CLUSTERING – AN UNSUPERVISED APPROACH

Introduction and Background

- Clustering in data mining is a technique used to group similar data points together based on their features and characteristics.

- It is an unsupervised learning method that helps to identify patterns in large datasets and segment them into smaller groups or subsets.

- Clustering can be used for various applications such as customer segmentation, image recognition, and anomaly detection.

- The goal of cluster analysis is to divide a dataset into groups (or clusters) such that the data points within each group are more similar to each other than to data points in other groups.

- This process is often used for exploratory data analysis and can help identify patterns or relationships within the data that may not be immediately obvious.

- Clustering algorithms typically require defining the number of clusters, similarity measures, and clustering methods.

- These algorithms aim to group data points together in a way that maximizes similarity within the groups and minimizes similarity between different groups, as shown in the picture below.

Introduction and Background

Introduction and Background

- In data mining, a cluster refers to a group of data points with similar characteristics or features.
- These characteristics or features can be defined by the analyst or identified by the clustering algorithm while grouping similar data points together.
- The data points within a cluster are typically more similar to each other than those outside the cluster. -

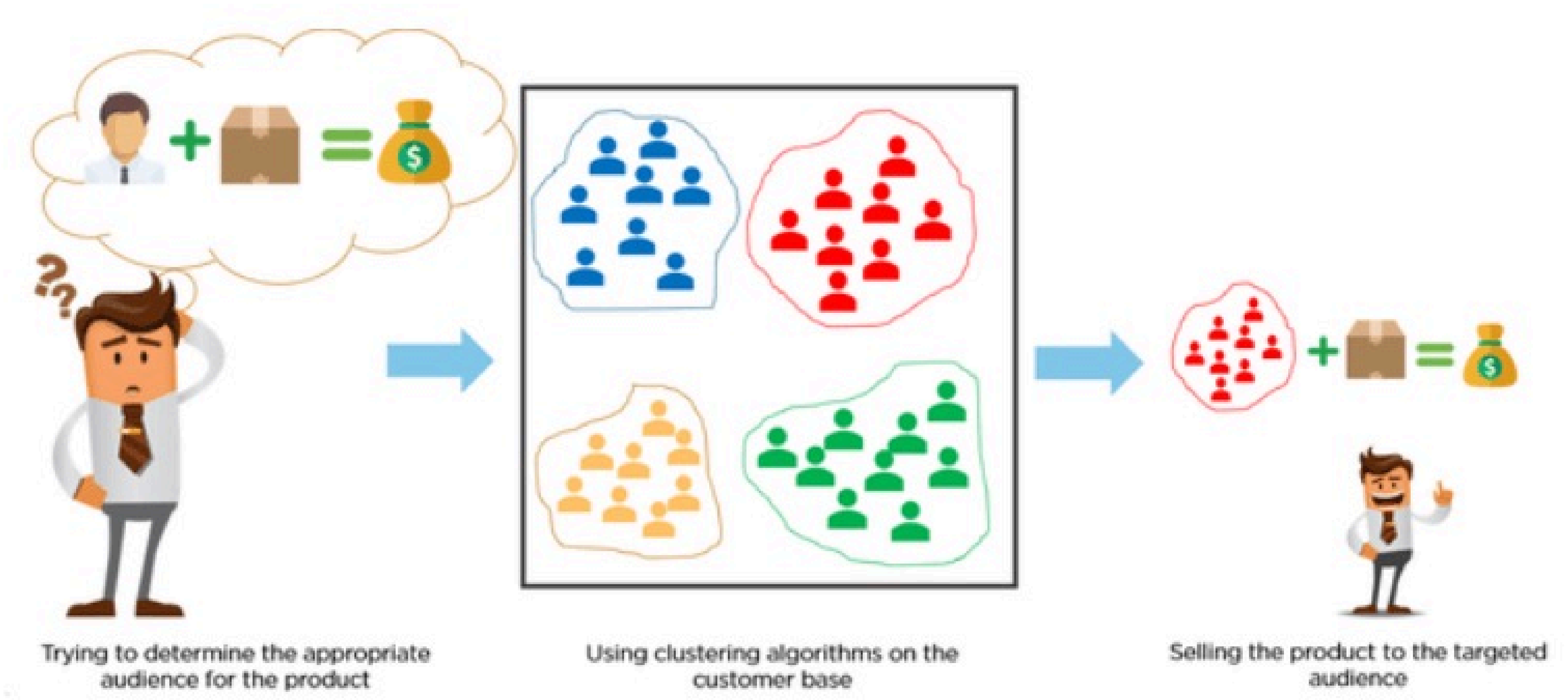
A cluster can have the following properties -

- The data points within a cluster are similar to each other based on some pre-defined criteria or similarity measures.
- The clusters are distinct from each other, and the data points in one cluster are different from those in another cluster.
- The data points within a cluster are closely packed together.
- A cluster is often represented by a centroid or a center point that summarizes the properties of the data points within the cluster.
- A cluster can have any number of data points, but a good cluster should not be too small or too large.
- It should be capable of dealing with different types of data like discrete, categorical and interval-based data, binary data etc.

Applications of Clustering

Applications of Clustering

- **Customer Segmentation:** Clustering techniques in data mining can be used to group customers with similar behavior, preferences, and purchasing patterns to create more targeted marketing campaigns.
- **Image Segmentation:** Clustering techniques in data mining can be used to segment images into different regions based on their pixel values, which can be useful for tasks such as object recognition and image compression.
- **Anomaly Detection:** Clustering techniques in data mining can be used to identify outliers or anomalies in datasets that deviate significantly from normal behavior.
- **Text Mining:** Clustering techniques in data mining can be used to group documents or texts with similar content, which can be useful for tasks such as document summarization and topic modeling.
- **Biological Data Analysis:** Clustering techniques in data mining can be used to group genes or proteins with similar characteristics or expression patterns, which can be useful for tasks such as drug discovery and disease diagnosis.
- **Recommender Systems:** Clustering techniques in data mining can be used to group users with similar interests or behavior to create more personalized recommendations for products or services.


Trying to determine the appropriate audience for the product
Using clustering algorithms on the customer base
Selling the product to the targeted audience

Good Clustering – Properties

High Intra-cluster Similarity: Data points within the same cluster should be highly similar or homogeneous. This ensures that the clusters are compact and well-separated from other clusters.

Low Inter-cluster Similarity: Data points in different clusters should be dissimilar or heterogeneous. This property ensures that the clusters are well-separated and distinct from each other.

High Cluster Cohesion and Separation: Clusters should exhibit high cohesion, meaning that the data points within a cluster are closely bound together. At the same time, clusters should be well-separated from each other, indicating a clear distinction between different groups.

Interpretability: A good clustering should provide meaningful and interpretable insights into the data. The resulting clusters should correspond to underlying patterns, structures, or properties that can be explained and understood in the context of the problem domain.

Stability and Robustness: A good clustering should be stable and robust to small perturbations or changes in the data. Small variations in the input data should not significantly impact the clustering results, ensuring consistency and reliability.

Scalability: For large-scale datasets, a good clustering algorithm should be computationally efficient and scalable, allowing it to handle the increasing volume and complexity of data effectively.

Good Clustering – Properties

High Intra-cluster Similarity: Data points within the same cluster should be highly similar or homogeneous. This ensures that the clusters are compact and well-separated from other clusters.

Low Inter-cluster Similarity: Data points in different clusters should be dissimilar or heterogeneous. This property ensures that the clusters are well-separated and distinct from each other.

High Cluster Cohesion and Separation: Clusters should exhibit high cohesion, meaning that the data points within a cluster are closely bound together. At the same time, clusters should be well-separated from each other, indicating a clear distinction between different groups.

Interpretability: A good clustering should provide meaningful and interpretable insights into the data. The resulting clusters should correspond to underlying patterns, structures, or properties that can be explained and understood in the context of the problem domain.

Stability and Robustness: A good clustering should be stable and robust to small perturbations or changes in the data. Small variations in the input data should not significantly impact the clustering results, ensuring consistency and reliability.

Scalability: For large-scale datasets, a good clustering algorithm should be computationally efficient and scalable, allowing it to handle the increasing volume and complexity of data effectively.

Types of Clustering

- Hard and soft clustering are two fundamental concepts in the field of machine learning and data analysis, particularly in the realm of clustering algorithms.

1. Hard Clustering:

1. In hard clustering, each data point belongs exclusively to one cluster.

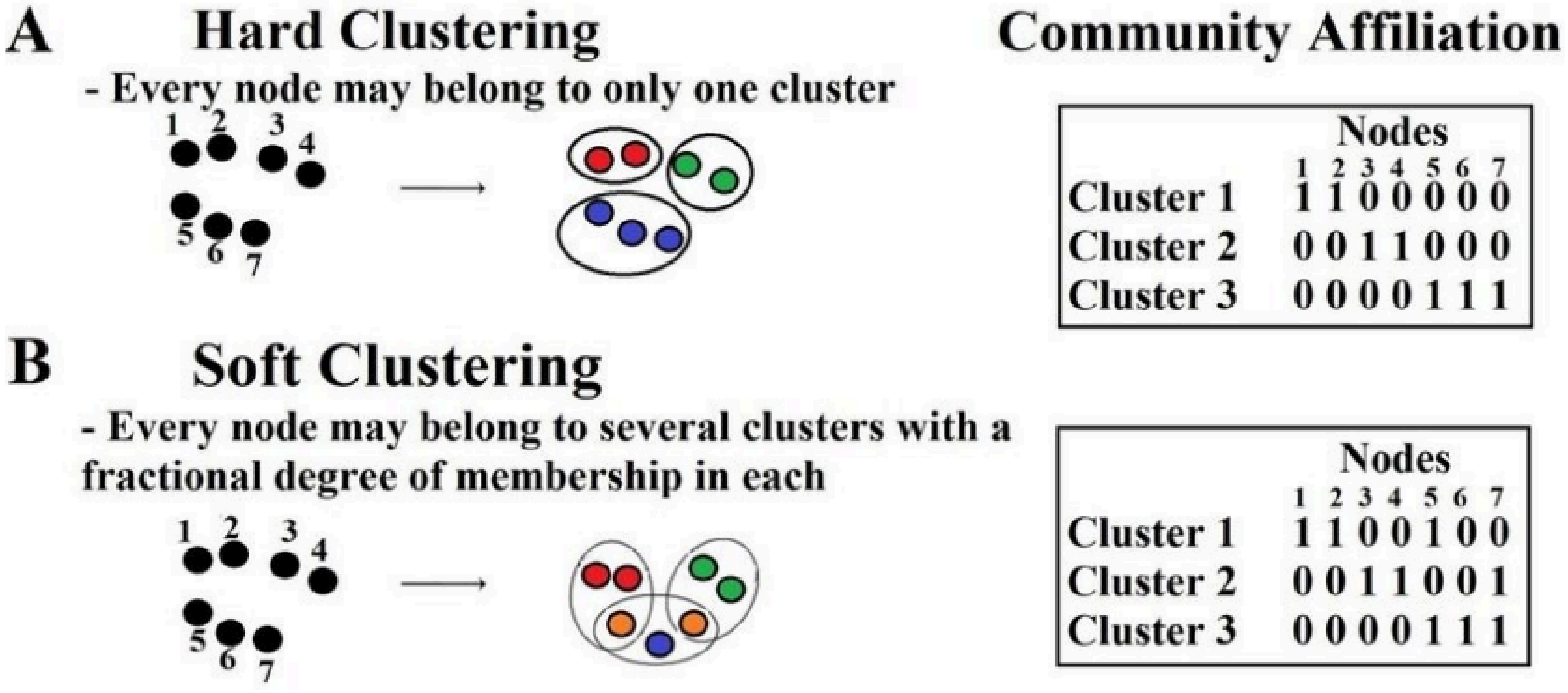
2. It involves partitioning the data into a set of distinct clusters where each data point is assigned to exactly one cluster. 3. Examples of hard clustering algorithms include K-means clustering and hierarchical clustering. 4. In K-means clustering, each data point is assigned to the cluster with the nearest centroid.

2. Soft Clustering (also known as fuzzy clustering):

1. In soft clustering, each data point can belong to multiple clusters with varying degrees of membership. 2. It assigns a membership value to each data point indicating the degree of belongingness to each cluster. 3. Examples of soft clustering algorithms include Fuzzy C-means clustering (FCM) and Gaussian Mixture Models (GMM).

4. In FCM, each data point has a membership value for each cluster, representing the degree of belongingness to that cluster.

Types of Clustering



A **Hard Clustering**

- Every node may belong to only one cluster

Community Affiliation

	Nodes						
	1	2	3	4	5	6	7
Cluster 1	1	1	0	0	0	0	0
Cluster 2	0	0	1	1	0	0	0
Cluster 3	0	0	0	0	1	1	1

B **Soft Clustering**

- Every node may belong to several clusters with a fractional degree of membership in each

	Nodes						
	1	2	3	4	5	6	7
Cluster 1	1	1	0	0	1	0	0
Cluster 2	0	0	1	1	0	0	1
Cluster 3	0	0	0	0	1	1	1

Types of Clustering – Based on Working Mechanism

1. Centroid based clustering

- Centroid-based clustering is a type of clustering method that partitions or splits a data set into similar groups based on the distance between their centroids.
- Each cluster's centroid, or center, is either the mean or median of all the points in the cluster depending on the data.
- One of the most commonly used centroid-based clustering techniques is the k-means clustering algorithm.
- K-means assumes that the center of each cluster defines the cluster using a distance measure, mostly commonly Euclidean distance, to the centroid.
- To initialize the clustering, you provide a number of expected clusters, which represents the 'K' in K-means, and the algorithm attempts to find reasonable clusters across the data to match that number.
- The optimal k clusters in a given dataset is identified by iteratively minimizing the total distance between each point and its assigned cluster centroid

Types of Clustering – Based on Working Mechanism

2. Hierarchical clustering

- Hierarchical clustering, sometimes called connectivity-based clustering, groups data points together based on the proximity and connectivity of their attributes.
- This method determines clusters based on how close data points are to one another across all of the dimensions.
- The idea is that objects that are nearer are more closely related than those that are far from each other.
- Unlike k-means, there is no need to pre-specify the number of clusters.
- Instead, the clustering algorithm creates a graph network of the clusters at each hierarchical level.
- Hierarchical clusters can be graphed with a dendrogram to help visually summarize and organize discovered clusters and the hierarchy that they may contain.

Types of Clustering – Based on Working Mechanism

3. Density-based clustering

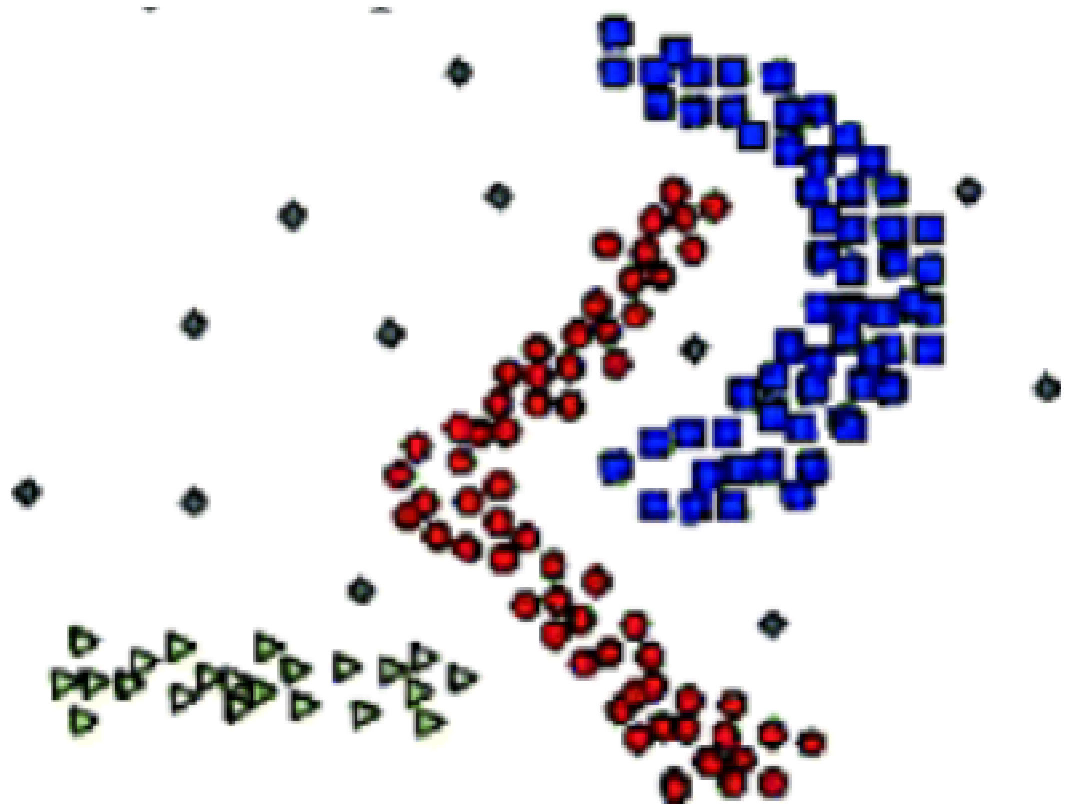
- Density-based clustering works by detecting areas where points
  are concentrated and where they are separated by areas that
  are empty or sparse.

- Unlike centroid based approaches, like K-means, or distribution
  based approaches, like Expectation Maximization, density
  based clustering can detect clusters of an arbitrary shape.

- This can be extremely helpful when clusters aren’t defined
  around a specific location or distribution.

- Unlike other clustering algorithms, such as K-means and
  hierarchical clustering, a density-based algorithm can discover
  clusters of any shape, size, or density in your data.

- Density-based clustering also can distinguish between data
  points which are part of a cluster and those which should be
  labeled as noise. Density-based clustering is especially useful
  when working with datasets with noise or outliers or when we
  don’t have prior knowledge about the number of clusters in the



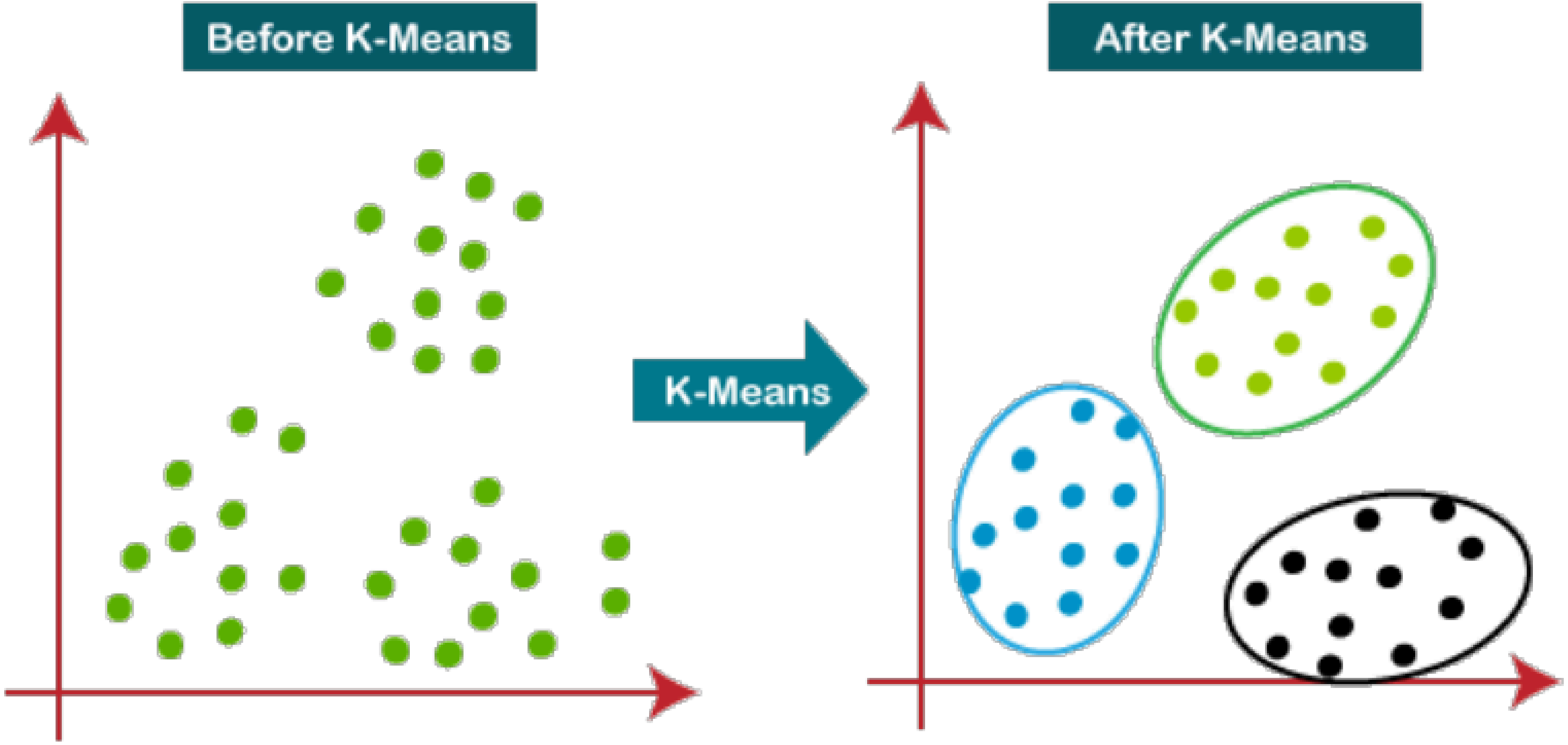
data.

K-Means Clustering

- K-means is a hard clustering approach, meaning each data point is assigned to a separate cluster and no probability associated with cluster membership.
- K-means works well when the clusters are of roughly equivalent size, and there are not significant outliers or changes in density across the data.
- K-means often performs poorly when the data is high dimensional or when clusters have significantly different sizes or densities.
- K-means is also especially sensitive to outliers since it tries to establish centroids based on the mean values of all values in the cluster and thus is susceptible to overfitting to include those outliers.
- Another centroid based approach to K-means is K-medoids.
- Medoids are representative objects of a dataset or a cluster within a dataset whose sum of distances to other objects in the cluster is minimal. Instead of having an arbitrary centroid be the center of the graph, the algorithm creates clusters by using individual data points as the medoid or center of the cluster.
- Since the K-medoids algorithm uses extant data points rather than arbitrary centroids it is less sensitive to outliers.
- 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.

K-Means Clustering



Pawan

Niroula[15]

K-Means Clustering – Algorithm

- 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 step 7.
  - Step-7: The model is ready.

K-Means Clustering – Example

- We are also given the information that we need to make3 clusters. Assign the given Points on the clusters usingK-means Algorithm for k=3.

Point Coordinates A1 (2,10) A2 (2,6)
A3 (11,11) A4 (6,9)
A5 (6,4)
A6 (1,2)
A7 (5,10) A8 (4,9)
A9 (10,12) A10 (7,5)
A11 (9,11) A12 (4,6)
A13 (3,10) A14 (3,8)
A15 (6,11)

K-Means Clustering – Example

- First, we will randomly choose 3 centroids from the given data. Let us consider A2 (2,6), A7 (5,10), and A15 (6,11) as the centroids of the initial clusters. Hence, we will consider that
    - Centroid 1=(2,6) is associated with cluster 1.
    - Centroid 2=(5,10) is associated with cluster 2.
    - Centroid 3=(6,11) is associated with cluster 3.

- Now we will find the Euclidean distance between each point and the centroids. Based on the minimum distance of each point from the centroids, we will assign the points to a cluster.

K-Means Clustering – Example

Point Distance from	Centroid 1 (2,6) Distance from Centroid 2	(5,10) Distance from Centroid 3	(6,11) Assigned Cluster

A1 (2,10) 4 3 4.123106 Cluster 2 A2 (2,6) 0 5 6.403124 Cluster 1 A3 (11,11) 10.29563 6.082763 5 Cluster 3 A4 (6,9) 5 1.414214 2 Cluster 2 A5 (6,4) 4.472136 6.082763 7 Cluster 1 A6 (1,2) 4.123106 8.944272 10.29563 Cluster 1 A7 (5,10) 5 0 1.414214 Cluster 2 A8 (4,9) 3.605551 1.414214 2.828427 Cluster 2 A9 (10,12) 10 5.385165 4.123106 Cluster 3 A10 (7,5) 5.09902 5.385165 6.082763 Cluster 1 A11 (9,11) 8.602325 4.123106 3 Cluster 3 A12 (4,6) 2 4.123106 5.385165 Cluster 1 A13 (3,10) 4.123106 2 3.162278 Cluster 2 A14 (3,8) 2.236068 2.828427 4.242641 Cluster 1 A15 (6,11) 6.403124 1.414214 0 Cluster 3

K-Means Clustering – Example

- At this point, we have completed the first iteration of the k-means clustering algorithm and assigned each point into a cluster.
- In the previous table, you can observe that the point that is closest to the centroid of a given cluster is assigned to the cluster.
- Now, we will calculate the new centroid for each cluster.
    - In cluster 1, we have 6 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), A12 (4,6), A14 (3,8).
    - To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster.
    - Hence, the new centroid for cluster 1 is (3.833, 5.167).
    - Hence, the new centroid for cluster 2 is (4, 9.6) In cluster 2, we have 5 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), and A13 (3,10).
    - In cluster 3, we have 4 points i.e. A3 (11,11), A9 (10,12), A11 (9,11), and A15 (6,11).
    - Hence, the new centroid for cluster 3 is (9, 11.25).
- Now that we have calculated new centroids for each cluster, we will calculate the distance of each data point from the new centroids.
- Then, we will assign the points to clusters based on their distance from the centroids.

K-Means Clustering – Example

Point Distance from Centroid 1 (3.833, 5.167) Distance from centroid 2 (4, 9.6) Distance from centroid 3 (9, 11.25) Assigned Cluster

A1 (2,10) 5.169 2.040 7.111 Cluster 2 A2 (2,6) 2.013 4.118 8.750 Cluster 1 A3 (11,11) 9.241 7.139 2.016 Cluster 3 A4 (6,9) 4.403 2.088 3.750 Cluster 2 A5 (6,4) 2.461 5.946 7.846 Cluster 1 A6 (1,2) 4.249 8.171 12.230 Cluster 1 A7 (5,10) 4.972 1.077 4.191 Cluster 2 A8 (4,9) 3.837 0.600 5.483 Cluster 2 A9 (10,12) 9.204 6.462 1.250 Cluster 3 A10 (7,5) 3.171 5.492 6.562 Cluster 1 A11 (9,11) 7.792 5.192 0.250 Cluster 3 A12 (4,6) 0.850 3.600 7.250 Cluster 1 A13 (3,10) 4.904 1.077 6.129 Cluster 2 A14 (3,8) 2.953 1.887 6.824 Cluster 2 A15 (6,11) 6.223 2.441 3.010 Cluster 2

K-Means Clustering – Example

- Now, we have completed the second iteration of the k-means clustering algorithm and assigned each point into an updated cluster.
- In the previous table, you can observe that the point closest to the new centroid of a given cluster is assigned to the cluster.
- Now, we will calculate the new centroid for each cluster for the third iteration.

- In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
- In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
- In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).

- At this point, we have calculated new centroids for each cluster.
- Now, we will calculate the distance of each data point from the new centroids.
- Then, we will assign the points to clusters based on their distance from the centroids.

K-Means Clustering – Example

Point	Distance from Centroid 1 (4, 4.6)	Distance from centroid 2 (4.143, 9.571)	Distance from centroid 3 (10, 11.333)	Assigned Cluster
A1 (2,10)	5.758	2.186	8.110	Cluster 2
A2 (2,6)	2.441	4.165	9.615	Cluster 1
A3 (11,11)	9.485	7.004	1.054	Cluster 3
A4 (6,9)	4.833	1.943	4.631	Cluster 2
A5 (6,4)	2.088	5.872	8.353	Cluster 1
A6 (1,2)	3.970	8.197	12.966	Cluster 1
A7 (5,10)	5.492	0.958	5.175	Cluster 2
A8 (4,9)	4.400	0.589	6.438	Cluster 2
A9 (10,12)	9.527	6.341	0.667	Cluster 3
A10 (7,5)	3.027	5.390	7.008	Cluster 1
A11 (9,11)	8.122	5.063	1.054	Cluster 3
A12 (4,6)	1.400	3.574	8.028	Cluster 1
A13 (3,10)	5.492	1.221	7.126	Cluster 2
A14 (3,8)	3.544			

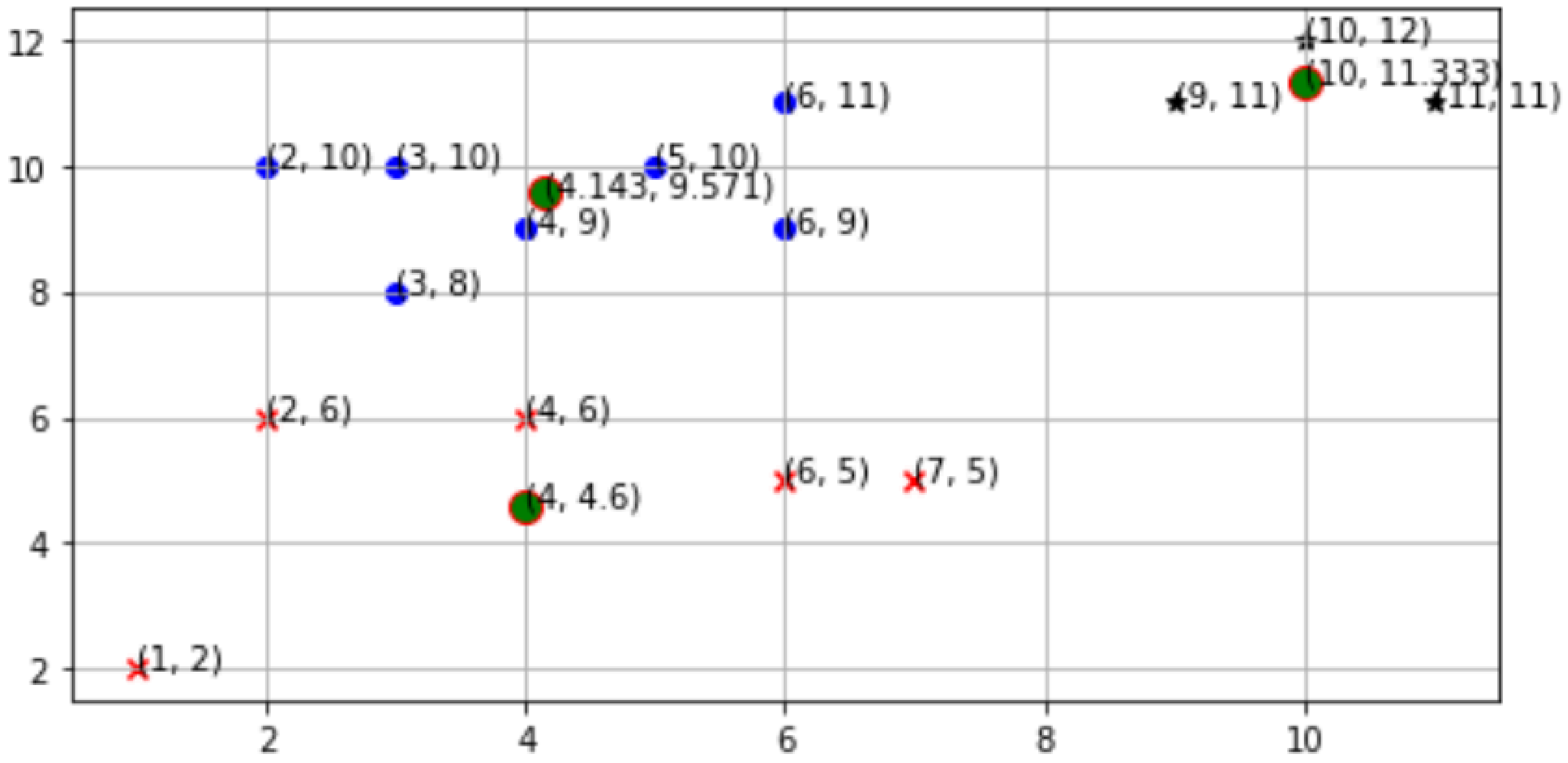
1.943 7.753 Cluster 2 A15 (6,11) 6.705 2.343 4.014 Cluster 2

K-Means Clustering – Example

- Now, we have completed the third iteration of the k-means clustering algorithm and assigned each point into an updated cluster.
- In the previous table, you can observe that the point that is closest to the new centroid of a given cluster is assigned to the cluster.
- Now, we will calculate the new centroid for each cluster for the third iteration.
  - In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
  - In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
  - In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).
- Here, you can observe that no point has changed its cluster compared to the previous iteration. Due to this, the centroid also remains constant.
- Therefore, we will say that the clusters have been stabilized. Hence, the clusters obtained after the third iteration

are the final clusters made from the given dataset.

K-Means Clustering – Example


(10, 12)
(10, 11.333)
(9, 11)
(11, 11)
(6, 11)
(2, 10)
(3, 10)
(5, 10)
(4.143, 9.571)
(4, 9)
(6, 9)
(3, 8)
(2, 6)
(4, 6)
(6, 5)
(7, 5)
(4, 4.6)
(1, 2)
2
4
6
8
10
12

K-Means Clustering – Advantages

- Following are some of the advantages of the k-means clustering algorithm.

  - **Easy to implement**: K-means clustering is an iterable algorithm and a relatively simple algorithm. In fact, we can also perform k-means clustering manually as we did in the numerical example.

  - **Scalability**: We can use k-means clustering for even 10 records or even 10 million records in a dataset. It will give us results in both cases.

  - **Convergence**: The k-means clustering algorithm is guaranteed to give us results. It guarantees convergence. Thus, we will get the result of the execution of the algorithm for sure.

  - **Generalization**: K-means clustering doesn’t apply to a specific problem. From numerical data to text documents, you can use the k-means clustering algorithm on any dataset to perform clustering. It can also be applied to datasets of different sizes having entirely different distributions in the dataset. Hence, this algorithm is completely generalized.

  - **Choice of centroids**: You can warm-start the choice of centroids in an easy manner. Hence, the algorithm allows you to choose and assign centroids that fit well with the dataset.

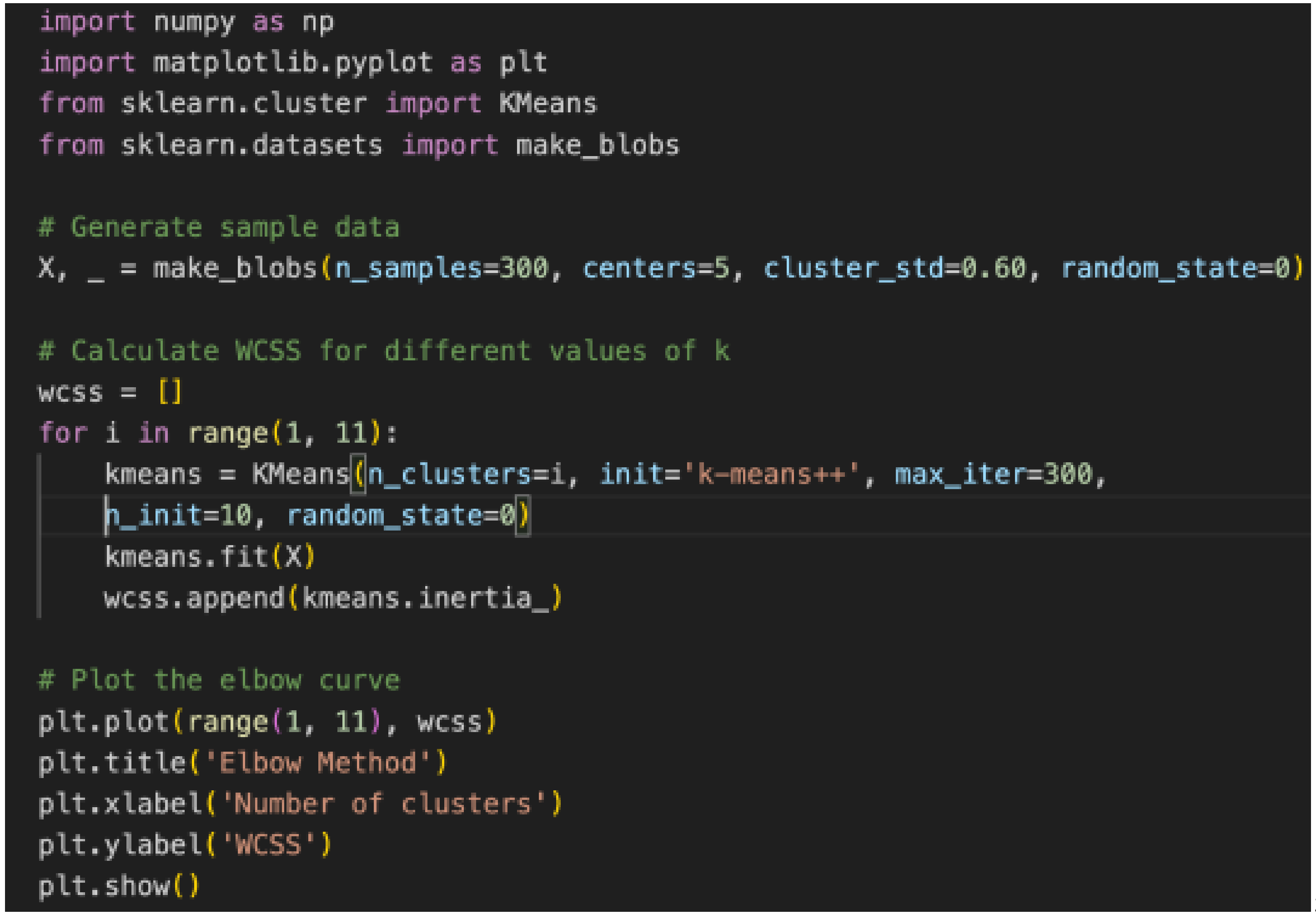
K-Means Clustering – Disadvantages

- With all the advantages, the k-means algorithm has certain disadvantages too which are discussed below.
  - **Deciding the number of clusters**: In k-means clustering, you need to decide the number of clusters by using the elbow method.
  - **Choice of initial centroids**: The number of iterations in the clustering process completely depends on the choice of centroids. Hence, you need to properly choose the centroids in the initial step for maximizing the efficiency of the algorithm.
  - **Effect of outliers**: In the execution of the k-means clustering algorithm, we use all the points in a cluster to determine the centroids for the next iteration. If there are outliers in the dataset, they highly affect the position of the centroids. Due to this, the clustering becomes inaccurate. To avoid this, you can try to identify outliers and remove them in the data cleaning process.
  - **Curse of Dimensionality**: If the number of dimensions in the dataset increases, the distance of the data points from a given point starts converging to a specific value. Due to this, k-means clustering that calculates the clusters based on the distance between the points becomes inefficient. To overcome this problem, you can use advanced clustering algorithms like spectral clustering. Alternatively, you can also try to reduce the dimensionality of the dataset while data preprocessing

K-Means Clustering – Choosing Number of Clusters

- The elbow method is a heuristic used to determine the optimal number of clusters in partitioning clustering algorithms such as k-means, k-modes, and k-prototypes clustering algorithms.
- With the increase in the number of clusters, the total cluster variance for a given dataset decreases rapidly.
- After a point, the total cluster variance becomes almost constant or the rate of decrease in total cluster variance becomes very low.
- In such a case, when we plot total cluster variance vs the number of clusters, the graph takes a shape of a bent elbow.
- The point after which the decrease in cluster variance becomes stagnant is termed the elbow.
- The point at the elbow is chosen as the optimal number of clusters.

K-Means Clustering – Choosing Number of Clusters - The method involves plotting the within-cluster sum of squares (WCSS) against the number of clusters and identifying the "elbow" point, where the rate of decrease in WCSS slows down.



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# Generate sample data
X, _ = make_blobs(n_samples=300, centers=5, cluster_std=0.60, random_state=0)

# Calculate WCSS for different values of k
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300,
    n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

# Plot the elbow curve
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Hierarchical Clustering

- A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed.
- Unlike K-means, hierarchical clustering doesn't require the number of clusters to be specified in advance. Instead, it generates a tree-like hierarchy of clusters, known as a dendrogram, which can be cut at different levels to obtain different numbers of clusters.
- There are two approaches to performing hierarchical cluster analysis:
  - **Agglomerative Clustering**:
    - In agglomerative clustering a bottom-up approach starts with individual data points and successively merges clusters by compute the proximity matrix of all the clusters at the current level of the hierarchy to create a tree-like structure.
    - Once one level of clusters has been created where all the clusters have no or low inter-cluster similarity, the algorithm moves to the set of newly created clusters and repeats the process until there is one root node at the top of the hierarchical graph.

Hierarchical Clustering – Agglomerative Clustering

- 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.

Hierarchical Clustering – Agglomerative Clustering

- Algorithm for Agglomerative Hierarchical Clustering is:
  - Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
  - Consider every data point as an individual cluster
  - Merge the clusters which are highly similar or close to each other.
  - Recalculate the proximity matrix for each cluster
  - Repeat Step 3 and 4 until only a single cluster remains.
- The proximity matrix is a matrix consisting of the distance between each pair of data points. The distance is computed by a distance function.
- Euclidean distance is one of the most commonly used distance functions.

Hierarchical Clustering – Agglomerative Clustering

- To group the data points in a cluster, a linkage function is used where the values in the proximity matrix are taken and the data points are grouped based on similarity.

- The newly formed clusters are linked to each other until they form a single cluster containing all the data points.

- The most common linkage methods are as follows:

- Complete linkage: The maximum of all pairwise distance between elements in each pair of clusters is used to measure the distance between two clusters.

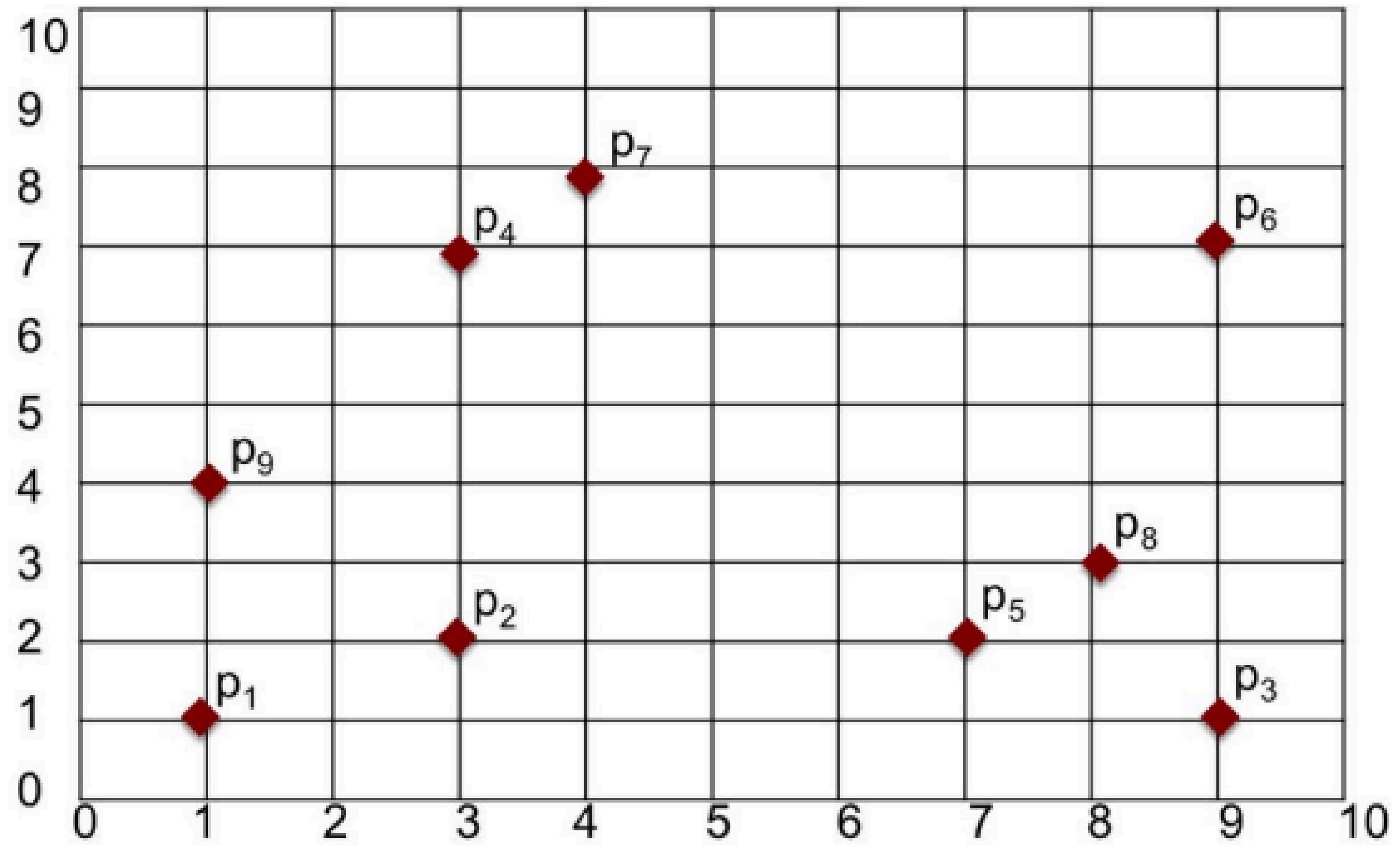
- Single linkage: The minimum of all pairwise distance between elements in each pair of clusters is used to measure the distance between two clusters.

- Average linkage: The average of all pairwise distances between elements in each pair of clusters is used to measure the distance between two clusters.

- Centroid linkage: Before merging, the distance between the two clusters' centroids are considered.

Hierarchical Clustering – Agglomerative Clustering

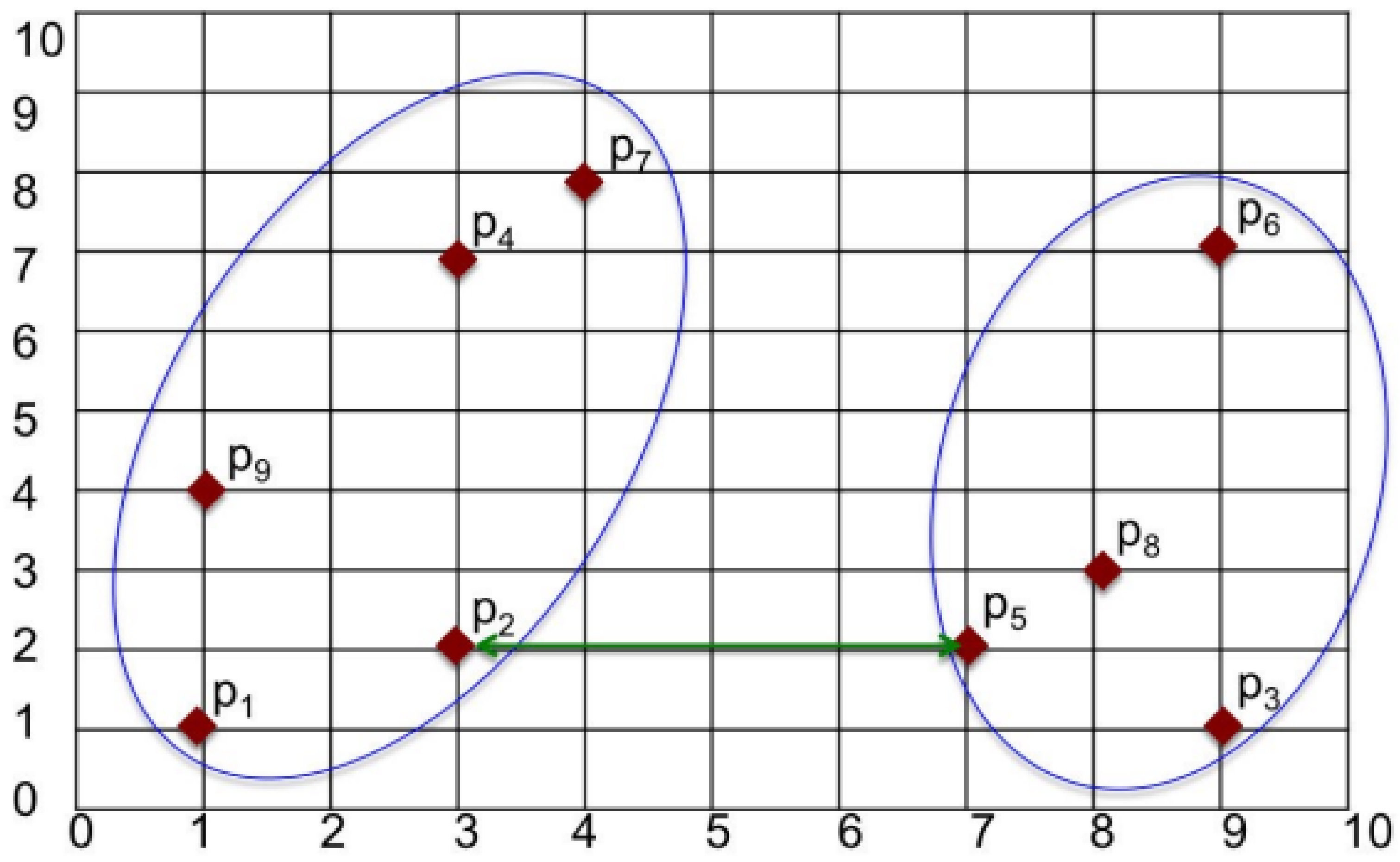
- Initial Data



Point	x	y
p_1	1	1
p_2	3	2
p_3	9	1
p_4	3	7
p_5	7	2
p_6	9	7
p_7	4	8
p_8	8	3
p_9	1	4

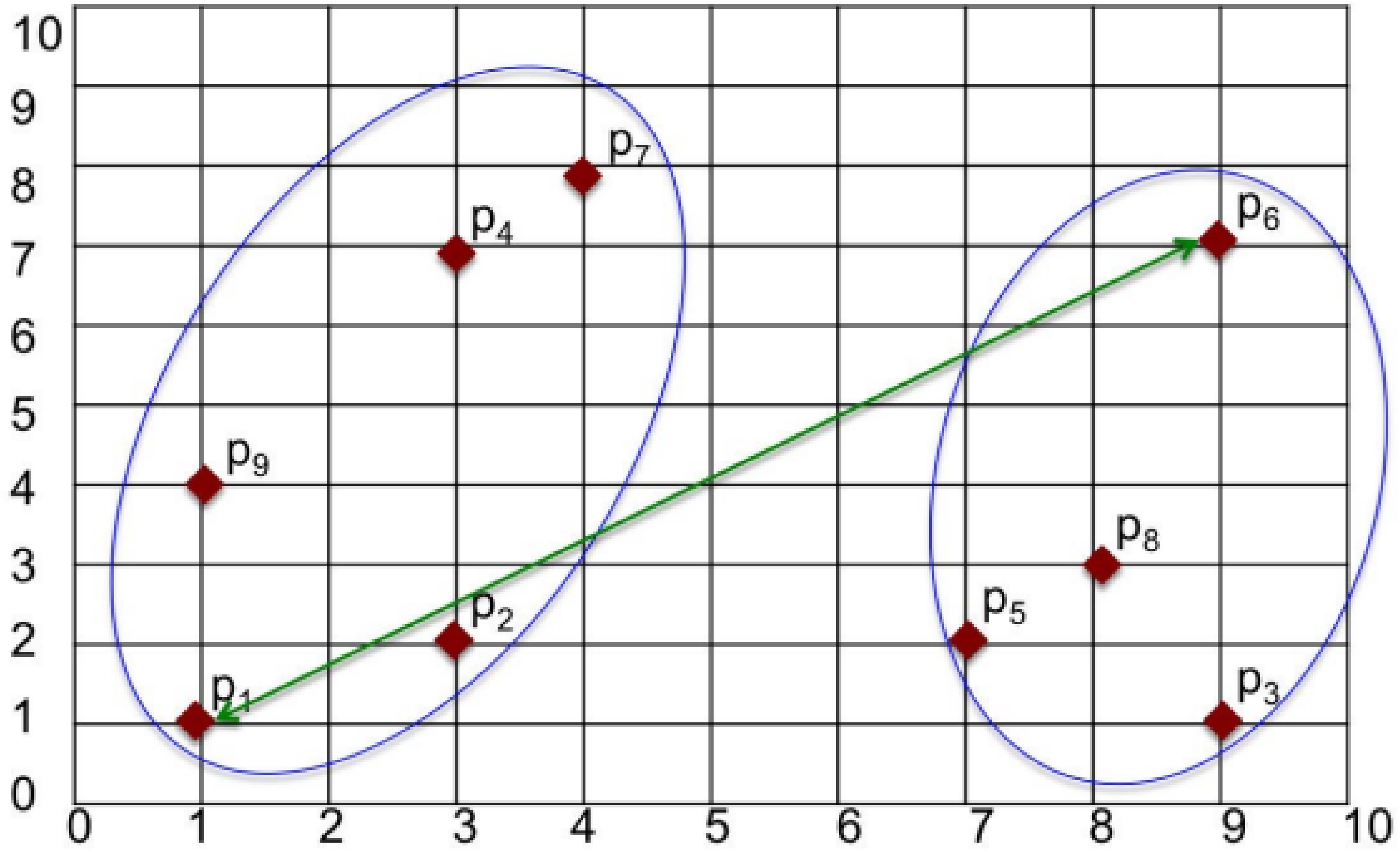
Hierarchical Clustering – Agglomerative Clustering

- **Single Linkage**



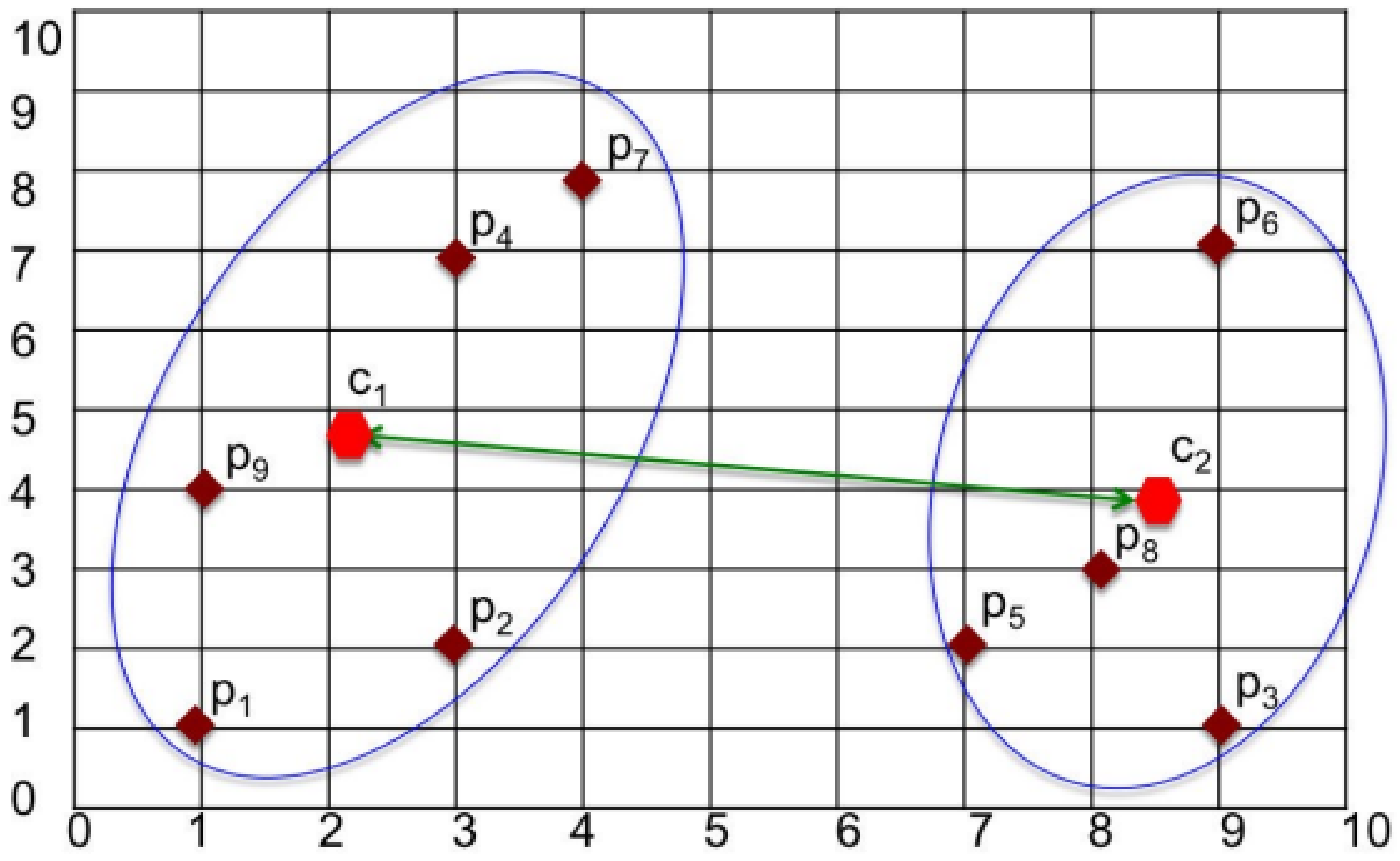
Hierarchical Clustering – Agglomerative Clustering

- **Complete Linkage**



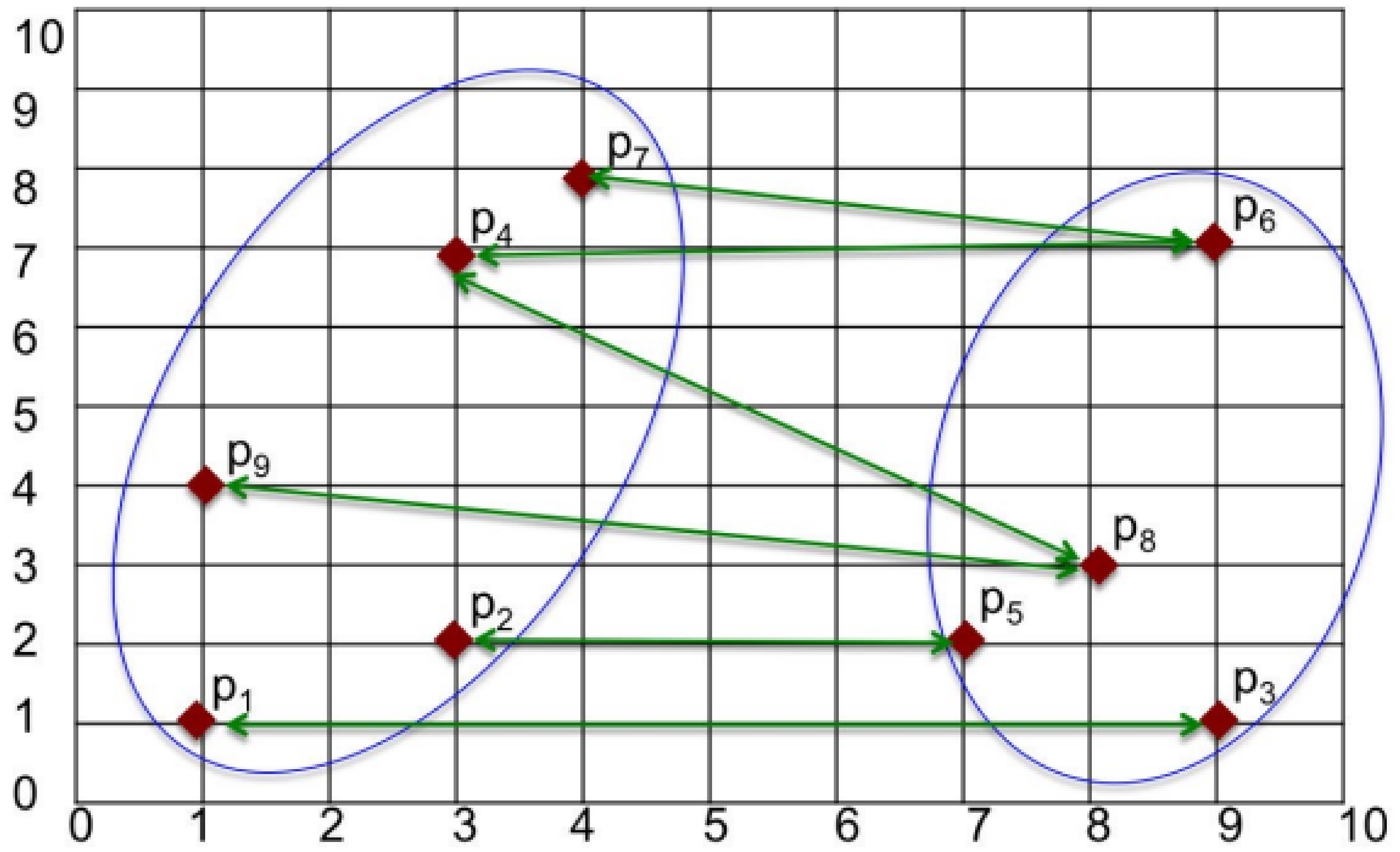
Hierarchical Clustering – Agglomerative Clustering

- Centroid Linkage

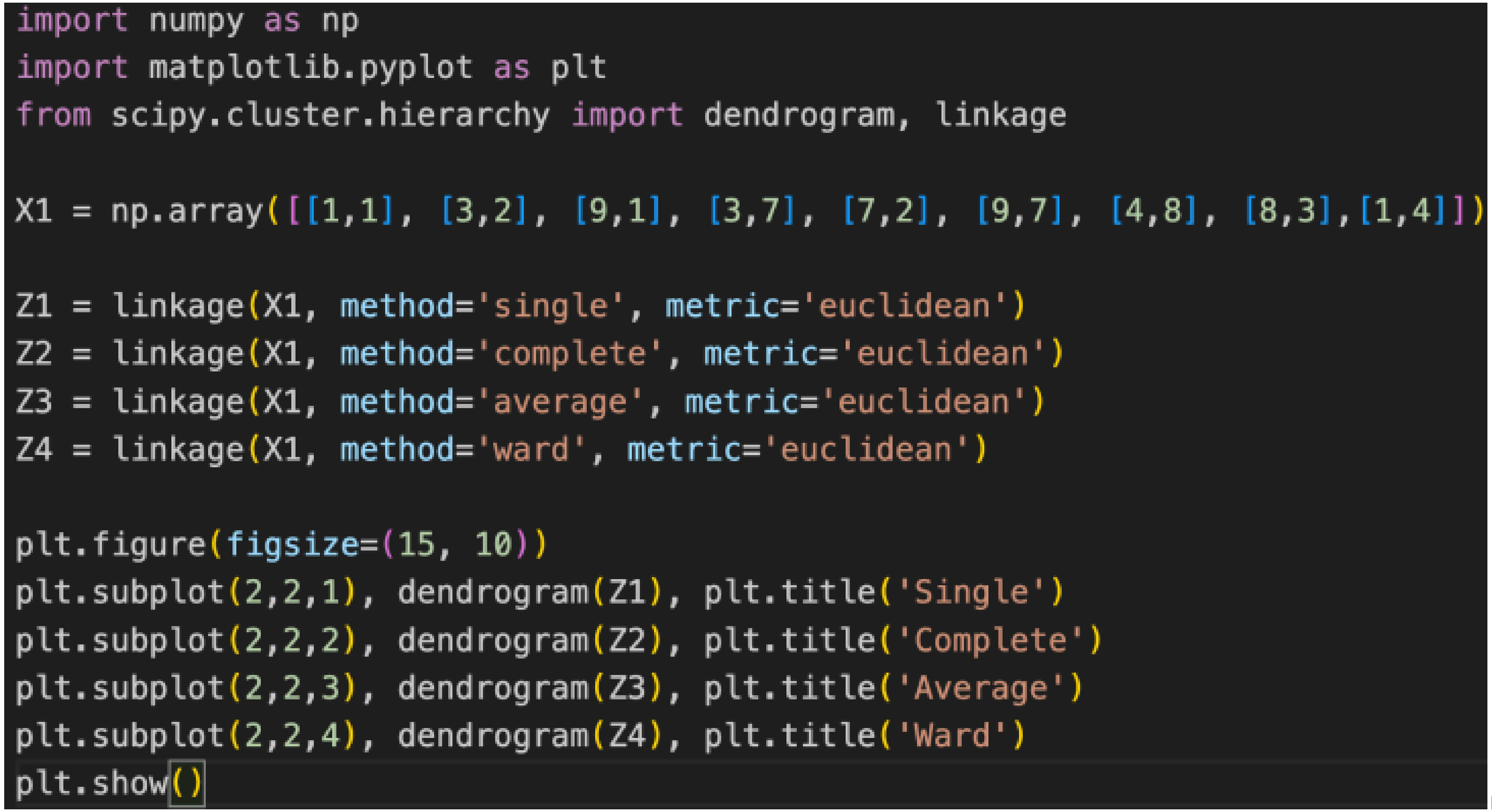


Hierarchical Clustering – Agglomerative Clustering

- Average Linkage



Hierarchical Clustering – Agglomerative Clustering



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

X1 = np.array([[1,1], [3,2], [9,1], [3,7], [7,2], [9,7], [4,8], [8,3],[1,4]])

Z1 = linkage(X1, method='single', metric='euclidean')
Z2 = linkage(X1, method='complete', metric='euclidean')
Z3 = linkage(X1, method='average', metric='euclidean')
Z4 = linkage(X1, method='ward', metric='euclidean')

plt.figure(figsize=(15, 10))
plt.subplot(2,2,1), dendrogram(Z1), plt.title('Single')
plt.subplot(2,2,2), dendrogram(Z2), plt.title('Complete')
plt.subplot(2,2,3), dendrogram(Z3), plt.title('Average')
plt.subplot(2,2,4), dendrogram(Z4), plt.title('Ward')
plt.show()
```

Hierarchical Clustering – Agglomerative Clustering

Hierarchical Clustering – Divisive Clustering

- We can say that the Divisive Hierarchical clustering is precisely the opposite of the Agglomerative Hierarchical clustering.

- In Divisive Hierarchical clustering, we take into account all of the data points as a single cluster and in every iteration, we separate the data points from the clusters which aren’t comparable.

- At the end, we are left with N clusters.

- The clusters with the largest Sum of Squared Errors (SSE) are then partitioned further using a flat clustering method.

- The algorithm stops either when it reaches individual nodes or some minimum SSE. Divisive partitioning allows greater flexibility in terms of both the hierarchical structure of the tree and the level of balance in the different clusters.

- Divisive hierarchical clustering can be faster than agglomerative hierarchical clustering, especially when the data doesn’t require constructing the tree all the way down to individual data points.

Hierarchical Clustering – Divisive Clustering

Hierarchical Clustering – Advantages

- **No Need for Predefined Number of Clusters:** Unlike K-means clustering, hierarchical clustering doesn't require specifying the number of clusters in advance. It produces a hierarchy of clusters, allowing the user to choose the number of clusters based on the dendrogram.

- **Interpretability:** The hierarchical structure of clusters provided by dendrograms allows for easy interpretation. It provides insights into the relationships and similarities between data points at different levels of granularity.

- **Robustness to Outliers:** Hierarchical clustering can be robust to outliers since it aggregates clusters based on distances, rather than centroids. Outliers may not significantly affect the overall clustering structure.

- **Flexibility in Cluster Shape:** Hierarchical clustering can handle clusters of various shapes and sizes, as it doesn't assume any specific cluster shape. It is suitable for datasets with non-linear boundaries and irregular cluster shapes.

- **Hierarchical Representation:** It offers a hierarchical representation of the data, allowing users to explore clusters at different levels of granularity. This can be particularly useful for exploratory data analysis.

Hierarchical Clustering – Disadvantages

1. **Computationally Intensive**: Hierarchical clustering can be computationally expensive, especially for large datasets. The time complexity is typically �(n^3) *n* is the number of data points. This makes it less practical for very large datasets.

2. **Memory Requirement**: Building the entire dendrogram requires storing the pairwise distances between all data points, leading to high memory requirements, especially for large datasets. This can limit its scalability.

3. **Sensitivity to Distance Metric and Linkage Method**: The choice of distance metric and linkage method can significantly impact the clustering results. Different combinations may lead to different clustering structures, making it challenging to determine the optimal configuration.

4. **Lack of Scalability**: Due to its computational complexity and memory requirements, hierarchical clustering may not be suitable for very large datasets with millions of data points. Other clustering algorithms, such as K-means or DBSCAN, are often preferred for such cases.

5. **Difficulty in Interpreting Large Dendrograms**: Dendrograms can become complex and difficult to interpret, especially for datasets with a large number of data points. Understanding the hierarchical structure and identifying meaningful clusters may require expert knowledge and domain expertise.

Density Based Clustering – DBSCAN Algortihm

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular unsupervised clustering algorithm used for discovering clusters of arbitrary shape and size in spatial data.
- Unlike other clustering algorithms like K-Means, which assume a specific shape or number of clusters, DBSCAN is designed to discover clusters based on the density of data points.
- Parameters Required For DBSCAN Algorithm
  - Eps (ε) :
    - It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors.
    - If the eps value is chosen too small then a large part of the data will be considered as an outlier.
    - If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters.
    - One way to find the eps value is based on the k-distance graph.
  - MinPts:
    - Minimum number of neighbors (data points) within eps radius.
    - The larger the dataset, the larger value of MinPts must be chosen.
    - As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

Density Based Clustering – DBSCAN Algortihm

- DBSCAN Algorithm consists of three types of points:
  - Core Point(x): Data point that has at least minPoints (n) within epsilon (ε) distance.
  - Border Point(y): Data point that has at least one core point within epsilon (ε) distance and lower than minPoints (n) within epsilon (ε) distance from it.
  - Noise Point(z): Data point that has no core points within epsilon (ε) distance.

Density Based Clustering – DBSCAN Algorithm

- **Algorithm Steps**:
    - **Parameter Selection**:
        - The DBSCAN algorithm requires two main parameters:
            - eps (epsilon): The radius that defines the neighborhood around a data point.
            - minPts: The minimum number of points required within the eps radius to form a dense region (cluster). -

    Cluster Formation:
        - The algorithm starts by randomly selecting an unvisited data point.
        - If the point has at least minPts points within the eps radius, it is considered a core point, and a new cluster is formed.
        - The core point and its neighboring points (within eps radius) are added to the new cluster.
        - The algorithm then recursively visits the neighbors of the core point's neighbors, adding them to the same cluster if they are also dense points or border points.
        - This process continues until all points in the cluster are found.
    - **Noise Handling**:
        - After a cluster is formed, the algorithm moves to the next unvisited point and repeats the process.
        - If a point is not a core point and does not belong to any cluster, it is labeled as noise.

Density Based Clustering – DBSCAN Algorithm

- **Advantages**:
    - DBSCAN can discover clusters of arbitrary shape and size.
    - It is robust to outliers and noise points, as they are automatically labeled as noise.
    - It does not require the number of clusters to be specified in advance.
- **Limitations**:
    - DBSCAN is sensitive to the choice of eps and minPts parameters, which can significantly affect the clustering results.
    - It may struggle with clusters of varying densities, as a single eps value may not be suitable for all clusters. - DBSCAN can be computationally expensive for large datasets, as it needs to calculate the distances between all pairs of points.

Density Based Clustering – DBSCAN Algorithm

K-means Clustering DBSCAN Distance based clustering Density based

clustering

Every observation becomes a part of some cluster eventually

Clearly separates outliers and clusters observations in high density areas

Build clusters that have a shape of a hypersphere within clusters.

Build clusters that have an arbitrary shape or clusters

Sensitive to outliers Robust to outliers

Require no. of clusters as input Doesn’t require no. of clusters as input

Density Based Clustering – Applications

- Geospatial Data Analysis

- Geographic Information Systems (GIS): DBSCAN is used to identify clusters of geographical data points, such as finding areas with high densities of certain species, detecting hotspots of crime, or clustering regions based on similar environmental conditions.

- Earthquake Analysis: Clustering seismic activity to identify patterns and regions with high seismic activities.

- Image Processing and Computer Vision

- Image Segmentation: Segmenting images into meaningful clusters based on pixel intensity or color, useful in medical imaging (e.g., tumor detection) and object recognition.

- Feature Detection: Identifying regions of interest in an image, such as detecting clusters of features in satellite imagery for urban planning or agriculture.

- Market Basket Analysis

- Customer Segmentation: Grouping customers based on purchasing behavior, which helps in targeted marketing and personalized recommendations.

- Product Recommendation: Identifying products that are frequently bought together and suggesting them to customers.

Pawan Niroula

Density Based Clustering – Applications

- Anomaly Detection

- Fraud Detection: Detecting fraudulent activities in financial transactions by identifying transactions that do not fit into any cluster (i.e., anomalies).
- Network Security: Identifying unusual patterns in network traffic that may indicate security breaches or attacks.

- Social Network Analysis
  - Community Detection: Identifying communities or groups within social networks, such as detecting groups of friends or users with similar interests.
  - Influence Analysis: Analyzing patterns of influence and information spread within social networks.
- Environmental Science
  - Climate Data Analysis: Clustering weather stations based on climate patterns to identify regions with similar weather conditions.
  - Pollution Monitoring: Detecting clusters of pollution measurements to identify polluted areas and analyze pollution sources.
- Retail and E-commerce
  - Store Location Optimization: Identifying optimal store locations based on customer density and purchasing patterns.
  - Inventory Management: Grouping products based on sales patterns to optimize inventory levels and storage.
- Transportation and Mobility
  - Traffic Analysis: Analyzing traffic patterns to identify congested areas and optimize traffic flow.
  - Public Transport Optimization: Clustering passenger usage patterns to optimize routes and schedules.