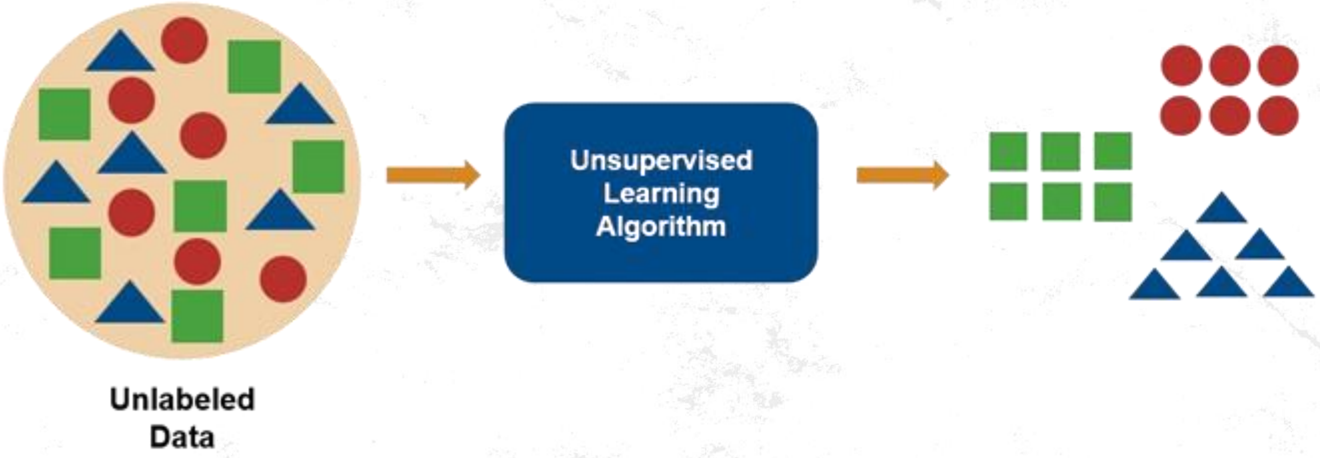


Unsupervised Learning

NIELIT ROPAR

What We Will Cover

1 **What is Unsupervised Learning?**
(Supervised vs. Unsupervised)

2 **Introduction to Clustering**
(Goals & Real-World Applications)

3 **Measuring "Closeness"**
(Euclidean & Manhattan Distance)

4 **Hierarchical Clustering**
(Agglomerative, Divisive, & Dendrograms)

5 **Partitional Clustering**
(The K-Means Algorithm)

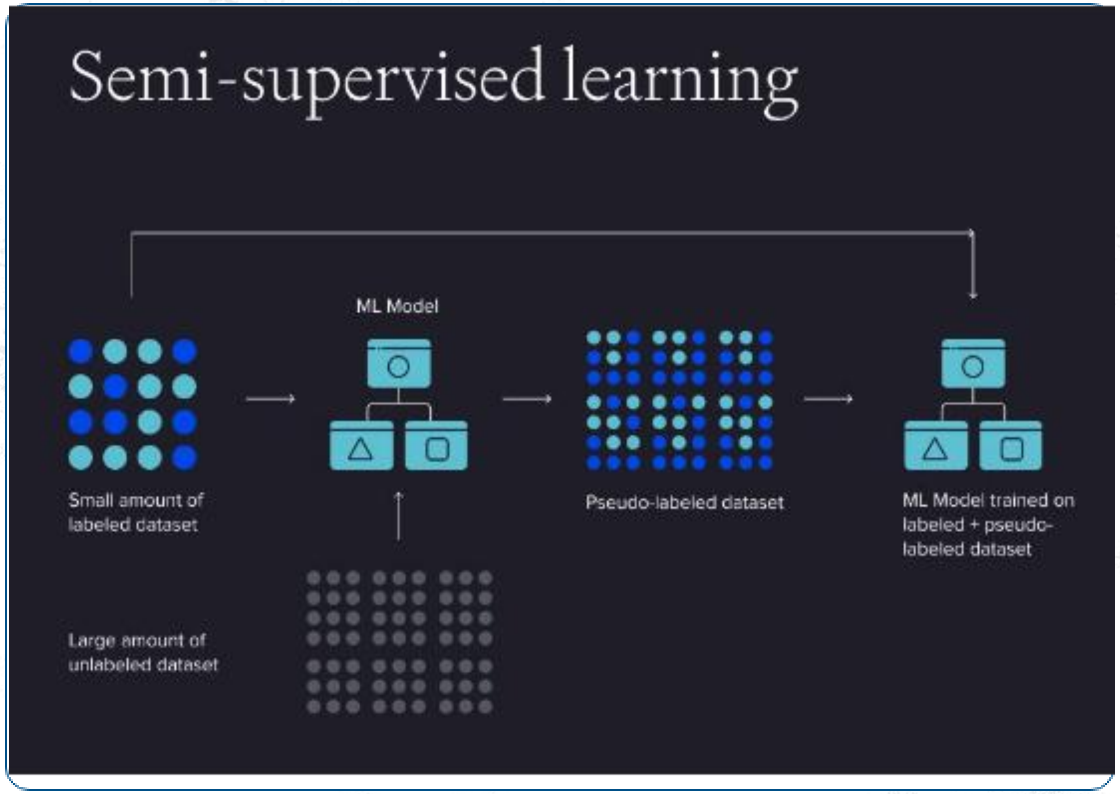
6 **Summary & Comparison**

“The brain has far more parameters than labeled data — so it must learn almost everything without supervision.”— **Geoffrey Hinton**

Supervised vs. Unsupervised Learning

Supervised ("Learning with a Teacher")

- **Data:** Labeled data (e.g., photos tagged "cat," "dog").
- **Goal:** Learn a mapping from inputs (X) to outputs (Y).
- **Example:** Predicting house prices based on past sales data.

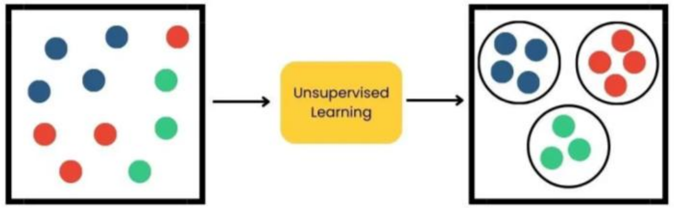


Unsupervised ("Learning on Your Own")

- **Data:** Unlabeled data (e.g., a folder full of photos).
- **Goal:** Find hidden structure or patterns *in* the data.
- **Example:** Grouping similar photos together automatically.

Unsupervised Learning

- Learning on Your Own
- Unsupervised learning finds patterns in data without labeled answers.

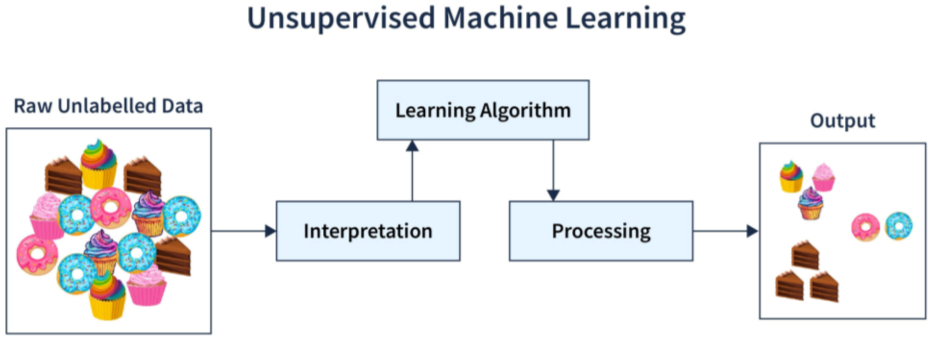


What is Unsupervised Learning?

- A machine learning method with no predefined labels.
- The algorithm identifies structure and patterns independently.
- Example: Grouping customers by buying behavior.

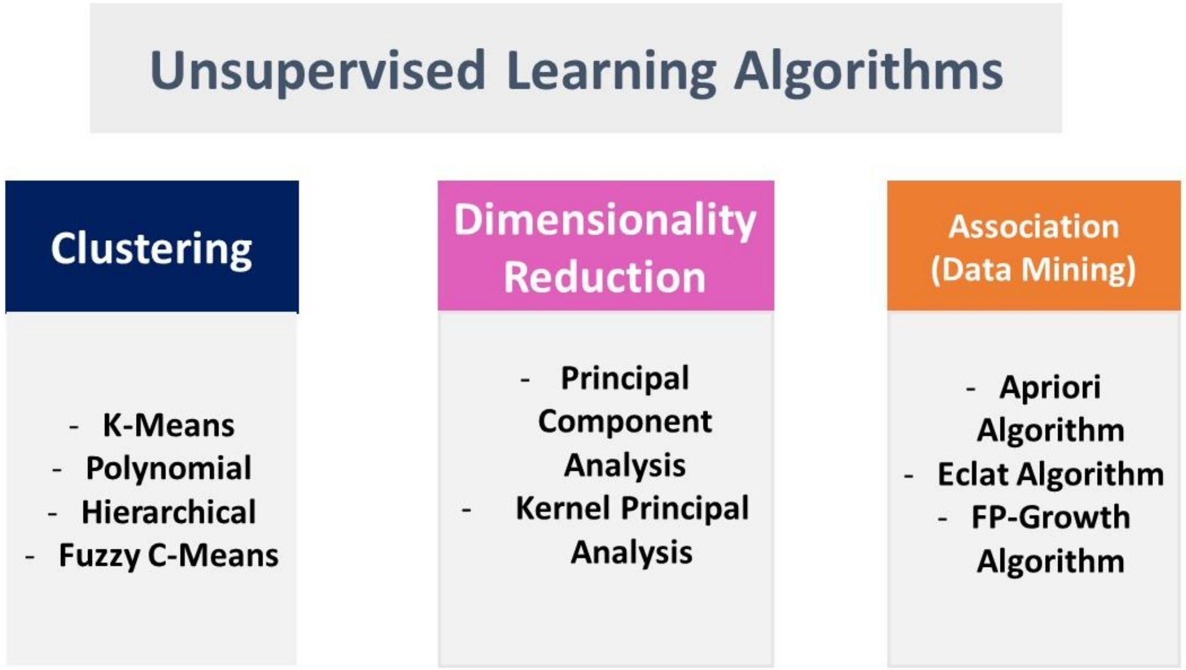
Why Use Unsupervised Learning?

- Helps discover unknown patterns.
- Useful when labeled data is not available.
- Supports data exploration and preprocessing.



Key Types

1. Clustering
2. Association
3. Dimensionality Reduction



Introduction to Clustering

The Goal of Clustering

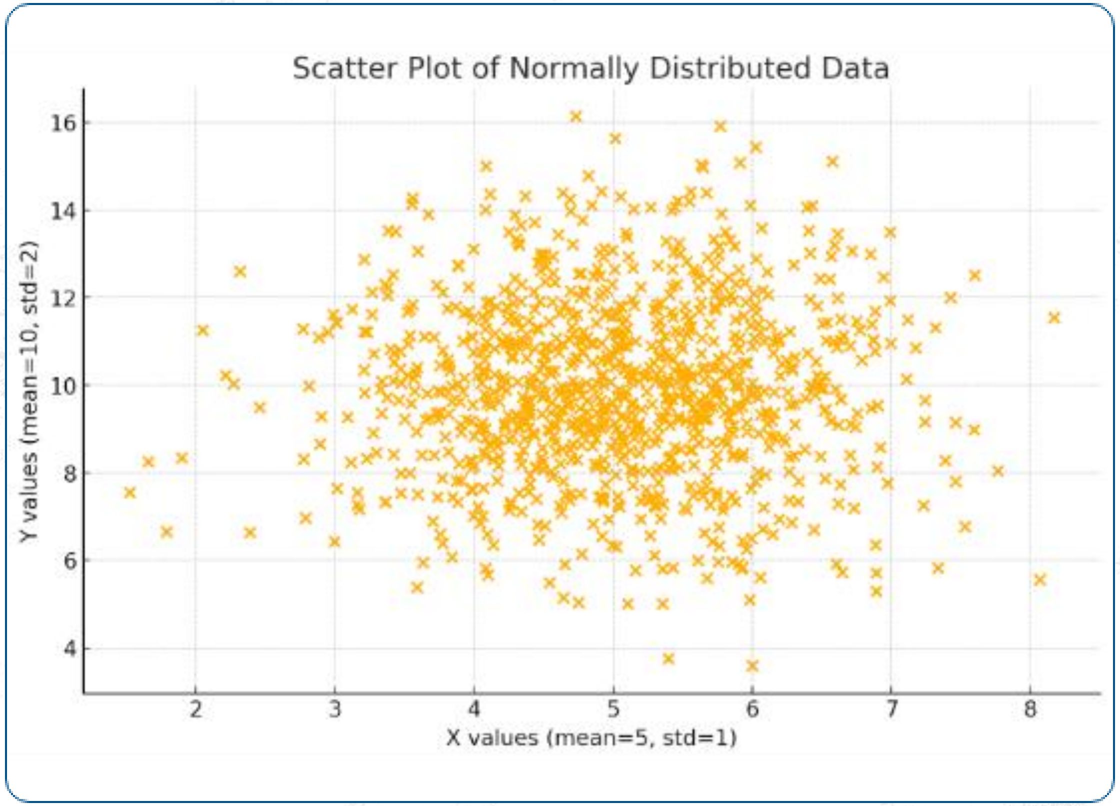
The fundamental task is to divide data points into groups (clusters) based on their similarity.

1. High Intra-cluster Similarity

Points **within** the same cluster should be as similar as possible.

2. Low Inter-cluster Similarity

Points in **different** clusters should be as dissimilar as possible.



Introduction to Clustering

- Technique to group similar data instances together
- Objective: maximize similarity within clusters and differences between clusters
- Examples: Customer segmentation, document grouping

Types of Clustering

1. Hierarchical Clustering

- Agglomerative Clustering (Bottom-Up)
- Divisive Clustering (Top-Down)

2. Partitional Clustering

- K-Means Clustering

Hierarchical Clustering

Hierarchical Clustering is a type of unsupervised machine learning technique that groups data into clusters by building a **hierarchy (tree-like structure)** of clusters. It either starts with every data point as its own cluster and then merges them step-by-step (**Agglomerative – bottom-up**) or starts with one large cluster and splits it into smaller clusters (**Divisive – top-down**). The result is usually visualized using a **dendrogram**, which helps decide the number of clusters.

- Builds a hierarchy of clusters
- Visualized using a Dendrogram
- Does not require specifying number of clusters initially

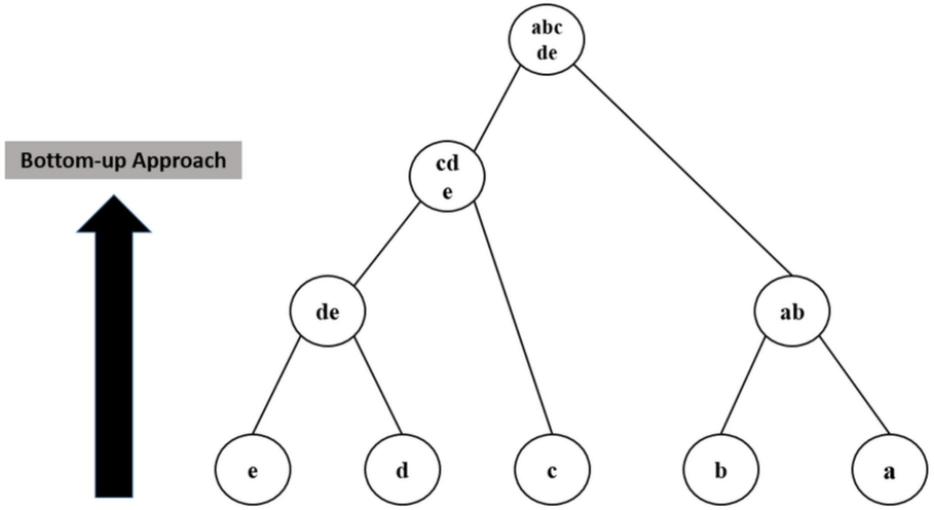
Hierarchical clustering does not form all clusters at once. Instead, it builds them gradually, creating a **tree** that shows how data points are grouped together. This is why it is called *hierarchical*.

Agglomerative Clustering (Bottom-Up)

- Each data point starts as an individual cluster
- Clusters are repeatedly merged based on similarity
- Continues until one final cluster remains

Agglomerative Clustering is the most common type of hierarchical clustering. It is a **bottom-up** approach, meaning:

Every data point starts as its **own separate cluster**, and then pairs of the most similar clusters are **merged step-by-step** until only one cluster (or a desired number of clusters) remains.



An illustration of Agglomerative Clustering (bottom-up approach)

How it Works (Step-by-Step)

1. Start with **N clusters** (each data point = 1 cluster)
2. Compute the **distance** between all clusters
3. **Merge** the two closest clusters
4. Recalculate cluster distances
5. Repeat until desired number of clusters is reached

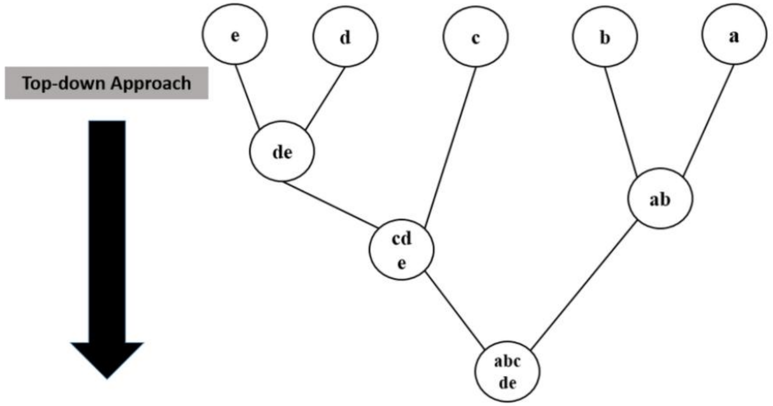
Divisive Clustering (Top-Down)

- Starts with all data points in a single cluster
- Splits cluster into smaller clusters step-by-step
- Less common but conceptually opposite of agglomerative

Divisive Clustering is a hierarchical clustering technique that follows a **top-down** approach. It begins with **all data points grouped into one large cluster**, and then repeatedly **splits** the cluster into smaller clusters until each data point stands alone or the desired number of clusters is reached.

How it Works (Step-by-Step)

Start with **one single cluster** containing all data points
Identify the cluster that is most dissimilar internally
Split this cluster into two smaller clusters
Select a cluster to split again (usually the most heterogeneous one)
Repeat until the desired number of clusters is reached



Why is it called "Top-Down"?

Because it works like breaking a big branch into smaller branches

Real-World Applications

Customer Segmentation

Grouping customers by purchasing habits for targeted marketing.

Genomic Analysis

Classifying genes or proteins that have similar functions.

Anomaly Detection

Identifying outliers that don't belong to any cluster (e.g., fraud).

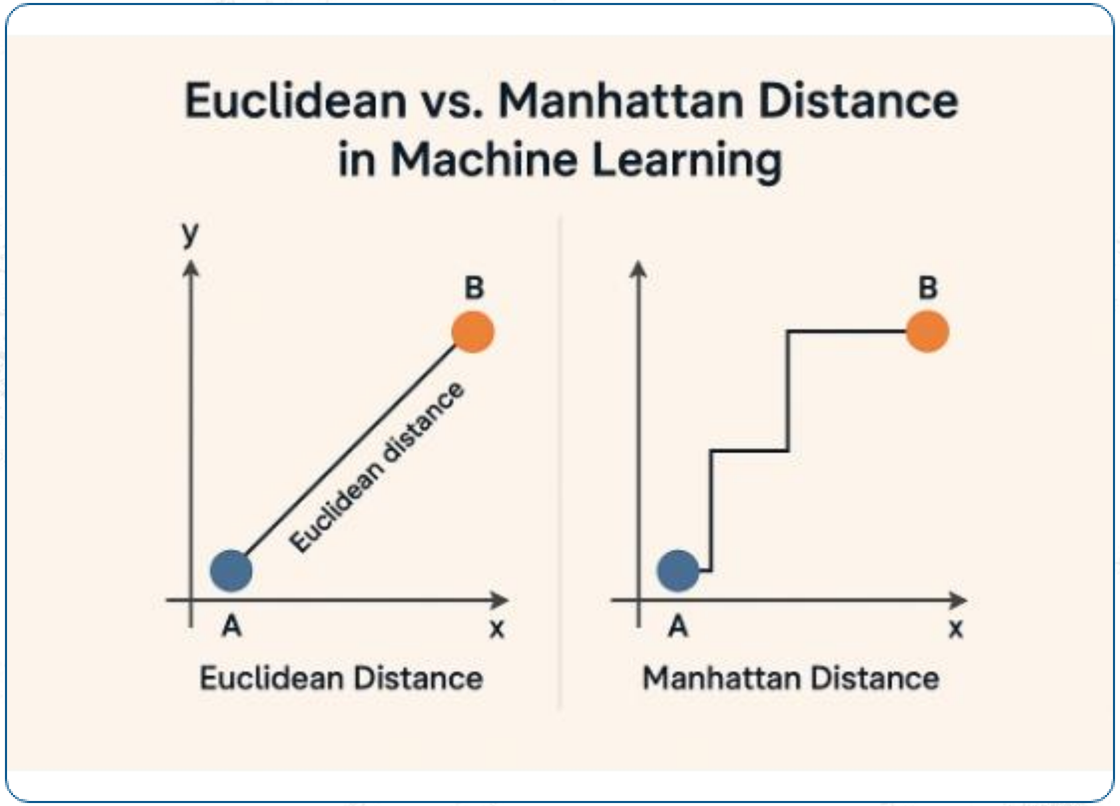
How We Measure "Closeness" (Distance Metrics)

Euclidean Distance

The "straight-line" distance between two points (like using a ruler). It's the most common metric.

Manhattan Distance

The "city block" distance. The sum of the absolute differences of their coordinates.

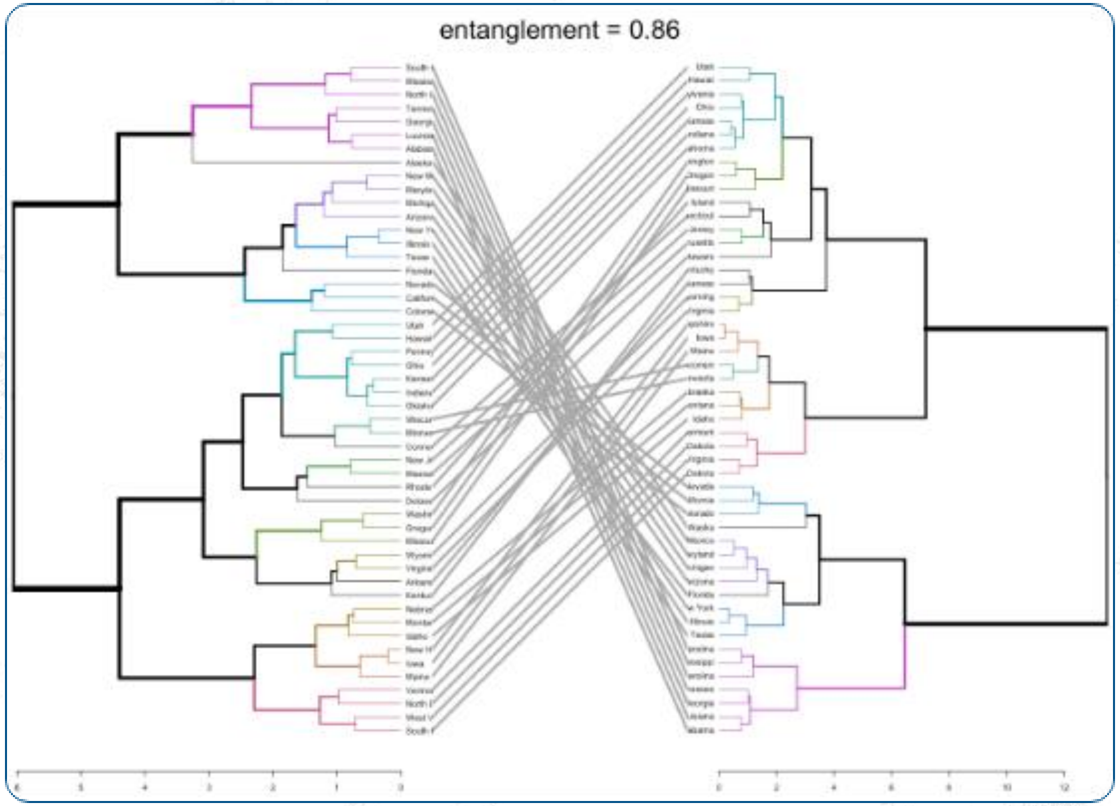


The Two Main Families of Clustering

1. Hierarchical Clustering

Builds a tree-like hierarchy (a dendrogram) of clusters.

- **Pro:** Does **not** require the number of clusters ('K') to be specified in advance.
- **Methods:** Agglomerative (bottom-up) or Divisive (top-down).



2. Partitional Clustering

Divides the dataset into a fixed number ('K') of non-overlapping clusters.

- **Pro:** Fast and efficient, especially for large datasets.
- **Famous Algorithm:** K-Means.

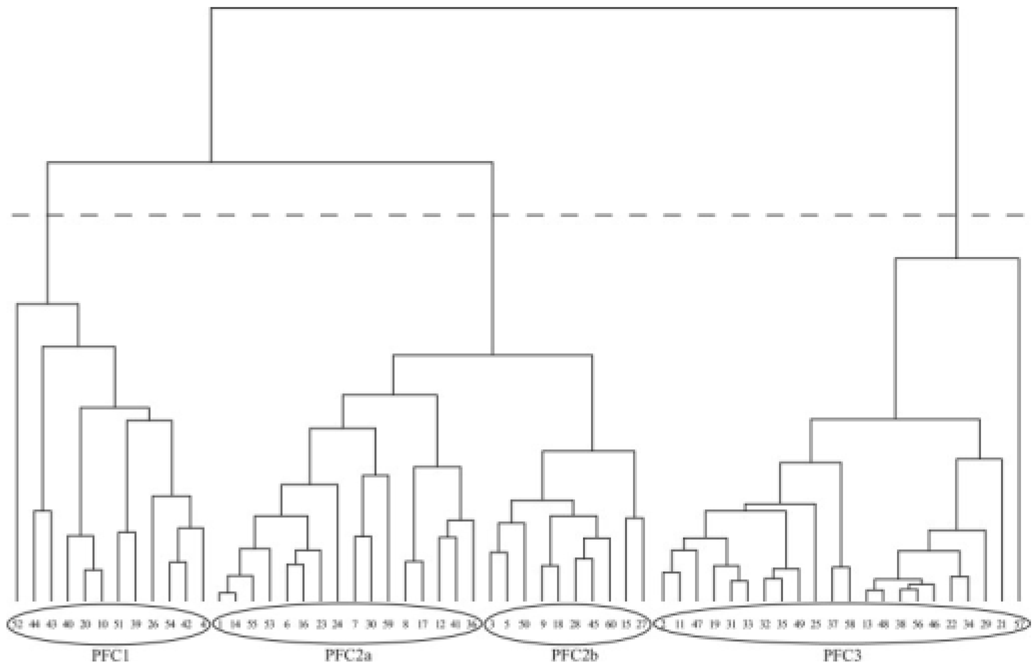
Hierarchical Clustering

Understanding the Dendrogram

This is the **output** of hierarchical clustering. It's a tree diagram that shows how clusters are merged.

How to read it:

- **X-axis:** The individual data points (samples).
- **Y-axis:** The "Distance" or dissimilarity.
- **Merges:** The horizontal lines show which clusters (or points) were merged.
- **Cut the Tree:** You can draw a horizontal line ("cut") across the tree to get a specific number of clusters.

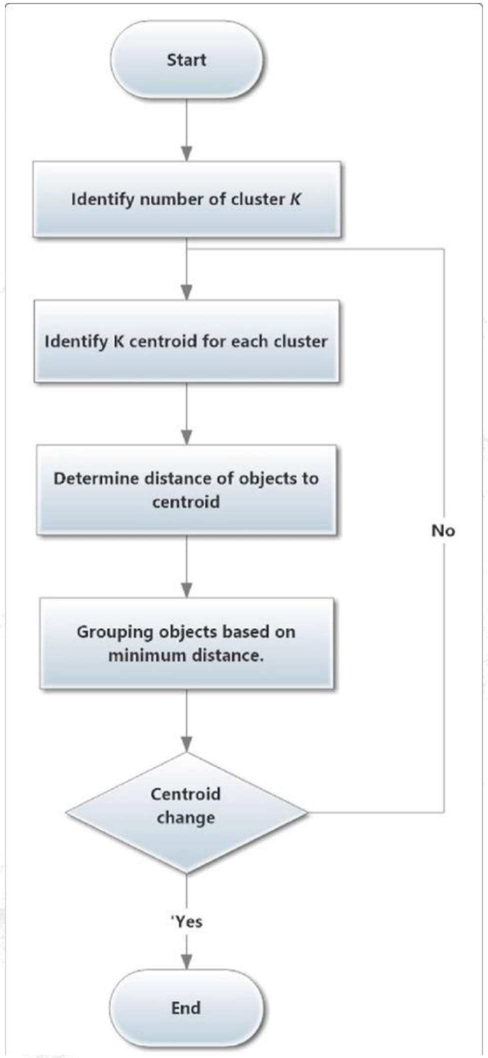


Partitional Clustering - K-Means

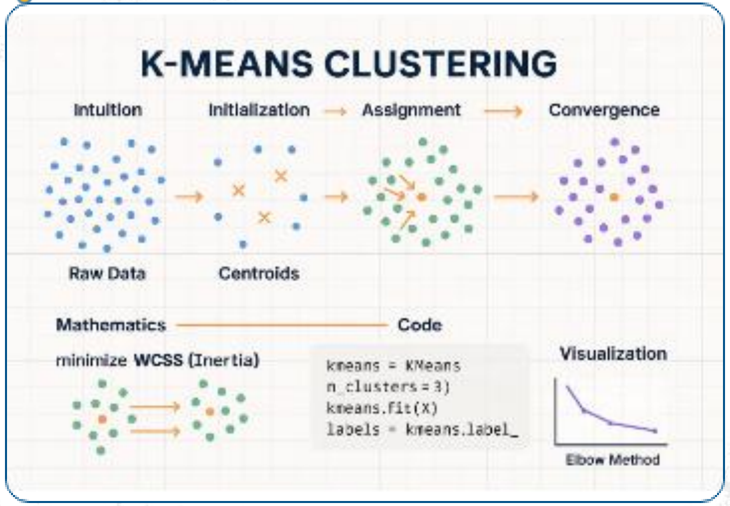
- Divides data into K distinct clusters
- Based on distance between data points and cluster centroids
- Requires pre-defining number of clusters (K)

K-Means: Algorithm Steps

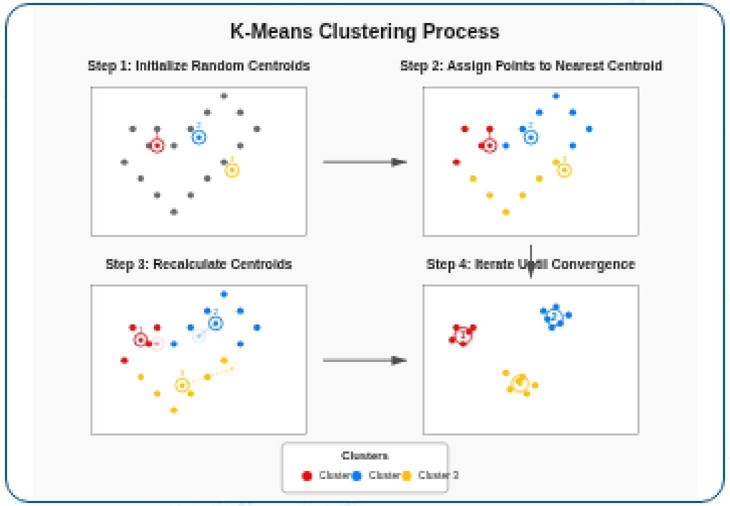
1. Select K cluster centers
2. Assign data points to nearest center
3. Recalculate cluster centroids
4. Repeat until convergence



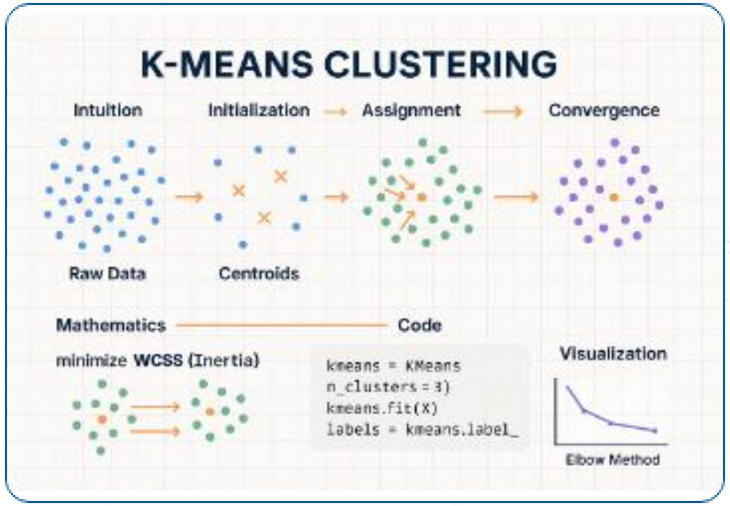
Deep Dive: The K-Means Algorithm Steps



Step 1: Initialize
Randomly place 'K' centroids.



Step 2: Assign
Assign each point to its nearest centroid.



Step 3 & 4: Update & Repeat
Move centroids to the mean. Repeat until stable.

Python Example: K-Means

A step-by-step walkthrough using Scikit-learn.

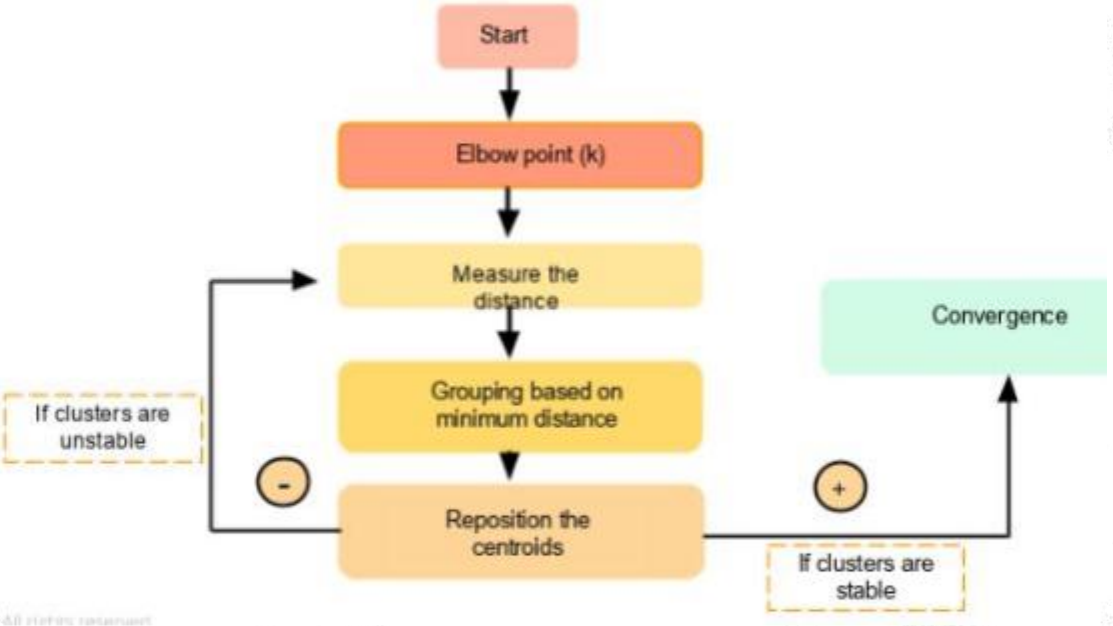
What is K-Means? A Quick Refresher

K-Means is an algorithm to find a specific number of groups (k) in a dataset.

Goal: To partition data points into 'k' distinct, non-overlapping clusters.

How: It iteratively assigns points to the nearest cluster "center" (centroid) and then updates the centroid's position based on the mean of the assigned points.

Use Cases: Customer segmentation, image compression, document clustering.



The Code, Part 1: Setup & Data

1. Setup & Imports

First, we import our libraries. We need:

- matplotlib for plotting.
- make_blobs to create a sample dataset of "blobs" (groups).
- KMeans from Scikit-learn, which is the algorithm itself.

We then generate 300 sample points grouped into 3 centers.

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

# 1. Generate Sample Data
X, y_true = make_blobs(
                n_samples=300,
                centers=3,
                cluster_std=0.8,
                random_state=42
        )
```

The Code, Part 2: Create & Fit Model

2. Create & Fit Model

This is the core of the algorithm. We create a KMeans object, telling it to find 3 clusters (n_clusters=3).

We use n_init=10 to run the algorithm 10 times with different starting points and pick the best result.

Then, we .fit() the model to our data X.

Finally, we get the .labels_ (which cluster each point belongs to) and the .cluster_centers_ (the final centroid coordinates).

```
# 2. Create and Fit K-Means
kmeans = KMeans(
            n_clusters=3,
            n_init=10,
            random_state=42
         )
kmeans.fit(X)

# 3. Get the Results
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

The Code, Part 3: Visualization

3. Plot the Results

Now, we use matplotlib to plot our results.

- The first plt.scatter plots all our data points (X) and colors them (c=labels) based on the cluster they were assigned to.
- The second plt.scatter plots the final centroids as large red 'X's.

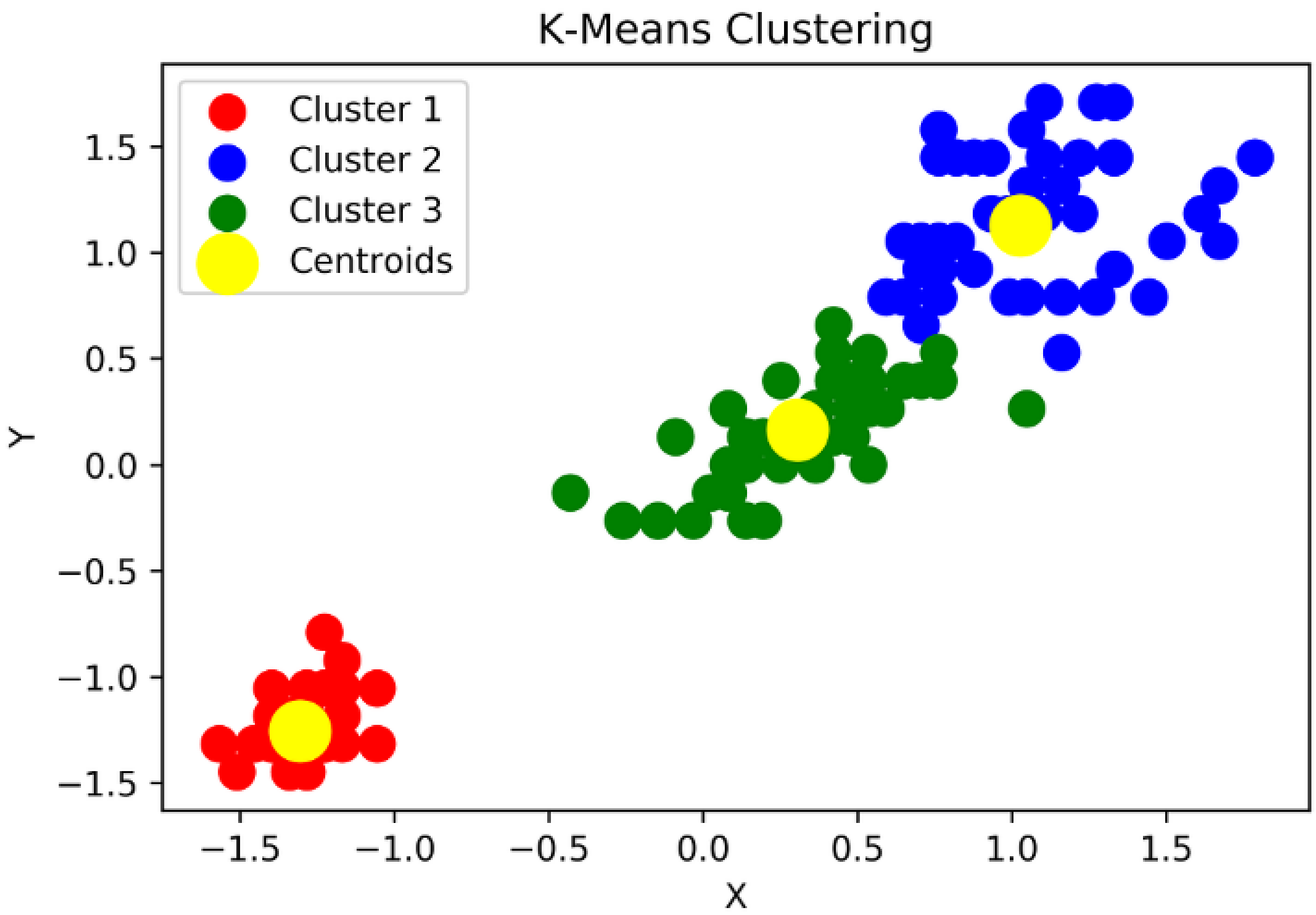
```
# 4. Plot the Results
plt.figure(figsize=(8, 6))

# Plot data points
plt.scatter(
            X[:, 0], X[:, 1],
            c=labels,
            s=50,
            cmap='viridis'
        )

# Plot centroids
plt.scatter(
            centroids[:, 0],
            centroids[:, 1],
            c='red', s=200,
            marker='X',
            label='Centroids'
        )

plt.title('K-Means Clustering' )
plt.legend()
plt.show()
```

The Final Result



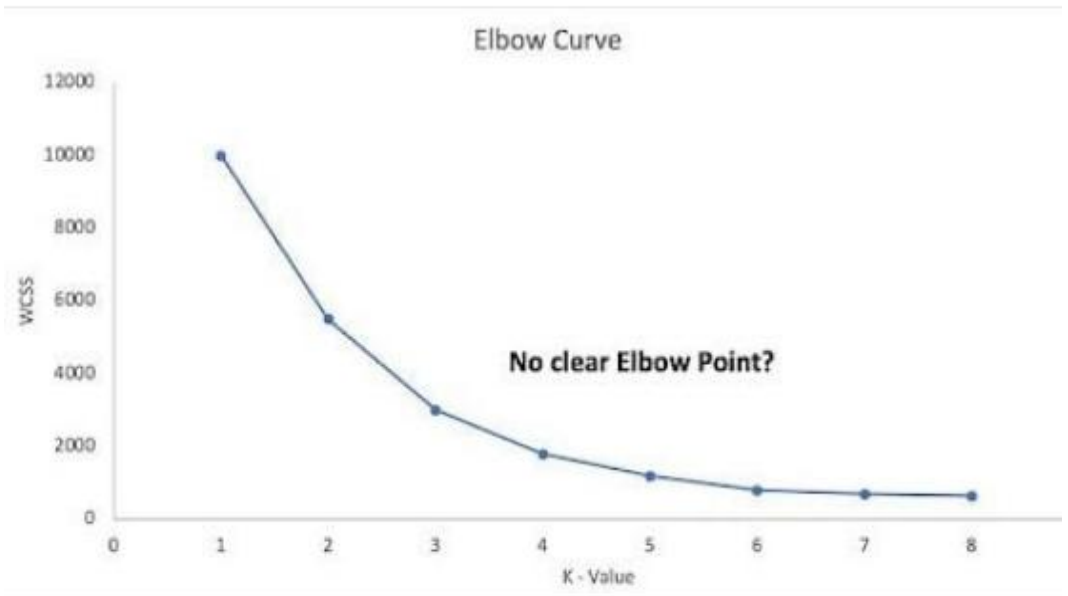
K-Means Challenge: How to Choose 'K'?

The Solution: The Elbow Method

K-Means requires **you** to specify 'K' in advance. The Elbow Method helps you find a good value.

How it works:

1. Run K-Means multiple times with different 'K' values (e.g., K=1, 2, ...10).
2. Plot 'K' vs. WCSS (Within-Cluster Sum of Squares). WCSS measures how compact the clusters are.
3. Look for the "elbow": the point where WCSS stops decreasing rapidly. This is a good estimate for the optimal 'K'.



Comparison: K-Means vs. Hierarchical

Feature	K-Means (Partitional)	Hierarchical (Agglomerative)
Number of Clusters (K)	Must be specified in advance.	Not required. Determined from dendrogram.
Computational Speed	Fast ($O(n)$). Good for large data.	Very slow ($O(n^3)$). Not for large data.
Output	A single set of K clusters.	A full hierarchy (dendrogram).
Cluster Shape	Assumes clusters are spherical.	Can handle any cluster shape.

Summary

- Unsupervised learning finds patterns without labels
- Clustering groups similar data points
- Hierarchical: Agglomerative (bottom-up) and Divisive (top-down)
- Partitional: K-Means algorithm
- Widely used across data-driven industries

Google Colab Notebook: https://colab.research.google.com/drive/1uocUh0pEgFOOGICUIZJglKvD8N5otluh?usp=sharing

Scan for Google Colab Link

Questions?

Thank you.

NIELIT ROPAR