Machine Learning

Unsupervised Learning

Eng. Gihan Weerasekara
Consultant/ Lecturer
National Institute of Business Management

Definition of Unsupervised Learning

Unsupervised Learning is a type of machine learning that deals with analyzing and modeling data without predefined labels. The algorithm works on unlabeled data to find hidden structures or patterns.

Key Points:

- No labeled training data or ground truth is provided.
- The model explores the data and tries to understand its structure or relationships.
- Useful for finding similarities and differences in data points without external guidance.

Difference Between Supervised and Unsupervised Learning

Supervised Learning:

- The model learns from labeled data, where each input has a corresponding correct output.
- Example: Predicting house prices based on features like square footage, location, and number of bedrooms.

Unsupervised Learning:

- The model learns from data that doesn't have explicit labels or outcomes.
- Example: Grouping customers based on their purchasing behavior without predefining customer categories.

Purpose of Unsupervised Learning

Key Objective:

To explore and identify the hidden structure of data and discover insights, patterns, or relationships that are not obvious.

Examples:

Identifying clusters in customer data, detecting anomalies in network traffic, or finding associations between products in a market basket analysis.

Why Use Unsupervised Learning

Often, labeled datasets are scarce or expensive to create.

Useful for exploratory data analysis and pre-processing before applying supervised learning techniques.

Enables pattern discovery in unstructured or large-scale datasets without needing human annotation.

Key Characteristics of Unsupervised Learning

Finds Hidden Patterns and Relationships

- **Objective:**

  Unsupervised learning focuses on identifying and uncovering hidden patterns, structures, or relationships in data.

- **Example:**

  It can cluster similar data points together or find correlations between features without needing explicit instructions.

  (Grouping customers based on purchasing behavior or discovering groups of similar articles in a news dataset.)

Key Characteristics of Unsupervised Learning

No Explicit Feedback

- **Explanation:**

  There is no direct feedback or output provided to validate the learning process. Unlike supervised learning, where the model adjusts based on feedback (like classification accuracy or error rate), unsupervised learning models refine their approach without direct comparisons to correct answers.

- **Consequence:**

  Algorithms rely on internal criteria like distance measures or data similarity to group data points or find patterns.

Key Characteristics of Unsupervised Learning

Data Exploration and Preprocessing

- **Purpose:**

  Unsupervised learning is often used in exploratory data analysis (EDA). It helps in understanding the dataset's structure and characteristics before applying any predictive modeling or classification.

- **Example:**

  Visualizing high-dimensional data by reducing dimensions (like using PCA or t-SNE) to discover underlying trends or clusters.

Key Characteristics of Unsupervised Learning

Examples of where Unsupervised Learning is applied:

- **Clustering:**

  Segmenting customers based on behavior, grouping images with similar patterns.

- **Dimensionality Reduction:**

  Reducing features for better data visualization or processing efficiency.

.

Key Characteristics of Unsupervised Learning

Examples of where Unsupervised Learning is applied:

- **Anomaly Detection:**

  Detecting unusual behavior in network traffic or spotting defective items in manufacturing.

- **Association Mining:**

  Finding relationships between products in a market basket analysis.

Types of Unsupervised Learning

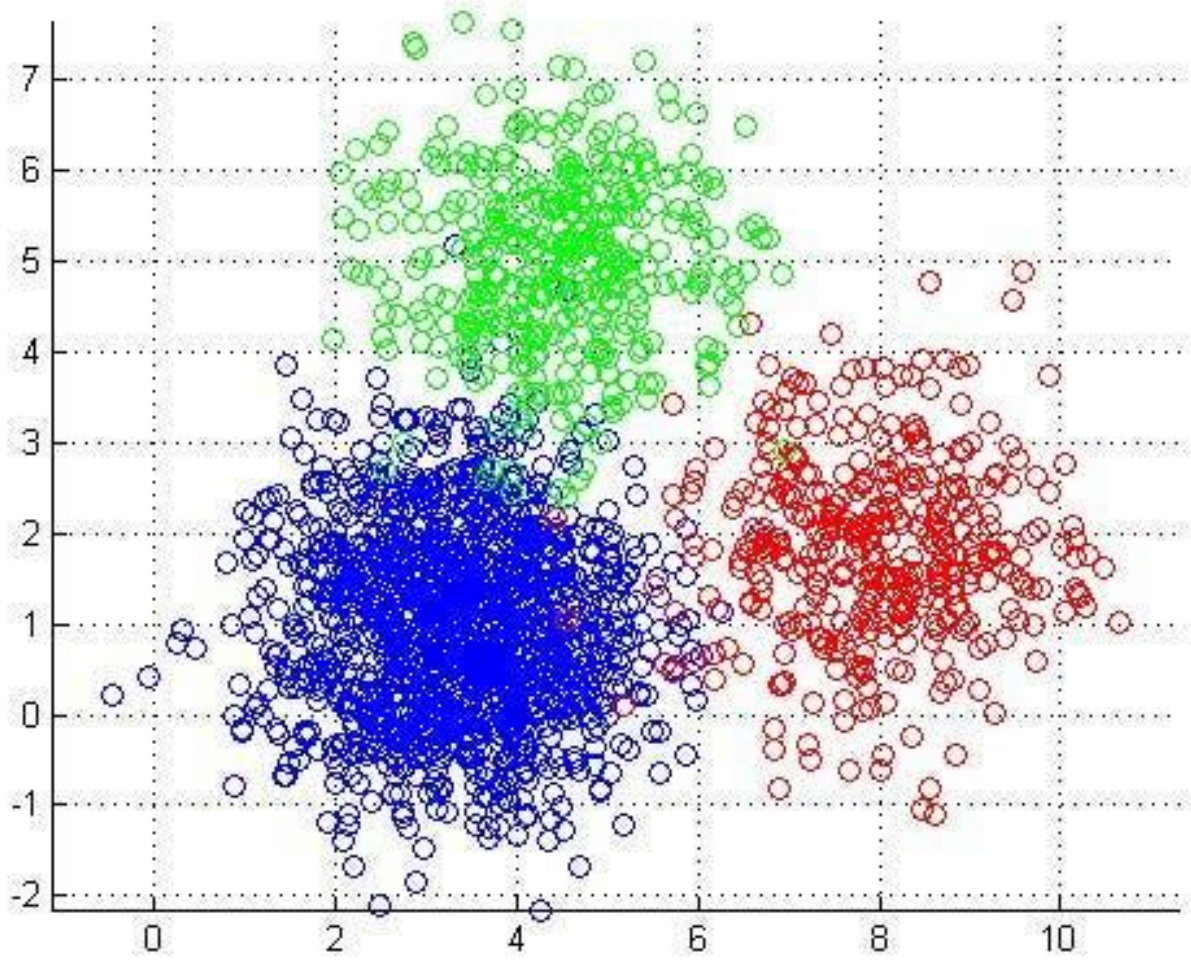
Clustering

- **Definition:**

  Clustering groups similar data points into clusters based on some similarity criteria.

  It organizes data points that share common features or characteristics into groups, without prior knowledge of the data.

Types of Unsupervised Learning



Types of Unsupervised Learning

Key Algorithms:

- **K-Means Clustering:**

  Partitions data into *K* clusters by finding centroids and assigning data points to the nearest centroid.

- **Hierarchical Clustering:**

  Builds a hierarchy of clusters through an iterative merging (agglomerative) or splitting (divisive) approach.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**

  Finds clusters based on the density of data points, allowing the identification of arbitrary-shaped clusters and outliers.

Types of Unsupervised Learning

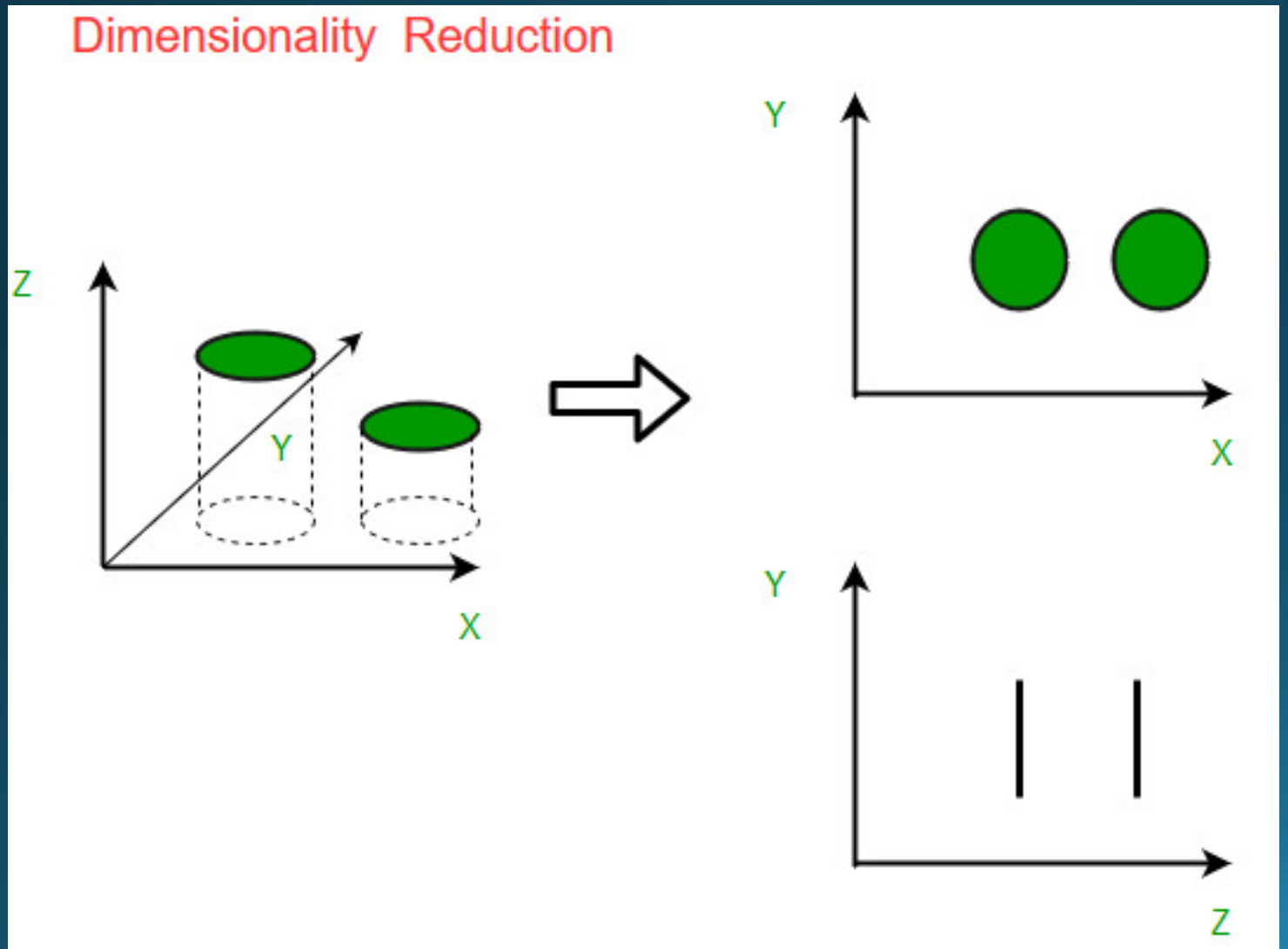
Dimensionality Reduction

- **Definition:**

  Reduces the number of features or dimensions in a dataset while retaining essential information.

  This technique is useful for simplifying complex datasets and visualizing high-dimensional data.

Types of Unsupervised Learning



Types of Unsupervised Learning

Key Algorithms:

- **Principal Component Analysis (PCA):**

  Transforms the dataset into a lower-dimensional space using orthogonal components that capture the most variance.

- **t-Distributed Stochastic Neighbor Embedding (t-SNE):**

  Visualizes high-dimensional data in 2D or 3D by preserving the local similarities between data points.

- **Linear Discriminant Analysis (LDA):**

  Finds a linear combination of features that best separates classes in labeled datasets, but it's also used in some unsupervised contexts.

Types of Unsupervised Learning

Applications:

- Data visualization
- Noise reduction
- Feature extraction

Types of Unsupervised Learning

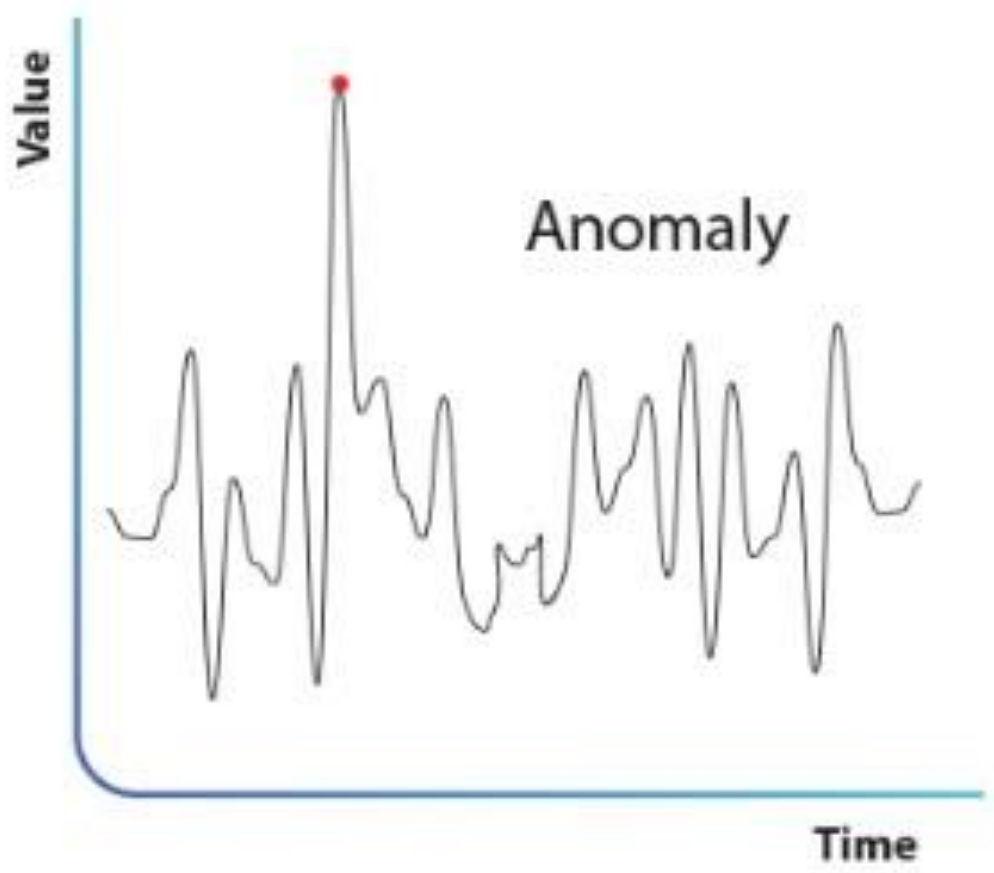
Anomaly Detection

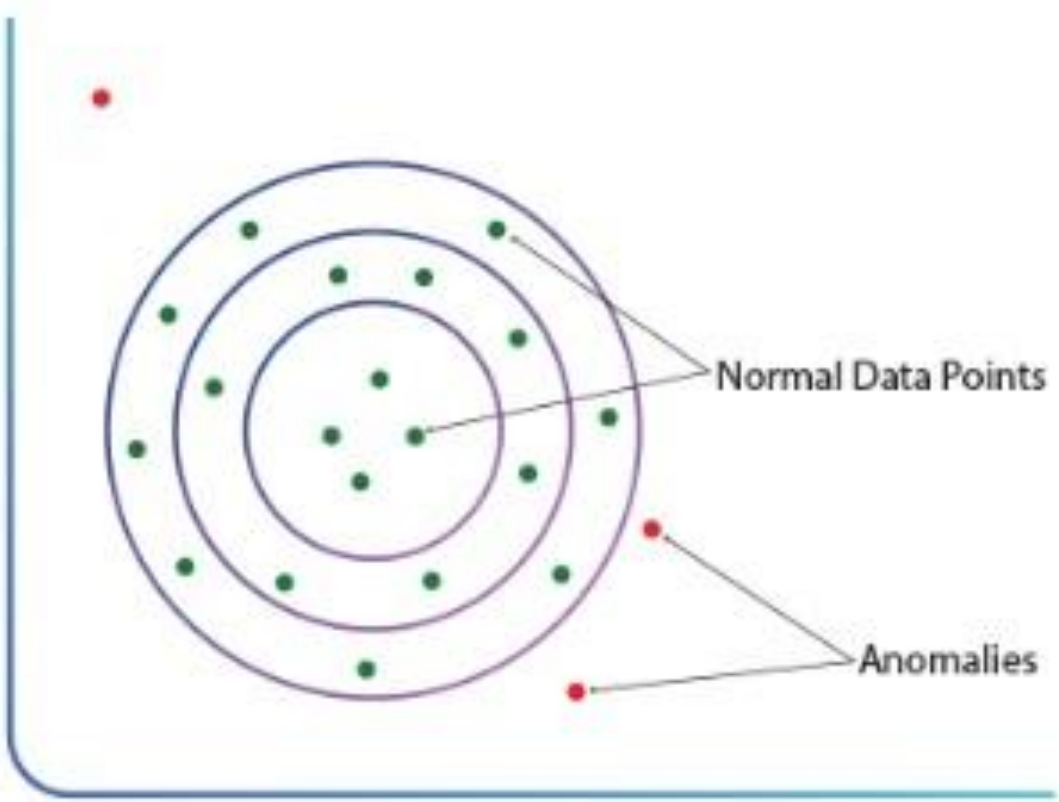
- **Definition:**

  Identifies data points that deviate significantly from the norm or expected pattern.

  These anomalies (or outliers) can represent rare events, errors, or fraudulent activities.

Types of Unsupervised Learning





Types of Unsupervised Learning

Key Algorithms:

- **Isolation Forest:**

  Randomly partitions data points and isolates anomalies based on how easily a point can be isolated.

- **Gaussian Mixture Models (GMM):**

  Models the dataset as a mixture of multiple Gaussian distributions and identifies outliers based on the likelihood of occurrence.

Types of Unsupervised Learning

Applications:

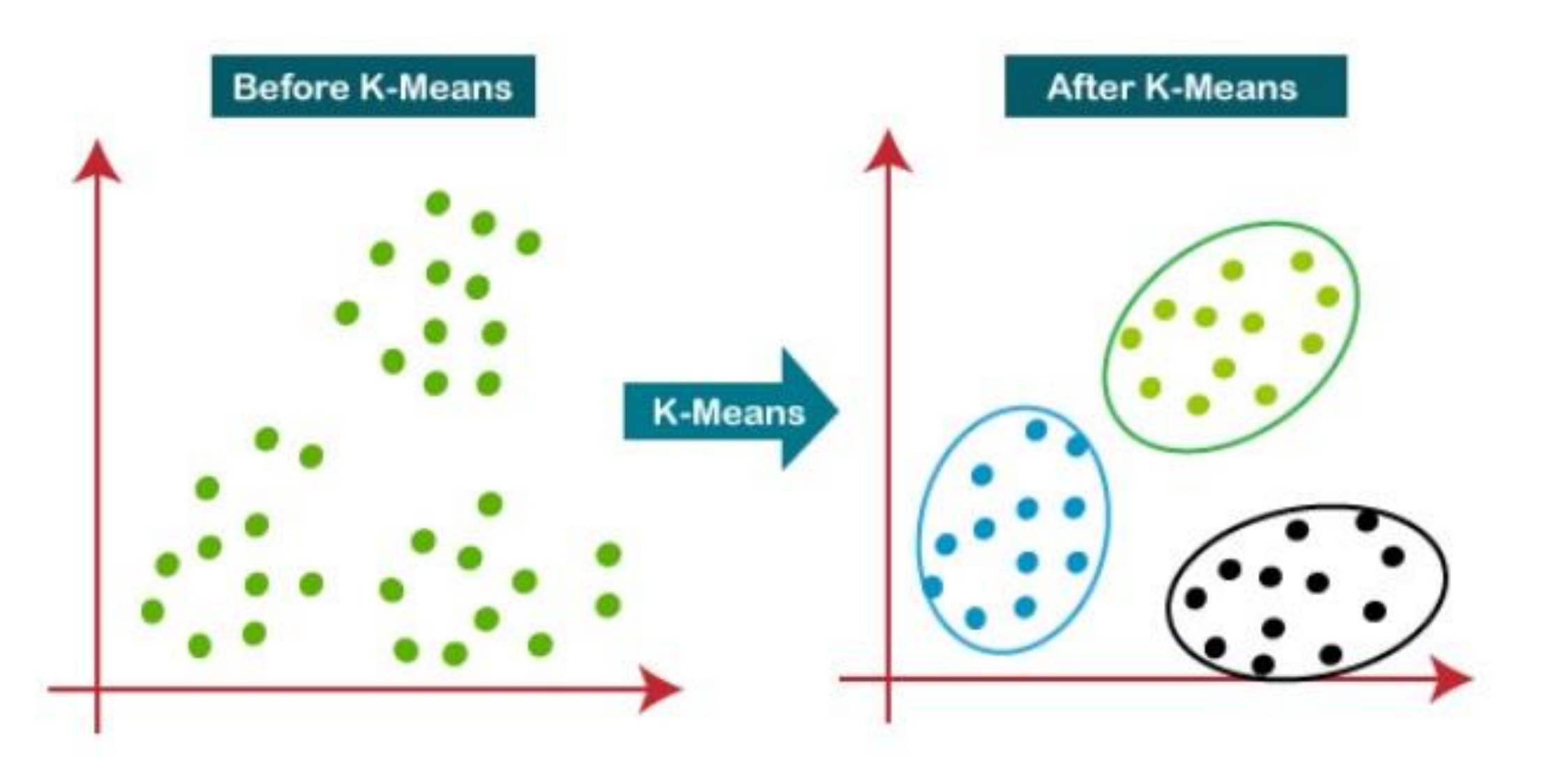
- Fraud detection
- Network security monitoring
- Fault detection in manufacturing

K-Means Clustering

K-Means is a centroid-based clustering algorithm where the objective is to divide *N* data points into *K* clusters in such a way that each data point belongs to the cluster with the nearest mean (centroid).

The algorithm minimizes the variance within each cluster by adjusting the cluster centroids iteratively.

K-Means Clustering



How K-Means Clustering Works

Initialization:

Randomly select *K* initial centroids (cluster centers) from the dataset.

Assignment:

Assign each data point to the nearest centroid based on a distance metric (usually Euclidean distance). This forms *K* clusters.

Update:

Repeat: Repeat the assignment and update steps until the centroids no longer change or converge (i.e., the clusters are stable).

How K-Means Clustering Works

Repeat:

Repeat the assignment and update steps until the centroids no longer change or converge (i.e., the clusters are stable).

Example:

If you have a dataset of customer spending habits, K-Means can segment customers into *K* different groups based on their spending characteristics.

Key Concepts in K-Means

Centroid:

The center of a cluster, represented by the mean value of all the points in that cluster. It is a critical concept since data points are assigned to clusters based on their distance to the centroids.

Distance Metric:

Typically, the Euclidean distance is used to determine the distance between data points and centroids. Other distance metrics, like Manhattan distance, can be used depending on the data type and application.

Key Concepts in K-Means

Objective Function:

The goal of K-Means is to minimize the *within-cluster sum of squares* (WCSS), which is defined as the sum of squared distances between each point and its cluster centroid.

$$WCSS = \sum_{i=1}^{k} \sum_{x \in c_i} (x - \mu_i\ )^2$$

Where:

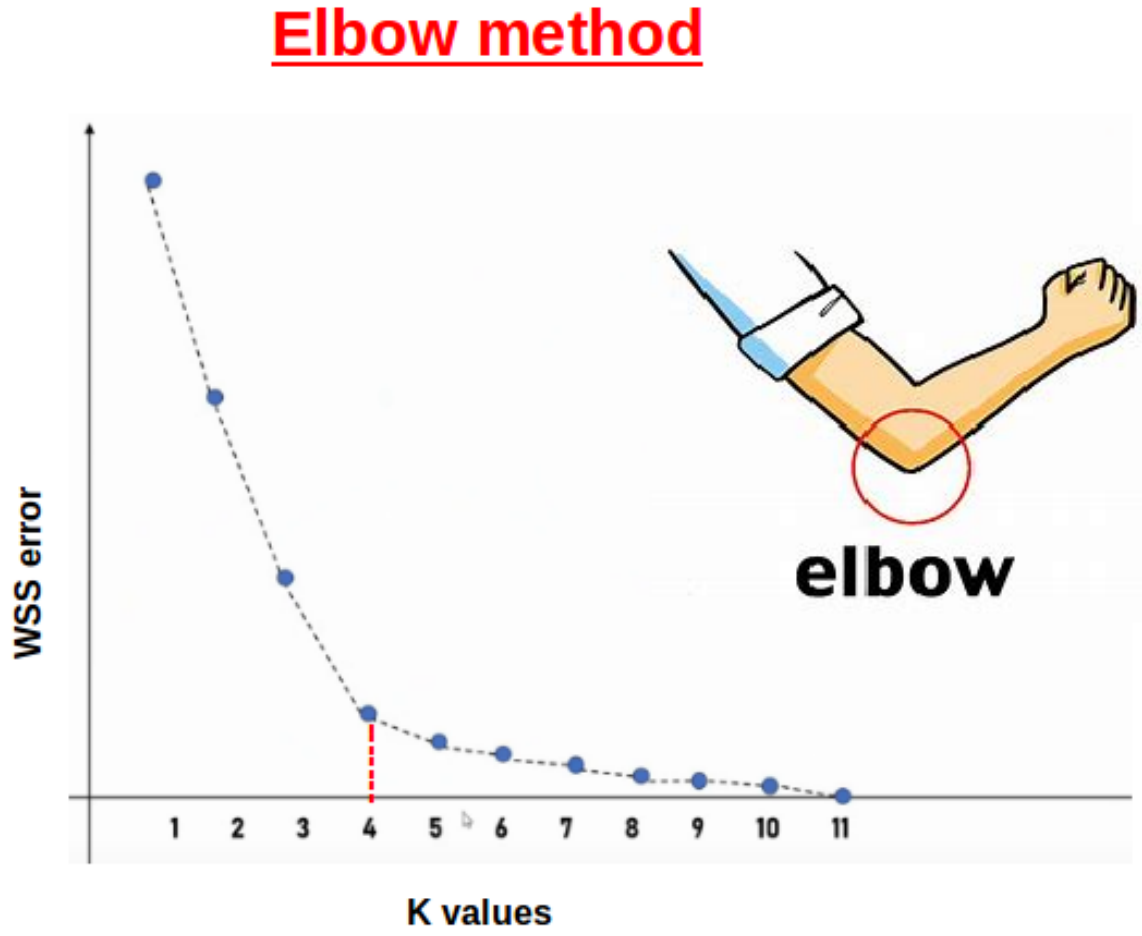
- K is the number of clusters.
- Ci represents the *i*-th cluster.
- μ_i is the centroid of cluster Ci

Choosing the Number of Clusters (*K*)

Elbow Method:

Plots the sum of squared errors (SSE) against the number of clusters. The "elbow" point, where the rate of decrease sharply slows, is considered the optimal *K*.

Choosing the Number of Clusters (*K*)



Advantages of K-Means Clustering

Simple and Fast:

The algorithm is computationally efficient and easy to implement. It scales well with large datasets.

Works Well for Well-Separated Clusters:

It effectively separates clusters that have spherical shapes and are well-separated.

Advantages of K-Means Clustering

Simple and Fast:

The algorithm is computationally efficient and easy to implement. It scales well with large datasets.

Works Well for Well-Separated Clusters:

It effectively separates clusters that have spherical shapes and are well-separated.

Real-World Applications of K-Means

Customer Segmentation:

Identifying different groups of customers based on purchasing behavior or demographics for targeted marketing.

Image Compression:

Reducing image size by grouping similar colors and replacing each group with its centroid color.

Document Clustering:

Organizing articles or documents based on similar topics or content.

Pattern Recognition:

Grouping similar data points to identify underlying patterns or anomalies.

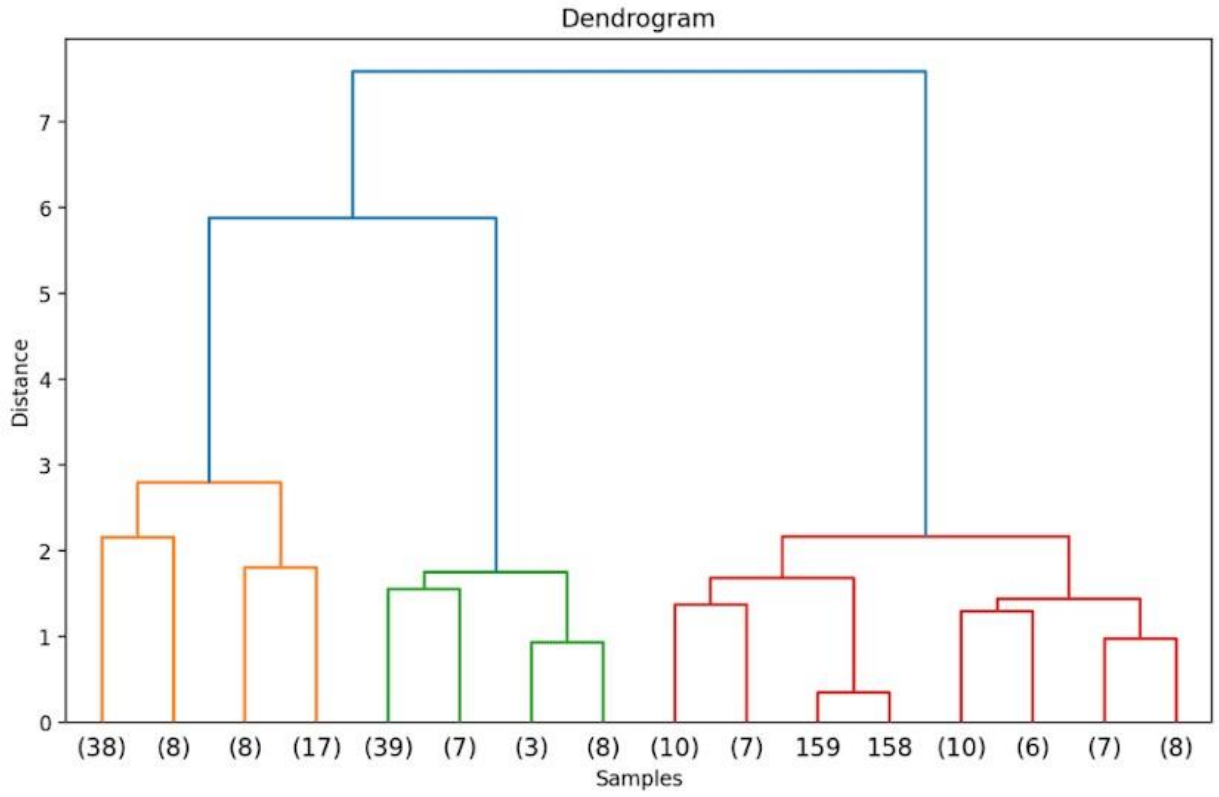
Hierarchical Clustering

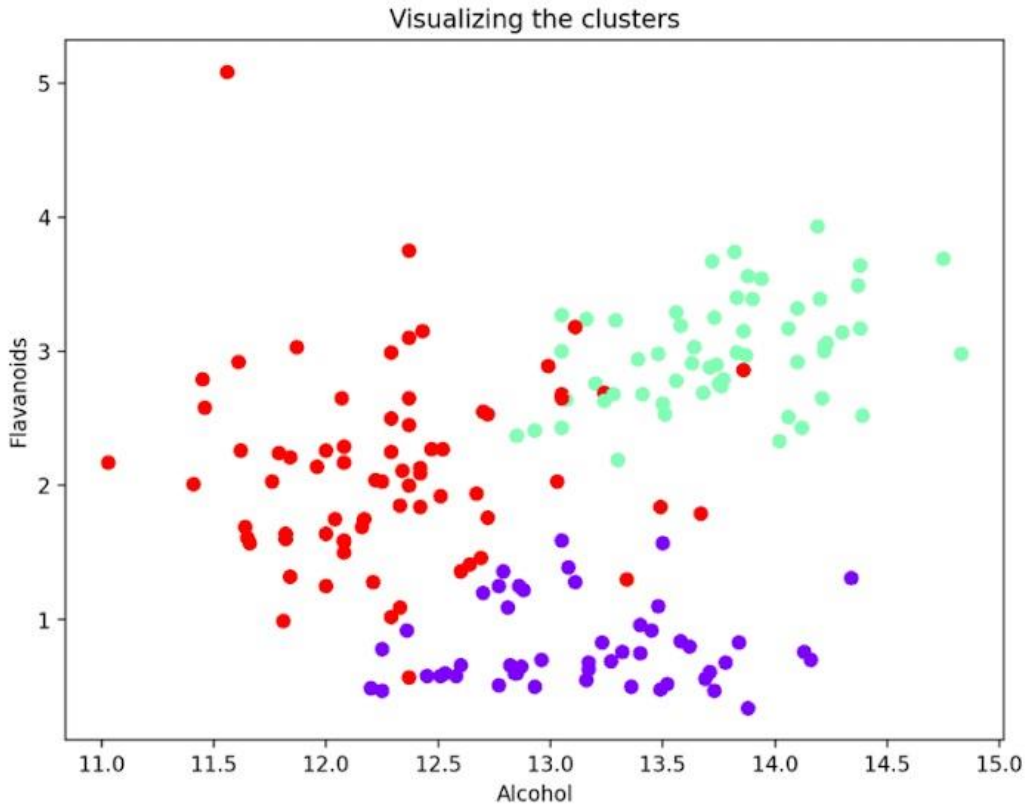
Hierarchical Clustering organizes data points into a nested hierarchy of clusters based on their similarity. It iteratively merges or splits clusters to create a tree-like structure that captures relationships between clusters.

Output:

The result of hierarchical clustering is a dendrogram, which is a tree diagram that visually represents the arrangement of the clusters and sub-clusters.

Hierarchical Clustering





Types of Hierarchical Clustering

Agglomerative Hierarchical Clustering (Bottom-Up Approach)

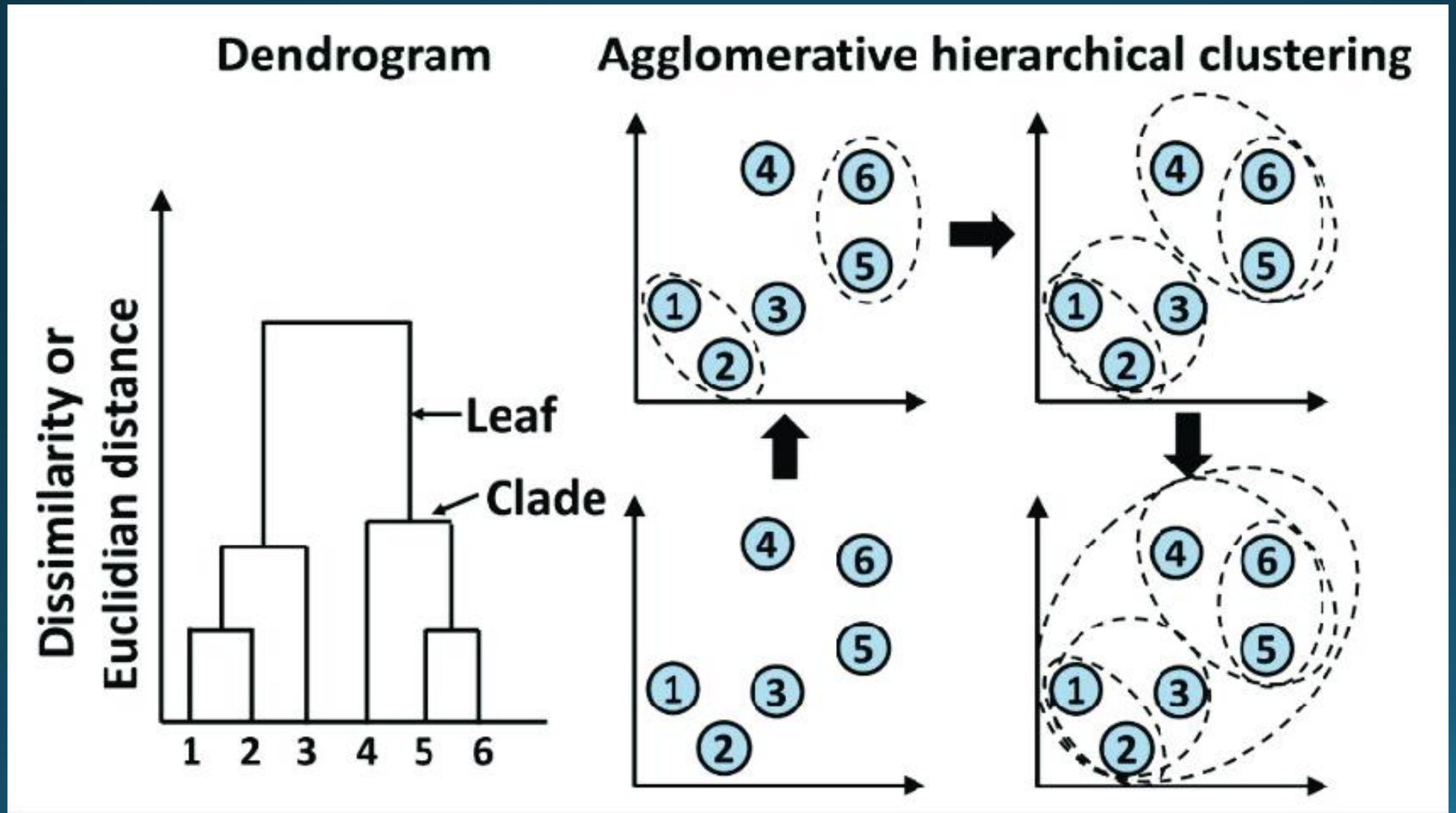
Process:

Starts with each data point as its own cluster and iteratively merges the closest pairs of clusters until all points are in a single cluster or a specified number of clusters is reached.

Commonly Used:

Agglomerative is the more popular form of hierarchical clustering

Types of Hierarchical Clustering



Divisive Hierarchical Clustering (Top-Down Approach)

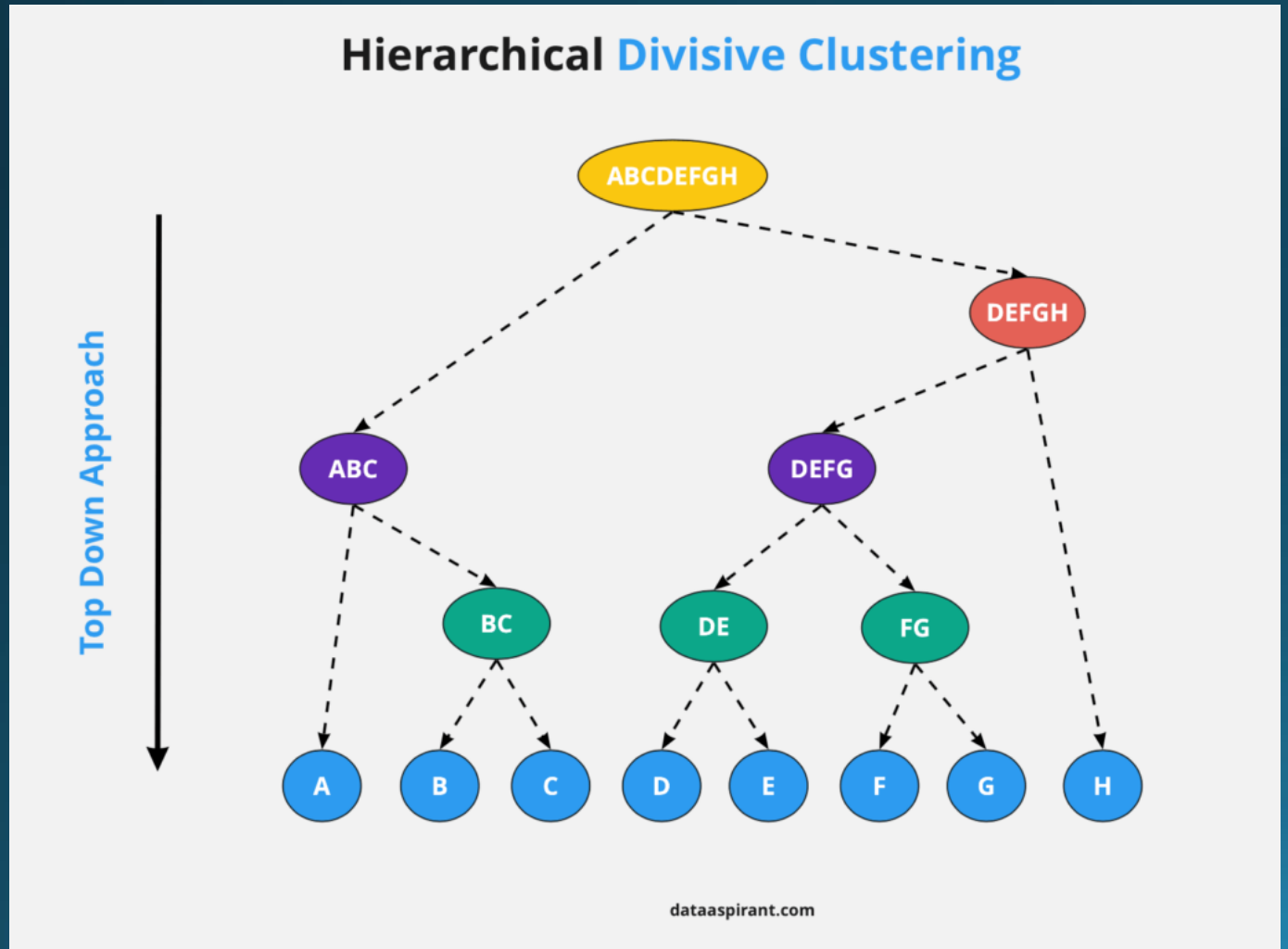
Process:

Starts with all data points in one single cluster and iteratively splits the cluster into smaller sub-clusters until each data point forms its own cluster.

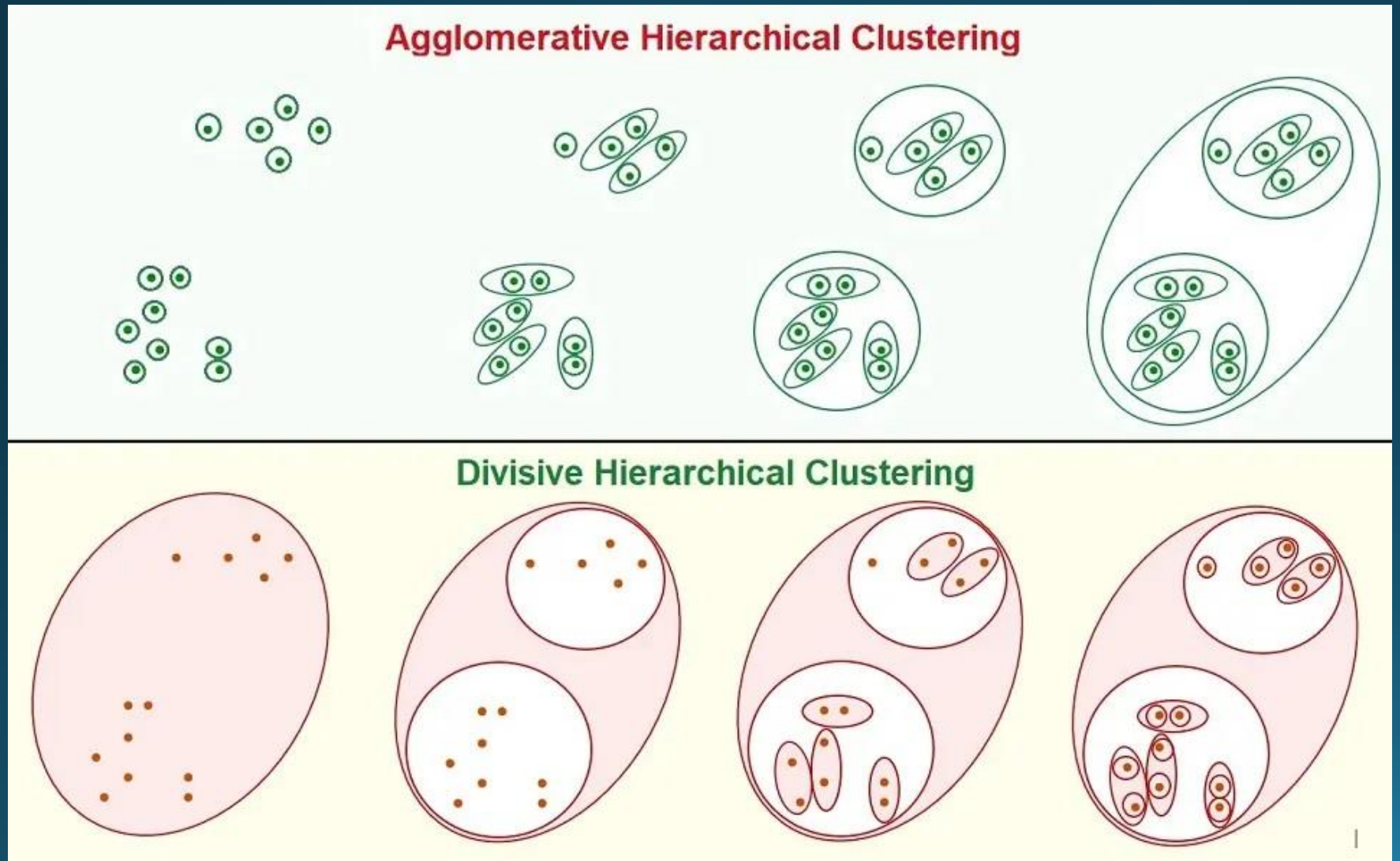
Less Common:

This approach is less frequently used due to its computational complexity.

Divisive Hierarchical Clustering (Top-Down Approach)



Comparison


Agglomerative Hierarchical Clustering
Divisive Hierarchical Clustering

How Hierarchical Clustering Works

Step 1: Calculate Distance Matrix:

Compute a distance matrix that measures the pairwise distances (e.g., Euclidean distance) between all data points.

Step 2: Merge or Split Clusters:

Based on the type (agglomerative or divisive), the algorithm either merges the closest clusters or splits the largest cluster.

Step 3: Update Distance Matrix:

Recalculate the distance between newly formed clusters and other clusters using a chosen linkage method.

.

How Hierarchical Clustering Works

Step 4: Repeat:

Continue merging or splitting until the desired number of clusters or a stopping criterion is reached.

Advantages of Hierarchical Clustering

No Need to Specify the Number of Clusters in Advance:

Unlike K-Means, you do not have to define *K* beforehand.

Flexibility in Distance and Linkage Metrics:

You can choose different linkage criteria and distance measures based on your dataset and problem.

Produces a Dendrogram:

The dendrogram provides a comprehensive view of the cluster structure and the relationships between clusters, helping in making decisions about the number of clusters.

Real-World Applications of Hierarchical Clustering

Customer Segmentation:

Used in market research to segment customers based on purchasing behavior or demographics.

Document Organization:

Categorizing documents based on content similarity for better information retrieval.

Gene Expression Analysis:

Grouping genes with similar expression patterns to understand biological processes.

Real-World Applications of Hierarchical Clustering

Social Network Analysis:

Identifying communities or sub-groups in a network of social connections.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise

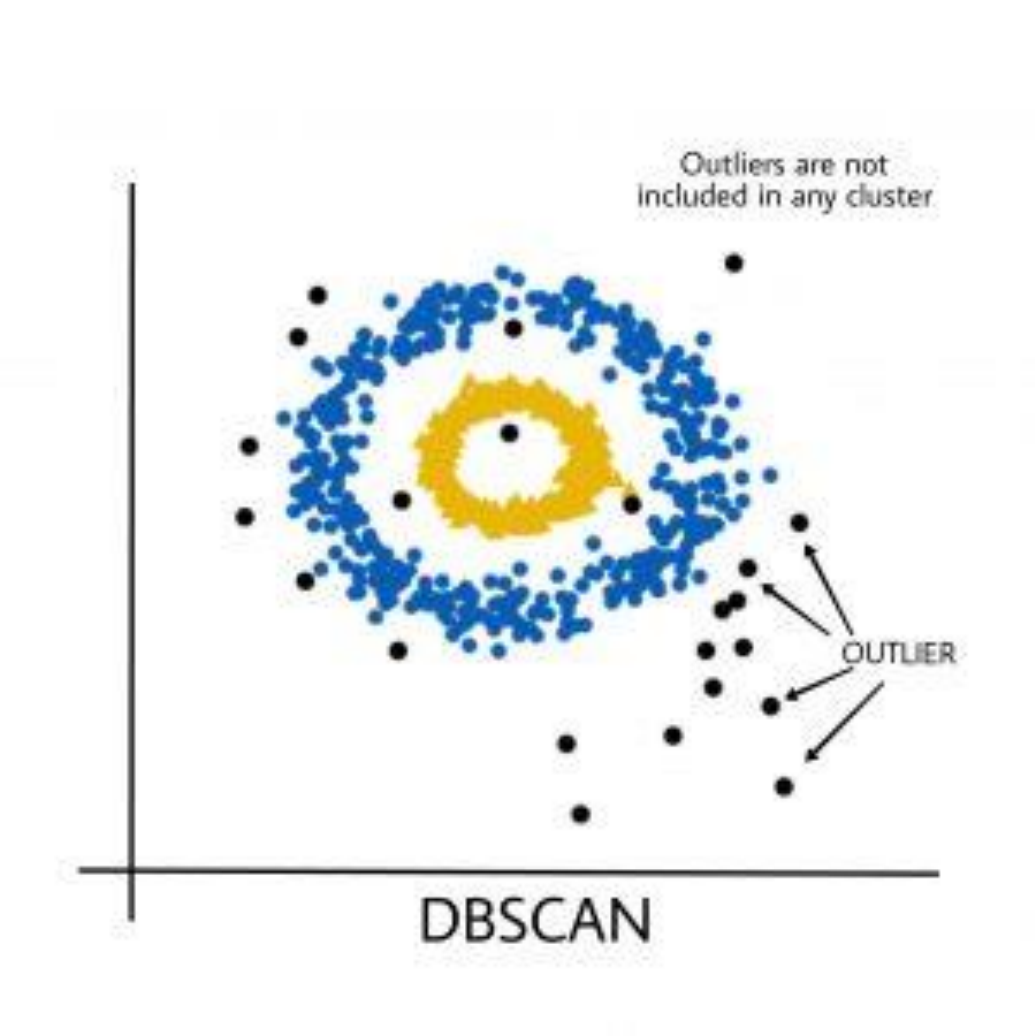
DBSCAN is a density-based clustering algorithm that groups data points based on their density within a certain neighborhood.

It works by expanding clusters from high-density areas and marking any isolated points as noise (outliers).

Objective:

To find clusters of any shape and detect outliers by classifying dense areas as clusters and sparse areas as noise.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise



Key Concepts in DBSCAN

Core Points, Border Points, and Noise

- **Core Point:** A point with at least a specified minimum number of neighbors (MinPts) within a specified distance (ε). Core points are central to forming clusters.
- **Border Point:** A point within the neighborhood of a core point but with fewer neighbors than MinPts. Border points are part of clusters but not cluster initiators.
- **Noise (Outliers):** Any point that is not a core point and is not within the neighborhood of a core point.

Key Concepts in DBSCAN

Parameters in DBSCAN

- **Epsilon (ε):** The radius within which points are considered neighbors.
- **MinPts:** The minimum number of points required to form a dense region around a core point. This parameter determines the density threshold for clusters.

How DBSCAN Works

Select an Unvisited Point:

Start with an arbitrary, unvisited point in the dataset.

Check Density:

Calculate the number of points within the ε-radius neighborhood.

- If the point has at least MinPts neighbors, it is labeled a core point, and a new cluster is initiated.
- If it does not have enough neighbors, it is labeled as noise initially (though it could later become part of a cluster as more points are visited).

How DBSCAN Works

Expand Cluster:

From each core point, iteratively expand the cluster by including all reachable core and border points within ε.

Repeat:

Continue the process until all points are visited. Any unvisited points that do not meet the density criteria are marked as noise or outliers.

How DBSCAN Works

Example:

In a dataset of customer geolocations, DBSCAN can identify clusters of customers within close proximity, like those frequenting nearby stores, without knowing the number of clusters in advance.

Advantages of DBSCAN

Automatic Outlier Detection:

Points that do not belong to any dense region are labeled as noise, which makes DBSCAN effective at identifying outliers.

No Need for Predefined Number of Clusters:

Unlike K-Means, DBSCAN does not require specifying the number of clusters, making it suitable for exploratory data analysis.

Flexible Cluster Shapes:

DBSCAN can detect clusters of arbitrary shapes, which is beneficial for datasets where clusters aren't spherical or uniform in size.

Real-World Applications of DBSCAN

Geospatial Analysis:

Identifying densely populated regions, like finding hotspots in a city map or mapping customer clusters near a retail location.

Anomaly Detection:

Detecting fraud or unusual patterns by labeling sparse regions as noise in financial transactions or network traffic.

Image Analysis:

Grouping similar pixels in an image, especially where natural clusters exist in varying shapes and densities.

Real-World Applications of DBSCAN

Astronomy:

Identifying star clusters and galaxy structures in astronomical data based on the density of celestial objects.

Advantages and Challenges of Clustering

Advantages:

- Provides insights and organizes complex data.
- Useful for pattern recognition, anomaly detection, and exploratory data analysis.

Challenges:

- Determining the optimal number of clusters (*K*) is often subjective.
- Different clustering algorithms can yield varying results on the same dataset.
- Performance can be sensitive to outliers and noise in the data.

Dimensionality Reduction

Dimensionality Reduction is the process of transforming data from a high-dimensional space (many features) into a low-dimensional space while maintaining as much relevant information as possible.

Objective:

Reduce the dataset's complexity to improve computational efficiency, reduce noise, and enable visualization in two or three dimensions.

Purpose of Dimensionality Reduction

Simplifies Data Visualization:

High-dimensional data is difficult to visualize, but dimensionality reduction enables visualization of key patterns in 2D or 3D.

Improves Model Performance:

By eliminating redundant or irrelevant features, dimensionality reduction can help improve model training times and performance, especially for algorithms sensitive to feature counts.

Reduces Noise:

It can remove or reduce noise in the data, making it easier to detect meaningful patterns or relationships.

Principal Component Analysis (PCA)

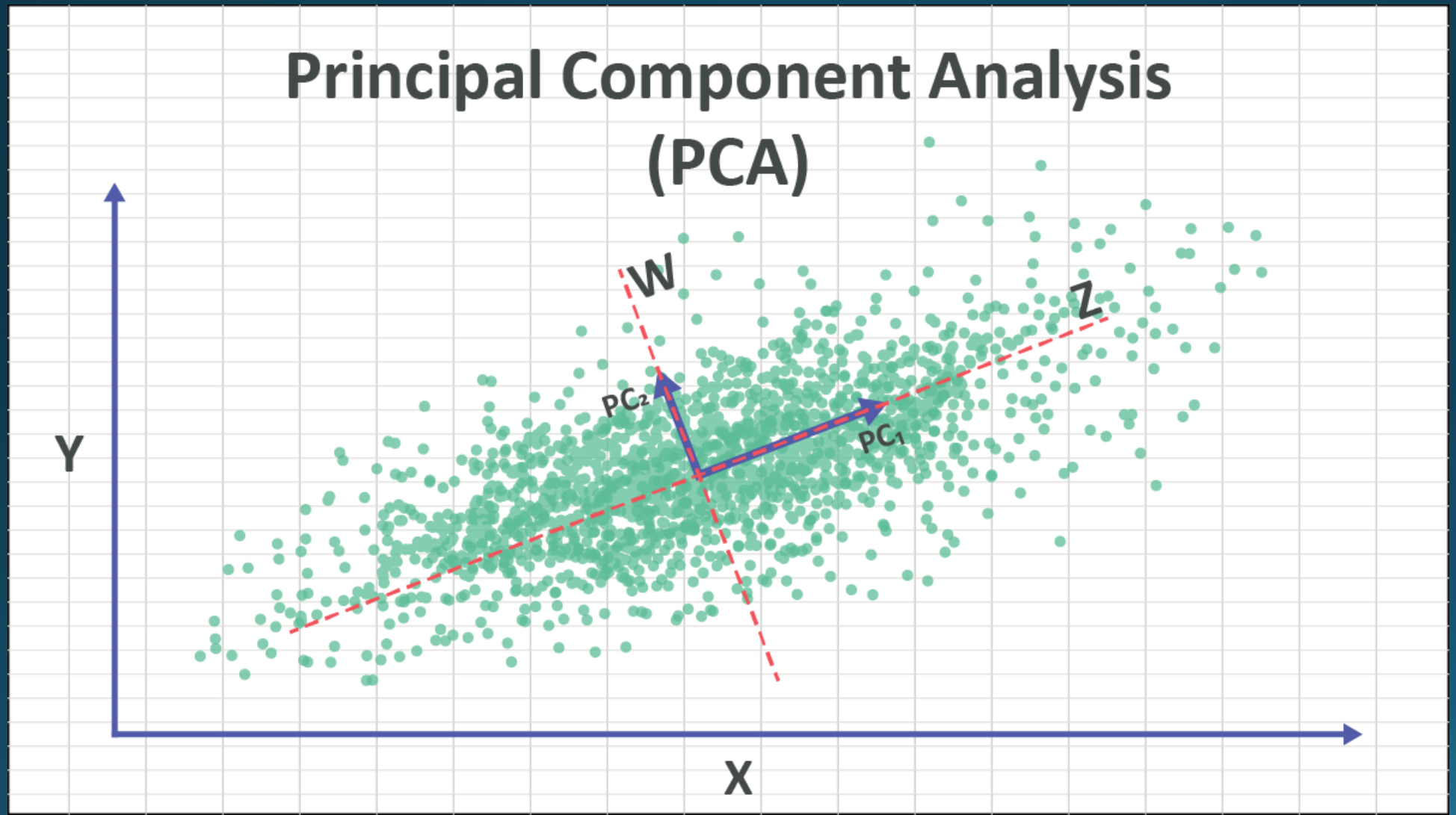
PCA transforms a dataset into a new coordinate system, where each axis (principal component) captures a different aspect of the data's variance.

The primary goal of PCA is to reduce the number of features in a dataset while retaining as much variability as possible.

Output:

PCA generates a set of uncorrelated components ranked by how much variance they capture, allowing for a lower-dimensional data representation with minimal information loss.

Principal Component Analysis (PCA)



Key Concepts in PCA

Principal Components:

These are new, uncorrelated variables that represent the directions in which the data varies the most. The first principal component captures the highest variance, followed by the second, and so forth.

Variance and Dimensionality Reduction:

By retaining only the first few principal components that explain most of the variance, PCA reduces dimensionality while preserving data structure.

Linear Transformation:

PCA is a linear method, meaning it works well when data has a linear structure, but may not capture complex non-linear relationships.

Real-World Applications of PCA

Image Compression:

Reducing the dimensionality of pixel data allows for compressed storage and faster image processing.

Gene Expression Analysis:

PCA can reduce thousands of gene expression measurements to a few components, making patterns and relationships easier to detect.

Customer Segmentation:

In marketing, PCA can reduce complex customer behavior data, revealing patterns and similarities for better segmentation.

Real-World Applications of PCA

Finance:

In portfolio management, PCA helps reduce the dimensionality of asset returns, focusing on principal components that capture the majority of market variance.

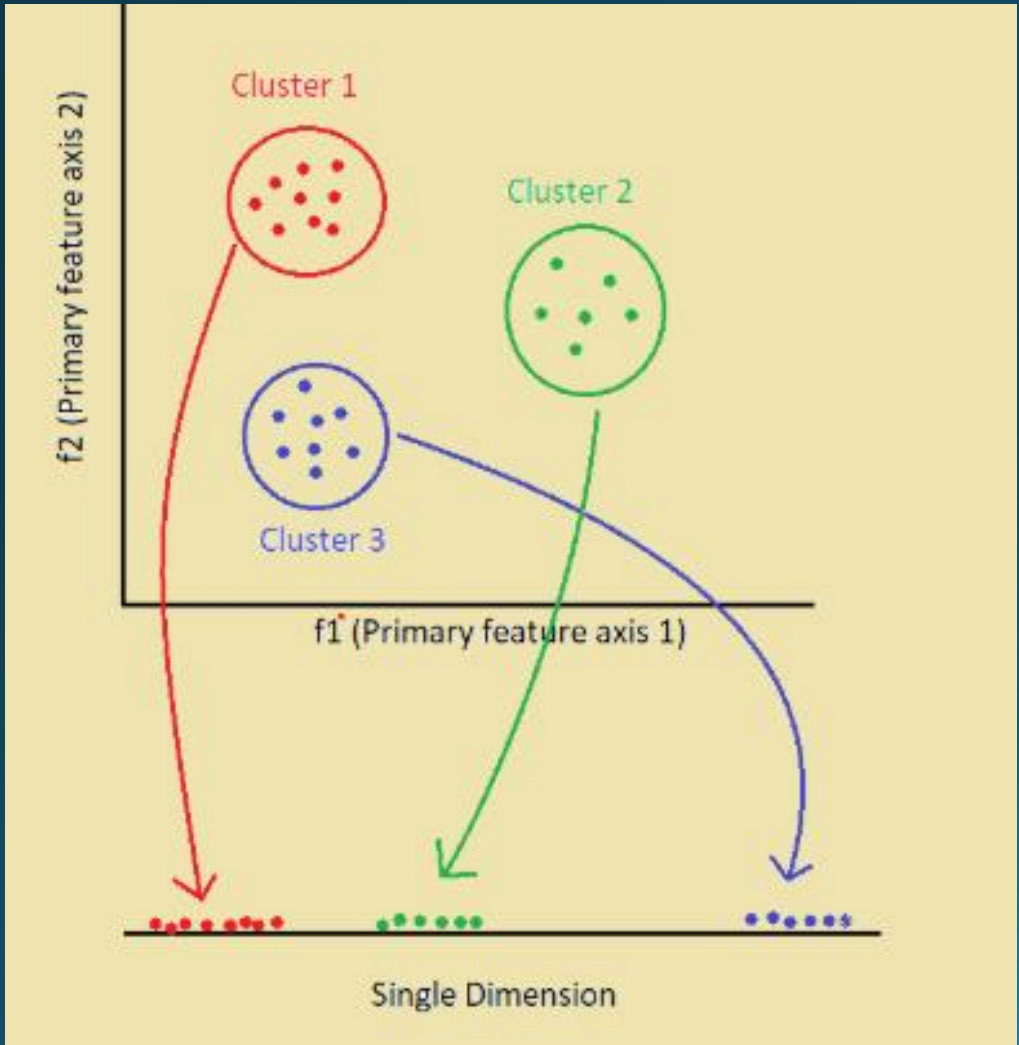
t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a method that reduces the dimensionality of high-dimensional data by placing similar data points close together in the reduced space while pushing dissimilar points farther apart.

Objective:

The goal of t-SNE is to create a map that reflects the structure of the data in a low-dimensional space, focusing on preserving *local* similarities between points.

t-Distributed Stochastic Neighbor Embedding (t-SNE)



How t-SNE Works

Convert High-Dimensional Distances to Probabilities:

- In the high-dimensional space, t-SNE converts distances between pairs of points into conditional probabilities. These probabilities reflect the likelihood that a point *A* would pick point *B* as a neighbor, based on a Gaussian distribution centered at *A*.
- This transformation captures the local structure of data by assigning high probabilities to nearby points and low probabilities to distant ones.

How t-SNE Works

Construct a Similar Probability Distribution in Low Dimensions:

- In the lower-dimensional space, t-SNE tries to replicate these pairwise probabilities by mapping points into a new configuration.
- To prevent "crowding" of distant points, t-SNE uses a *Student's t-distribution* instead of a Gaussian in the low-dimensional space, which has heavier tails and better separates clusters.

How t-SNE Works

Example:

Imagine you have a dataset of handwritten digits (e.g., the MNIST dataset with each digit image having hundreds of features).

t-SNE can reduce this data to two dimensions, allowing a 2D scatter plot where each digit cluster can be identified by its unique pattern, even though the data started in a very high-dimensional space.

Key Parameters in t-SNE

Perplexity:

A parameter related to the number of nearest neighbors t-SNE considers for each point, usually ranging between 5 and 50. Higher perplexity values account for broader neighborhoods, balancing local and global aspects of the data.

Learning Rate:

Controls how fast or slow the optimization progresses. A too-small learning rate can lead to slow convergence, while a too-large learning rate may cause poor clustering.

Key Parameters in t-SNE

Iterations:

The number of optimization iterations (default is 1000-5000). More iterations can lead to better embedding quality but increase computation time.

Advantages of t-SNE

Effective at Capturing Complex Structures:

t-SNE is ideal for visualizing data with complex, non-linear relationships and multiple clusters.

Local Similarity Preservation:

By preserving local neighborhoods, t-SNE is particularly useful in tasks where clusters, such as categories in image or text data, need clear separation.

Highly Visual Interpretability:

t-SNE produces visually meaningful plots that make it easy to identify patterns, groupings, and outliers, especially in exploratory data analysis.

Real-World Applications of t-SNE

Genomics and Bioinformatics:

In gene expression analysis, t-SNE helps visualize high-dimensional gene or protein expression data, identifying clusters or patterns associated with specific biological states or diseases.

Natural Language Processing (NLP):

t-SNE visualizes high-dimensional word embeddings or document representations, helping reveal semantic relationships between words or documents.

Real-World Applications of t-SNE

Image Recognition and Classification:

Reduces high-dimensional image data, allowing visualization of complex patterns. In the MNIST handwritten digits dataset, for example, t-SNE can reveal distinct clusters for each digit.

Customer Segmentation:

In marketing analytics, t-SNE can visualize customer segments based on high-dimensional behavior data, making it easier to identify and target clusters of similar customers.