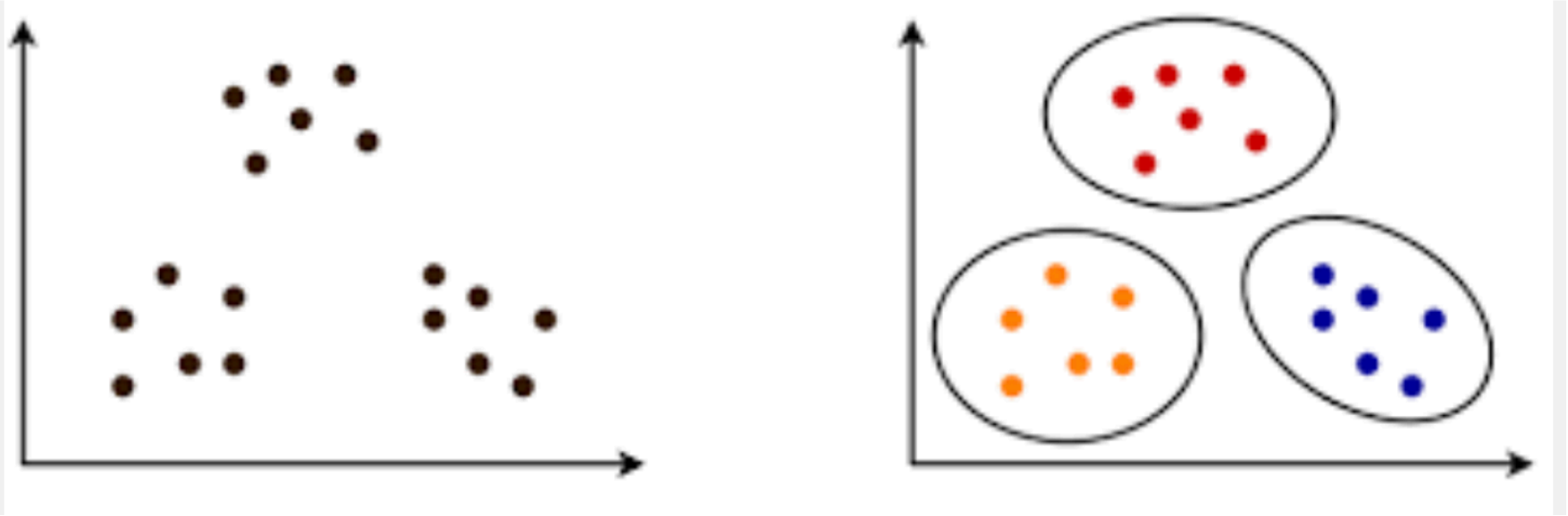
CLUSTERING



Introduction:

- We cluster our data into groups based on the similarities of the individual.

- Allow a buisiness to target different group (high profits & low risk, ..)

Example of clustering:

- **Retail & Marketing**: Identify buying patterns / recommendation systems
- **Banking**: Fraud detection / identify clusters (loyal, churn, …)
- **Insurance**: Fraud detection / Risk
- **Publication**: Auto-categorize / recommend
- **Medicine**: Characterize behavior
- **Biology**: Group genes / cluster genetic markers (family ties)

1. Insurance: Fraud Detection / Risk

Example 1 – Fraud Detection:
An insurance company applies K-means clustering to detect fraud in health insurance claims. By analyzing past claims, the model identifies clusters of normal versus suspicious claims. A new claim that falls into a suspicious cluster (e.g., unusually high medical expenses, multiple claims from different locations) is flagged for further investigation.

Example 2 – Risk Assessment:
In car insurance, a company groups customers based on their risk levels. Using K-means clustering, it identifies high-risk drivers based on factors such as accident history, age, vehicle type, and driving behavior. This helps in determining appropriate premium rates for different customer segments.

2. Publication: Auto-Categorize / Recommend

Example 1 – Auto-Categorization:
A **scientific journal** uses **clustering** to group submitted research papers into different categories (e.g.,astrophysics, machine learning, medicine). Instead of manually classifying each paper, **K-means groups papersbased on keywords, citations, and abstract content**, making the publication process more efficient.

Example 2 – Recommendation System:
An **online news website** recommends articles based on user reading behavior. By clustering users into different interest groups (e.g., politics, sports, technology), the system suggests articles that are most relevant to each user, improving engagement and content personalization.

Clustering methods:

- **Partitioned-based** (K-means, K-Median, Fuzzy c-means, ...):
  - Sphere-like clusters
  - Medium or large data

- **Hierarchical** (Agglomerative, Divisive):
  - Trees of clusters
  - Small size datasets

- **Density-based** (DBSCAN):
  - Arbitrary shaped clusters
  - Good for special clusters or noisy data

K-means

The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* patitions, where $k < n$.

- We need to understand the similarity and dissimilarity.
- **Goal:** It minimizes the distance between points in the same cluster while maximizing the distance between different clusters.
- It is always good to **Normalize**!
- Different formulas: **Euclidean, Cosine, Average distance**, ... so first understand the domain knowledge.

Clustering - K-Means

- Decide the number of clusters (K).

- In K-means, centroids are initialized by:
  - Selecting random points from the dataset to be the center of the cluster.

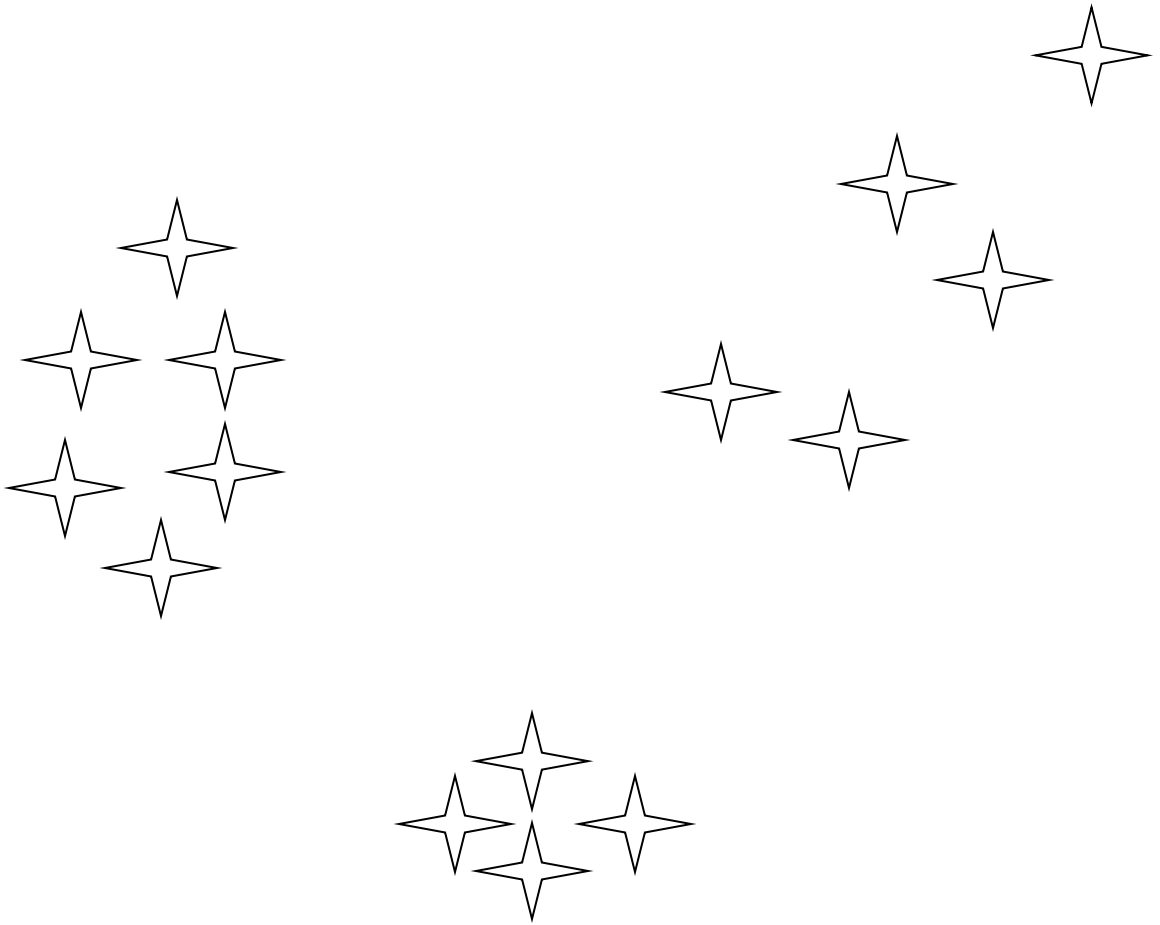
- Assign each data point to the closest centroid and compute the distance matrix.

- Update the centroid to the mean of its assigned data points.

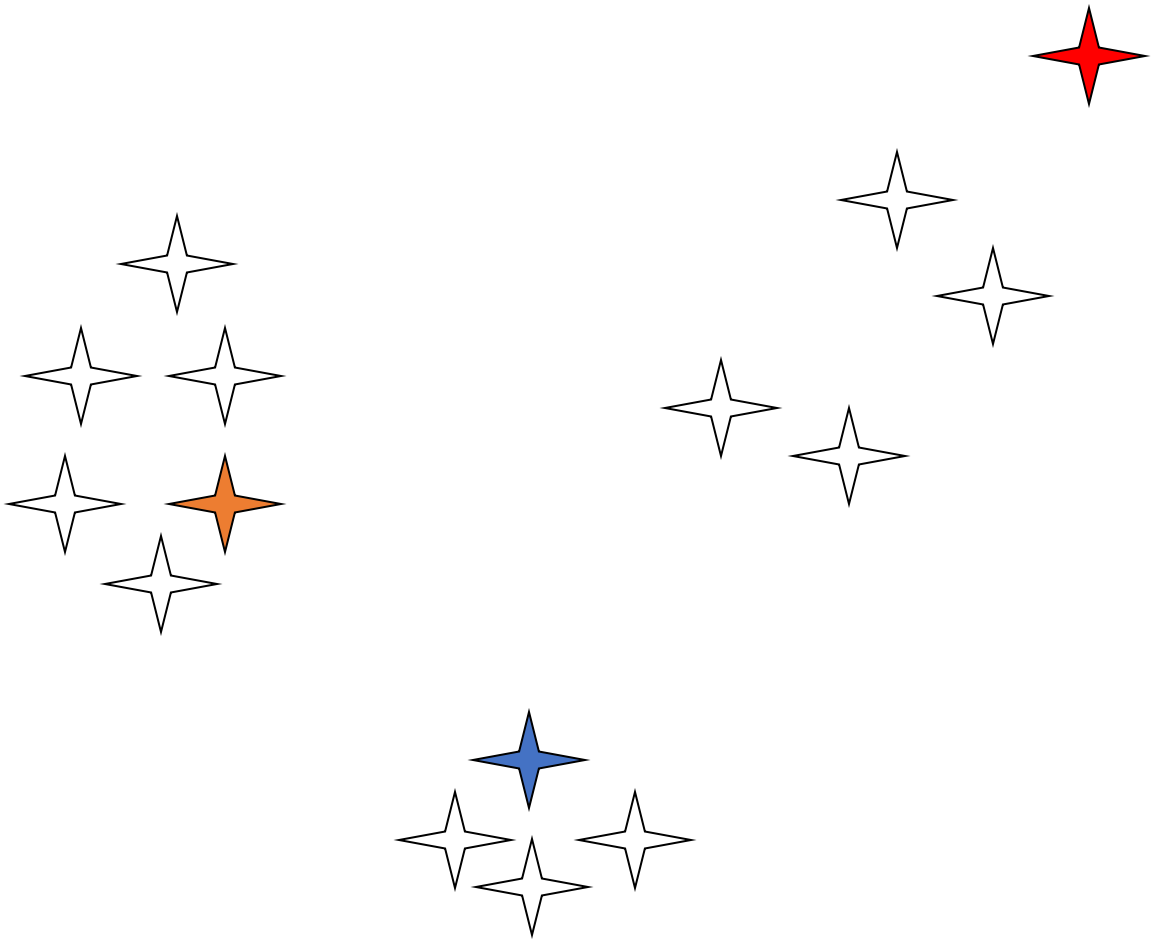
- Repeat until the centroids stop moving.

Notes:
- K-Means does not guarantee the best result, as it may get stuck in a local optimum.
- It is fast, so we can run it multiple times to find a better solution.

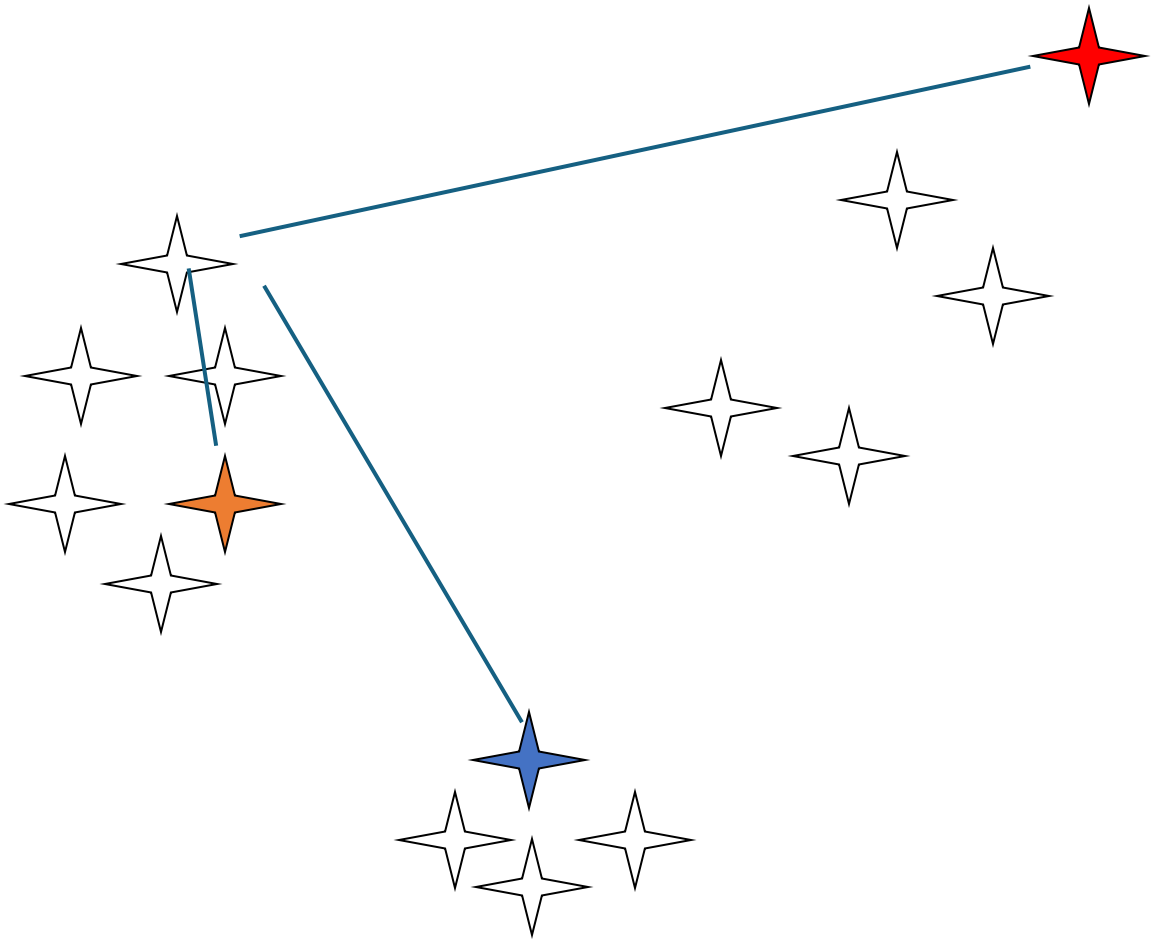
Step 1: Choose how many cluster



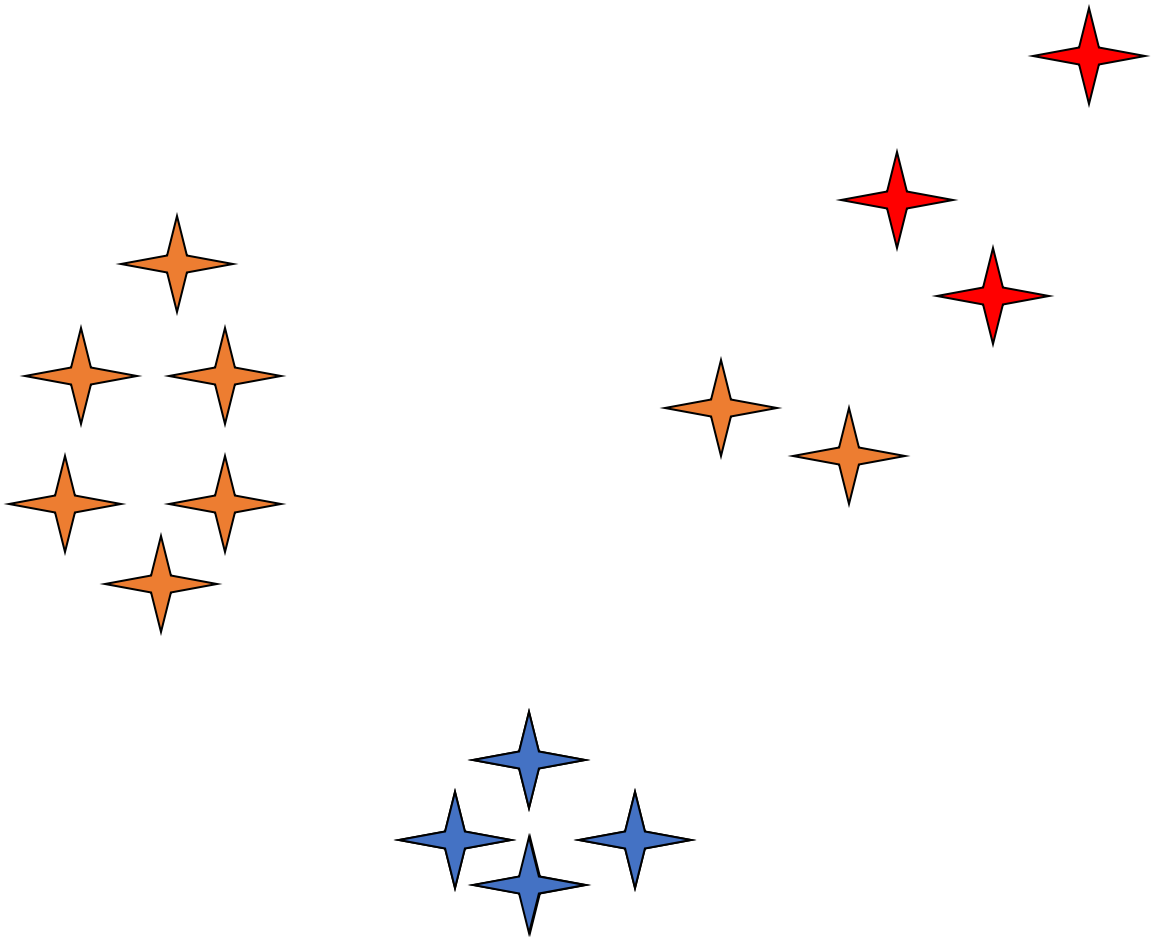
Step 2: Choose three random data



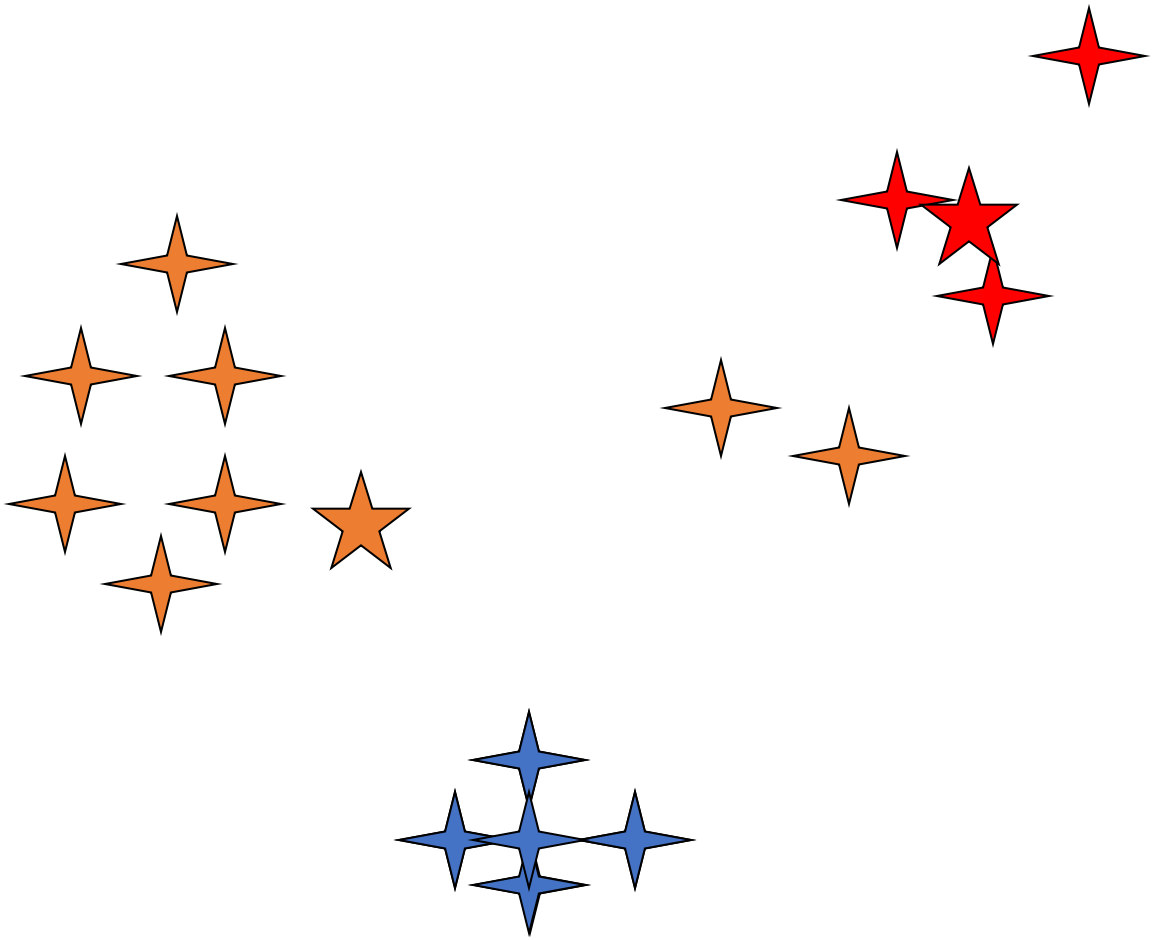
Step 3: calculated the distance and put the close points into a cluster



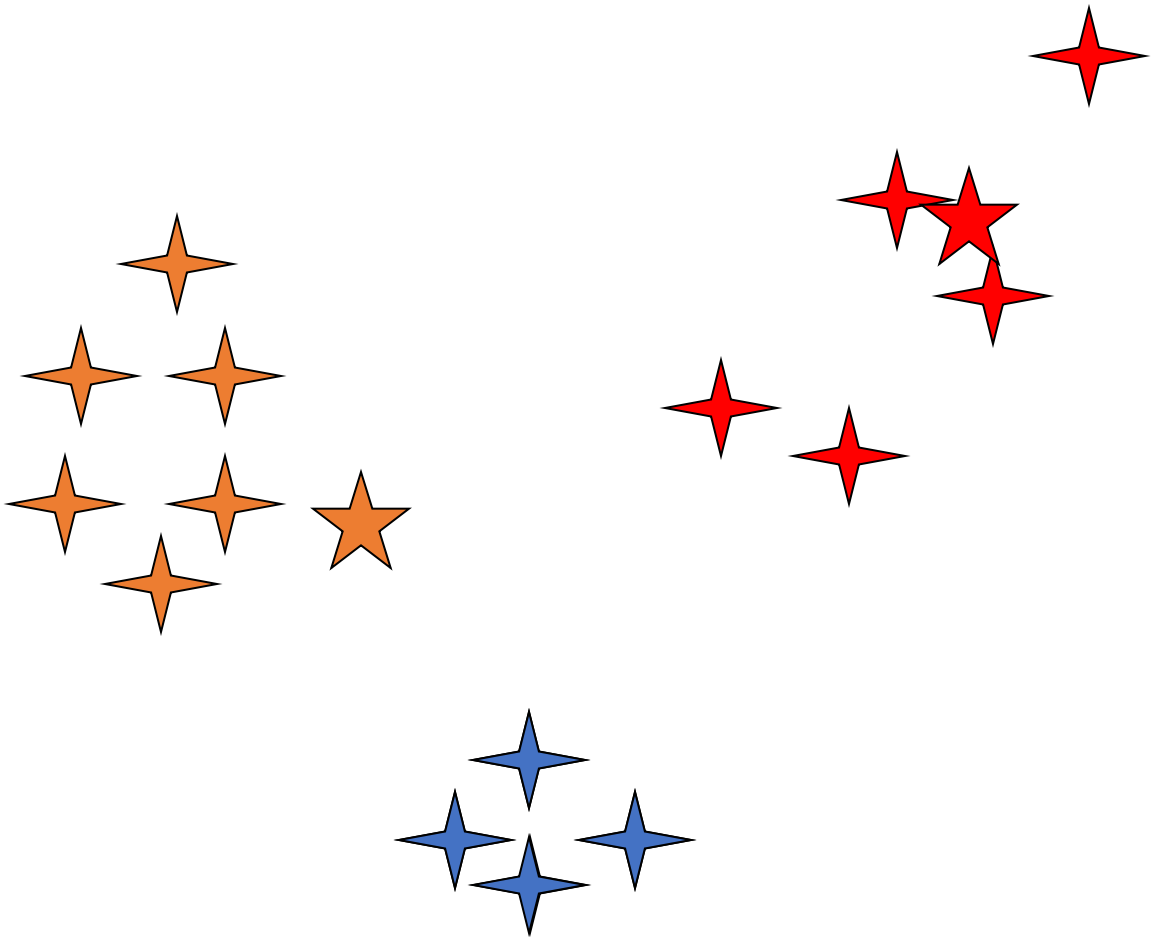
Form clusters around them - Step 2

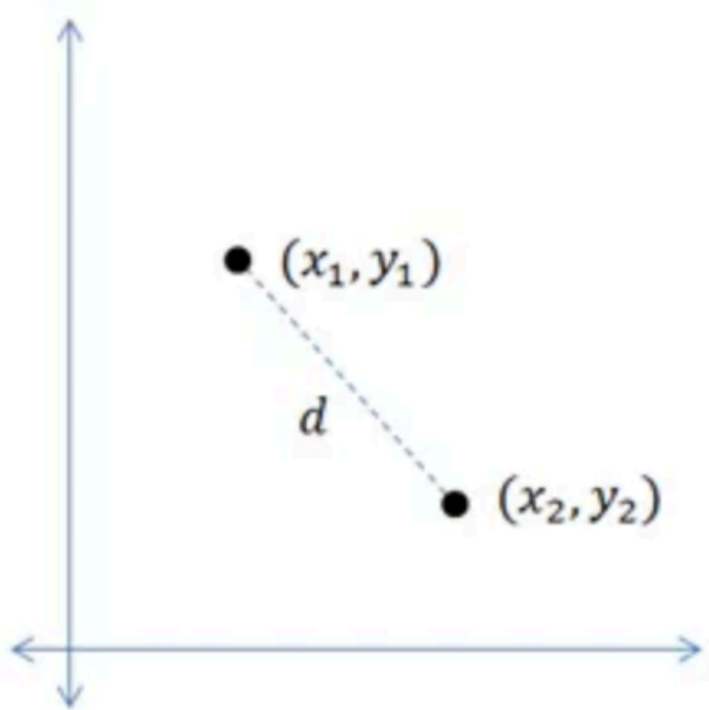


Step 3: Calculate the new centroids of the clusters



Step 4: Define new clusters around the centroids





$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

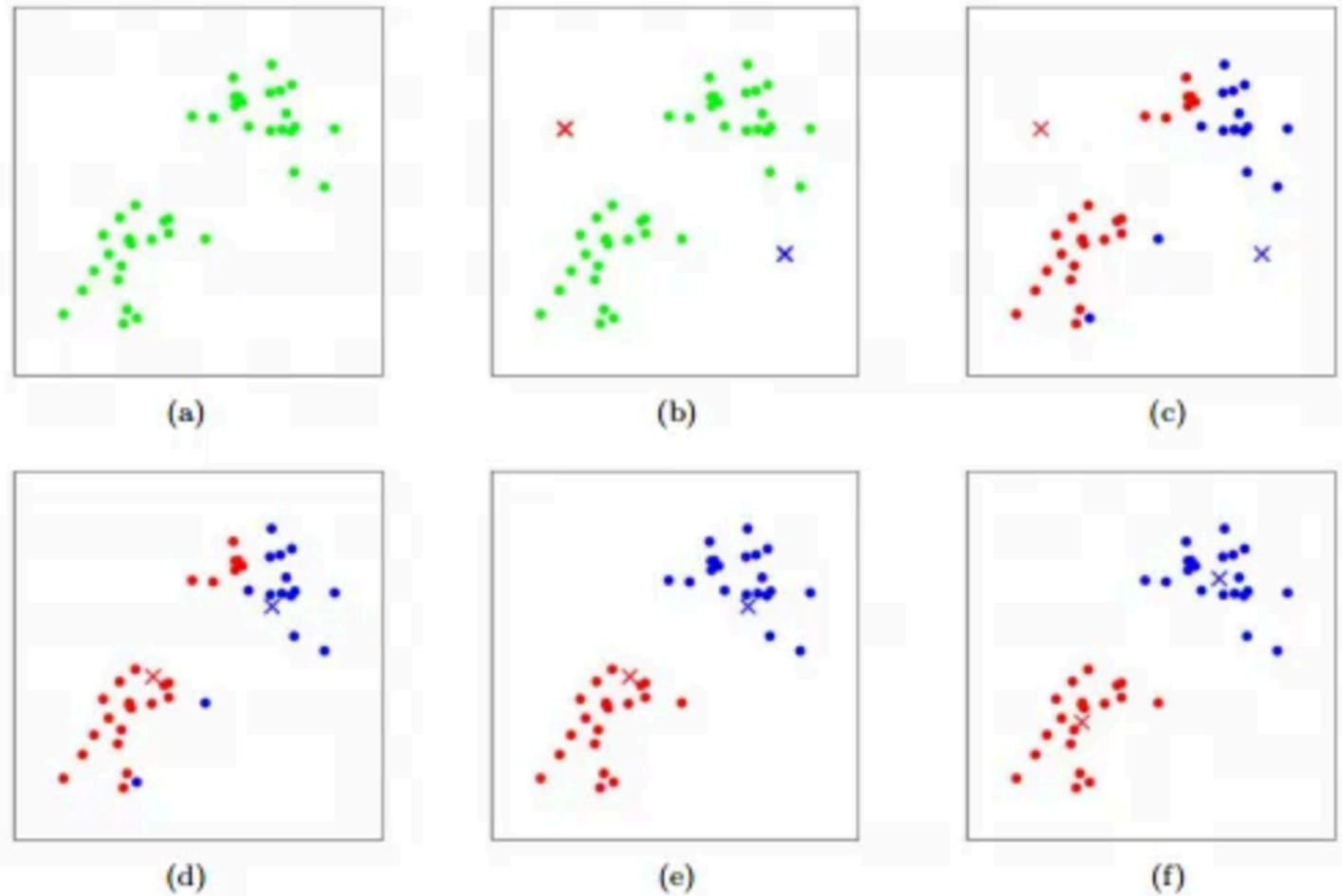


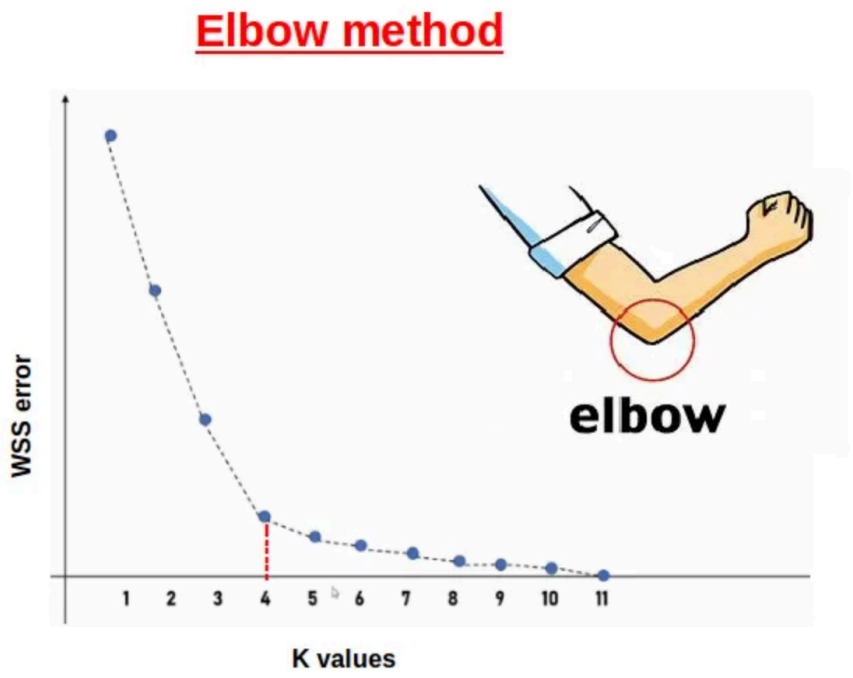
image credit: standford.edu

Elbow Method for Evaluating K-Means

The **Elbow Method** helps determine the optimal number of clusters (**K**) for K-Means clustering.

It works by analyzing how the **Sum of Squared Errors (SSE)**, also called **inertia**,

changes as K increases.



1. Graphical Representation of Elbow Method

Steps:

1. **Run K-Means** for different values of K
2. **Compute SSE** for each K: $$SSE = \sum_{i} (x_i - c_j)^2$$
3. **Plot SSE vs. K**:
    - The SSE decreases as K increases because adding more clusters reduces the variance within each cluster.
    - However, after a certain point, the reduction in SSE slows down.
4. **Find the "elbow" point**:
    - The "elbow" is where the SSE starts decreasing at a slower rate.
    - This indicates that adding more clusters beyond this point does not significantly improve the clustering.

Hierarchical

Hierarchical Clustering:

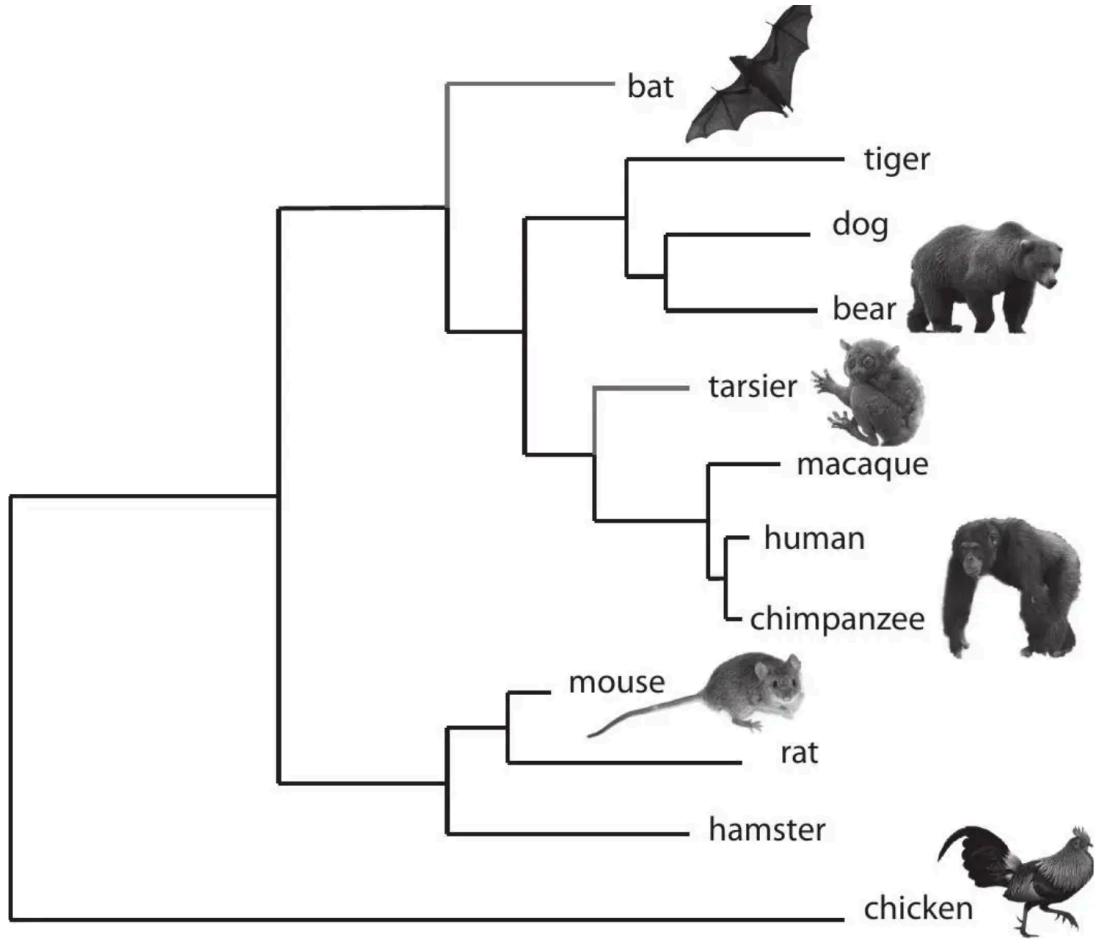
Hierarchical clustering is a method used to group data points into clusters based on their similarity. Unlike K-Means, you **don't need to specify the number of clusters in advance**.

✨ Key Idea:

- It **builds a hierarchy** of clusters by **either merging smaller clusters into larger ones (agglomerative)** or **splitting larger clusters into smaller ones (divisive)**.

- This process is often visualized using a **dendrogram**, a tree-like diagram that shows how clusters are formed.

Example



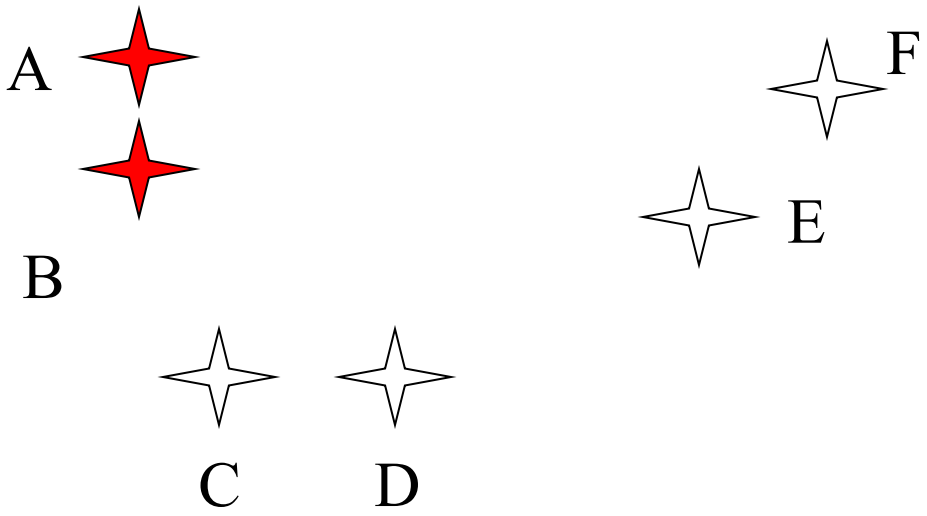
https://machinelearningknowledge.ai/agglomerative-hierarchical-clustering-in-python-sklearn-scipy/

Types of Hierarchical Clustering

1. **Agglomerative (Bottom-Up) [Most Common]**
    - Each data point starts as its **own cluster**.
    - The closest clusters are merged step by step until only **one cluster remains**.
    - Distance is calculated using **linkage methods** (e.g., single-link, complete-link, average-link).

2. **Divisive (Top-Down)**
    - Starts with **all points in one big cluster**.
    - Recursively **splits** the clusters until each data point is its **own cluster**.
    - Less common due to higher computational cost.

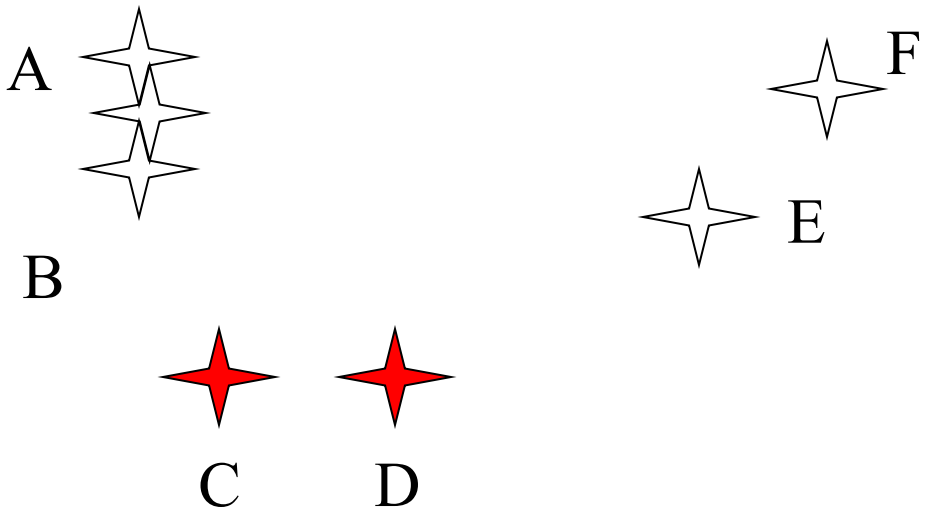
Example



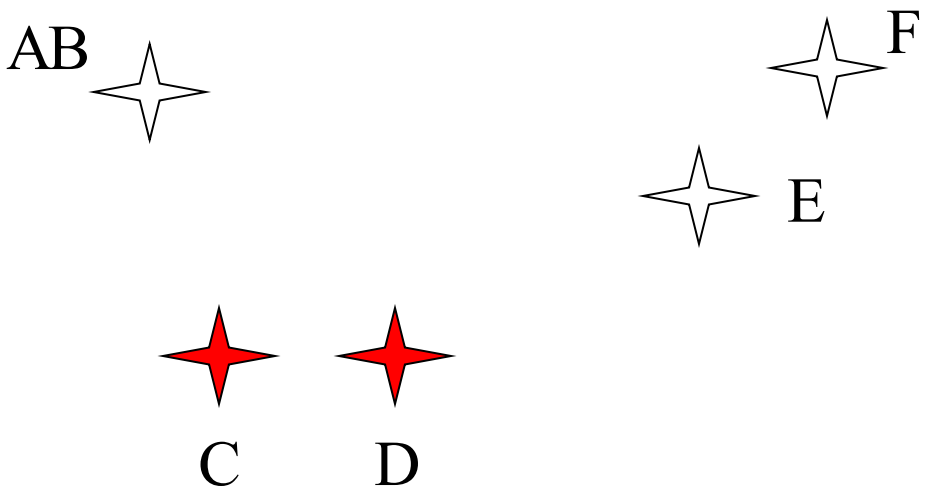
Example

Set up the tree: step 1

Example

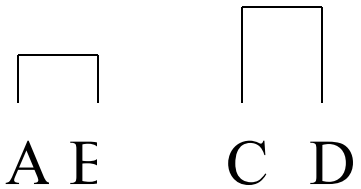


Example

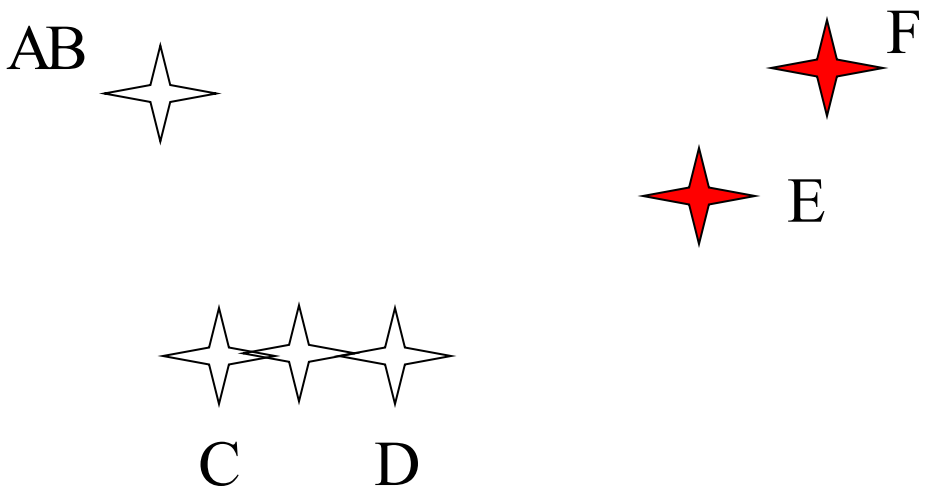


Example

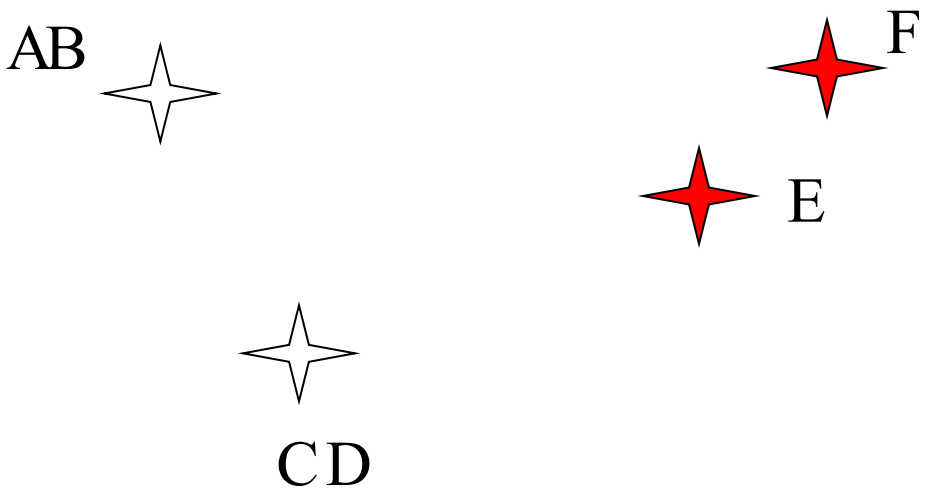
Set up the tree: step 2



Example

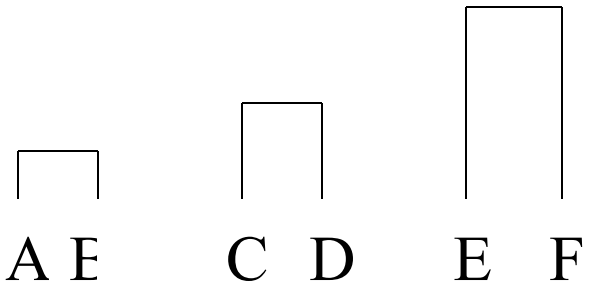


Example

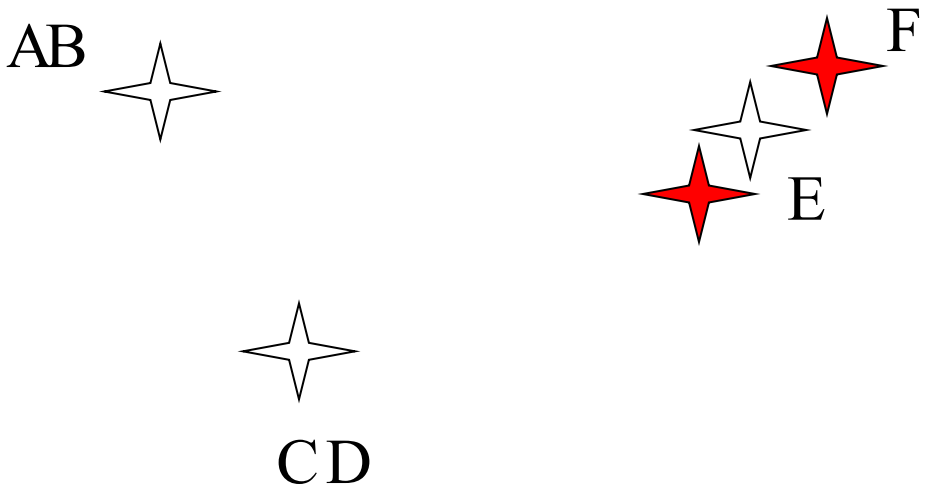


Example

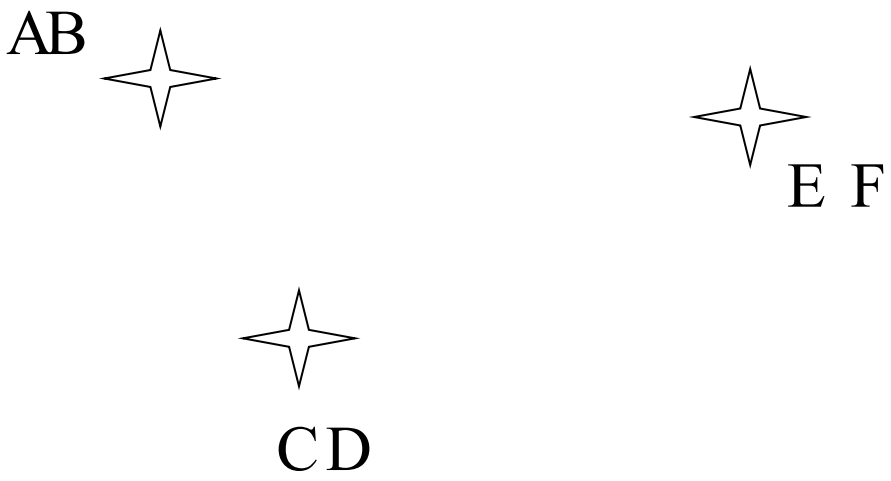
Set up the tree: step 3



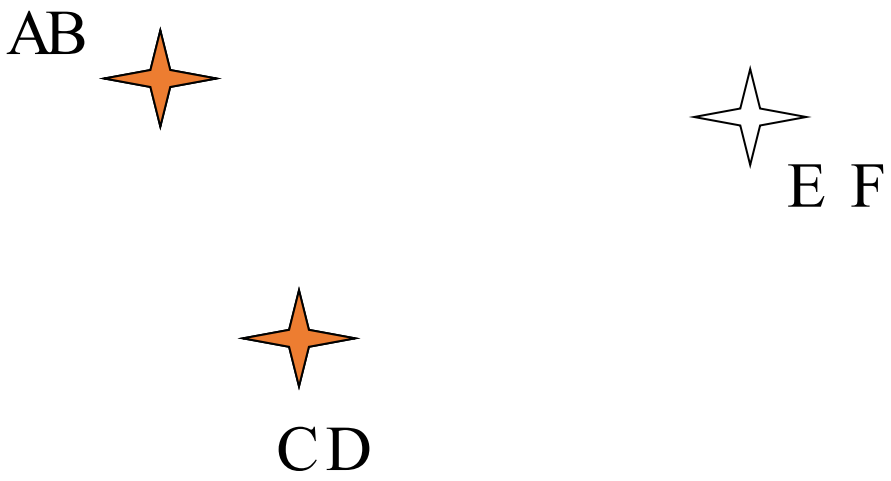
Example



Example

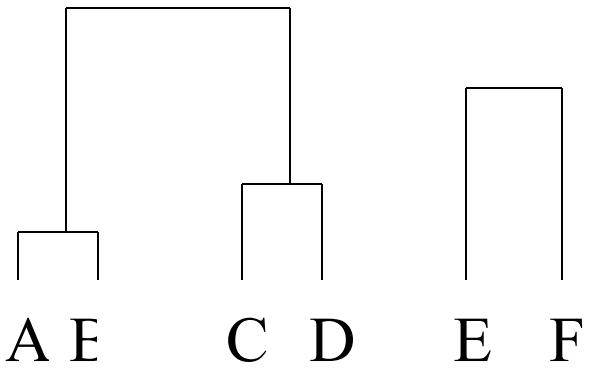


Example

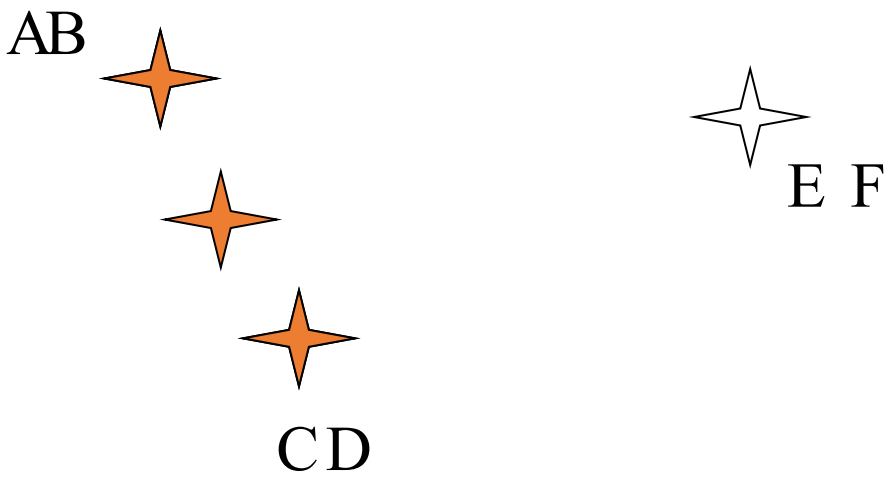


Example

Set up the tree: step 4

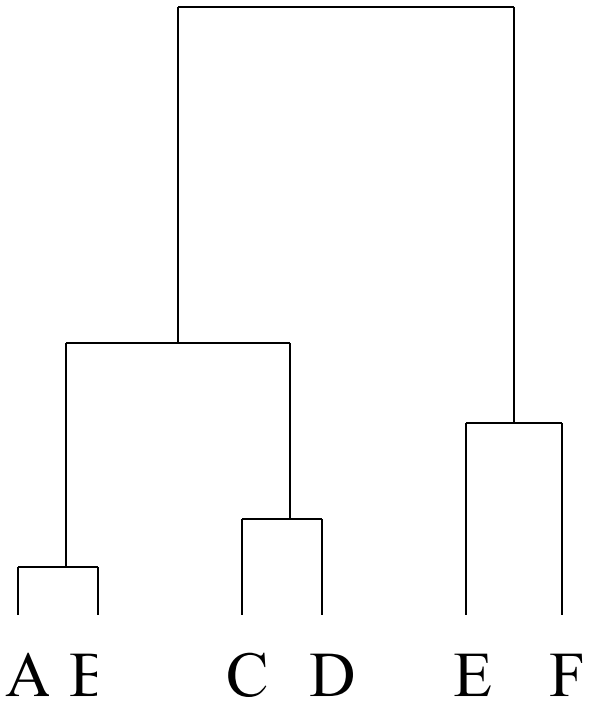


Example

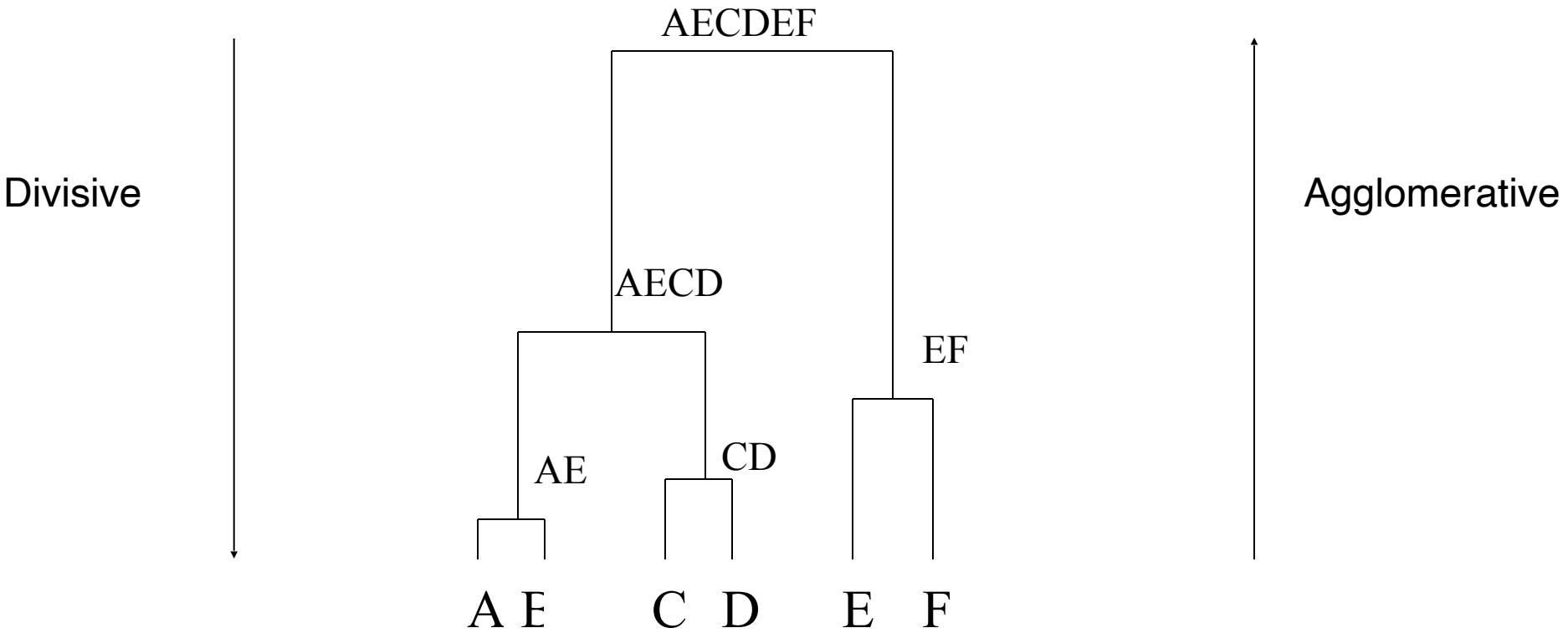


Example

Set up the tree: step 5



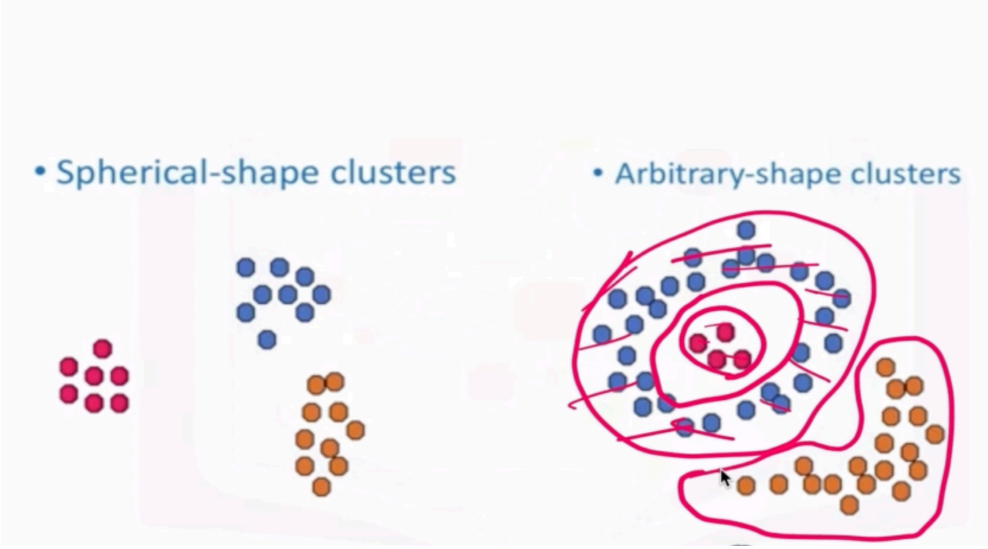
Dendrogram: node ranks



DBSCAN

What is DBSCAN?

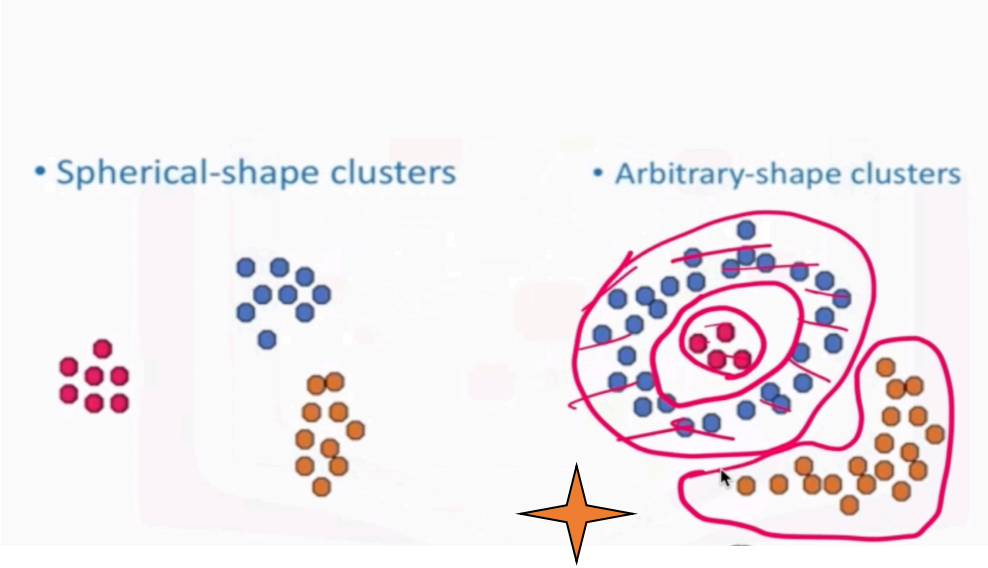
DBSCAN is a **density-based clustering algorithm**. The core idea behind DBSCAN is that clusters are regions of high density separated by regions of low density.



Unlike K-Means, which assumes clusters are spherical and of roughly the same size, DBSCAN can find clusters of **arbitrary shapes** and handle noise points (outliers) very well.

What is DBSCAN?

DBSCAN is a **density-based clustering algorithm**. The core idea behind DBSCAN is that clusters are regions of high density separated by regions of low density.



Unlike K-Means, which assumes clusters are spherical and of roughly the same size, DBSCAN can find clusters of **arbitrary shapes** and handle noise points (outliers) very well.

Key Concepts of DBSCAN:

1. **Core Points**: Points that have at leat M neighboring points within a specified distance (`M`, r).

2. **Border Points**: Points that are close to core points but do not have enough neighbors to be a core point themselves.

3. **Noise Points**: Points that are not close enough to any core points and thus are considered outliers.

How DBSCAN works:

1. **Choose a point**.

2. **Check if it's a core point**: Does it have at least `M neighboring` points within a distance of `R`?

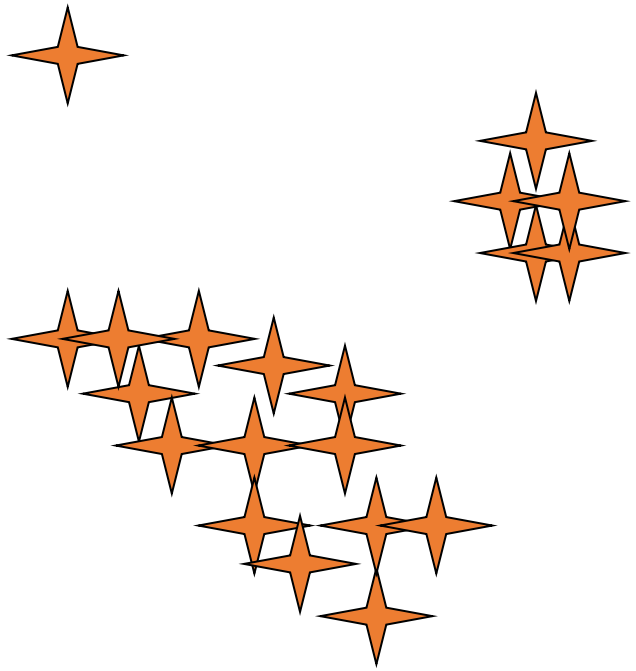
3. If it is a core point, **expand the cluster** by looking for other points within `ε`.

4. If it’s not a core point, mark it as **noise** (unless it’s a border point, in which case it’s assigned to a neighboring core point’s cluster).

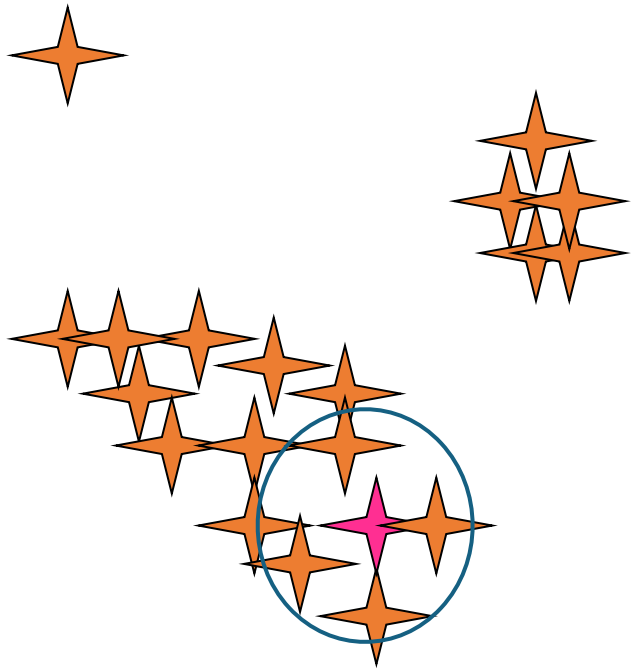
5. Repeat this process for all points in the dataset.

Step 1: choose a distance and minimum number of neighbour

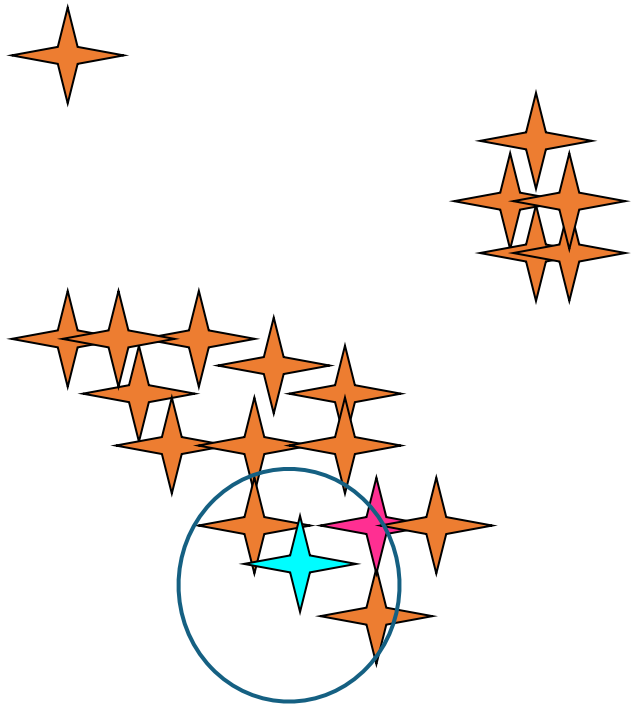
Let’s say 3 neighbour and distance 2



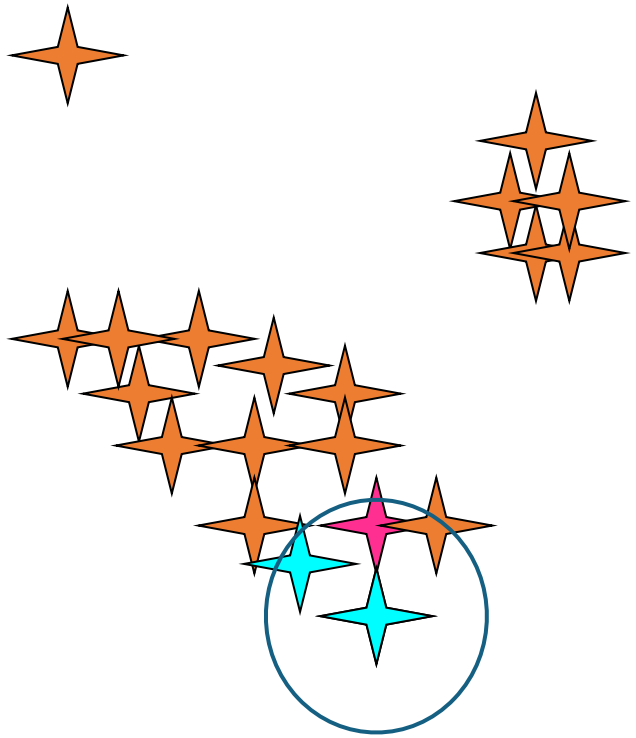
Step 2: Check point by point if they are core or brother or outlier



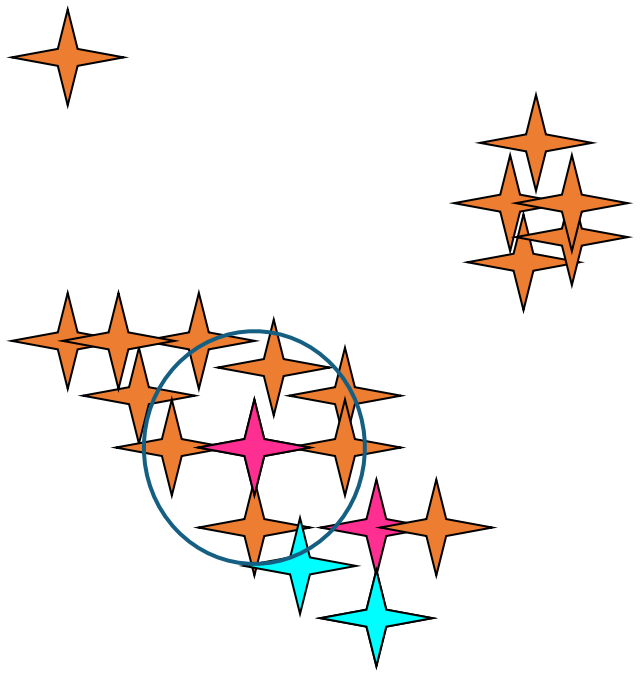
Step 2: Check point by point if they are core or brother or outlier



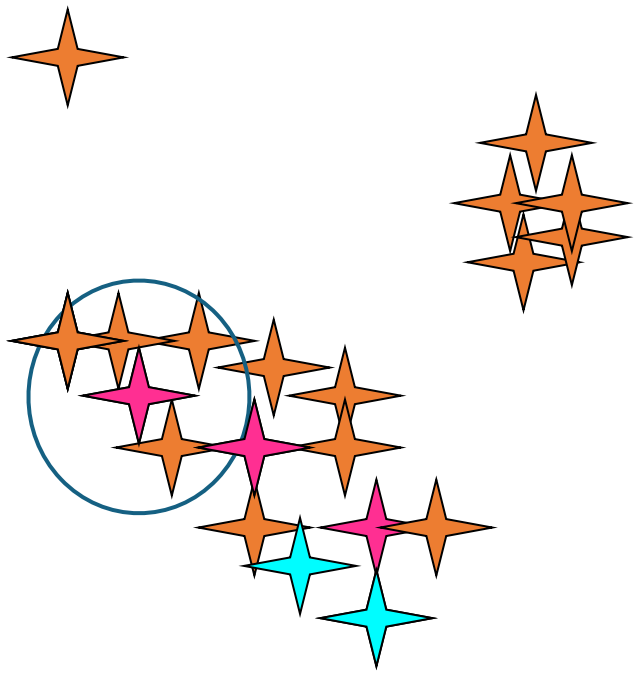
Step 2: Check point by point if they are core or brother or outlier



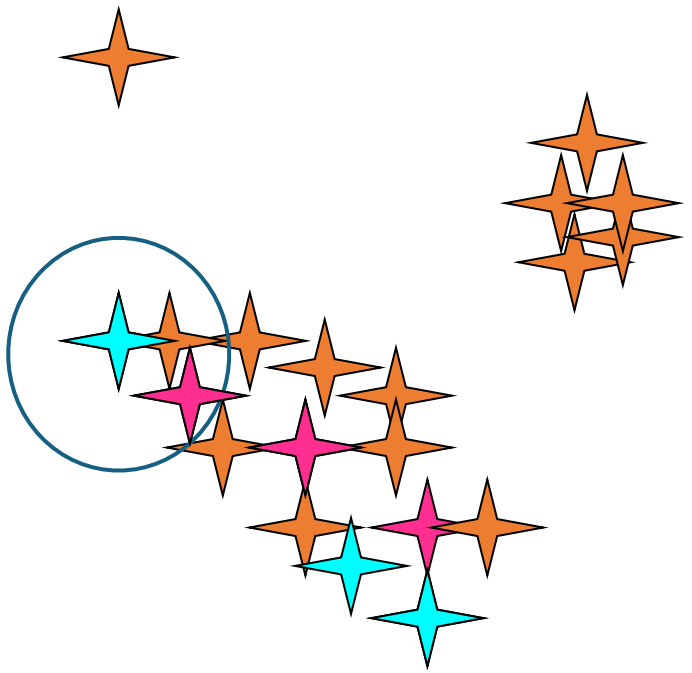
Step 2: Check point by point if they are core or brother or outlier



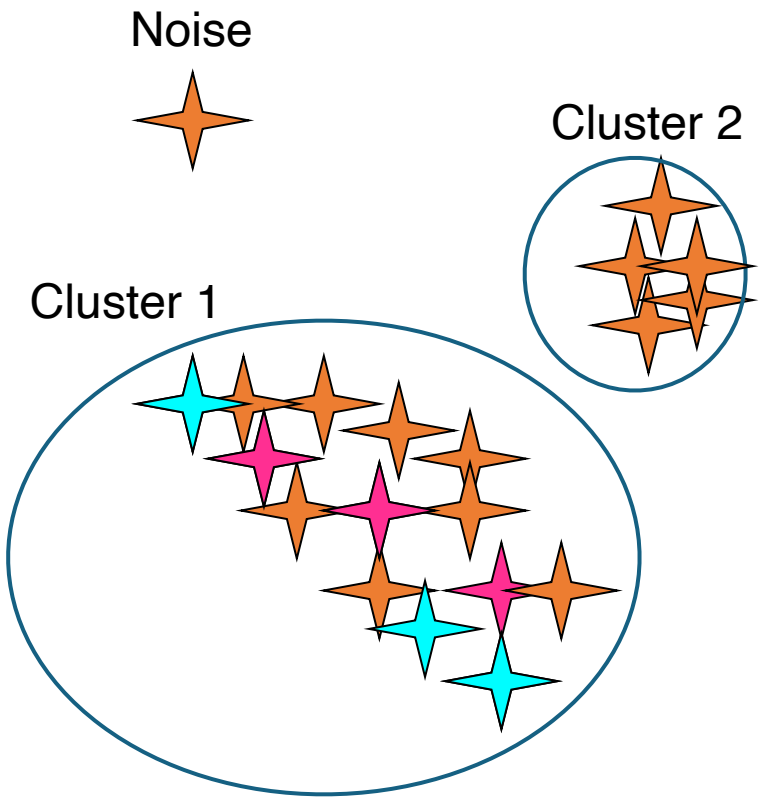
Step 2: Check point by point if they are core or brother or outlier



Step 2: Check point by point if they are core or brother or outlier



You are done!



Aspect	DBSCAN	K-Means
Number of Clusters	No need to specify the number of clusters. DBSCAN finds the number based on density.	Must specify the number of clusters `k` before running.
Cluster Shape	Can find clusters of arbitrary shape (e.g., circular, elongated, or complex).	Assumes clusters are spherical and of roughly equal size.
Outliers	Identifies outliers as noise points.	Does not handle outliers well; they are forced into the nearest cluster.
Density Handling	Works well with clusters of varying densities.	Struggles with clusters of varying densities.
Data Structure	Requires two parameters: `ε` (neighborhood radius) and `min_samples` (minimum points to form a cluster).	Requires only the number of clusters `k`.