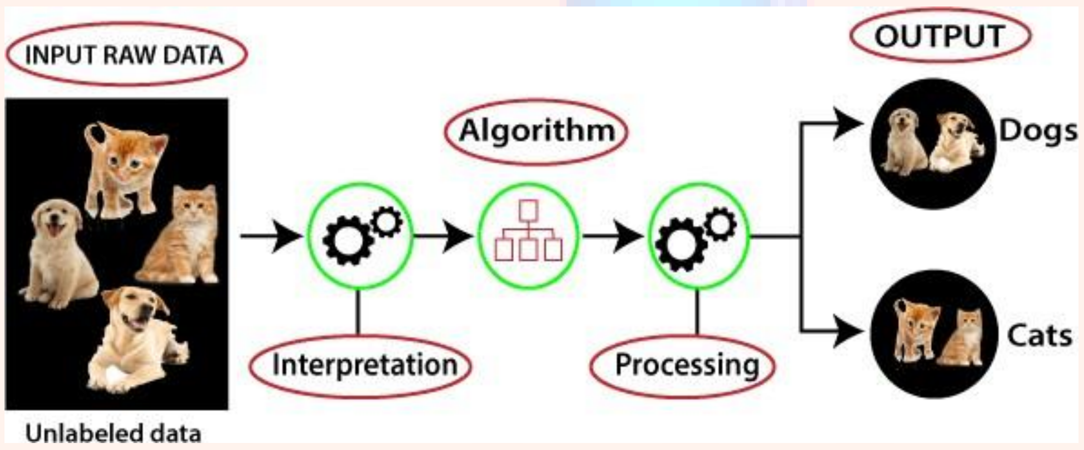
## Why use Unsupervised Learning?

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.

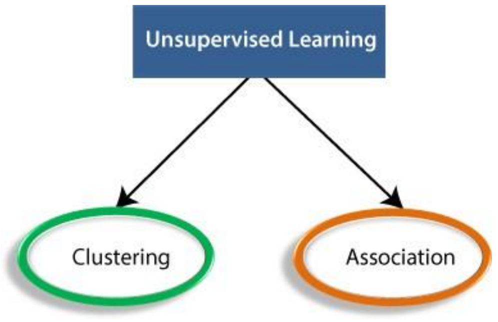
## Working of Unsupervised Learning

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
- 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 learningbecause, 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 labeleddata.

**Disadvantages of Unsupervised Learning**

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not havecorresponding output.

- The result of the unsupervised learning algorithm might be less accurate as input data is notlabeled, and algorithms do not know the exact output in advance.

| Supervised Learning | Unsupervised Learning |
|---|---|
| Supervised learning algorithms are trained using labeled data. | Unsupervised learning algorithms are trained using unlabeled data. |
| Supervised learning model takes direct feedback to check if it is predictingcorrect output or not. | Unsupervised learning model does not take any feedback. |
| Supervised learning model predicts the output. | Unsupervised learning model finds the hidden patterns in data. |
| In supervised learning, input data is provided to the model along with theoutput. | In unsupervised learning, only input data is provided to the model. |
| The goal of supervised learning is to train the model so that it can predictthe output when it is given new data. | The goal of unsupervised learning is to find the hidden patterns anduseful insights from the unknown dataset. |
| Supervised learning needs supervision to train the model. | Unsupervised learning does not need any supervision to train themodel. |
| Supervised learning can be categorized in **Classification** and **Regression** problems. | Unsupervised Learning can be classified in **Clustering** and **Associations** problems. |
| Supervised learning can be used for those cases where we know the inputas well as corresponding outputs. | Unsupervised learning can be used for those cases where we haveonly input data and no corresponding output data. |
| Supervised learning model produces an accurate result. | Unsupervised learning model may give less accurate result ascompared to supervised learning. |
| Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correctoutput. | Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routinethings by his experiences. |
| It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decisiontree, Bayesian Logic, etc. | It includes various algorithms such as Clustering, KNN, and Apriorialgorithm. |

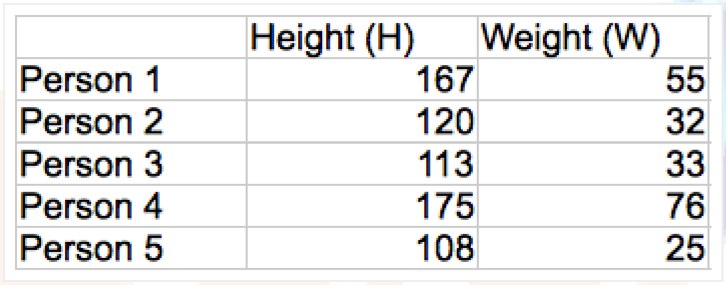
## K-Mean Clustering

**k-means clustering algorithm**

One of the most used clustering algorithm is *k-means*. It allows to group the data according to the existing similarities among them in *k* clusters, given as input to the algorithm. I’ll start with a simple example.

Let’s imagine we have 5 objects (say 5 people) and for each of them we know two features (height and weight). We want to group them into *k=2* clusters.

Our dataset will look like this:



| | Height (H) | Weight (W) |
|---|---|---|
| Person 1 | 167 | 55 |
| Person 2 | 120 | 32 |
| Person 3 | 113 | 33 |
| Person 4 | 175 | 76 |
| Person 5 | 108 | 25 |

First of all, we have to initialize the value of the centroids for our clusters. For instance, let’s choosePerson 2 and Person 3 as the two centroids *c1* and *c2*, so that *c1=(120,32)* and *c2=(113,33).*

Now we compute the euclidian distance between each of the two centroids and each point in the data.If you did all the calculations, you should have come up with the following numbers:

| | **Distance of object from *c1*** | **Distance of object from *c2*** |
|---|---|---|
| Person 1 | 52.3 | 58.3 |
| Person 2 | 0 | 7.1 |
| Person 3 | 7.1 | 0 |
| Person 4 | 70.4 | 75.4 |
| Person 5 | 13.9 | 9.4 |

At this point, we will assign each object to the cluster it is closer to (that is taking the minimum betweenthe two computed distances for each object).

We can then arrange the points as follows:

Person 1 → cluster 1
Person 2 → cluster 1
Person 3 → cluster 2
Person
4 →
cluster
1Person
5→
cluster
2

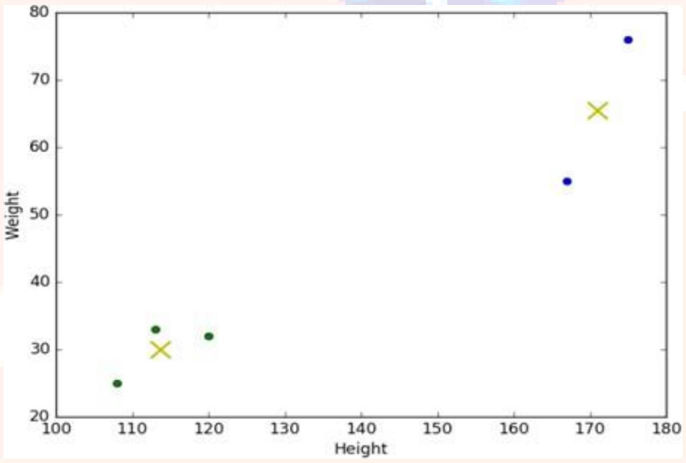
Let’s iterate, which means to redefine the centroids by calculating the mean of the members of each ofthe two clusters.

So *c’1* = ((167+120+175)/3, (55+32+76)/3) = (154, 54.3) and *c’2* = ((113+108)/2, (33+25)/2) = (110.5, 29)

Then, we calculate the distances again and re-assign the points to the new centroids.

We repeat this process until the centroids don’t move anymore (or the difference between them isunder a certain small threshold).

In our case, the result we get is given in the figure below. You can see the two different clusters labelledwith two different colours and the position of the centroids, given by the crosses.



**How to apply k-means?**

As you probably already know, I’m using Python libraries to analyze my data. The *k-means* algorithm is implemented in the *scikit-learn* package. To use it, you will just need the following line in your script:

**What if our data is… non-numerical?**

At this point, you will maybe have noticed something. The basic concept of *k-means* stands on mathematical calculations (means, euclidian distances). But what if our data is non-numerical or, in other words, *categorical*? Imagine, for instance, to have the ID code and date of birth of the five people of the previous example, instead of their heights and weights.

We could think of transforming our categorical values in numerical values and eventually apply *k- means*. But beware: *k-means* uses numerical distances, so it could consider close two really distant objects that merely have been assigned two close numbers.

*k-modes* is an extension of *k-means*. Instead of distances it uses *dissimilarities* (that is, quantification of the total mismatches between two objects: the smaller this number, the more similar the two objects). And instead of means, it uses *modes*. A mode is a vector of elements that minimizes the dissimilarities between the vector itself and each object of the data. We will have as many modes as the number of clusters we required, since they act as centroids.

## KModes Clustering Algorithm for Categorical data

Introduction:

Clustering is an unsupervised learning method whose task is to divide the population or data points into a number of groups, such that data points in a group are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects based on similarity and dissimilarity between them.

**KModes clustering** is one of the unsupervised Machine Learning algorithms that is used to cluster **categorical variables.**

You might be wondering, why KModes when we already have KMeans.

KMeans uses mathematical measures (distance) to cluster continuous data. The lesser the distance, the more similar our data points are. Centroids are updated by Means.
But for categorical data points, we cannot calculate the distance. So we go for KModes algorithm. It uses the dissimilarities(total mismatches) between the data points. The lesser the dissimilarities the more similar our data points are. It uses Modes instead of means.

- **How does the KModes algorithm work?**

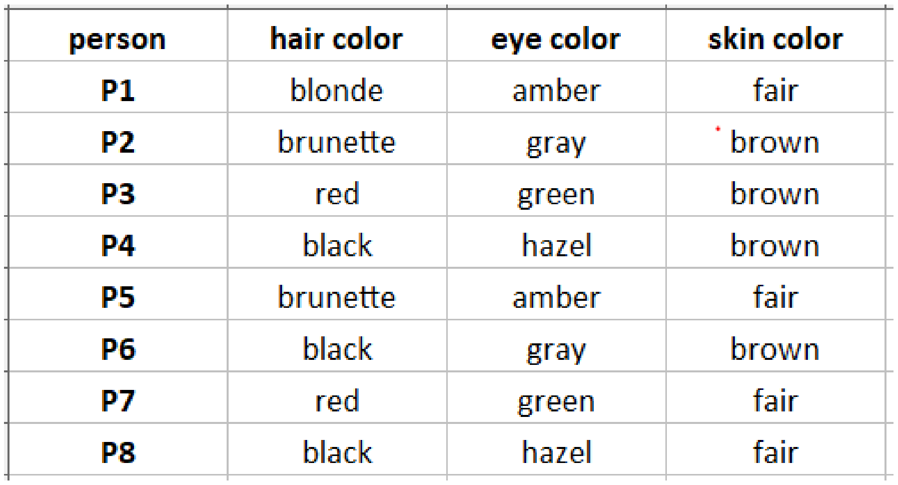
Unlike Hierarchical clustering methods, we need to upfront specify the K.

1. Pick K observations at random and use them as leaders/clusters
2. Calculate the dissimilarities and assign each observation to its closest cluster
3. Define new modes for the clusters
4. Repeat 2–3 steps until there are is no re-assignment required

I hope you got the basic idea of the KModes algorithm by now. So let us quickly take an example to illustrate the working step by step.

**Example:** Imagine we have a dataset that has the information about hair color, eye color, and skin color of persons. We aim to group them based on the available information(maybe we want to suggest some styling ideas)

Hair color, eye color, and skin color are all categorical variables. Below is how our dataset looks like.

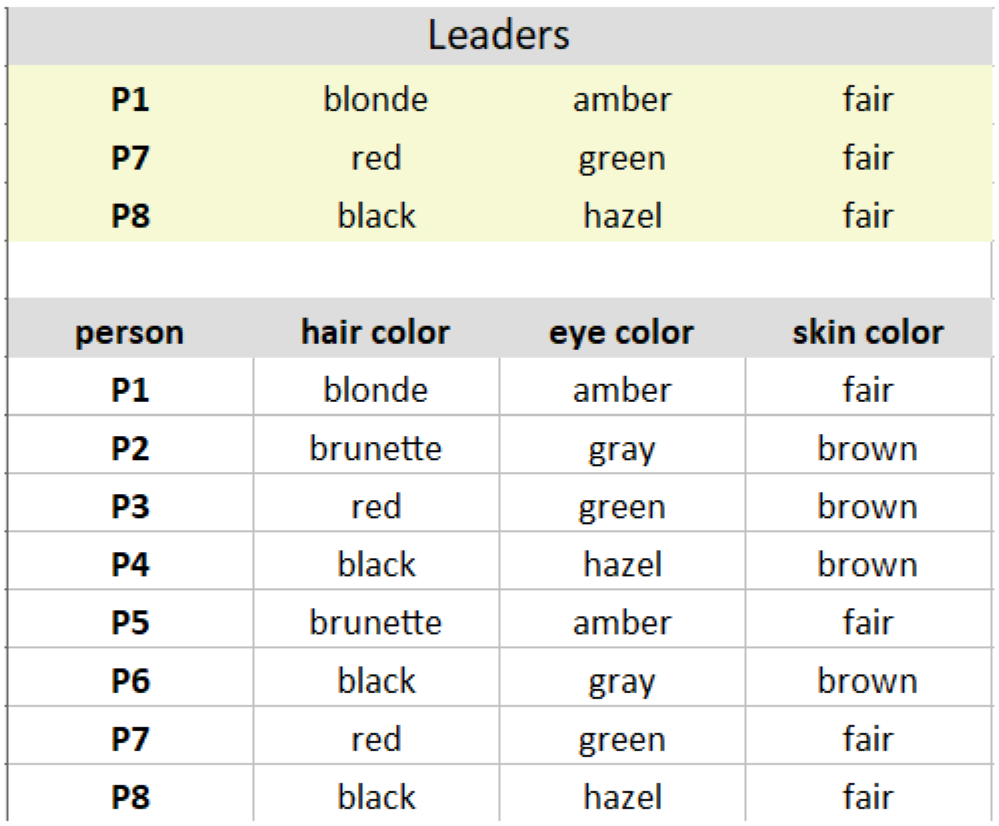


| person | hair color | eye color | skin color |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P2** | brunette | gray | brown |
| **P3** | red | green | brown |
| **P4** | black | hazel | brown |
| **P5** | brunette | amber | fair |
| **P6** | black | gray | brown |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

Alright, we have the sample data now. Let us proceed by defining the number of clusters(K)=3

***Step 1: Pick K observations at random and use them as leaders/clusters***

I am choosing P1, P7, P8 as leaders/clusters



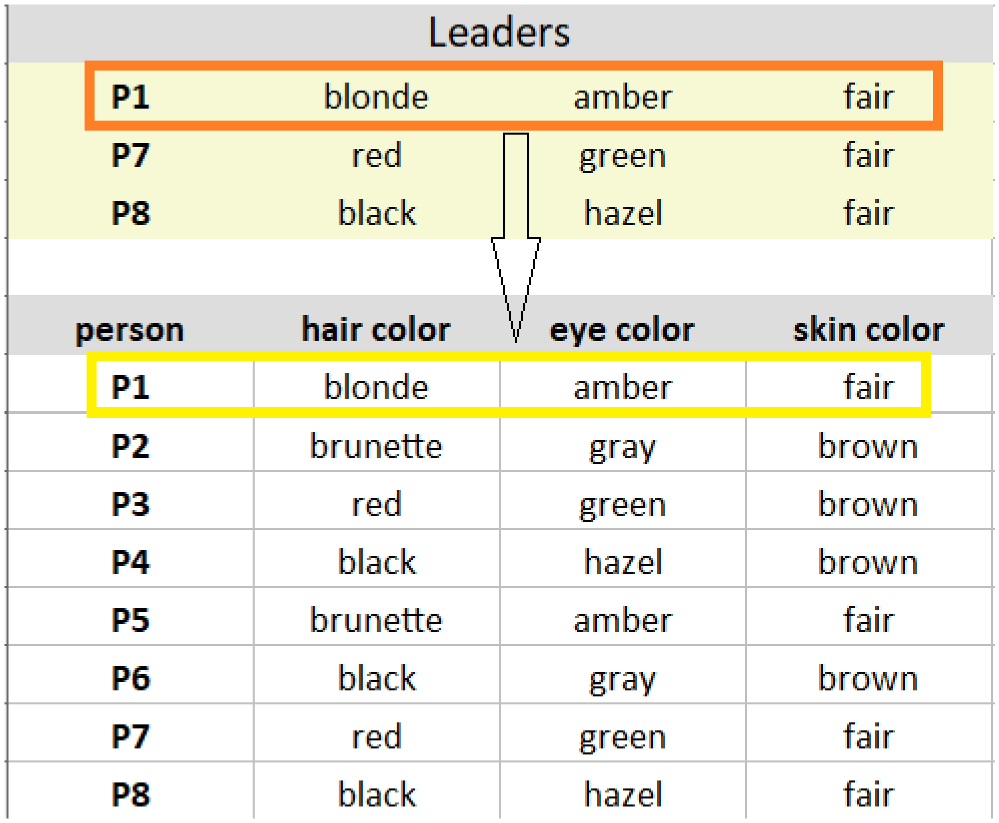
| Leaders | | | |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

| person | hair color | eye color | skin color |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P2** | brunette | gray | brown |
| **P3** | red | green | brown |
| **P4** | black | hazel | brown |
| **P5** | brunette | amber | fair |
| **P6** | black | gray | brown |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

Leaders and Observations

***Step 2: Calculate the dissimilarities(no. of mismatches) and assign each observation to its closest cluster***

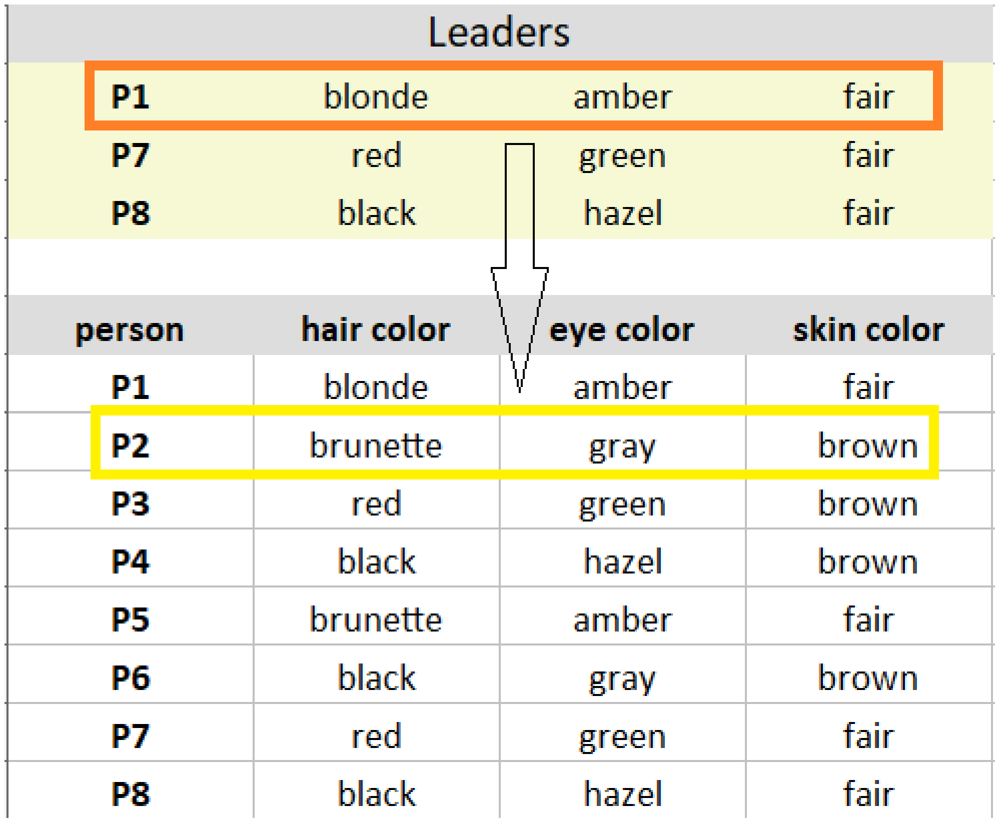
Iteratively compare the cluster data points to each of the observations. Similar data points give 0, dissimilar data points give 1.



| Leaders | | | |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

| person | hair color | eye color | skin color |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P2** | brunette | gray | brown |
| **P3** | red | green | brown |
| **P4** | black | hazel | brown |
| **P5** | brunette | amber | fair |
| **P6** | black | gray | brown |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

Comparing leader/Cluster P1 to the observation P1 gives 0 dissimilarities.

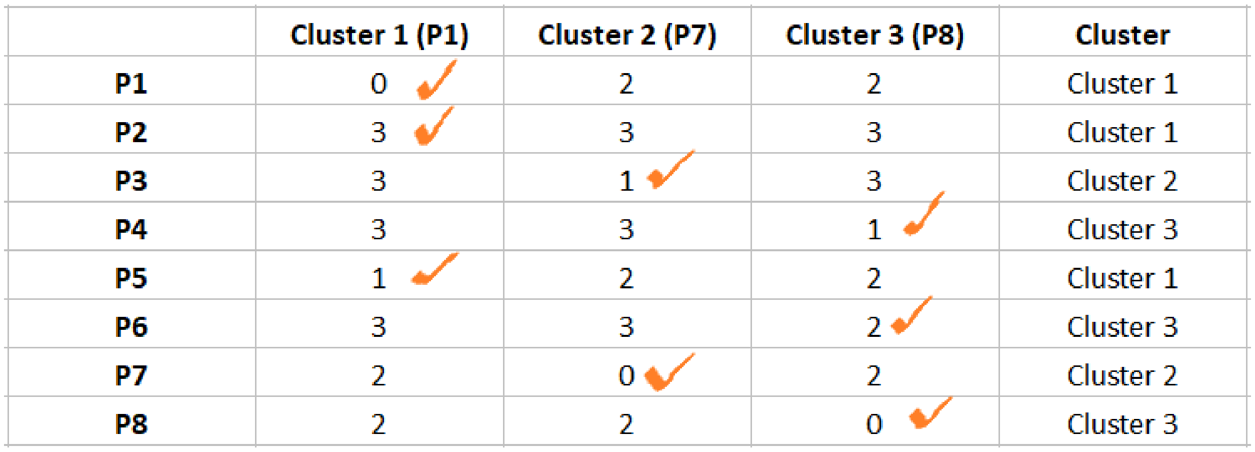


| Leaders | | | |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

| person | hair color | eye color | skin color |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P2** | brunette | gray | brown |
| **P3** | red | green | brown |
| **P4** | black | hazel | brown |
| **P5** | brunette | amber | fair |
| **P6** | black | gray | brown |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

Comparing leader/cluster P1 to the observation P2 gives 3(1+1+1) dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix as shown below and assign the observations to their closest cluster(cluster that has the least dissimilarity)



| | Cluster 1 (P1) | Cluster 2 (P7) | Cluster 3 (P8) | Cluster |
|---|---|---|---|---|
| P1 | 0 ✓ | 2 | 2 | Cluster 1 |
| P2 | 3 ✓ | 3 | 3 | Cluster 1 |
| P3 | 3 | 1 ✓ | 3 | Cluster 2 |
| P4 | 3 | 3 | 1 ✓ | Cluster 3 |
| P5 | 1 ✓ | 2 | 2 | Cluster 1 |
| P6 | 3 | 3 | 2 ✓ | Cluster 3 |
| P7 | 2 | 0 ✓ | 2 | Cluster 2 |
| P8 | 2 | 2 | 0 ✓ | Cluster 3 |

Dissimilarity matrix (Image by Author)

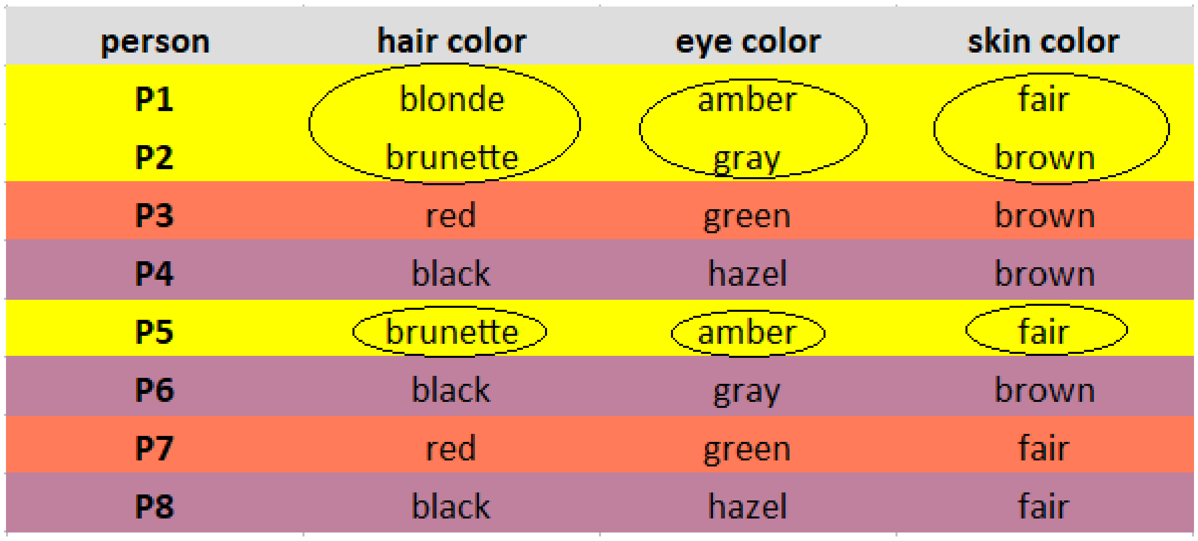
After step 2, the observations P1, P2, P5 are assigned to cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to cluster 3.

*Note: If all the clusters have the same dissimilarity with an observation, assign to any cluster randomly. In our case, the observation P2 has 3 dissimilarities with all the leaders. I randomly assigned it to Cluster 1.*

***Step 3: Define new modes for the clusters***

Mode is simply the **most observed value**.

Mark the observations according to the cluster they belong to. Observations of Cluster 1 are marked in Yellow, Cluster 2 are marked in Brick red, and Cluster 3 are marked in Purple.



| person | hair color | eye color | skin color |
|---|---|---|---|
| P1 | blonde | amber | fair |
| P2 | brunette | gray | brown |
| P3 | red | green | brown |
| P4 | black | hazel | brown |
| P5 | brunette | amber | fair |
| P6 | black | gray | brown |
| P7 | red | green | fair |
| P8 | black | hazel | fair |

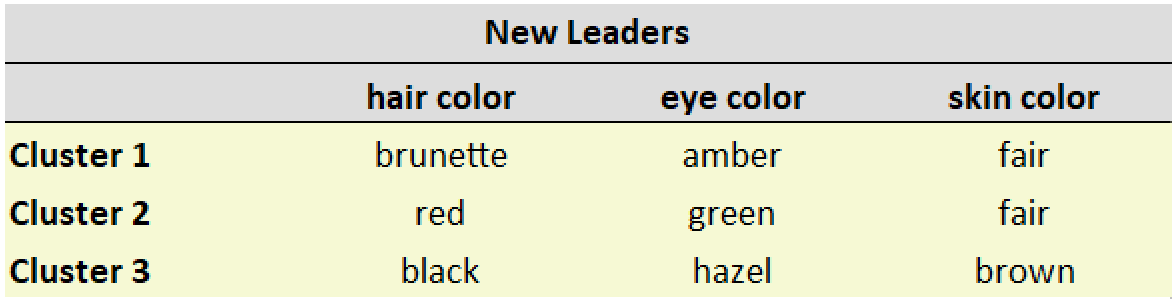
Looking for Modes (Image by author)

Considering one cluster at a time, for each feature, look for the Mode and update the new leaders.

**Explanation:** Cluster 1 observations(P1, P2, P5) has brunette as the most observed hair color, amber as the most observed eye color, and fair as the most observed skin color.

*Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 3(P3, P7) have one occurrence of brown, fair skin color. I randomly chose brown as the mode.*

Below are our new leaders after the update.

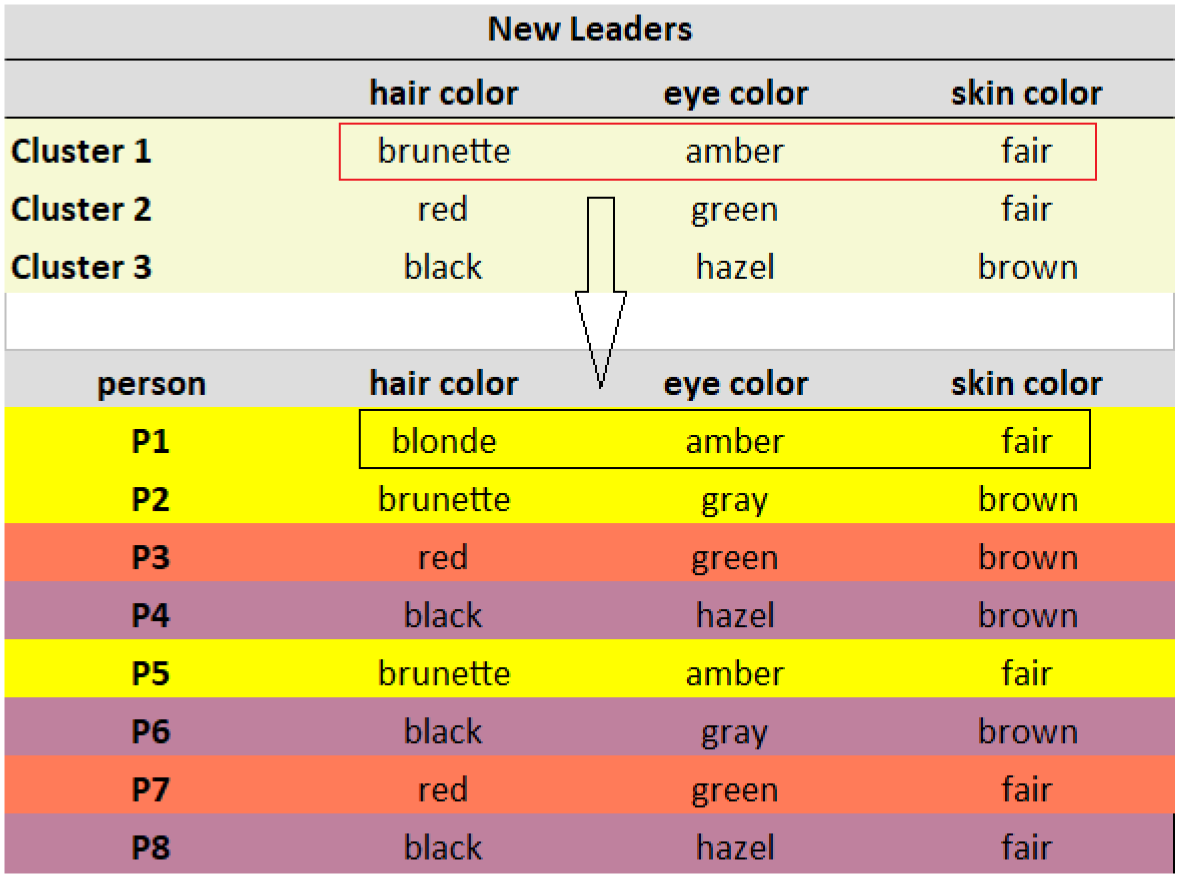


| New Leaders | | | |
|---|---|---|---|
| | **hair color** | **eye color** | **skin color** |
| **Cluster 1** | brunette | amber | fair |
| **Cluster 2** | red | green | fair |
| **Cluster 3** | black | hazel | brown |

Obtained new leaders

**Repeat steps 2–4**

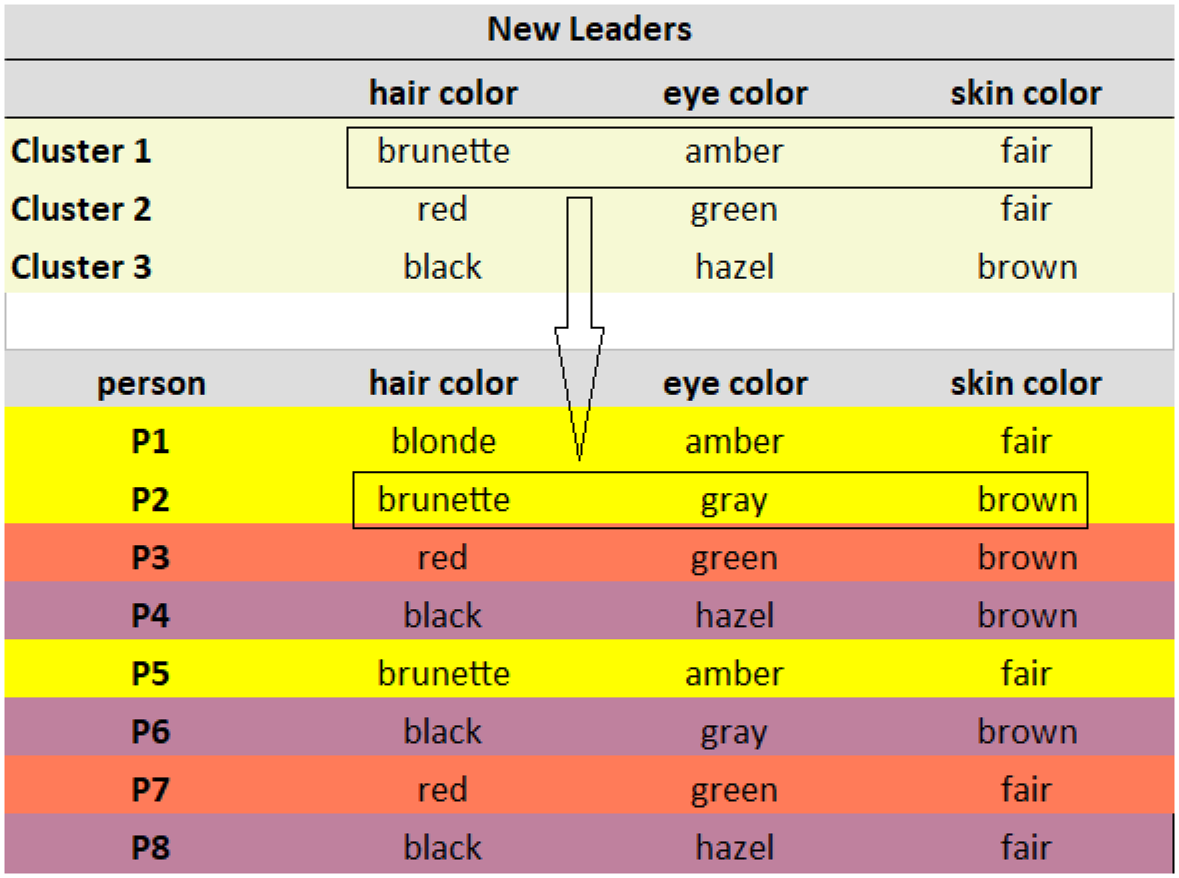
After obtaining the new leaders, again calculate the dissimilarities between the observations and the newly obtained leaders.



| New Leaders | | | |
|---|---|---|---|
| | **hair color** | **eye color** | **skin color** |
| **Cluster 1** | brunette | amber | fair |
| **Cluster 2** | red | green | fair |
| **Cluster 3** | black | hazel | brown |

| **person** | **hair color** | **eye color** | **skin color** |
|---|---|---|---|
| **P1** | blonde | amber | fair |
| **P2** | brunette | gray | brown |
| **P3** | red | green | brown |
| **P4** | black | hazel | brown |
| **P5** | brunette | amber | fair |
| **P6** | black | gray | brown |
| **P7** | red | green | fair |
| **P8** | black | hazel | fair |

Comparing Cluster 1 to the observation P1 gives 1 dissimilarity.



| New Leaders | hair color | eye color | skin color |
|---|---|---|---|
| Cluster 1 | brunette | amber | fair |
| Cluster 2 | red | green | fair |
| Cluster 3 | black | hazel | brown |

| person | hair color | eye color | skin color |
|---|---|---|---|
| P1 | blonde | amber | fair |
| P2 | brunette | gray | brown |
| P3 | red | green | brown |
| P4 | black | hazel | brown |
| P5 | brunette | amber | fair |
| P6 | black | gray | brown |
| P7 | red | green | fair |
| P8 | black | hazel | fair |

Comparing Cluster 1 to the observation P2 gives 2 dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix. Assign each observation to its closest cluster.

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster |
|---|---|---|---|---|
| P1 | 1 ✓ | 2 | 3 | Cluster 1 |
| P2 | 2 ✓ | 3 | 2 | Cluster 1 |
| P3 | 3 | 1 ✓ | 2 | Cluster 2 |
| P4 | 3 | 3 | 0 ✓ | Cluster 3 |
| P5 | 0 ✓ | 2 | 3 | Cluster 1 |
| P6 | 3 | 3 | 1 ✓ | Cluster 3 |
| P7 | 2 | 0 ✓ | 3 | Cluster 2 |
| P8 | 2 | 2 | 1 ✓ | Cluster 3 |

The observations P1, P2, P5 are assigned to Cluster 1; P3, P7 are assigned to Cluster 2; and P4, P6, P8 are assigned to Cluster 3.

We stop here as we see there is no change in the assignment of observations.

## ML | K-Medoids clustering with solved example

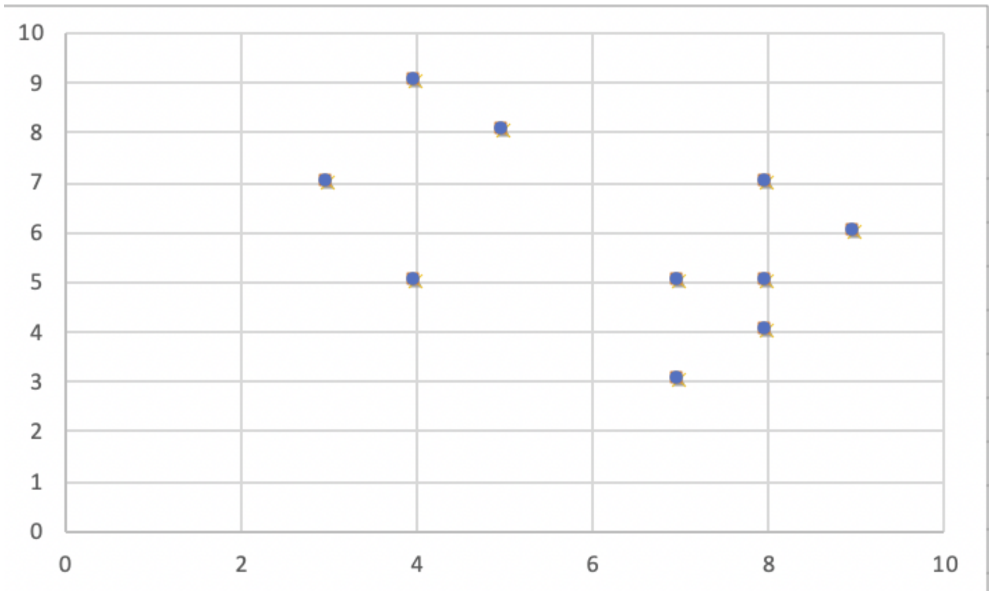
**K-Medoids** (also called Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster are minimum. The dissimilarity of the medoid(Ci) and object(Pi) is calculated by using E = |Pi – Ci|
*The cost in K-Medoids algorithm is given as*

$$c = \sum_{Ci} \sum_{Pi \in Ci} |Pi - Ci|$$

**Algorithm:**
1. Initialize: select *k* random points out of the *n* data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.
3. While the cost decreases: For each medoid m, for each data o point which is not a medoid:
   - Swap m and o, associate each data point to the closest medoid, recompute the cost.
   - If the total cost is more than that in the previous step, undo the swap.

| | X | Y |
|---|---|---|
| 0 | 8 | 7 |
| 1 | 3 | 7 |
| 2 | 4 | 9 |
| 3 | 9 | 6 |
| 4 | 8 | 5 |
| 5 | 5 | 8 |
| 6 | 7 | 3 |
| 7 | 8 | 4 |
| 8 | 7 | 5 |
| 9 | 4 | 5 |

Let’s consider the following example: If a graph is drawn using the above data points, we obtain the following:



**Step 1:** Let the randomly selected 2 medoids, so select k = 2 and let **C1 -(4, 5)** and **C2 -(8, 5)** are the two medoids.
**Step 2: Calculating cost.** The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:

| | X | Y | Dissimilarity from C1 | Dissimilarity from C2 |
|---|---|---|---|---|
| 0 | 8 | 7 | 6 | 2 |
| 1 | 3 | 7 | 3 | 7 |
| 2 | 4 | 9 | 4 | 8 |
| 3 | 9 | 6 | 6 | 2 |
| 4 | 8 | 5 | - | - |
| 5 | 5 | 8 | 4 | 6 |
| 6 | 7 | 3 | 5 | 3 |
| 7 | 8 | 4 | 5 | 1 |
| 8 | 7 | 5 | 3 | 1 |
| 9 | 4 | 5 | - | - |

Each point is assigned to the cluster of that medoid whose dissimilarity is less. The points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The Cost = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20

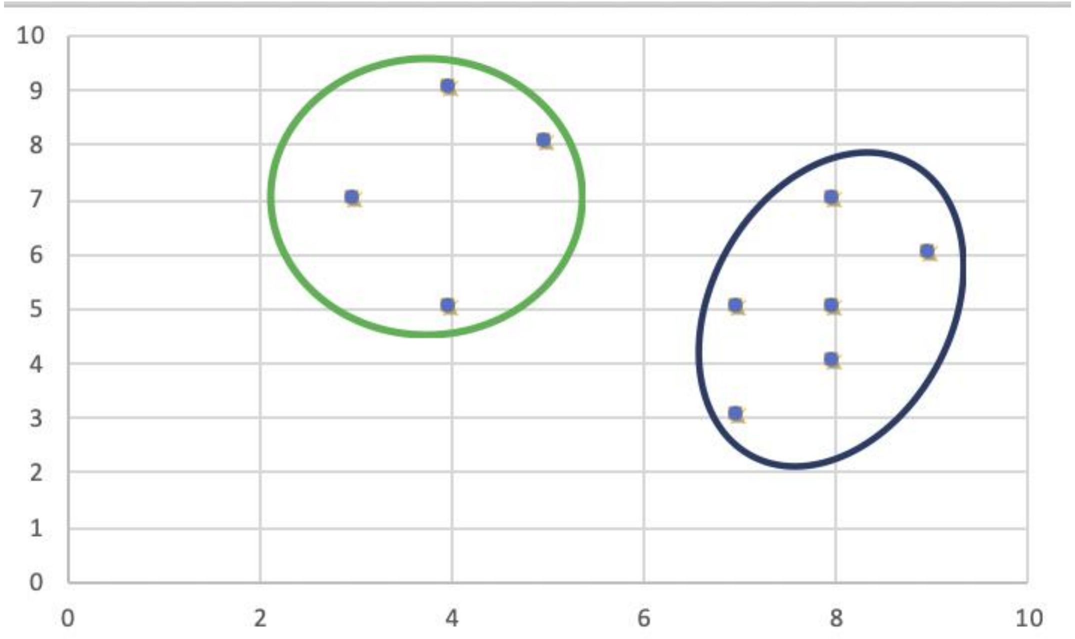
**Step 3: randomly select one non-medoid point and recalculate the cost.** Let the randomly selected point be (8, 4). The dissimilarity of each non-medoid point with the medoids – C1 (4, 5) and C2 (8, 4) is calculated and tabulated.

|  | X | Y | Dissimilarity from C1 | Dissimilarity from C2 |
|---|---|---|---|---|
| 0 | 8 | 7 | 6 | 3 |
| 1 | 3 | 7 | 3 | 8 |
| 2 | 4 | 9 | 4 | 9 |
| 3 | 9 | 6 | 6 | 3 |
| 4 | 8 | 5 | 4 | 1 |
| 5 | 5 | 8 | 4 | 7 |
| 6 | 7 | 3 | 5 | 2 |
| 7 | 8 | 4 | - | - |
| 8 | 7 | 5 | 3 | 2 |
| 9 | 4 | 5 | - | - |

Each point is assigned to that cluster whose dissimilarity is less. So, the points 1, 2, 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The New cost = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 22 Swap Cost = New Cost – Previous Cost = 22 – 20 and **2 >0** As the swap cost is not less than zero, we undo the swap. Hence (4, 5) and (8, 5) are the final medoids. The clustering would be in the following way.

The **time complexity** is

$O(k * (n - k)^2)$.



**Advantages:**

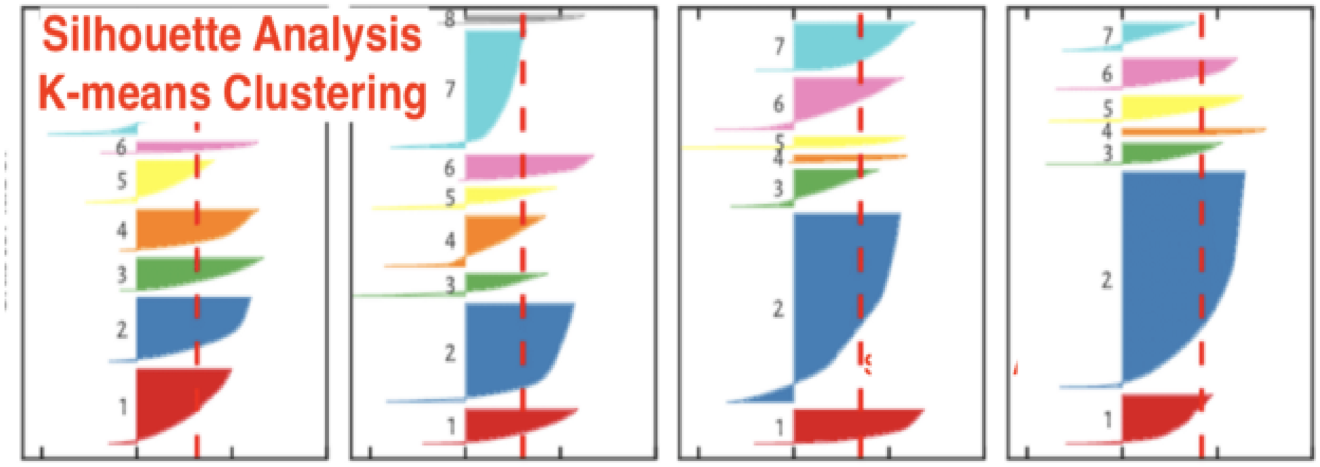
1. It is simple to understand and easy to implement.
2. K-Medoid Algorithm is fast and converges in a fixed number of steps.
3. PAM is less sensitive to outliers than other partitioning algorithms.

**Disadvantages:**

1. The main disadvantage of K-Medoid algorithms is that it is not suitable for clustering non-spherical (arbitrarily shaped) groups of objects. This is because it relies on minimizing the distances between the non-medoid objects and the medoid (the cluster center) – briefly, it uses compactness as clustering criteria instead of connectivity.
2. It may obtain different results for different runs on the same dataset because the first k medoids are chosen randomly.

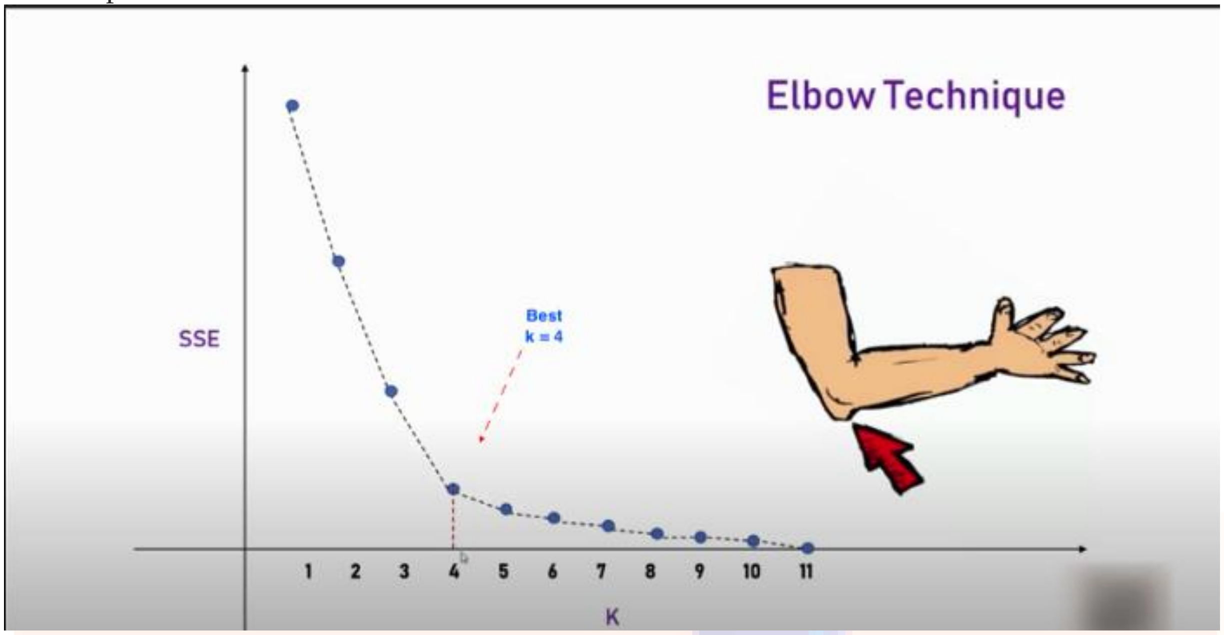
## Silhouette Analysis in K-means Clustering

Cluster Evaluation: silhouette coefficient

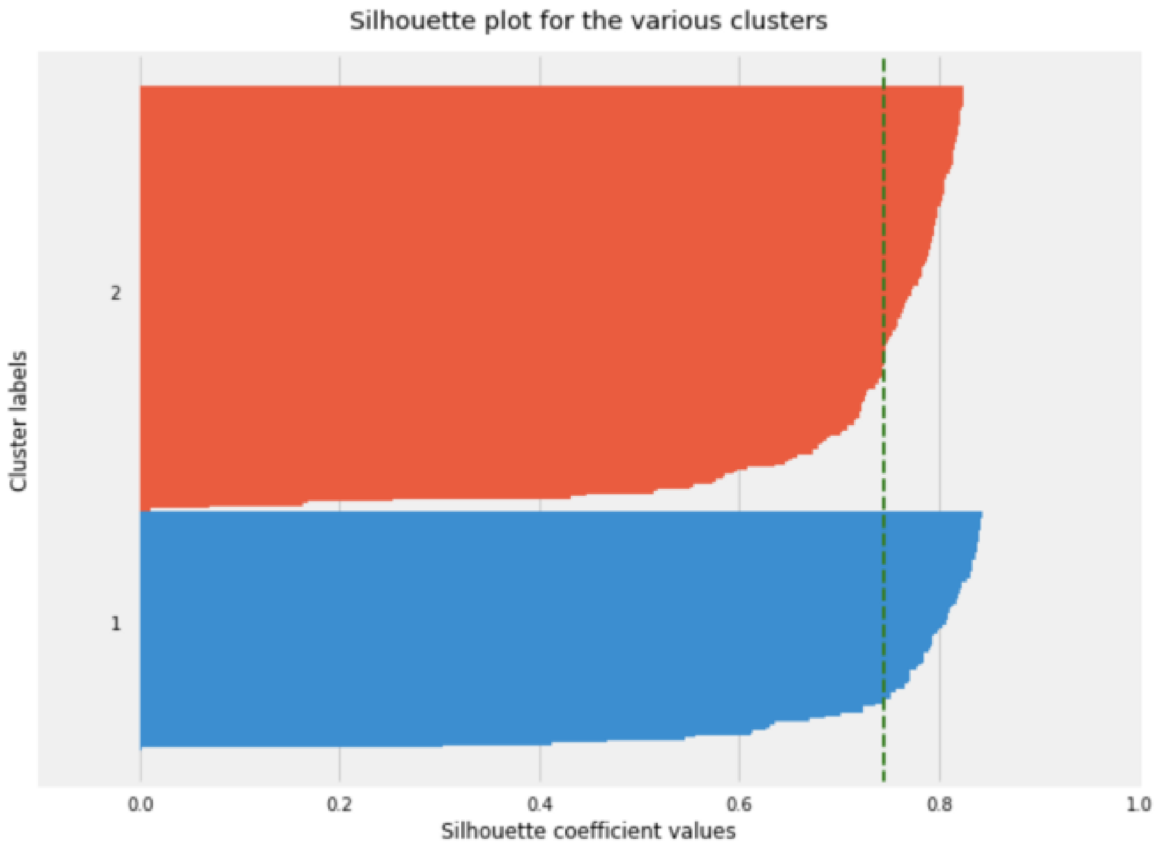


I am trying to explain tittle bit more on how to play more significant role in k-means clustering evaluation by silhouette analysis instead of elbow technique. Before go to this topic , we should know what k-mean clustering algorithm is about . I recommend to read my previous blog *here* to know about k-mean clustering algorithm. As we know that K-means clustering is a simplest and popular unsupervised machine learning algorithms. We can evaluate the algorithm by two ways . One is elbow technique and another is silhouette method. Here , I am trying to describe only silhouette analysis method and also trying to prove it is better than elbow method . I have already explain elbow method in my previous *blog* . Elbow is very simple method that it gives us plot like elbow shape . And we can easily guess optimal number of k from the plot . Maybe we become ambiguity to take decision when we get complex plot because i feel that sometime plot is vague . However , by silhouette method we can calculate silhouette coefficient and easily find exact number of k. let’s see picture that gives better concept:

Elbow method picture :



Silhouette method picture:



There are two things to consider here:

- If we have the ground truth labels (class information) of the data points available with us then we can make use of *extrinsic methods* like homogeneity score, completeness score and so on.

- But if we do not have the ground truth labels of the data points, we will have to use the *intrinsic methods* like silhouette score which is based on the *silhouette coefficient.* We now study this evaluation metric in a bit more details.

- Let's start with the equation for calculating the silhouette coefficient for a particular data point:

$$s(\boldsymbol{o}) = \frac{b(\boldsymbol{o}) - a(\boldsymbol{o})}{\max\{a(\boldsymbol{o}), b(\boldsymbol{o})\}}$$

where,

- ***s(o)*** is the silhouette coefficient of the data point **o**

- ***a(o)*** is the *average distance* between **o** and all the other data points in the cluster to which **o** belongs

- ***b(o)*** is the *minimum average distance* from **o** to all clusters to which **o** does not belong

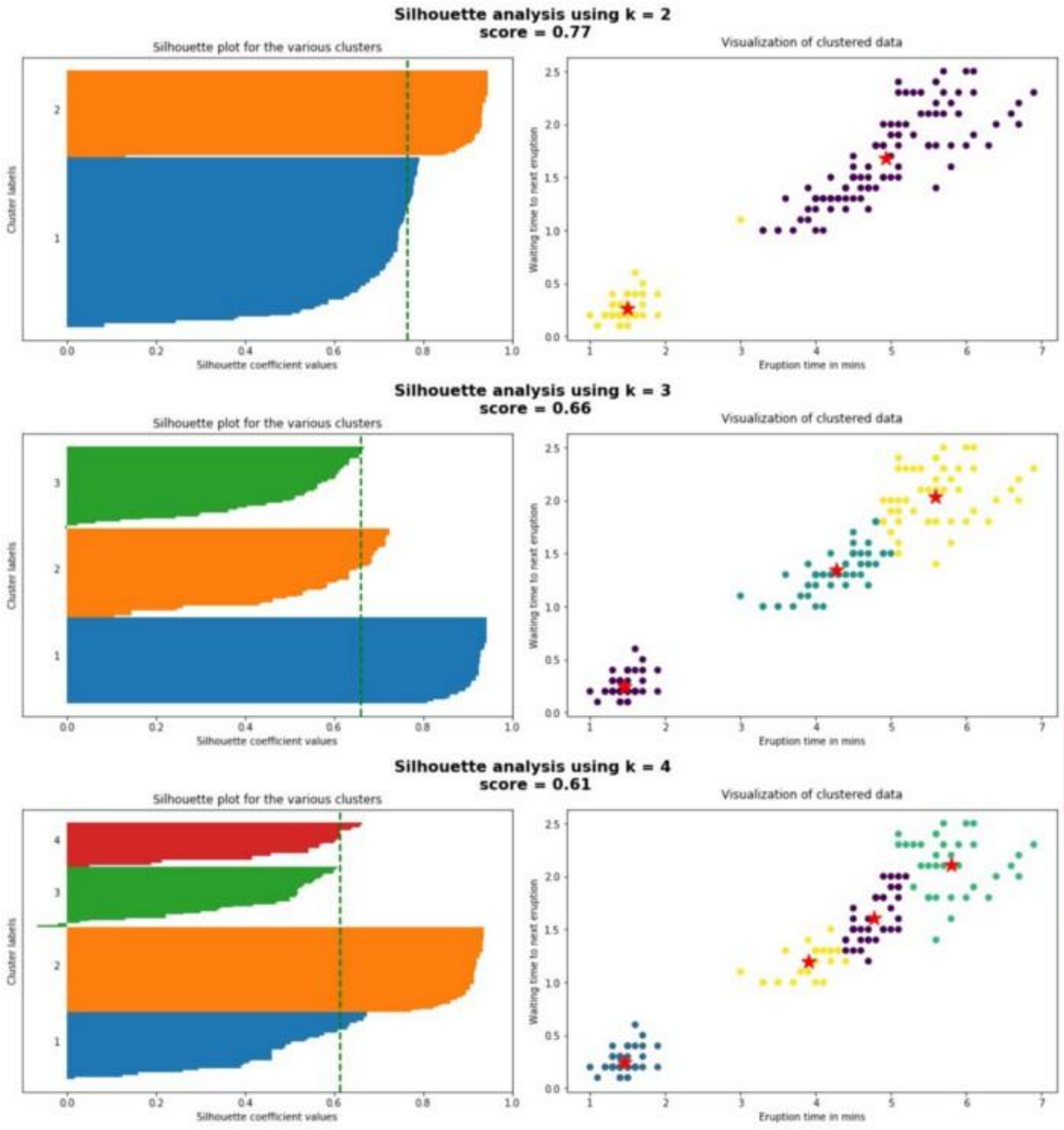
There are main points that we should remember during calculating silhouette coefficient .The value of the silhouette coefficient is between [-1, 1]. A score of 1 denotes the best meaning that the data point **o** is very compact within the cluster to which it belongs and far away from the other clusters**.** The worst value is -1. Values near 0 denote overlapping clusters.

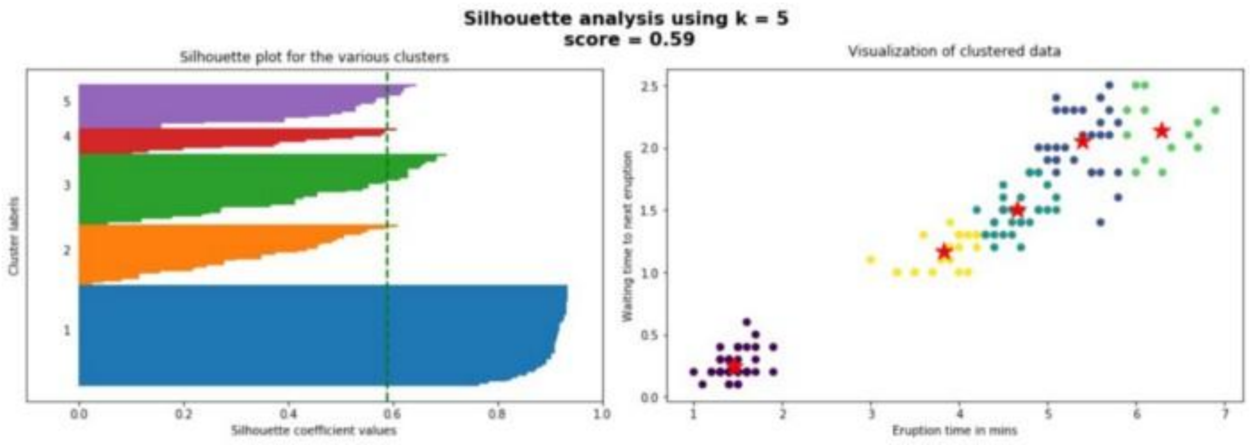
Let's try to understand through python code that make more easy.

```
# import neccessaries librariesimport pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inlinefrom sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples,silhouette_score# load petal data
data = datasets.load_iris()dir(data)# load into Dataframe
df = pd.DataFrame(data.data,columns = data.feature_names)
```

```
print(df.shape)
df.head()df1 = df.drop(['sepal length (cm)', 'sepal width (cm)'],axis = 'columns')
df1.head()# plot scatter plot
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'])# Now check silhouette coefficientfor i,k in
enumerate([2,3,4,5]):

    fig, ax = plt.subplots(1,2,figsize=(15,5))

    # Run the kmeans algorithm
    km = KMeans(n_clusters=k)
    y_predict = km.fit_predict(df1)
    centroids  = km.cluster_centers_# get silhouettesilhouette_vals = silhouette_samples(df1,y_predict)
    #silhouette_vals# silhouette ploty_ticks = []
y_lower = y_upper = 0for i,cluster in enumerate(np.unique(y_predict)):
   cluster_silhouette_vals = silhouette_vals[y_predict ==cluster]
   cluster_silhouette_vals.sort()
   y_upper += len(cluster_silhouette_vals)

   ax[0].barh(range(y_lower,y_upper),
   cluster_silhouette_vals,height =1);   ax[0].text(-0.03,(y_lower+y_upper)/2,str(i+1))
   y_lower += len(cluster_silhouette_vals)
   # Get the average silhouette score    avg_score = np.mean(silhouette_vals)
   ax[0].axvline(avg_score,linestyle ='--',
   linewidth =2,color = 'green')
   ax[0].set_yticks([])
   ax[0].set_xlim([-0.1, 1])
   ax[0].set_xlabel('Silhouette coefficient values')
   ax[0].set_ylabel('Cluster labels')
   ax[0].set_title('Silhouette plot for the various clusters');


    # scatter plot of data colored with labels

    ax[1].scatter(df2['petal length (cm)'],
    df2['petal width (cm)'] , c = y_predict);    ax[1].scatter(centroids[:,0],centroids[:,1],
    marker = '*' , c= 'r',s =250);
    ax[1].set_xlabel('Eruption time in mins')
    ax[1].set_ylabel('Waiting time to next eruption')
    ax[1].set_title('Visualization of clustered data', y=1.02)

    plt.tight_layout()
    plt.suptitle(f' Silhouette analysis using k = {k}',fontsize=16,fontweight = 'semibold')
    plt.savefig(f'Silhouette_analysis_{k}.jpg')
```

Output:


Silhouette analysis using k = 2
score = 0.77
Silhouette plot for the various clusters
Visualization of clustered data
Cluster labels
Silhouette coefficient values
Waiting time to next eruption
Eruption time in mins
Silhouette analysis using k = 3
score = 0.66
Silhouette plot for the various clusters
Visualization of clustered data
Silhouette analysis using k = 4
score = 0.61
Silhouette plot for the various clusters
Visualization of clustered data



In above all pictures , we can clearly see that how plot and score are different according to n_cluster(k) . So, we can easily choose high score and number of k via silhouette analysis technique instead of elbow technique.

**Conclusion:**

K-means clustering is a simplest and popular unsupervised machine learning algorithms . We can evaluate the algorithm by two ways such as elbow technique and silhouette technique . We saw differences between them above . I think silhouette technique gives us more precise score and number of k for k-means algorithm. However , we can also use elbow technique for quick response and intuition.

## Hierarchical Clustering in Machine Learning

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

*Why hierarchical clustering?*

As we already have other clustering algorithms such as **K-Means Clustering**, then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

Agglomerative Hierarchical clustering

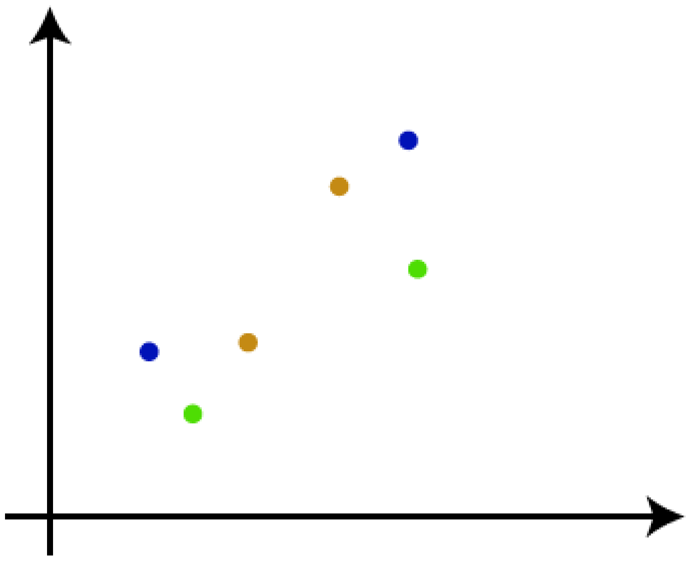
The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

This hierarchy of clusters is represented in the form of the dendrogram.

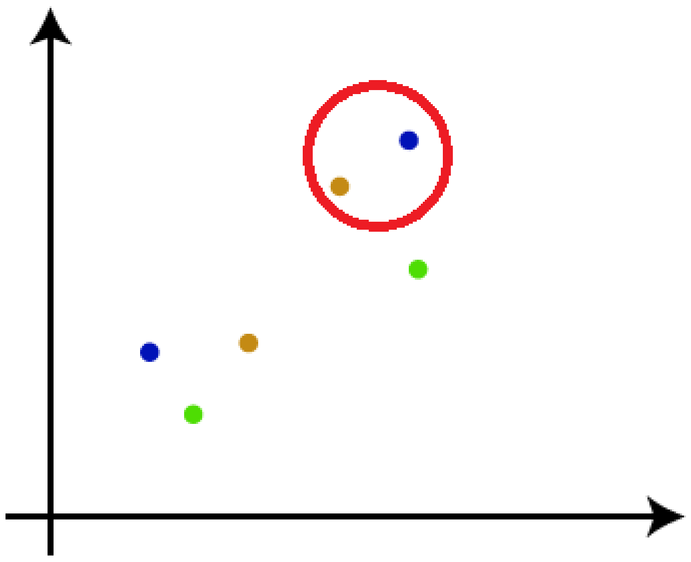
How the Agglomerative Hierarchical clustering Work?

The working of the AHC algorithm can be explained using the below steps:

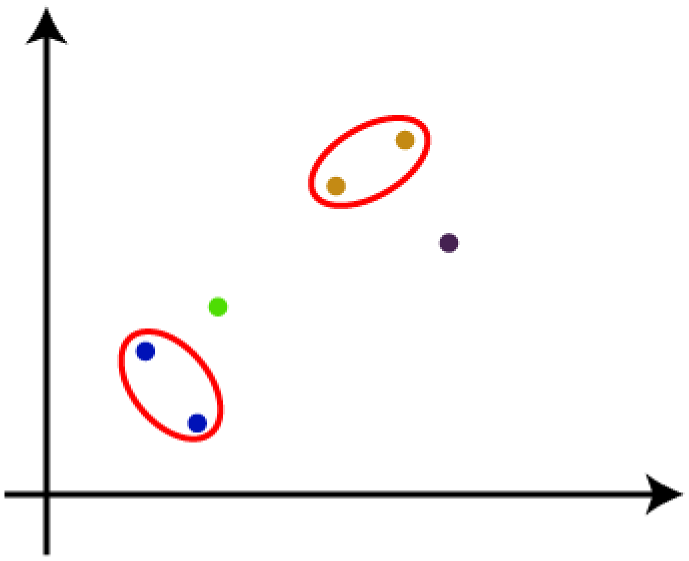
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



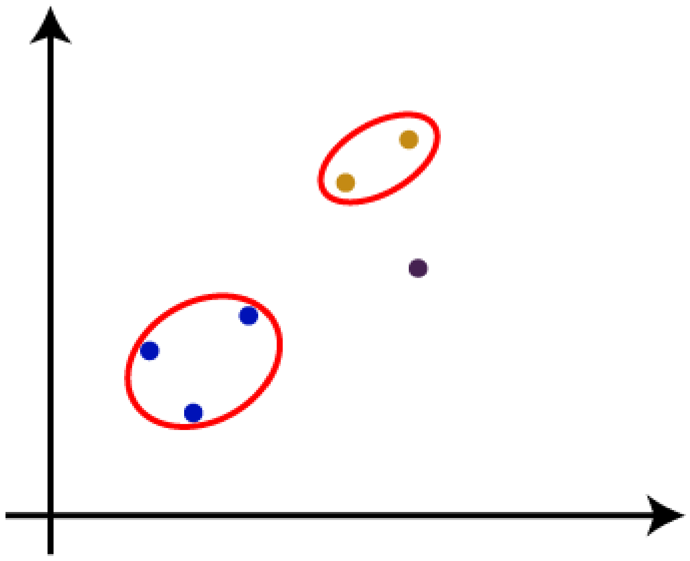
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.

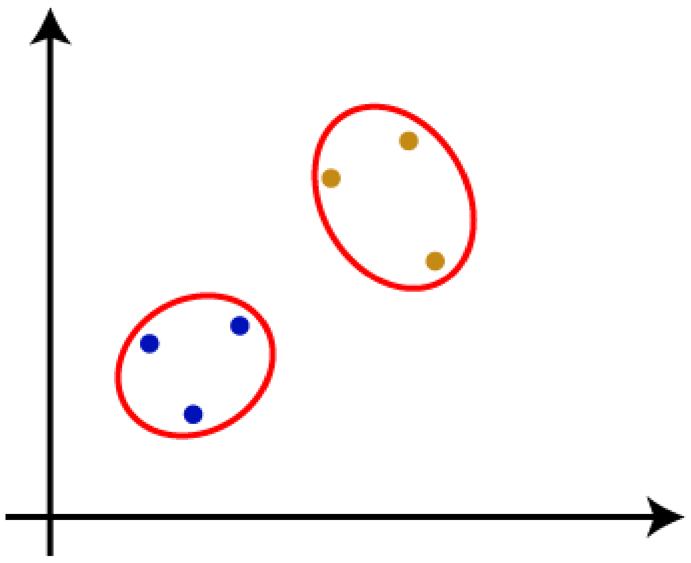


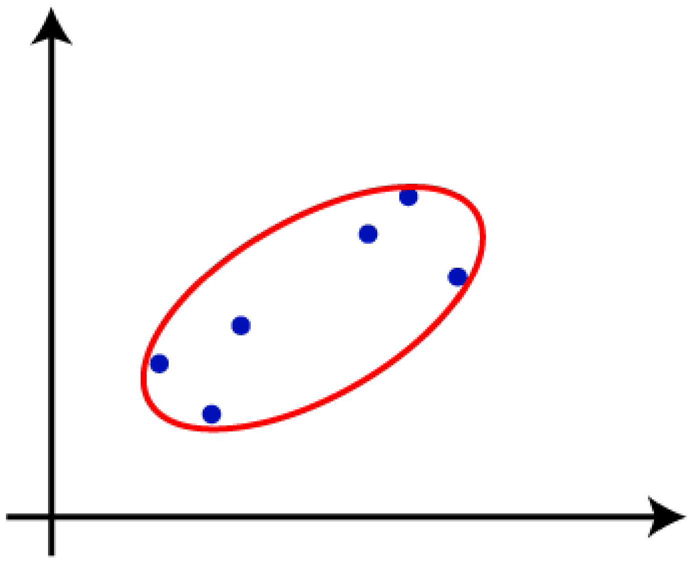
- **Step-3**: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 clusters.



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:







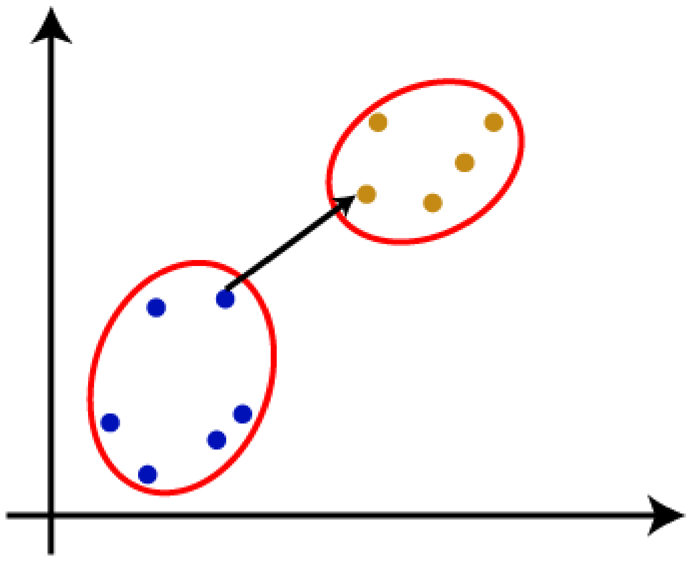
- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

*Note: To better understand hierarchical clustering, it is advised to have a look on k-means clustering*

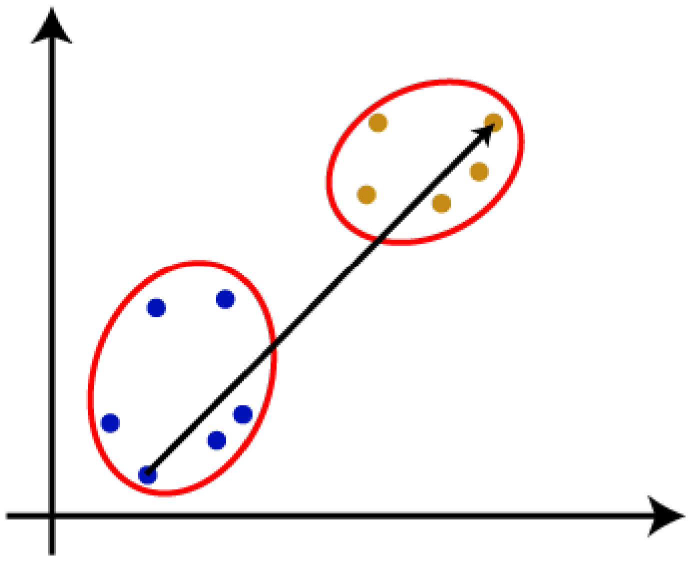
*Measure for the distance between two clusters*

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

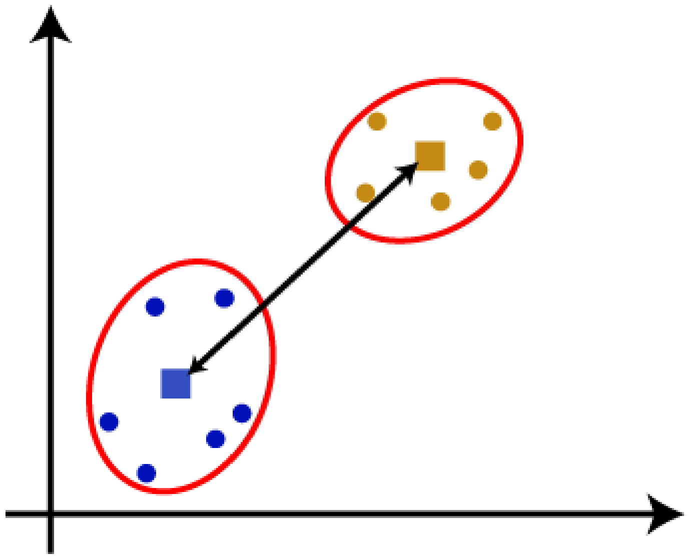
1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters. Consider the below image:



2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.
4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



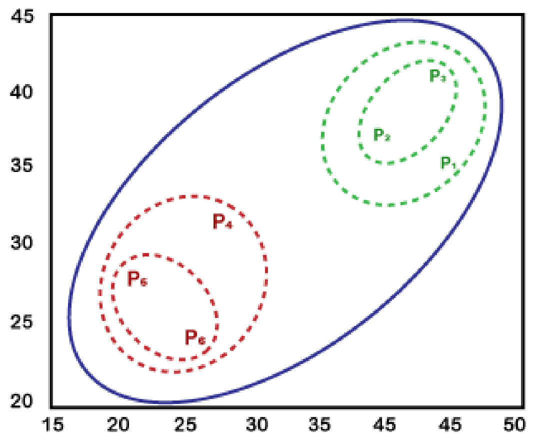
From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

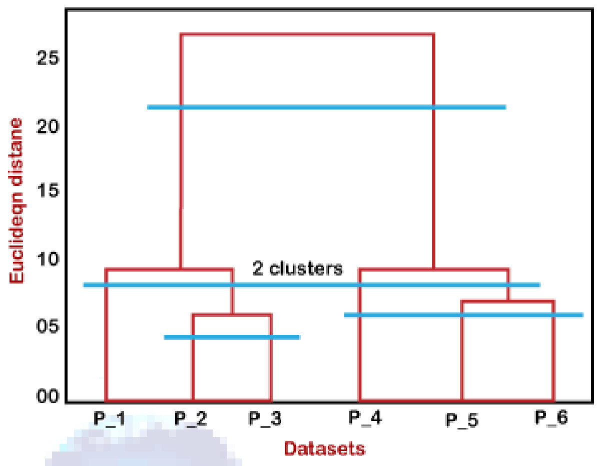
[illegible]

*Woking of Dendrogram in Hierarchical clustering*

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

The working of the dendrogram can be explained using the below diagram:





In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

- As we have discussed above, firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created, which connects P2 and P3 with a rectangular shape. The hight is decided according to the Euclidean distance between the data points.
- In the next step, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than of previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
- Again, two new dendrograms are created that combine P1, P2, and P3 in one dendrogram, and P4, P5, and P6, in another dendrogram.
- At last, the final dendrogram is created that combines all the data points together.

We can cut the dendrogram tree structure at any level as per our requirement.