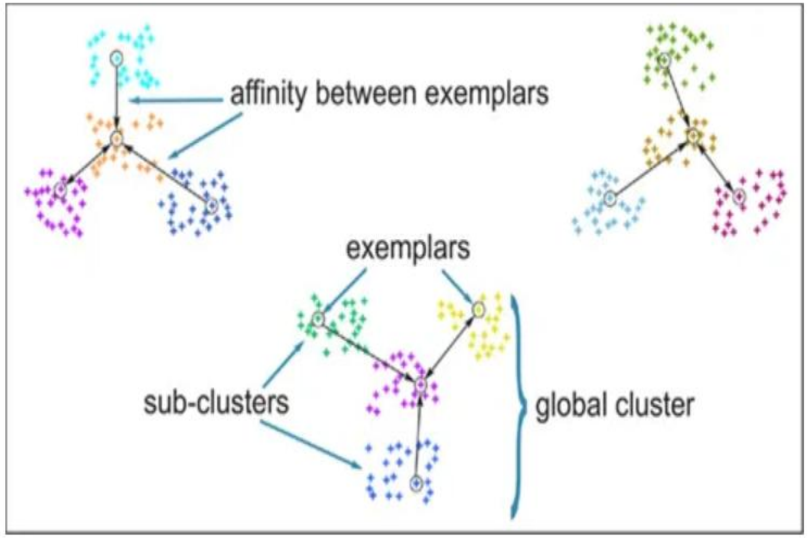
Types Of Clustering

Affinity Propagation

- **Affinity Propagation (AP)** is a **message-passing-based clustering algorithm** that finds **exemplars (cluster centers)** by exchanging messages between pairs of data points. Unlike algorithms like K-means, **AP does not require you to predefine the number of clusters**.
- It works by identifying points that are most "representative" of others (called **exemplars**) and assigning other points to them.




```
from sklearn.cluster import AffinityPropagation
af = AffinityPropagation(random_state=5)
af.fit(X)
```

Key Parameters

1. Similarity Matrix (S):

- Measures how similar two points are.
- Often calculated using negative squared Euclidean distance:

$$S(i,k) = -\|x_i - x_k\|^2$$

- Diagonal elements $S(k,k)$ represent **preferences** — higher values mean point k is more likely to be chosen as an exemplar.

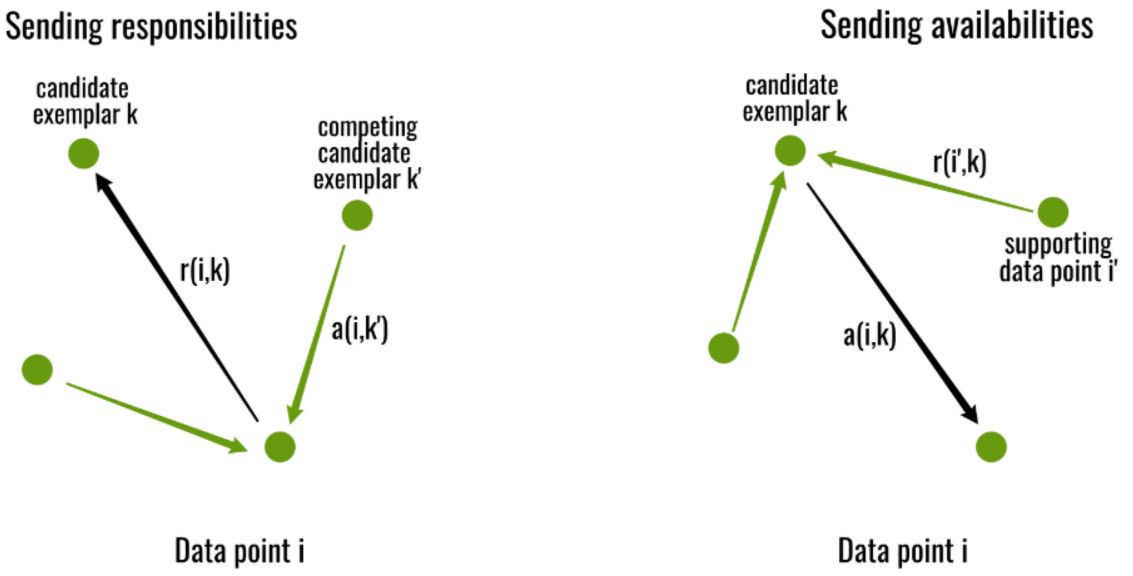
2. Messages Exchanged:

- **Responsibility (r)**: How well-suited point k is to serve as the exemplar for point i:

$$r(i,k) = S(i,k) - \max_{k' \neq k} [a(i,k') + S(i,k')]$$

- **Availability (a)**: How appropriate it would be for point i to choose point k as its exemplar:

$$a(i,k) = \min \left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k)) \right)$$



Advantages and Disadvantages of Affinity Propagation

✅ Advantages:

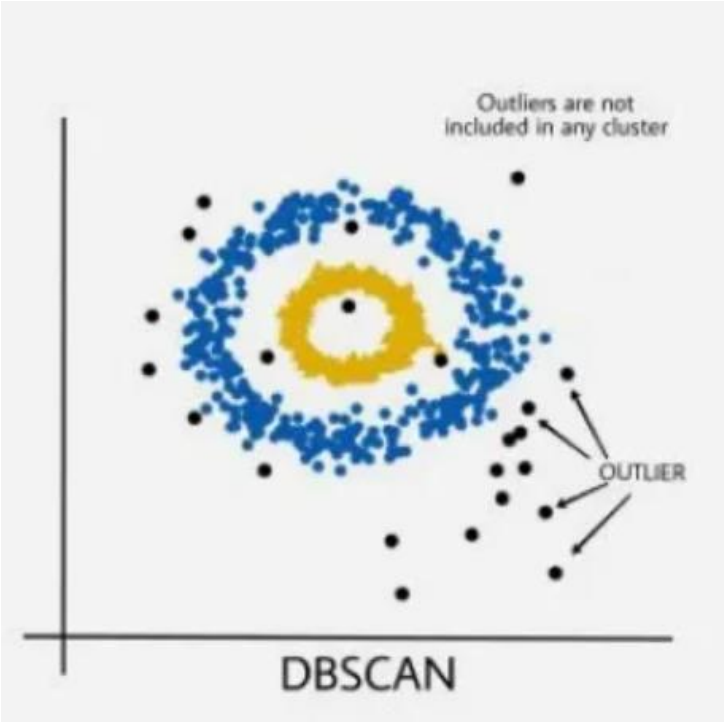
- No need to specify the number of clusters.
- Can find clusters of different sizes and shapes.
- Works well when a good similarity measure is available.
- Automatically identifies exemplars as real data points.

❌ Disadvantages:

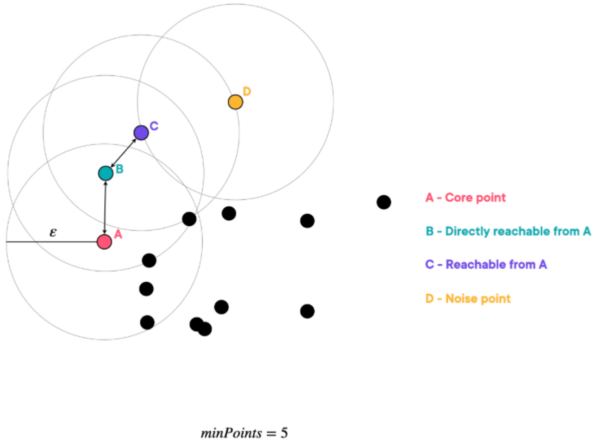
- High memory usage – requires computing and storing a full similarity matrix.
- Computationally expensive – not ideal for very large datasets.
- Performance depends heavily on the choice of similarity function and preference values.
- May converge slowly or not at all if not tuned properly.

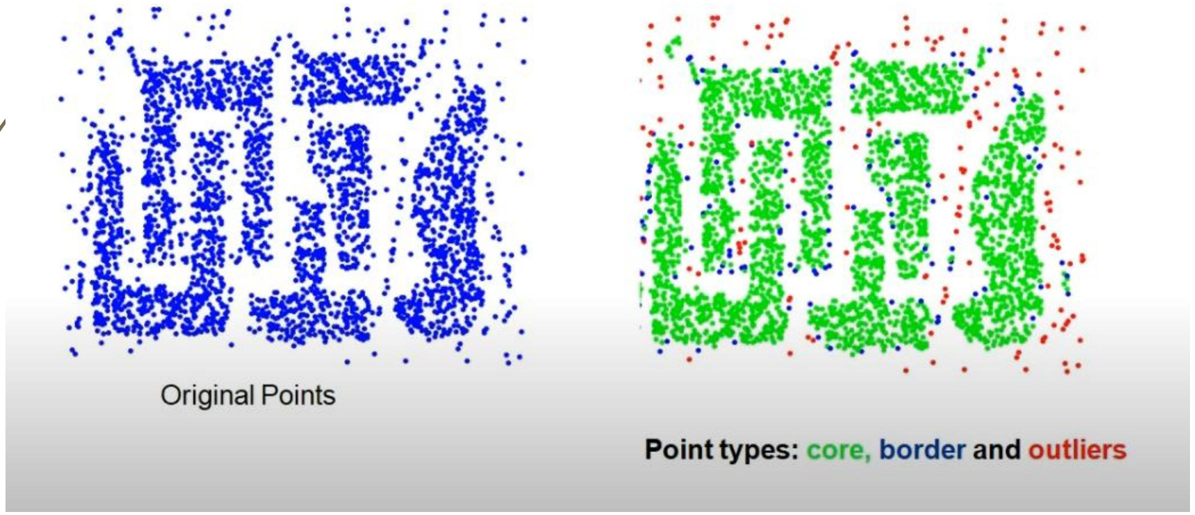
DBScan Clustering

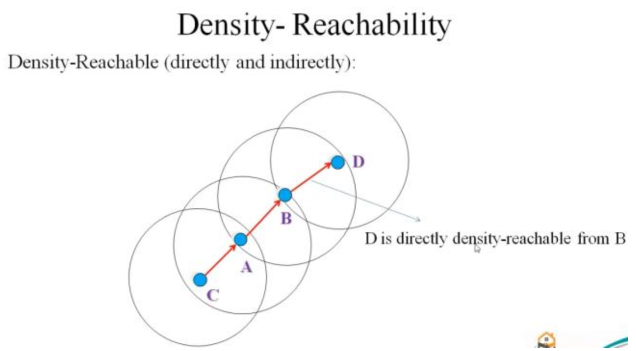
- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) is a **density-based clustering algorithm**. It groups together points that are **closely packed (dense regions)** and marks points that lie alone in low-density regions as **outliers (noise)**.
- It is useful for identifying clusters of arbitrary shape and handling noise in data.

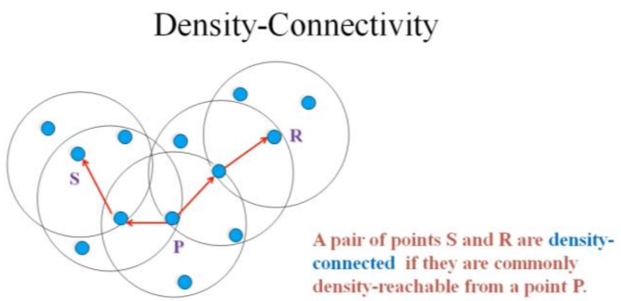



```python
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.2, min_samples=5)
db.fit(X)
```


A - Core point
B - Directly reachable from A
C - Reachable from A
D - Noise point
ε
minPoints = 5


Original Points
Point types: core, border and outliers


Density- Reachability
Density-Reachable (directly and indirectly):
D is directly density-reachable from B


Density-Connectivity
A pair of points S and R are density-connected if they are commonly density-reachable from a point P.

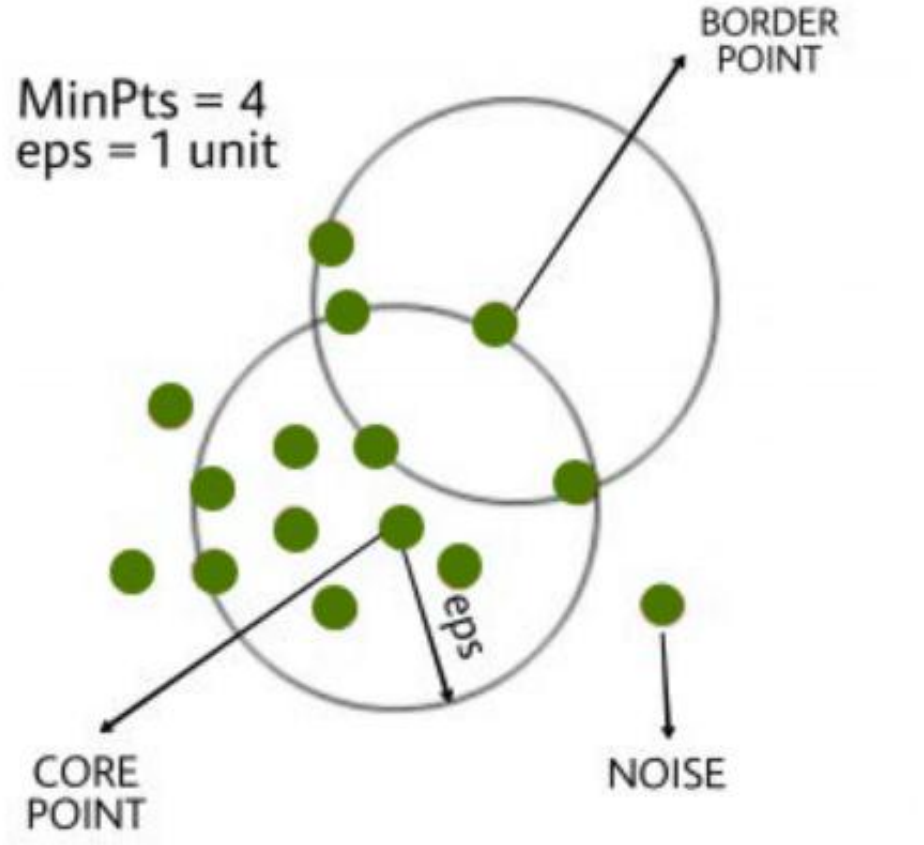
How Does DBSCAN Work?

Key Parameters in DBSCAN

- **ε (epsilon)**: Radius of neighborhood around a point.
- **MinPts**: Minimum number of points required to form a dense region (cluster).

Types of Points:

- **Core Point**: Has ≥ MinPts within ε radius.
- **Border Point**: Has < MinPts within ε but is in the neighborhood of a core point.
- **Noise Point (Outlier)**: Not a core or border point.



Advantages and Disadvantages of DBSCAN

✅ Advantages:

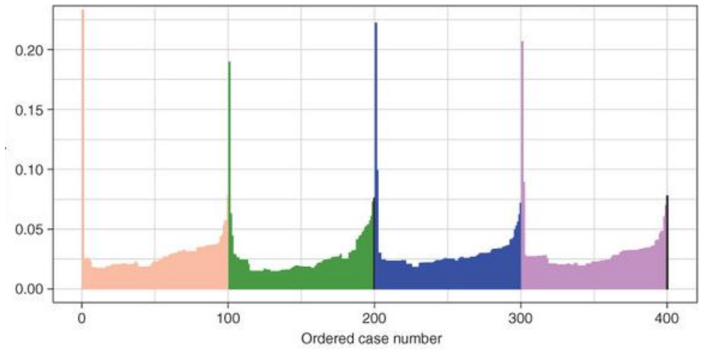
- No need to specify number of clusters in advance.
- Can find arbitrarily shaped clusters.
- Effectively identifies outliers/noise.
- Works well with spatial data and geographic data.

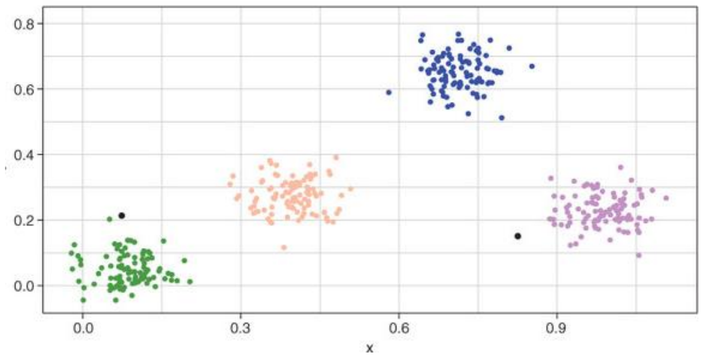
❌ Disadvantages:

- Choice of ε and MinPts is critical — poor values can lead to bad clustering.
- Not well suited for varying density clusters.
- Distance-based, so may struggle with high-dimensional data.
- Performance can degrade on large datasets without optimization.

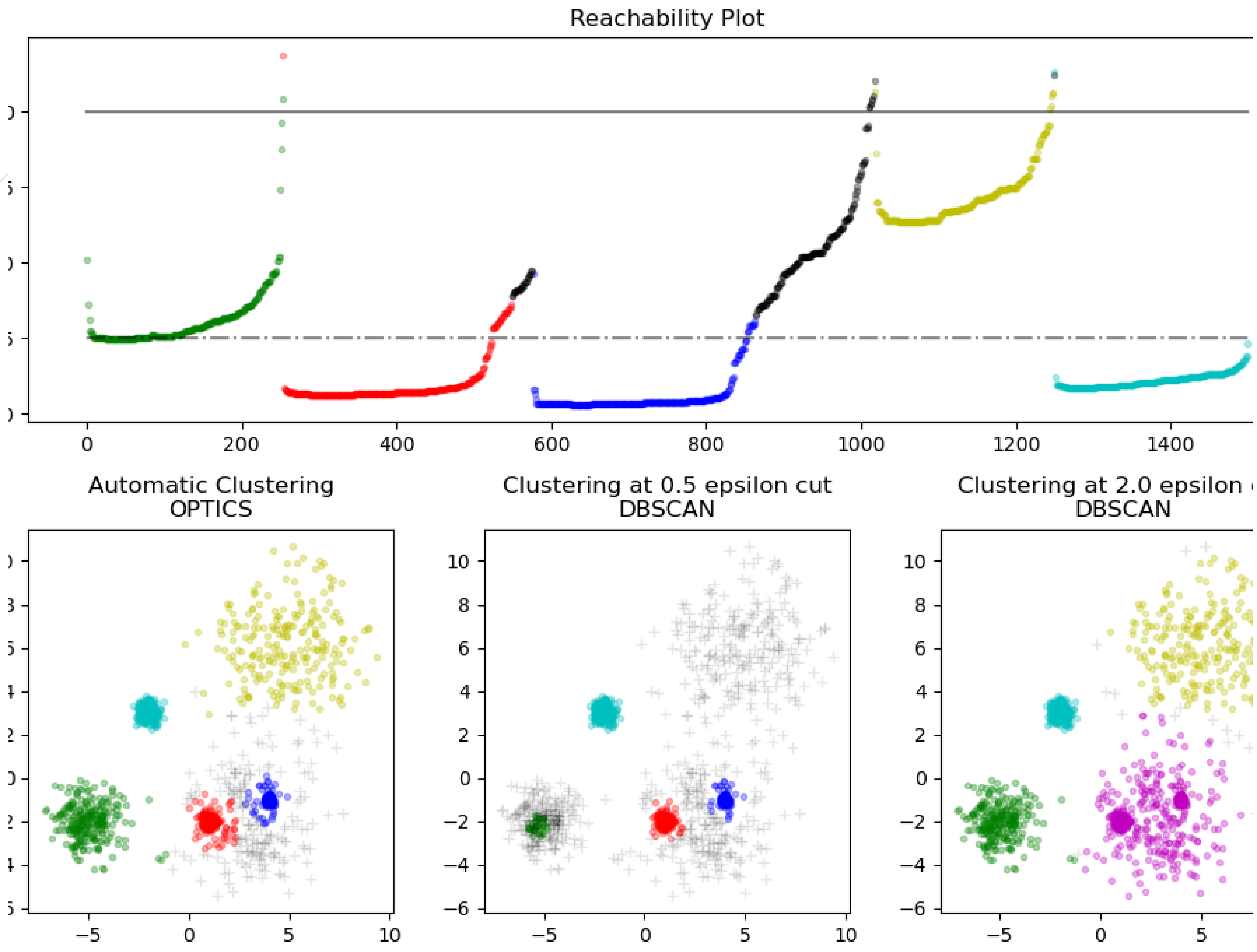
OPTICS Clustering

- **OPTICS** (**Ordering Points To Identify the Clustering Structure**) is a **density-based clustering algorithm**, like DBSCAN, but it overcomes one of DBSCAN's limitations by **identifying clusters of varying density**. Instead of producing an explicit clustering, OPTICS creates an **ordered list of data points** with **reachability distances**, which can be used extract clusters based on a chosen threshold.





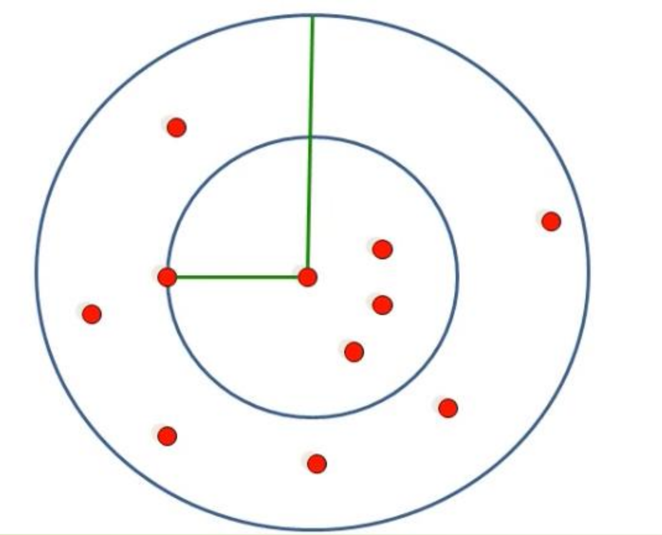

```python
from sklearn.cluster import OPTICS
optics_model = OPTICS(min_samples=10, xi=0.05, min_cluster_size=0.05)
optics_model.fit(X)
```



Reachability Plot
0
200
400
600
800
1000
1200
1400
Automatic Clustering
OPTICS
Clustering at 0.5 epsilon cut
DBSCAN
Clustering at 2.0 epsilon
DBSCAN
−5
0
5
10
−6
−4
−2
2
4
6
8

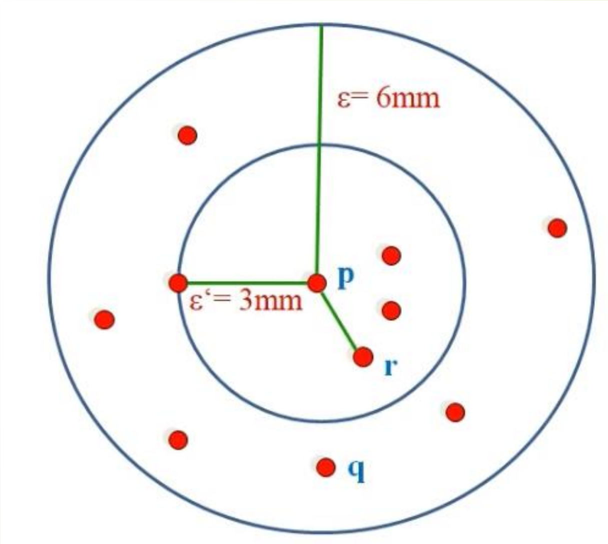
Core Distance

- It is the minimum value of radius required to classify a given point as a core point. If the given point is not a … nt, then it's core … is undefined



Reachability Distance

- It is defined with respect to another data point q. The Reachability distance … and q is … ore … the … p and q



Advantage and Disadvantage of OPTICS

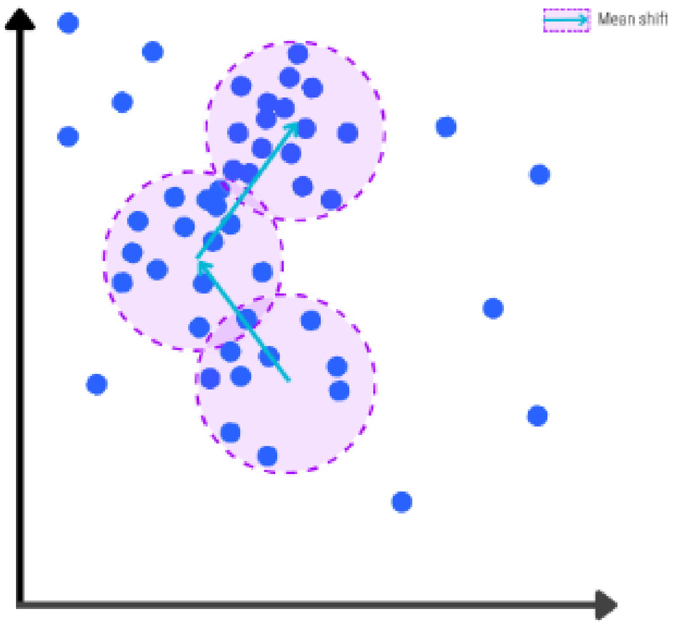
✅ Advantages:

- Finds clusters of varying density, unlike DBSCAN.
- No need to specify the number of clusters.
- Can detect outliers (like DBSCAN).
- Produces a reachability plot useful for analysis and tuning.

❌ Disadvantages:

- Slower than DBSCAN, especially for large datasets.
- More complex to implement and interpret.
- Extracting actual clusters requires additional steps or visual analysis.
- Still requires setting MinPts (and optionally ε).

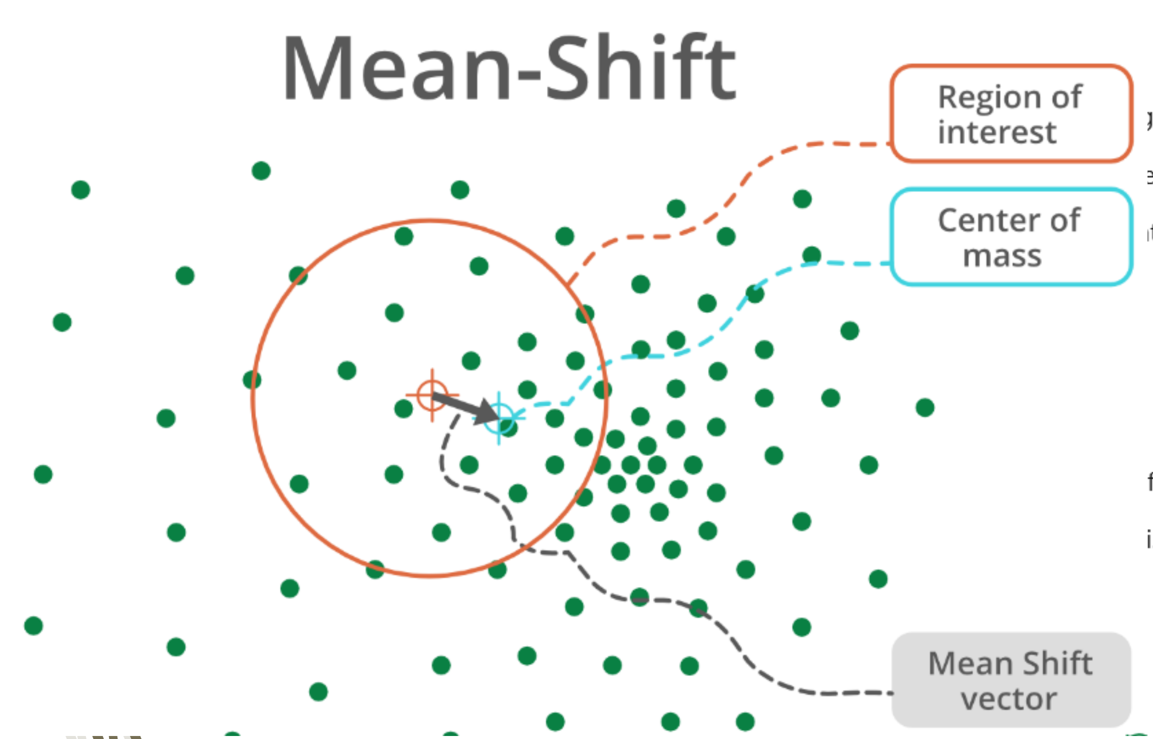
Mean Shift Clustering



- **Mean Shift Clustering** is a **centroid-based**, **non-parametric** clustering algorithm. It works by **shifting data points toward the mode (highest density point)** in the feature space. The goal is to find clusters by identifying **dense regions** (peaks) in the data distribution.
- It does not require specifying the number of clusters in advance.

```python
from sklearn.cluster import MeanShift, estimate_bandwidth
bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=300)
ms = MeanShift(bandwidth=bandwidth)
ms.fit(X)
```

How Does Mean shift work



J., Gaussian)

e)

it point x

$$m(x) = \frac{\sum_{i=1}^{n} x_i \cdot K\left(\frac{x - x_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)} - x$$

fts each point x toward the **mean of points** within the kernel window.

i.e., movement is very small or zero).

Advantages and Disadvantages of MeanShift

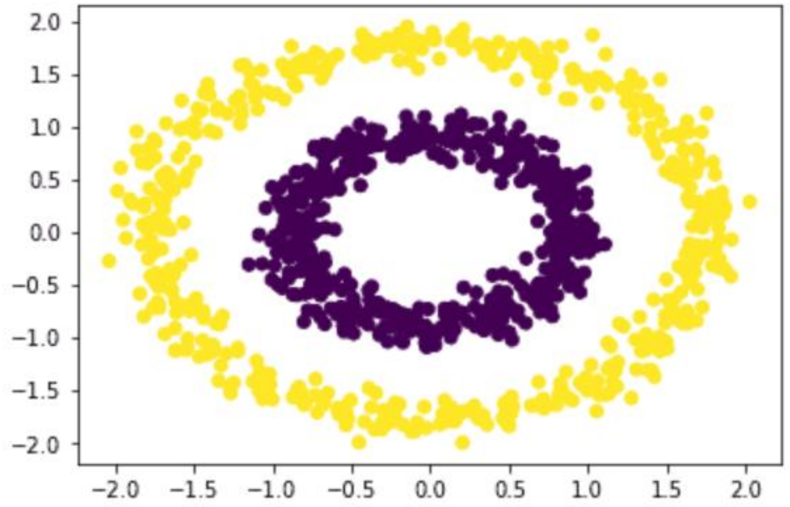
✅ Advantages:

- No need to specify number of clusters beforehand.
- Can find arbitrarily shaped clusters.
- Works well on smooth and continuous data.
- Good at locating cluster centers.

❌ Disadvantages:

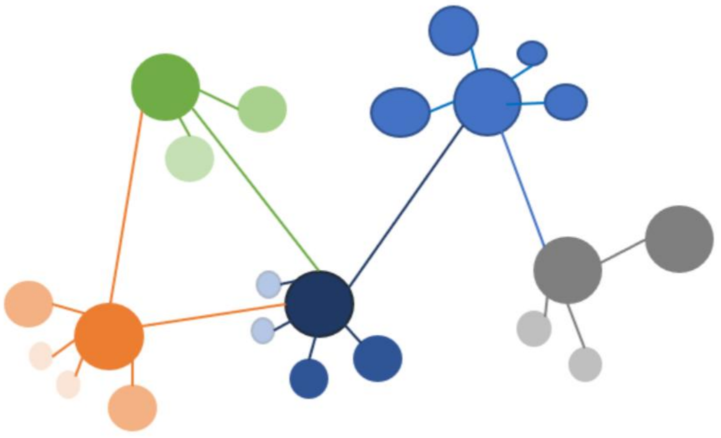
- High computational cost, especially with large datasets.
- Choice of bandwidth (h) is critical — too small → many clusters, too large → fewer clusters or merge.
- Not suitable for high-dimensional data due to curse of dimensionality.
- Sensitive to scale of features.

Spectral Clustering

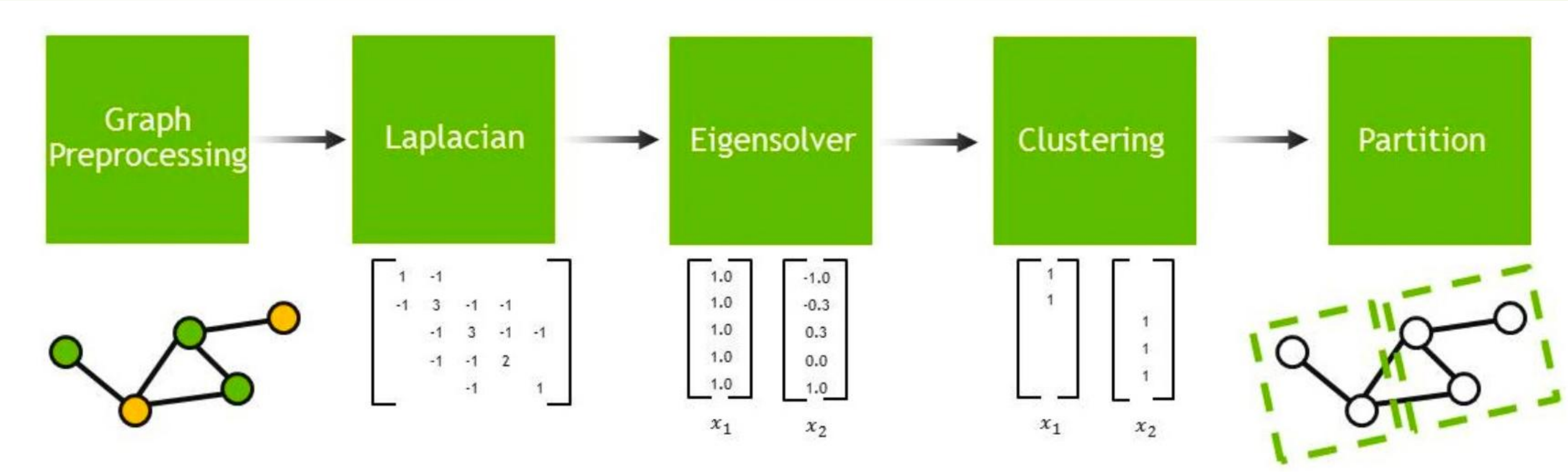


- **Spectral Clustering** is a graph-based clustering algorithm.
  It uses the **eigenvalues (spectrum) of a similarity matrix** to reduce dimensionality before applying a traditional clustering method (like K-means).
- It works well for **non-convex and complex-shaped clusters**, which traditional clustering

```
from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, affinity='rbf', assign_labels='kmeans', random_state=42)
labels = sc.fit_predict(X)
```



How Does Spectral Clustering work



Advantages and Disadvantages of Spectral Clustering

✅ Advantages

- Can find non-convex and complex clusters.
- Works well for image segmentation, social networks, etc.
- Uses graph theory, so clustering is based on connectivity.

❌ Disadvantages

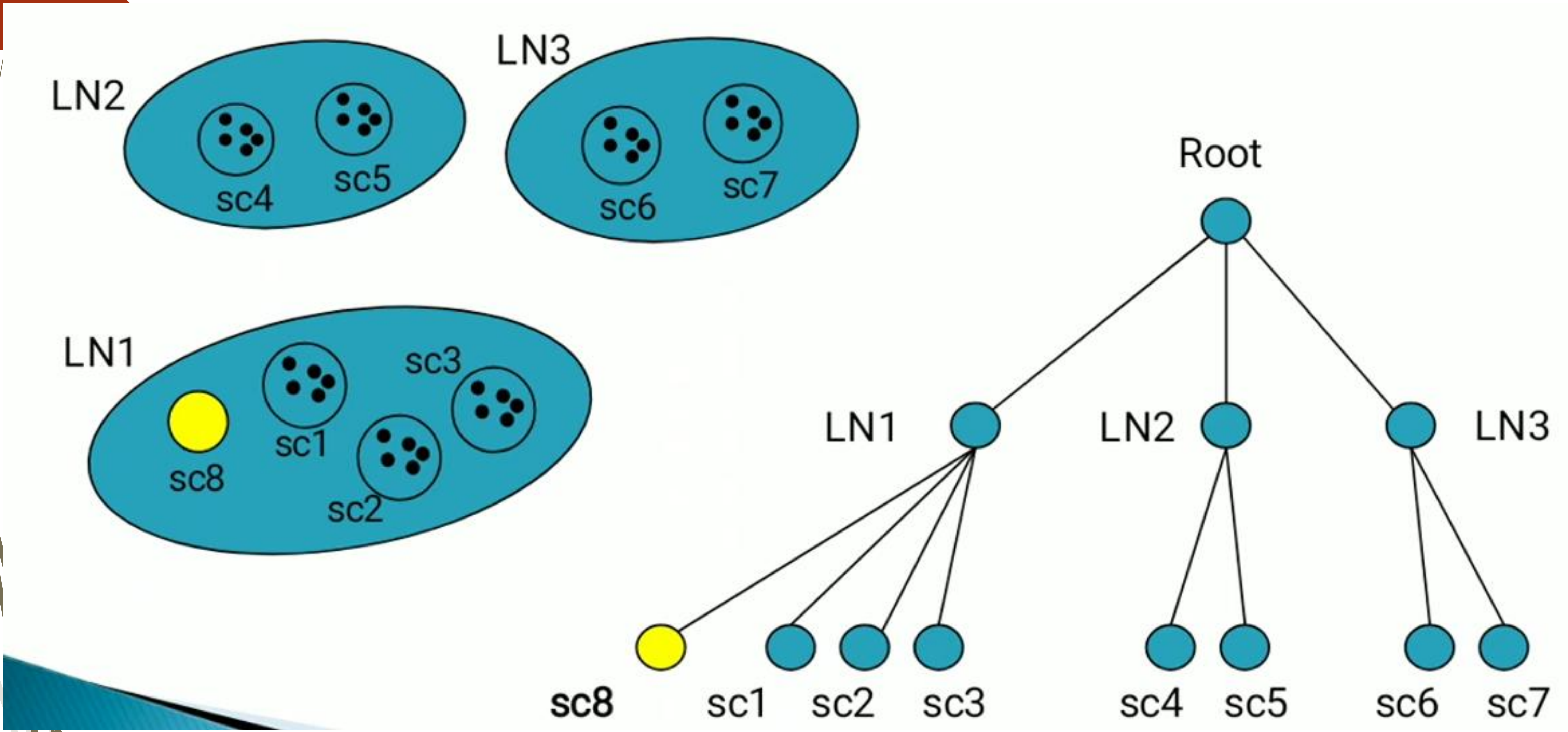
- Needs to set number of clusters (k) manually.
- Computationally expensive for large datasets (eigenvalue decomposition).
- Performance depends on similarity matrix and scaling parameters (e.g., σ).

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

The BIRCH clustering algorithm consists of two stages:

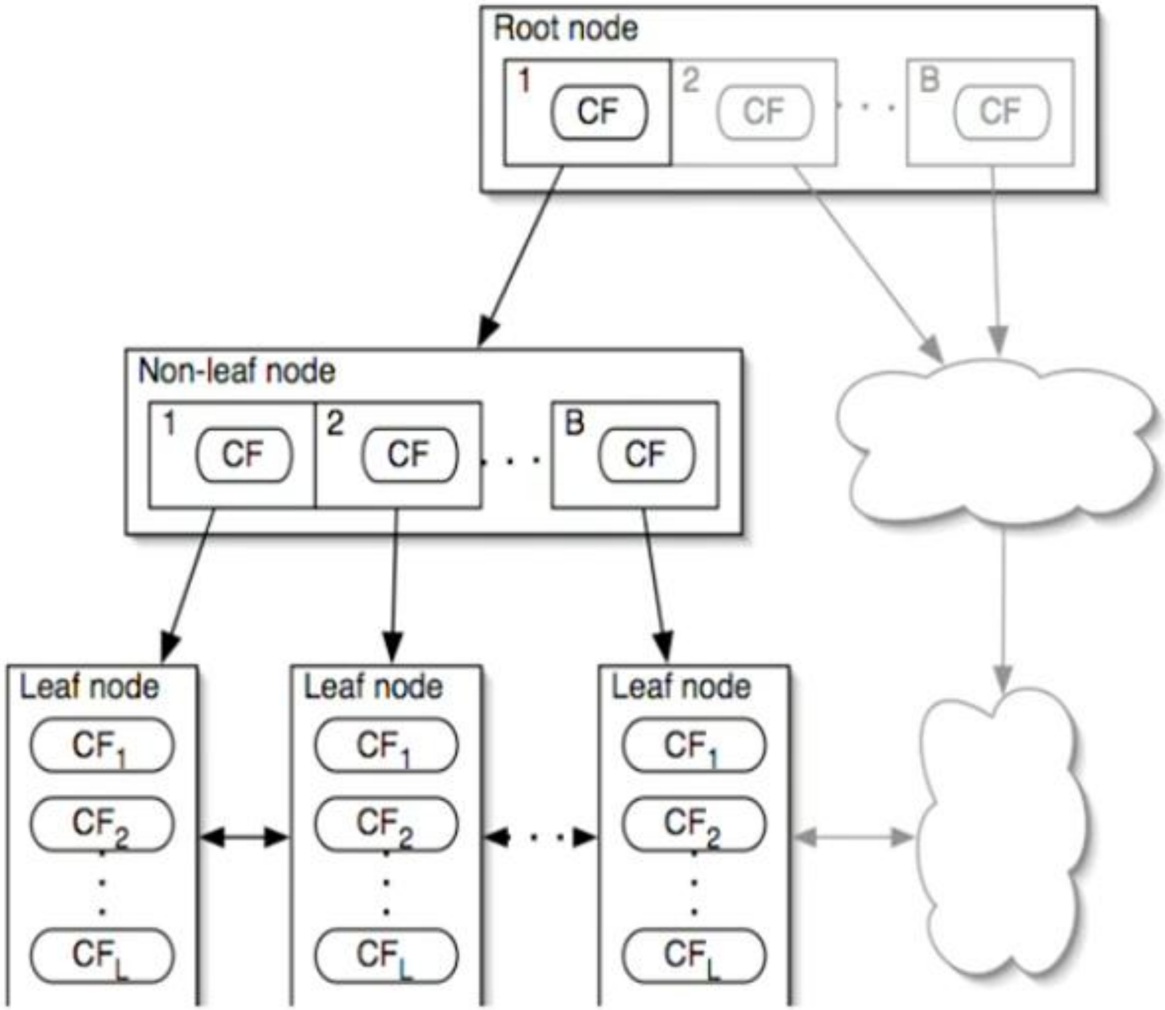
- **BIRCH** is a clustering algorithm designed for **large datasets**.
  It incrementally builds a **tree structure (CF Tree)** to summarize the data and performs clustering using those summaries.
- It is particularly effective for **large-scale**, **numeric data**.
- Can handle **streaming data** by updating the tree incrementally.

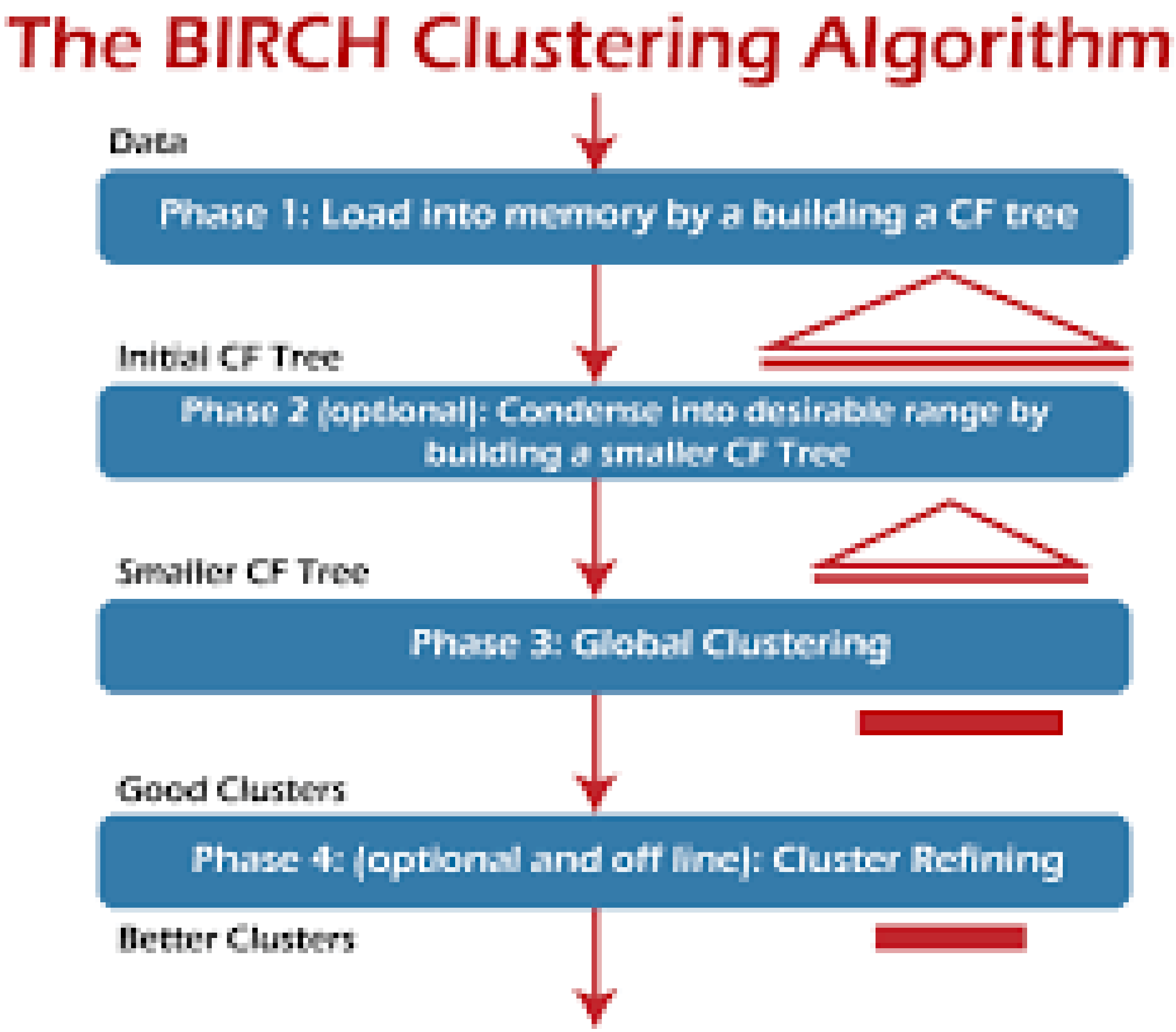
```
from sklearn.cluster import Birch
model = Birch(threshold=1.5, n_clusters=4)
labels = model.fit_predict(X)
```


LN2
sc4
sc5
LN3
sc6
sc7
Root
LN1
sc8
sc1
sc3
sc2
LN1
LN2
LN3
sc8
sc1
sc2
sc3
sc4
sc5
sc6
sc7

✅ CF Tree

- :
- A height-balanced tree where:
- Each **non-leaf node** contains a number of child nodes.
- Each **leaf node** holds clustering features and represents sub-clusters.
- Controlled by **threshold** (max radius of a sub-cluster) and **branching factor** (max children per node).




The BIRCH Clustering Algorithm
Data
Phase 1: Load into memory by a building a CF tree
Initial CF Tree
Phase 2 (optional): Condense into desirable range by building a smaller CF Tree
Smaller CF Tree
Phase 3: Global Clustering
Good Clusters
Phase 4: (optional and off line): Cluster Refining
Better Clusters

Advantages and Disadvantages of BIRCH

✅ Advantages

- Handles very large datasets efficiently.
- Suitable for incremental or streaming data.
- Can serve as a pre-clustering step to reduce data size.
- Supports multi-phase clustering (can apply KMeans on leaf nodes).

❌ Disadvantages

- Works best on numeric, metric data.
- Assumes spherical clusters (like K-Means).
- May struggle with high-dimensional or non-globular clusters.
- Performance depends on threshold and branching factor.