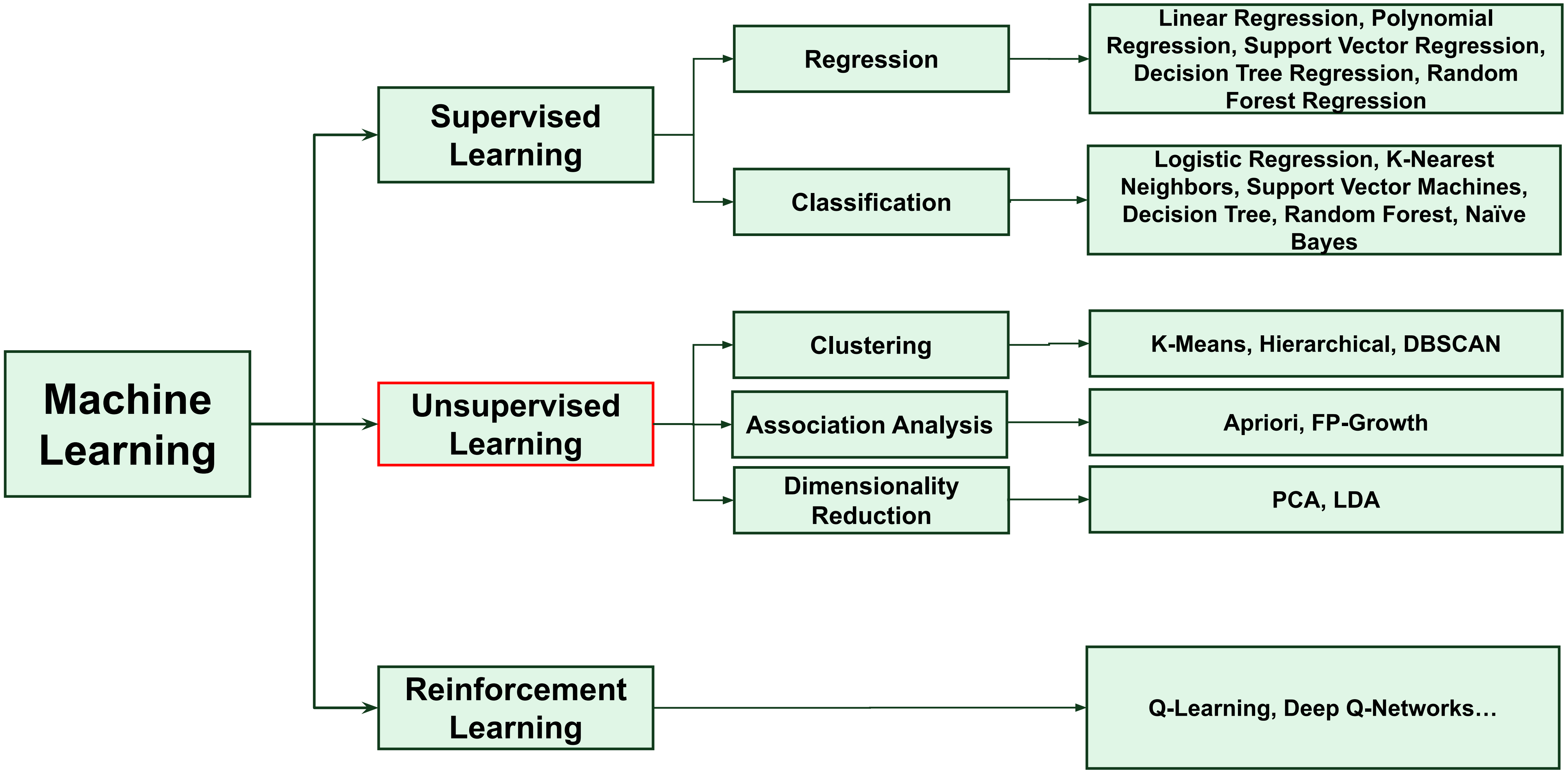
Machine Learning: Clustering Algorithms

Instructor: Sabina Mammadova

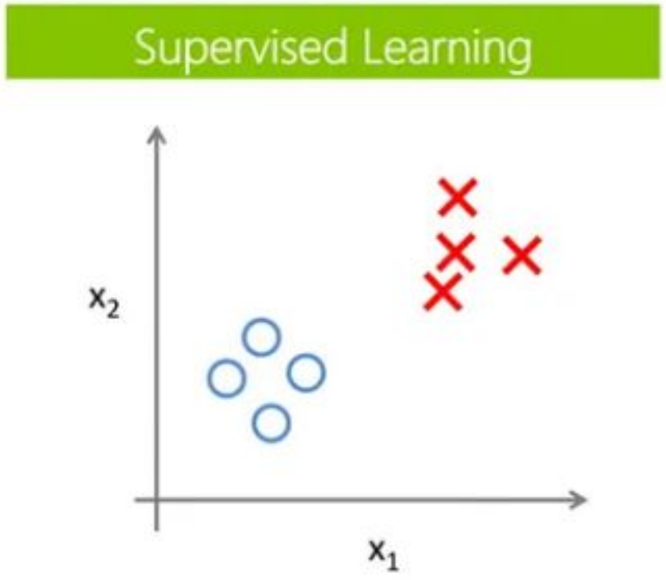
Agenda

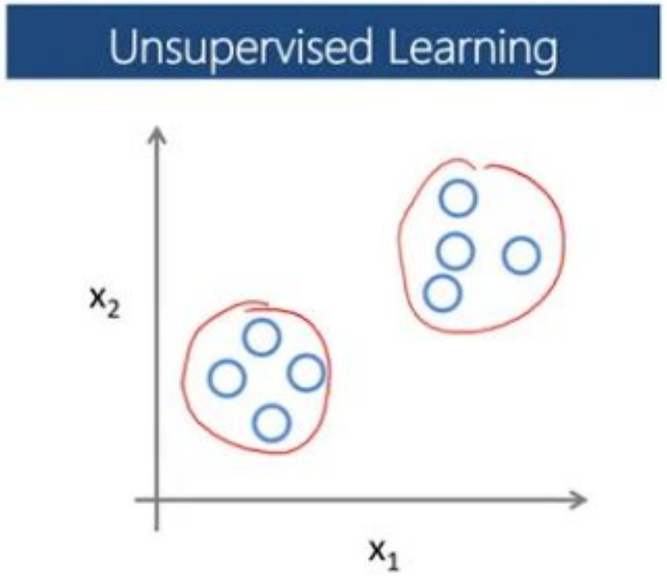
- K-Means Algorithm
- Choosing K – Elbow method & Silhouette Analysis
- Unsupervised Learning
- Agglomerative Clustering
- PCA

Machine Learning Algorithms


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Regression
Classification
Clustering
Association Analysis
Dimensionality Reduction
Linear Regression, Polynomial Regression, Support Vector Regression, Decision Tree Regression, Random Forest Regression
Logistic Regression, K-Nearest Neighbors, Support Vector Machines, Decision Tree, Random Forest, Naïve Bayes
K-Means, Hierarchical, DBSCAN
Apriori, FP-Growth
PCA, LDA
Q-Learning, Deep Q-Networks…

Difference between Supervised and Unsupervised Learning





- Input data is *labelled*
- There is *a training* phase
- Data is modelled based on training dataset
- Known number of classes (for classification)

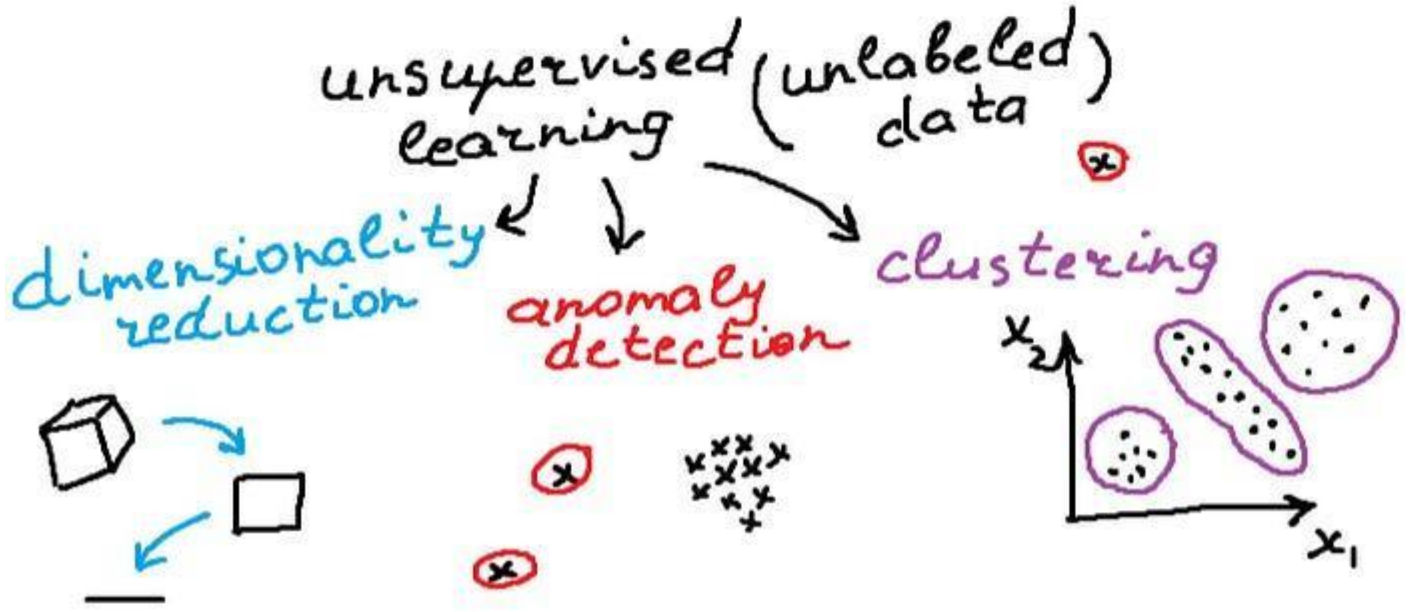
- Input data is *unlabeled*
- There is *no training* phase
- Uses properties of given data for clustering
- Unknown number of classes

What is Unsupervised Learning?

- **Unsupervised learning** is a type of machine learning where an algorithm learns patterns and structures from data **without labeled outputs**. Unlike supervised learning, where models are trained using labeled input-output pairs (e.g., images labeled as "cat" or "dog"), unsupervised learning works with **unlabeled data** and tries to discover hidden patterns or relationships within it.
- It is mainly used for **data exploration**, **feature learning**, and **preprocessing** in machine learning pipelines.
- **Hard to Evaluate**: No ground truth to compare results with.
- **Interpretability**: The patterns found by the algorithm may not always be meaningful or useful.

Unsupervised Learning Algorithms

- **Dimensionality Reduction**— the task of reducing the number of input features in a dataset,
- **Anomaly Detection**— the task of detecting instances that are very different from the norm, and
- **Clustering** — the task of grouping similar instances into clusters.



Unsupervised Learning Algorithms

- Unsupervised learning includes **transformations**, **clustering**, and **anomaly detection** algorithms, each with real-world applications.
- **Transformations** like **dimensionality reduction** help in **bioinformatics** for gene expression analysis, while **topic extraction** is used in **news categorization** and **social media monitoring**.
- **Clustering** groups similar data points, enabling **customer segmentation in marketing** and **facial recognition in social media**.
- **Anomaly detection algorithms** identify unusual patterns, making them essential for **fraud detection in banking**, **cybersecurity threat detection**, and **fault detection in manufacturing**, helping to recognize deviations from normal behavior.

Unsupervised Learning Algorithms

- One of the biggest challenges in unsupervised learning is evaluating whether the algorithm has learned something useful. Since there are no labels in the data, we don’t have a correct answer to compare the model’s output against. For example, imagine using a clustering algorithm to group customers based on their purchasing behavior. The algorithm might group people who buy luxury items separately from those who buy everyday essentials. While this is a valid way to categorize customers, it may not be what we were expecting—perhaps we wanted to group them based on shopping frequency instead. However, because there are no predefined labels, we cannot directly tell the algorithm what we want. The only way to assess the results is through manual inspection.
- Due to this challenge, unsupervised learning is mostly used for exploratory data analysis, helping data scientists uncover hidden patterns in the data rather than making final decisions in automated systems. Another important use of unsupervised learning is preprocessing for supervised learning. For example, dimensionality reduction can simplify complex data, making it easier for supervised algorithms to work efficiently while also improving their accuracy. Additionally, techniques like scaling and normalization, which adjust data values to a consistent range, are also considered unsupervised because they don’t rely on labeled data. These preprocessing steps are crucial for improving the performance of machine learning models.

K-Means Algorithm

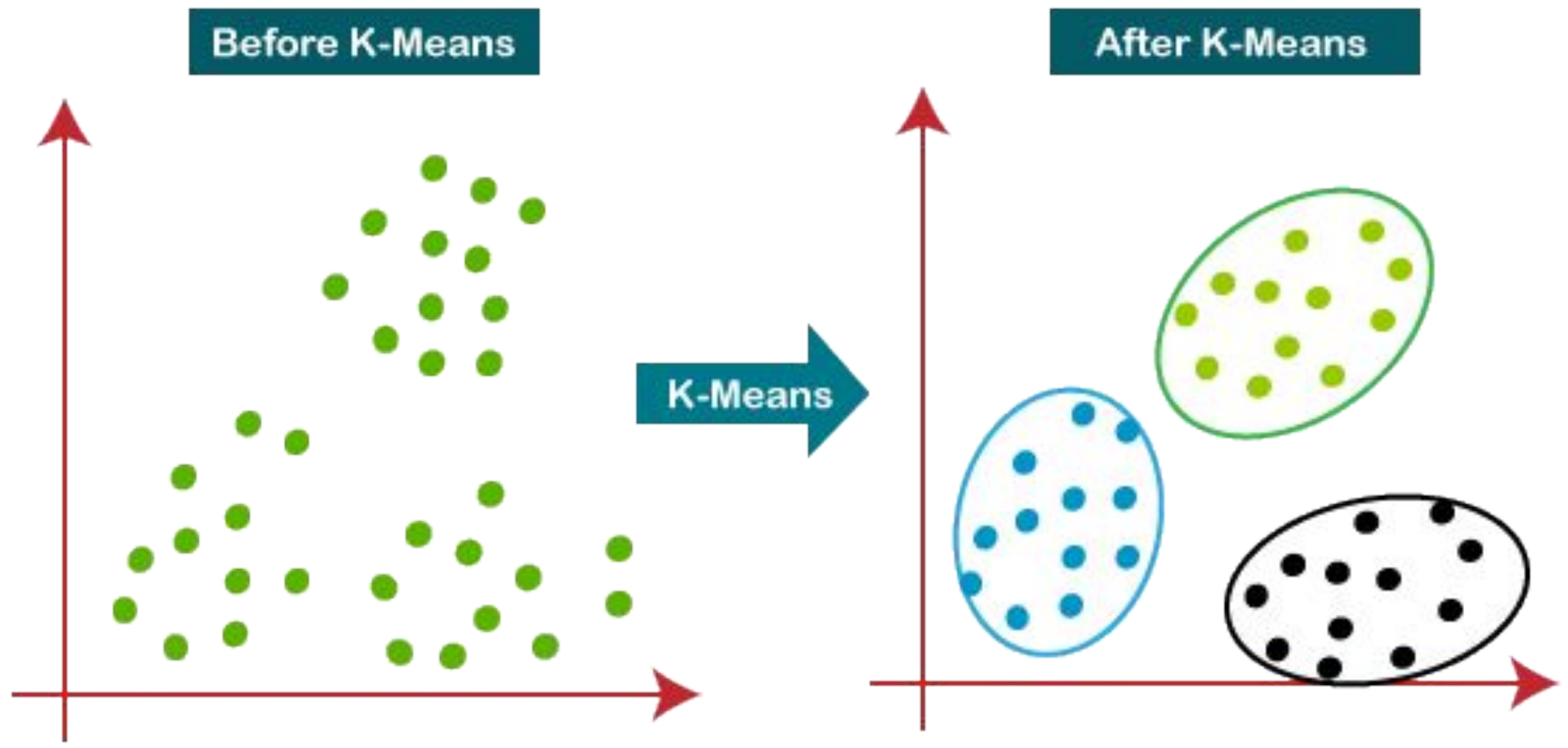
What is K-Means Algorithm?

- 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 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 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*.

What is K-Means Algorithm?

The k-means clustering algorithm mainly performs two tasks:

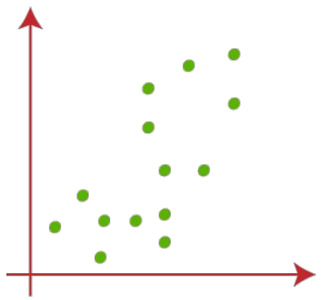
- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

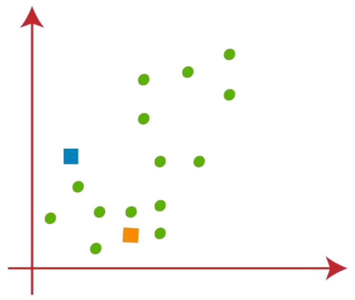


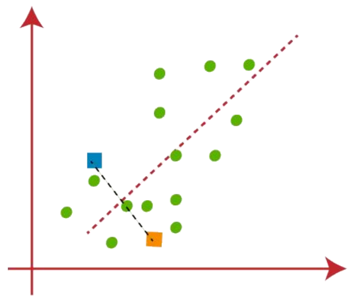
How does the K-Means Algorithm Work?

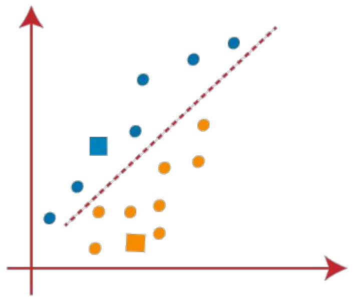
- **Step 1:** Select the number K to decide the number of clusters.
- **Step 2:** Select random K points or centroids. (It can be other from the input dataset).
- **Step 3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
- **Step 4:** Calculate the variance 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.

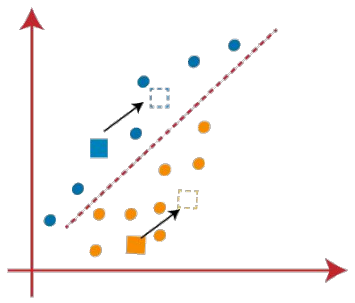
How does the K-Means Algorithm Work?

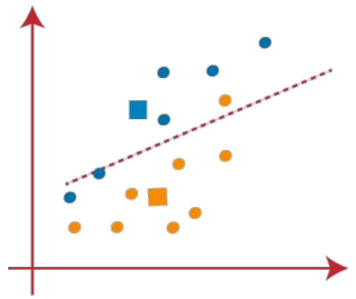

1

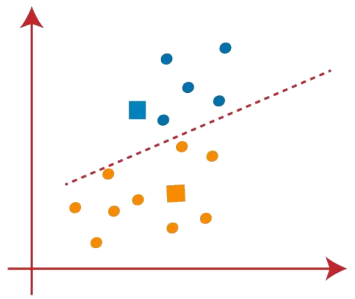

2

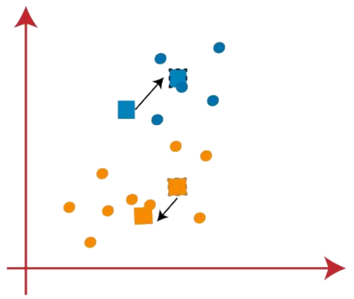

3

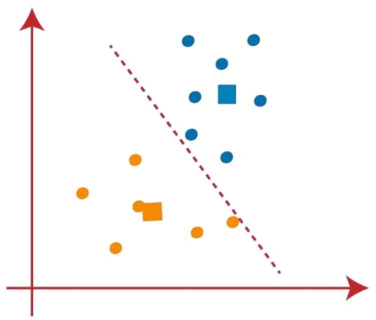

4

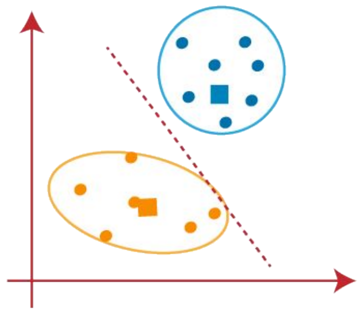

5


6


7


8

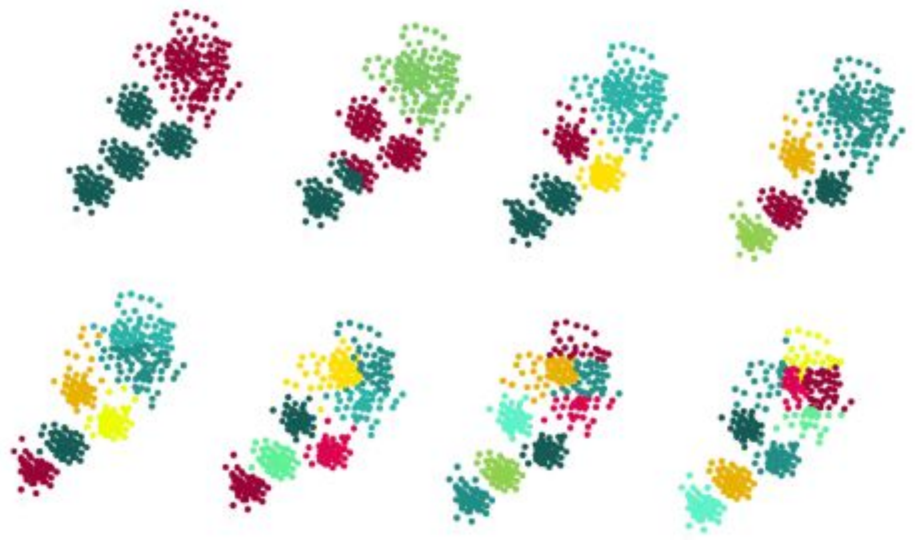

9

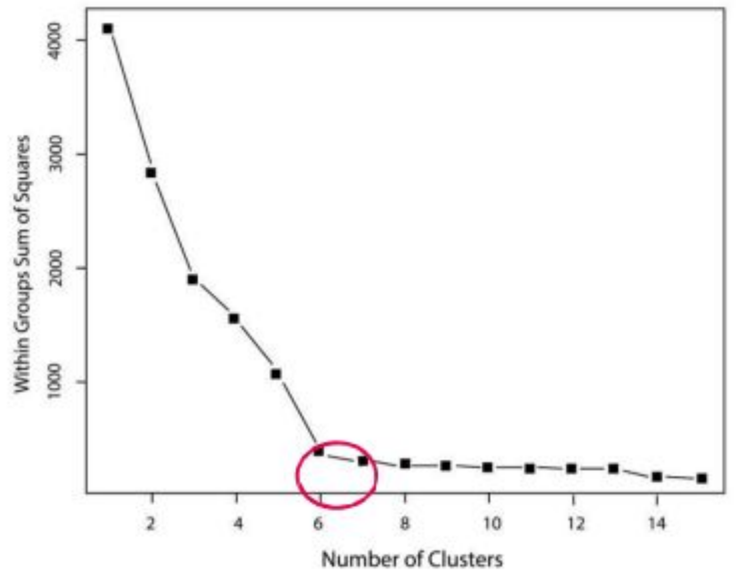

10

Choosing K – Elbow method & Silhouette Analysis

Elbow Method – optimal value of K

- Perform K-Means clustering on the dataset for a range of values of K (e.g., K = 1 to 10).
- As K increases, inertia will decrease, because adding more clusters reduces the distance between data points and their centroids.
- The "elbow" point in the plot is where the rate of decrease sharply slows down. This point suggests a good balance between the number of clusters and the amount of variance explained.



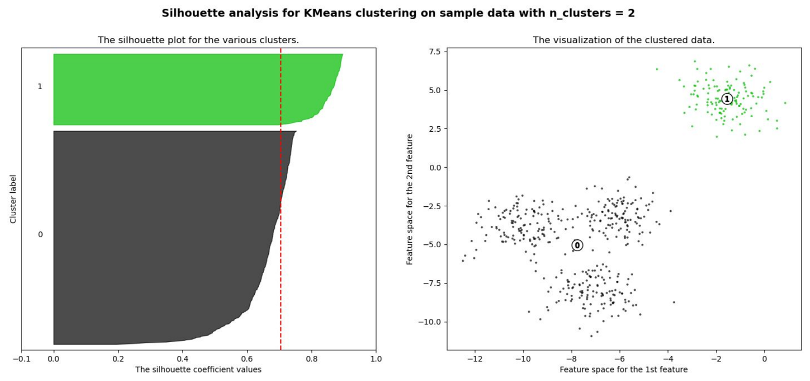


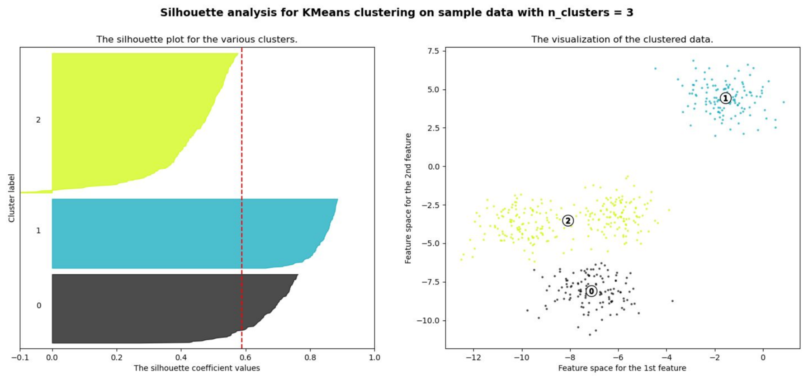
Silhouette Analysis

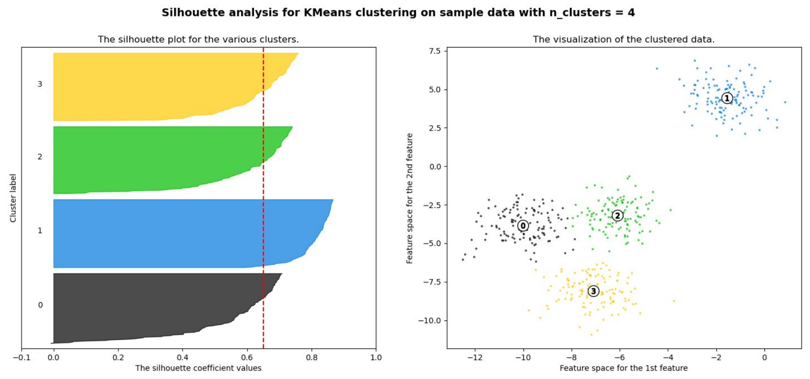
- Silhouette analysis can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1].
- Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.
- To calculate the ***Silhouette coefficient***, we need to define the mean distance of a point to all other points in its cluster (a(i)) and also define the mean distance to all other points in the closest cluster (b(i)).

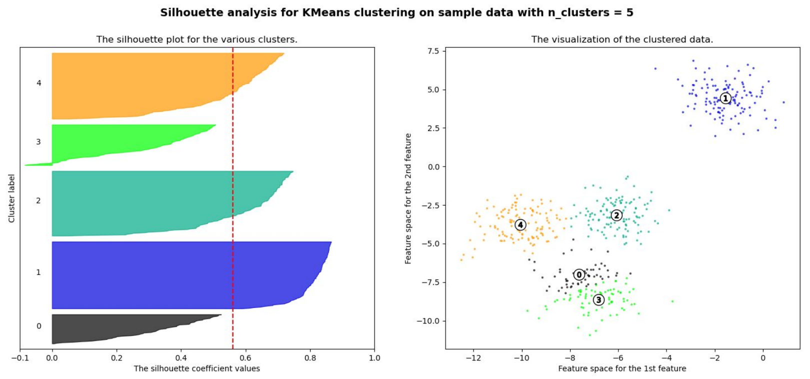
$$S(i) = \frac{(b(i) - a(i))}{\max(b(i), a(i))}$$

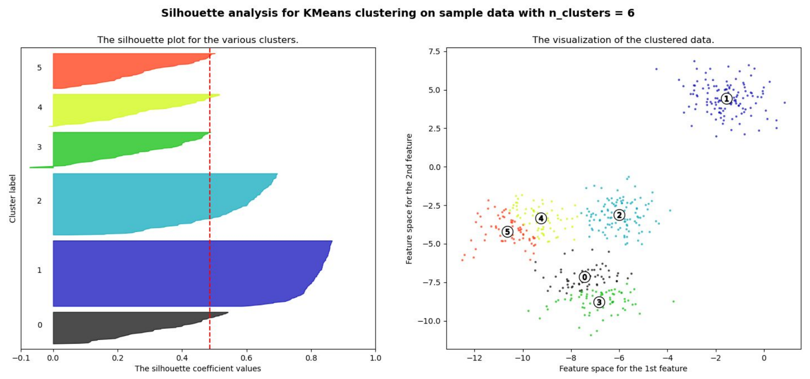
Silhouette Analysis











- For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
- For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
- **For n_clusters = 4 The average silhouette_score is : 0.6505186632729437**
- For n_clusters = 5 The average silhouette_score is : 0.561464362648773
- For n_clusters = 6 The average silhouette_score is : 0.4857596147013469

https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Agglomerative Clustering

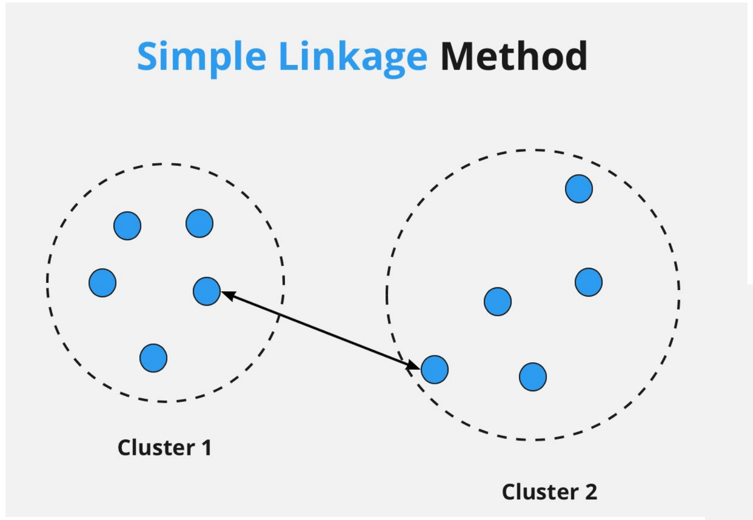
Agglomerative Clustering

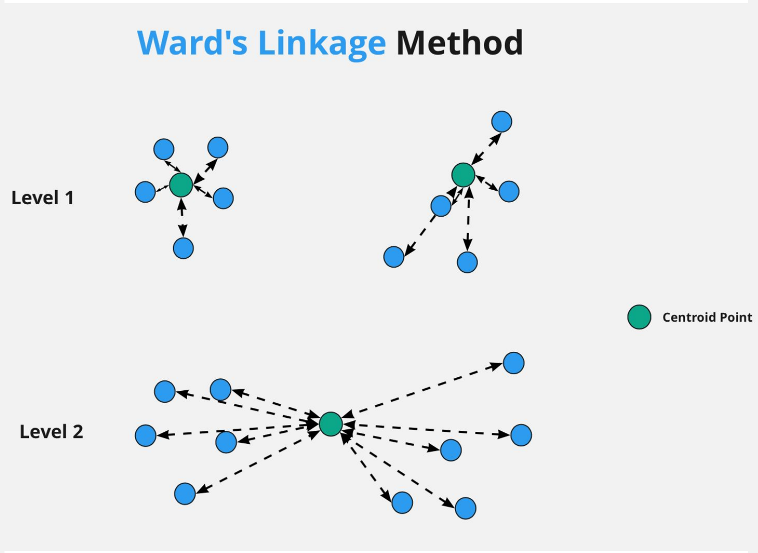
- Agglomerative clustering refers to a collection of clustering algorithms that all build upon the same principles: the algorithm starts by declaring each point its own cluster, and then merges the two most similar clusters until some stopping criterion is satisfied. The stopping criterion implemented in scikit-learn is the number of clusters, so similar clusters are merged until only the specified number of clusters are left. There are several linkage criteria that specify how exactly the “most similar cluster” is measured. This measure is always defined between two existing clusters.
- Not appropriate for large datasets

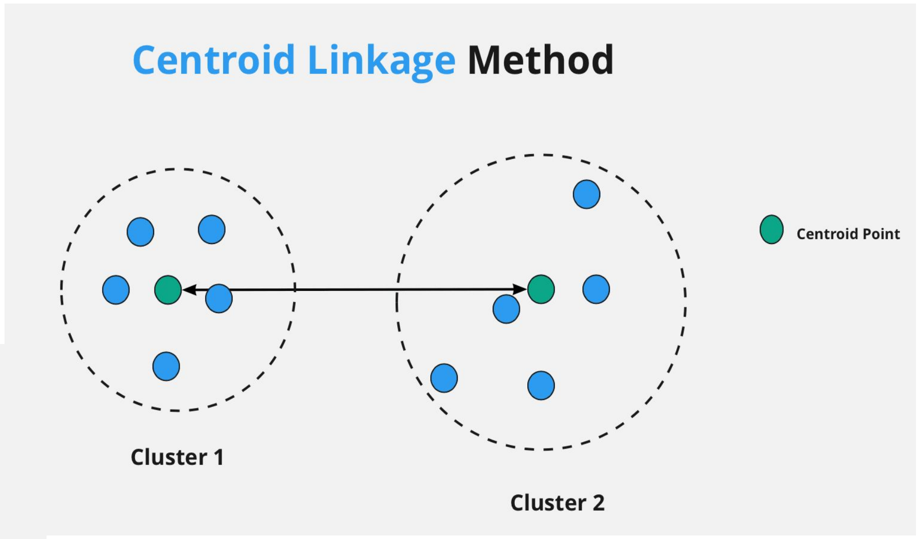
Computing Distance Matrix

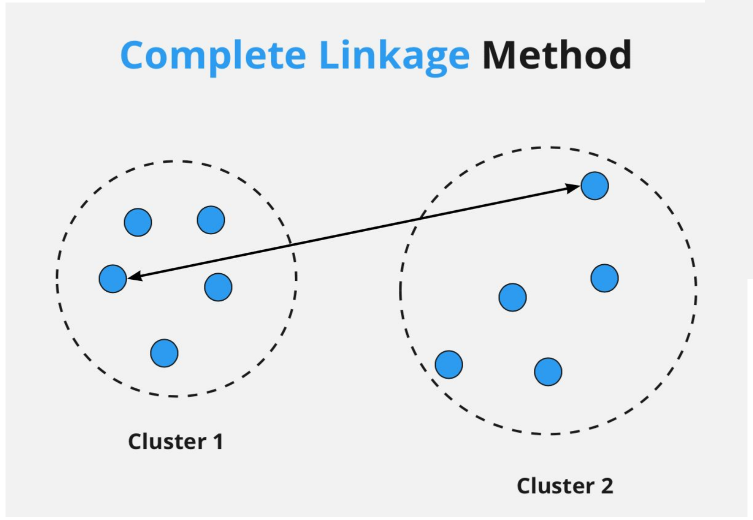
- The default choice, **ward** picks the two clusters to merge such that the variance within all clusters increases the least. This often leads to clusters that are relatively equally sized.
- **Average linkage** merges the two clusters that have the smallest average distance between all their points.
- **Complete linkage** (also known as maximum linkage) merges the two clusters that have the smallest maximum distance between their points.
- **Single linkage:** Merges the two clusters that have the **smallest minimum distance** between any of their points. Sensitive to noise and outliers.
- **Centroid linkage:** Merges clusters based on the distance between their centroids (mean points). Less sensitive to outliers than single linkage but can cause inversion (clusters merging in unexpected ways).
- **Median Linkage:** Similar to centroid linkage but uses the **median** instead of the mean when computing cluster distances.
- Ward works on most datasets. If the clusters have very dissimilar numbers of members (if one is much bigger than all the others, for example), average or complete might work better.

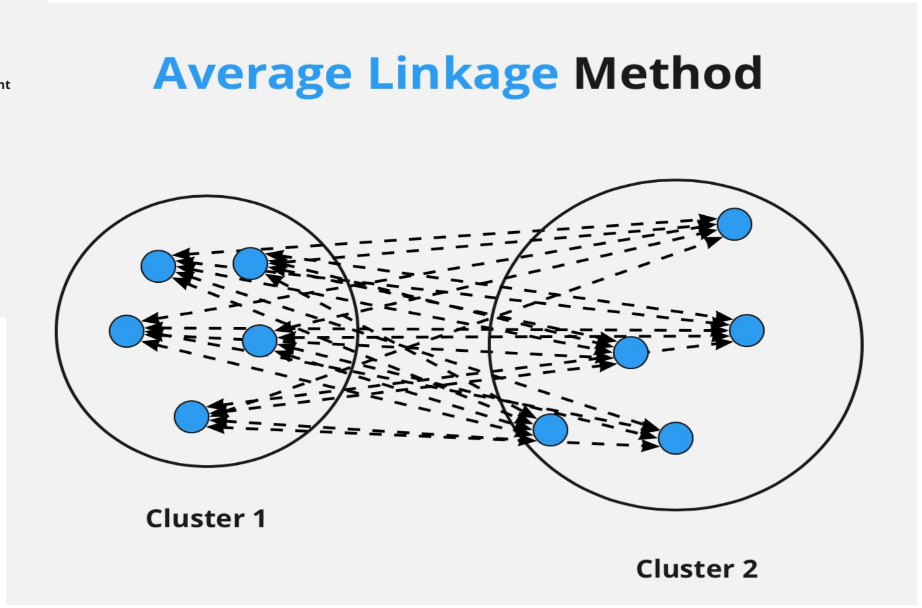
Computing Distance Matrix





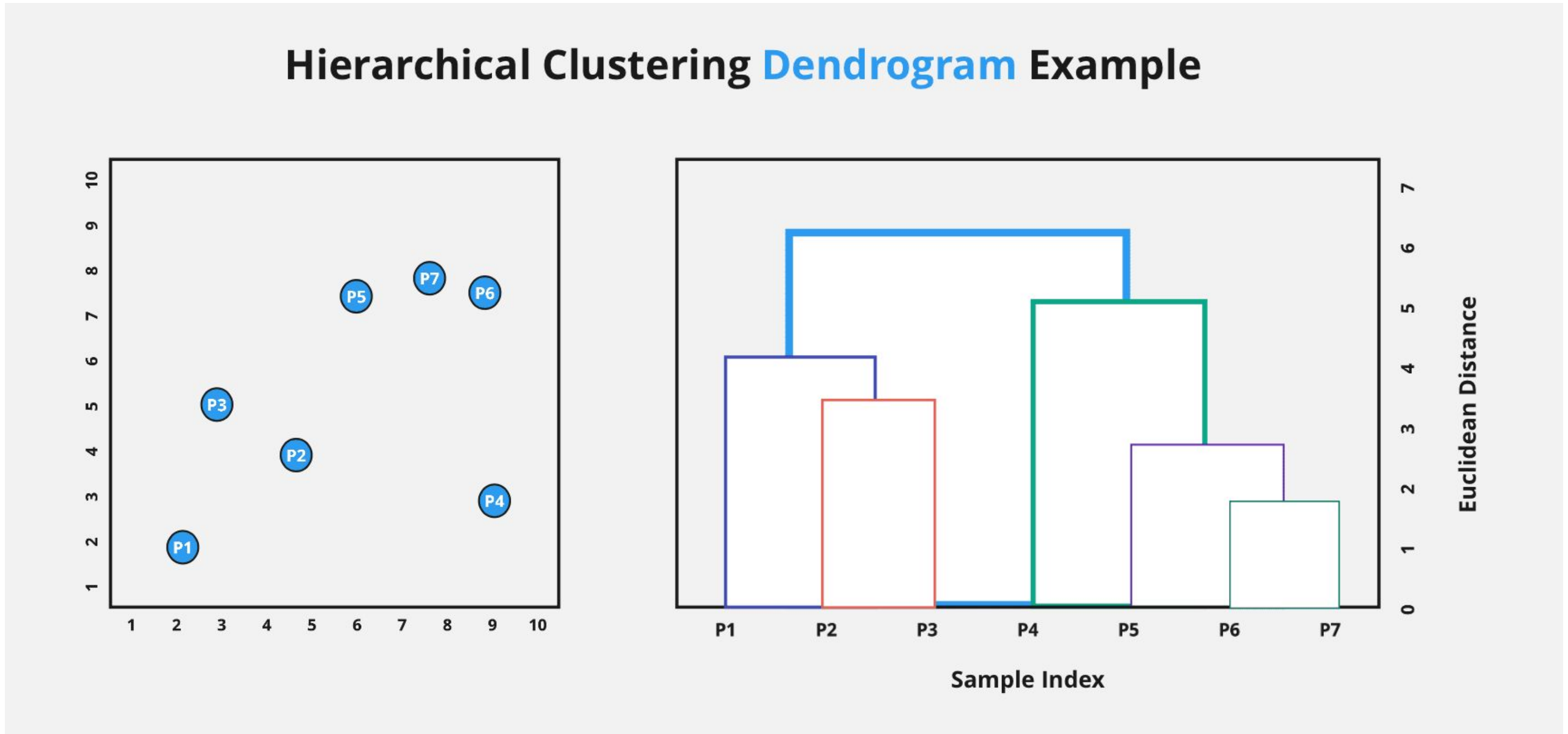






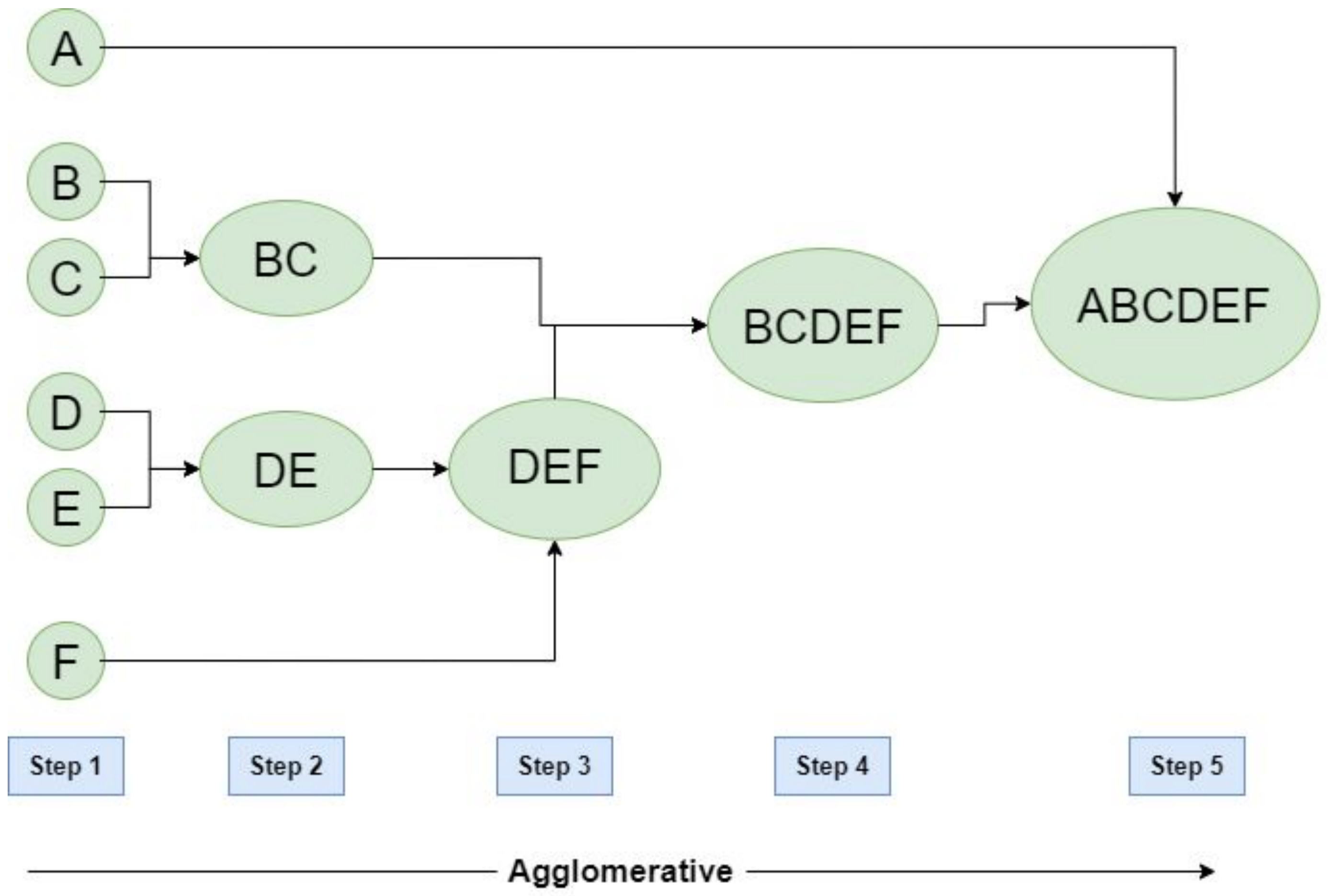
What is Dendrogram?

- A Dendrogram is a diagram that represents the hierarchical relationship between objects. The Dendrogram is used to display the distance between each pair of sequentially merged objects.
- These are commonly used in studying hierarchical clusters before deciding the number of clusters significant to the dataset.
- The distance at which the two clusters combine is referred to as the dendrogram distance.
- The primary use of a dendrogram is to work out the best way to allocate objects to clusters.



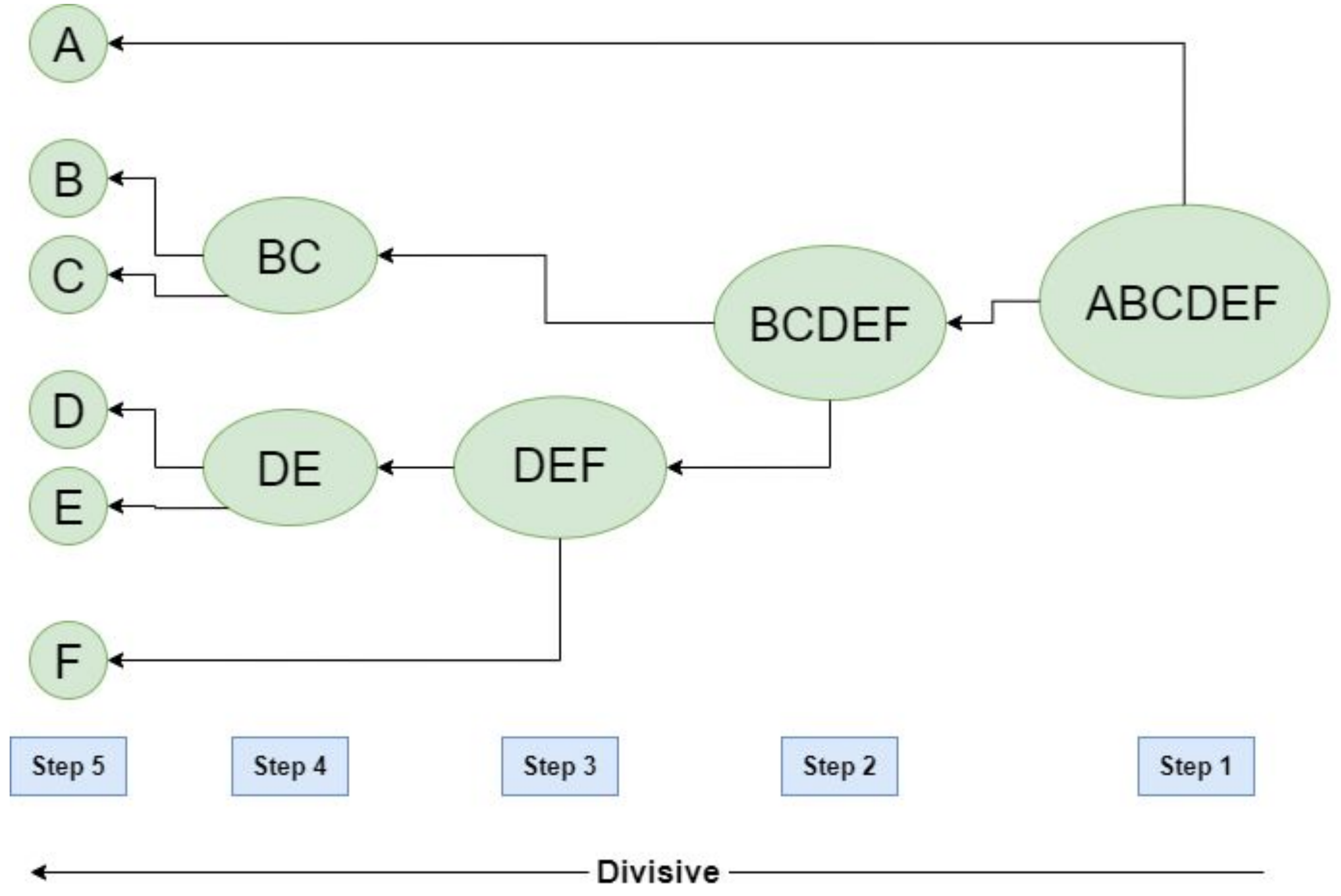
Hierarchical Agglomerative Clustering

- It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). Unlike flat clustering hierarchical clustering provides a structured way to group data. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.



Hierarchical Divisive Clustering

- It is also known as a top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton clusters.



- We form a new distance matrix by considering the single linkage decision rule.
- Using this linkage algorithm, we need to compute the distance from the newly formed cluster [B,C] to all the other objects.
- For example, with regard to the distance from the cluster [B,C] to object A, we need to check whether A is closer to object B or to object C. That is, we look for the minimum value in d(A,B) and d(A,C) from above As d(A,C)=36.249 is smaller than d(A,B)=49.010, the distance from A to the newly formed cluster is equal to d(A,C); that is, 36.249.
- We also compute the distances from cluster [B,C] to all the other objects (i.e., D, E, F, G). For example, the distance between [B,C] and D is the minimum of d(B,D)= 58.592 and d(C,D)= 38.275 (see the bottom table above)

$$d_{Euclidean}(B,C)=\sqrt{(x_B-x_C)^2+(y_B-y_C)^2}$$

Customer	A	B	C	D	E	F	G
x	33	82	66	30	79	50	10
y	95	94	80	67	60	33	17

Objects	A	B	C	D	E	F	G
A	0						
B	49.010	0					
C	36.249	21.260	0				
D	28.160	58.592	38.275	0			
E	57.801	34.132	23.854	40.497	0		
F	64.288	68.884	49.649	39.446	39.623	0	
G	81.320	105.418	84.291	53.852	81.302	43.081	0

Objects	A	B, C	D	E	F	G
A	0					
B, C	36.249	0				
D	28.160	38.275	0			
E	57.801	23.854	40.497	0		
F	64.288	49.649	39.446	39.623	0	
G	81.320	84.291	53.852	81.302	43.081	0

Objects	A	B, C, E	D	F	G
A	0				
B, C, E	36.249	0			
D	28.160	38.275	0		
F	64.288	39.623	39.446	0	
G	81.320	81.302	53.852	43.081	0

After third clustering step (single linkage)

Objects	A, D	B, C, E	F	G
A, D	0			
B, C, E	36.249	0		
F	39.446	39.623	0	
G	53.852	81.302	43.081	0

After fourth clustering step (single linkage)

Objects	A, B, C, D, E	F	G
A, B, C, D, E	0		
F	39.446	0	
G	53.852	43.081	0

After fifth clustering step (single linkage)

Objects	A, B, C, D, E, F	G
A, B, C, D, E, F	0	
G	43.081	0

Dimensionality Reduction

Dimensionality Reduction

- Dimensionality reduction is the process of reducing the number of features (variables) in a dataset while preserving as much important information as possible. In real-world data, many features can be correlated or redundant, making models more complex and harder to interpret. By reducing dimensions, we can simplify the data, improve computational efficiency, and enhance visualization.
- Imagine you have a dataset with 100 different features describing customer behavior in an online store. Many of these features might carry overlapping information—like "total money spent" and "average purchase value." Instead of analyzing all 100 features, dimensionality reduction methods help find the most informative ones or create new features that summarize the essential patterns in the data.

Dimensionality Reduction

- There are two main approaches to dimensionality reduction:
  - Feature Selection – Choosing a subset of the most important original features based on certain criteria (e.g., removing low-variance or highly correlated features).
  - Feature Extraction – Creating new, fewer features that capture the essential patterns of the data. Techniques like Principal Component Analysis (PCA) and t-SNE fall into this category.
- A major benefit of dimensionality reduction is that it helps in data visualization. If a dataset has 50 or 100 features, it's impossible to plot directly. But by reducing it to two or three dimensions, we can create scatter plots that reveal meaningful clusters and relationships.
- However, reducing dimensions also has risks—some details might be lost, and the transformed features may not always have clear interpretations. Therefore, it's important to balance simplicity with accuracy, choosing the right number of dimensions based on the specific problem and dataset.

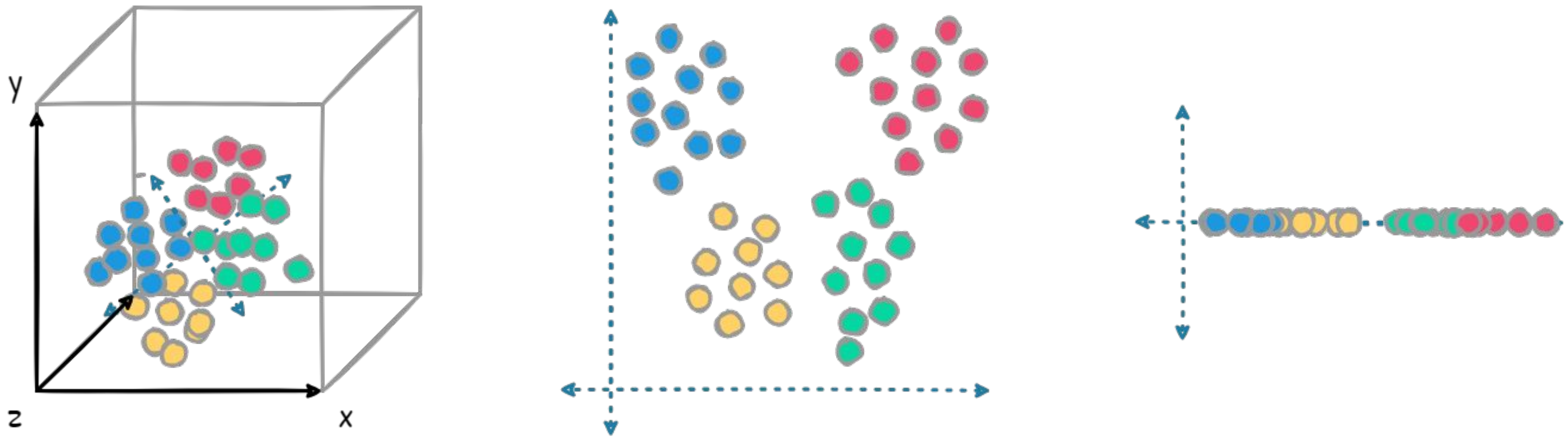
What is PCA?

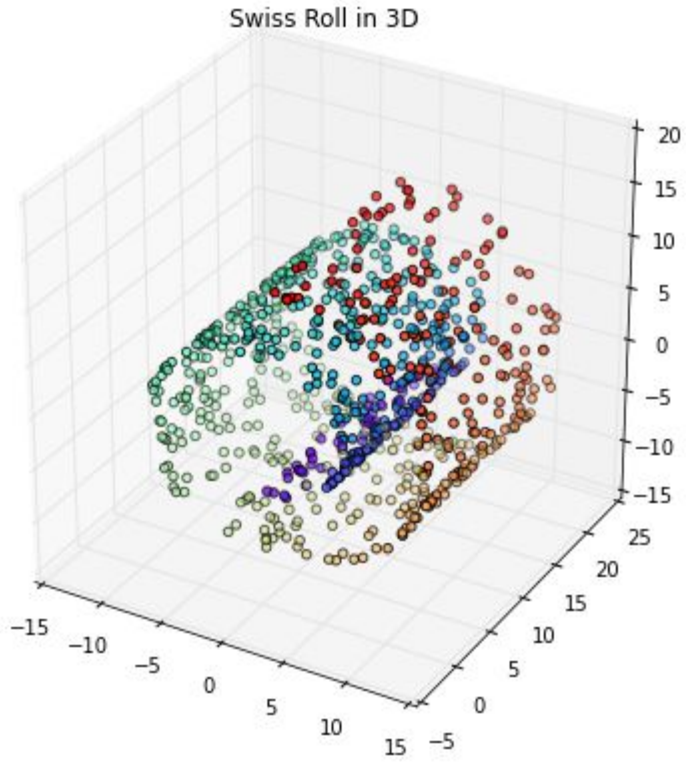
- Principal Component Analysis (PCA) is a technique used to **reduce the dimensionality** of a dataset while preserving as much variance (information) as possible. It **transforms** correlated features into a new set of **uncorrelated features** called **principal components**.
- PCA is widely used in:
  Data compression – Reducing storage needs.
  Feature selection – Removing less important features.
  Noise reduction – Eliminating redundant information.
  Visualization – Plotting high-dimensional data in 2D or 3D.
- Selecting only the top k principal components instead of using all original features.

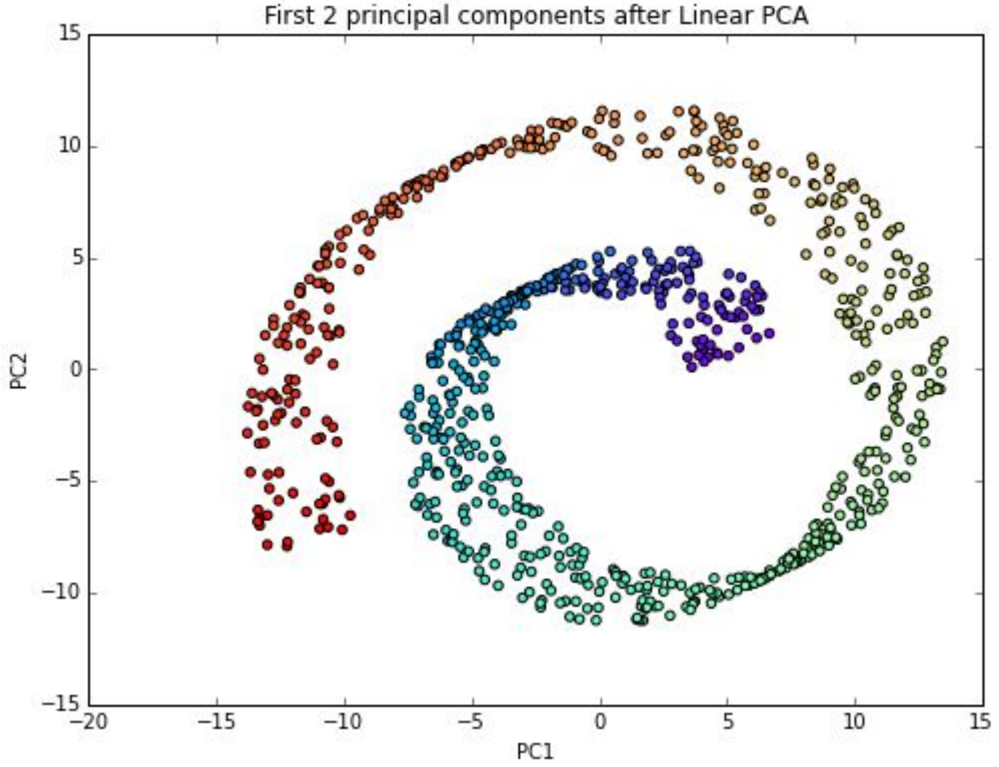
Eigenvectors and Eigenvalues

- Eigenvectors are the new directions or axes that we use to transform our data. In PCA, they represent the directions of maximum variance (the most important features of the data).
- Eigenvalues tell us how much variance (or "information") is captured by each eigenvector. A higher eigenvalue means the corresponding eigenvector (principal component) is more important because it captures more variance.
- In PCA:
  - We first calculate the covariance matrix of our data.
  - We find eigenvectors and eigenvalues of the covariance matrix.
  - The eigenvectors are the new axes for the transformed data (principal components).
  - The eigenvalues tell us how much each principal component explains the variance in the data.
  - We pick the principal components with the largest eigenvalues to reduce the number of dimensions while keeping most of the data's information.

Dimensionality Reduction




Swiss Roll in 3D
20
15
10
5
0
−5
−10
−15
25
20
15
10
5
0
−5
−15
−10
−5
0
5
10
15


First 2 principal components after Linear PCA
15
10
5
0
−5
−10
−15
PC2
−20
−15
−10
−5
0
5
10
15
PC1

Thank you!