Data Mining and Text Analytics

- IULM UNIVERSITY – POSTGRADUATE PROGRAMME IN "ARTIFICIAL INTELLIGENCE FOR BUSINESS AND SOCIETY"
- LECTURE_02
- TOPIC: CLUSTERING
- PROFESSOR ALESSANDRO BRUNO

Unearthing Structure in the Unknown

- The word "**Unknown**" refers to the **lack** of **labelled data**.
- Because of the missing labels, we must dive into the "Unknown" **without supervision**.
- That takes us to a journey to **Unsupervised Learning**. What is it about?
- **Unsupervised learning** is a branch of machine learning where algorithms try to find patterns in data without labelled examples or direct guidance.
- **Other than supervised learning**, which works with labelled data (**input-output pairs**), unsupervised learning works with unlabelled data, discovering the **inherent structure** on its own.

Unearthing Structure in the Unknown

What are the key characteristics of unsupervised learning?

- No labels or target outputs are provided during training
- The algorithm must identify patterns, relationships, and structures independently
- There's often no single "correct" answer – different algorithms may find different valid structures
- Evaluation can be challenging since there's no ground truth to compare against

Supervised Learning and Ground Truth

In data science, **Ground Truth** refers to the **true, verified values or labels** used as a benchmark to train and evaluate models.

It represents the actual outcome or classification that a model should ideally predict.

Ground truth is crucial in **supervised learning**, where models learn from labelled data.

Here is an example: images labelled as "cat" or "dog," or customer reviews tagged as "positive" or "negative."

During evaluation, **model predictions** are **compared against** the **ground truth** to calculate performance **metrics** like **accuracy**, **precision**, and **recall**. Essentially, the ground truth acts as the gold standard for validating how well a model generalizes from the data.

Reviewing the importance of Unsupervised Learning

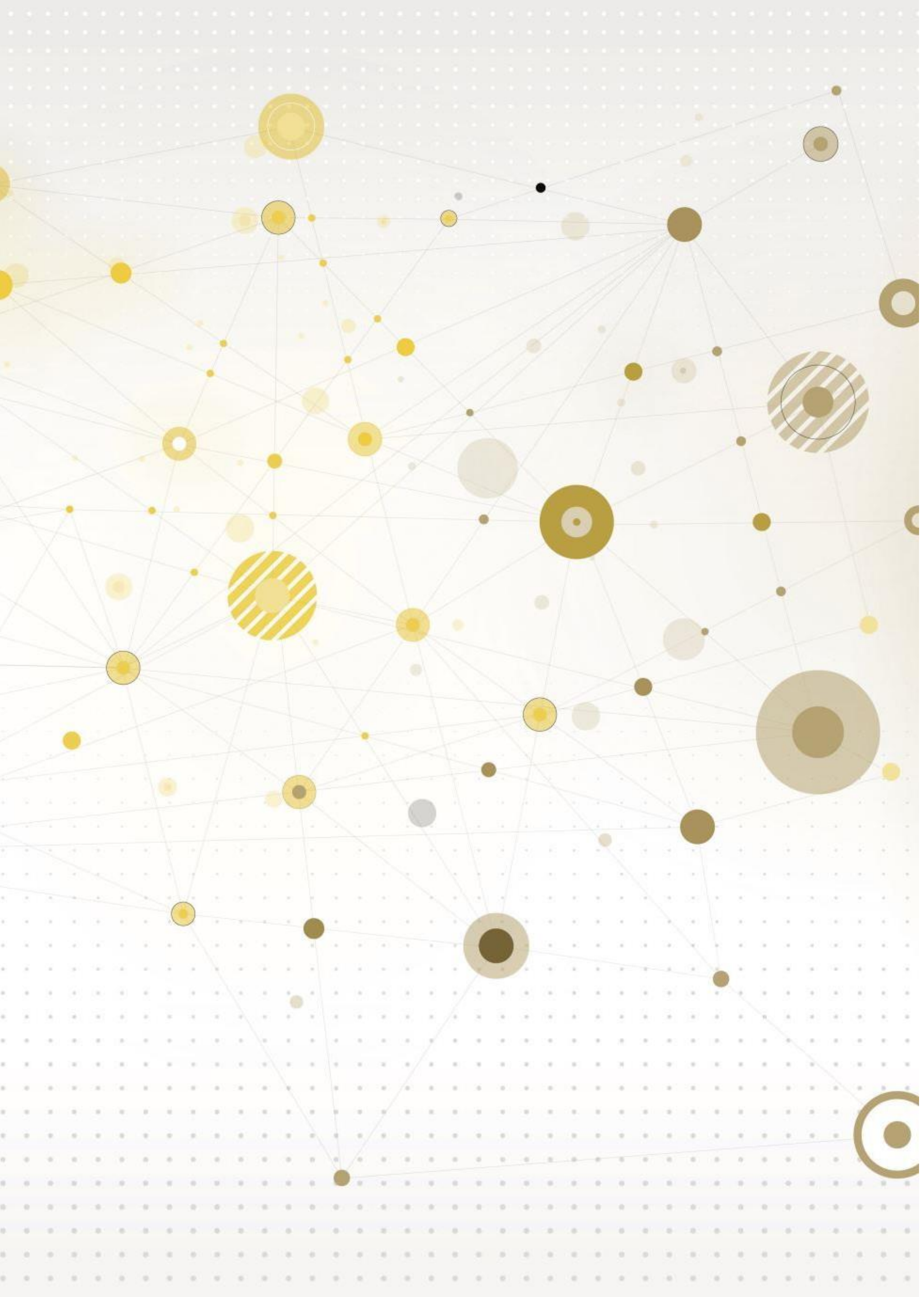
- Unsupervised learning is particularly valuable when:
- You have large amounts of unlabelled data
- You want to discover hidden patterns without preconceived notions
- Labelling data would be prohibitively expensive or time-consuming
- You need to understand the natural structure of your data before applying other techniques
- Some examples follow to let you have a better grasp of the above-mentioned concepts.

Discover hidden patterns without preconceived notions

An e-commerce site analyzes customer purchase data.

Instead of testing predefined marketing hypotheses, unsupervised learning might unexpectedly reveal that customers who buy gardening tools also frequently purchase bird feeders.

That's a connection that marketers did not anticipate, which opens up new cross-selling opportunities.



You have large amounts of unlabelled data

- A social media platform tracks millions of daily user interactions.
- Rather than **manually** categorizing users, **unsupervised learning** can be used to automatically identify different user segments based on behaviour patterns.
- For instance, clustering might reveal distinct groups like "**content creators**," "**passive scrollers**," and "**social connectors**" without anyone having to define these categories in advance.

Labelling data would be prohibitively expensive or time-consuming

- There is a medical research project featuring 10,000 brain scans.
- Having specialists manually label each scan could take months and cost hundreds of thousands of dollars.
- Unsupervised learning can **group similar scans together**, allowing experts to examine just a few representative examples from each cluster rather than every individual scan.
- Using unsupervised learning techniques might help save a lot of time and working hours.

Understanding the natural structure of data before applying other techniques

- A company analyzes customer service calls.
- Before building a supervised classification system, they use unsupervised learning to discover the natural categories of calls.
- This reveals there are **seven** distinct types of customer issues rather than the three categories they were using in their manual system.
- This insight **helps them design a better supervised model** with more appropriate target labels.

Let's get back on track with Clustering!

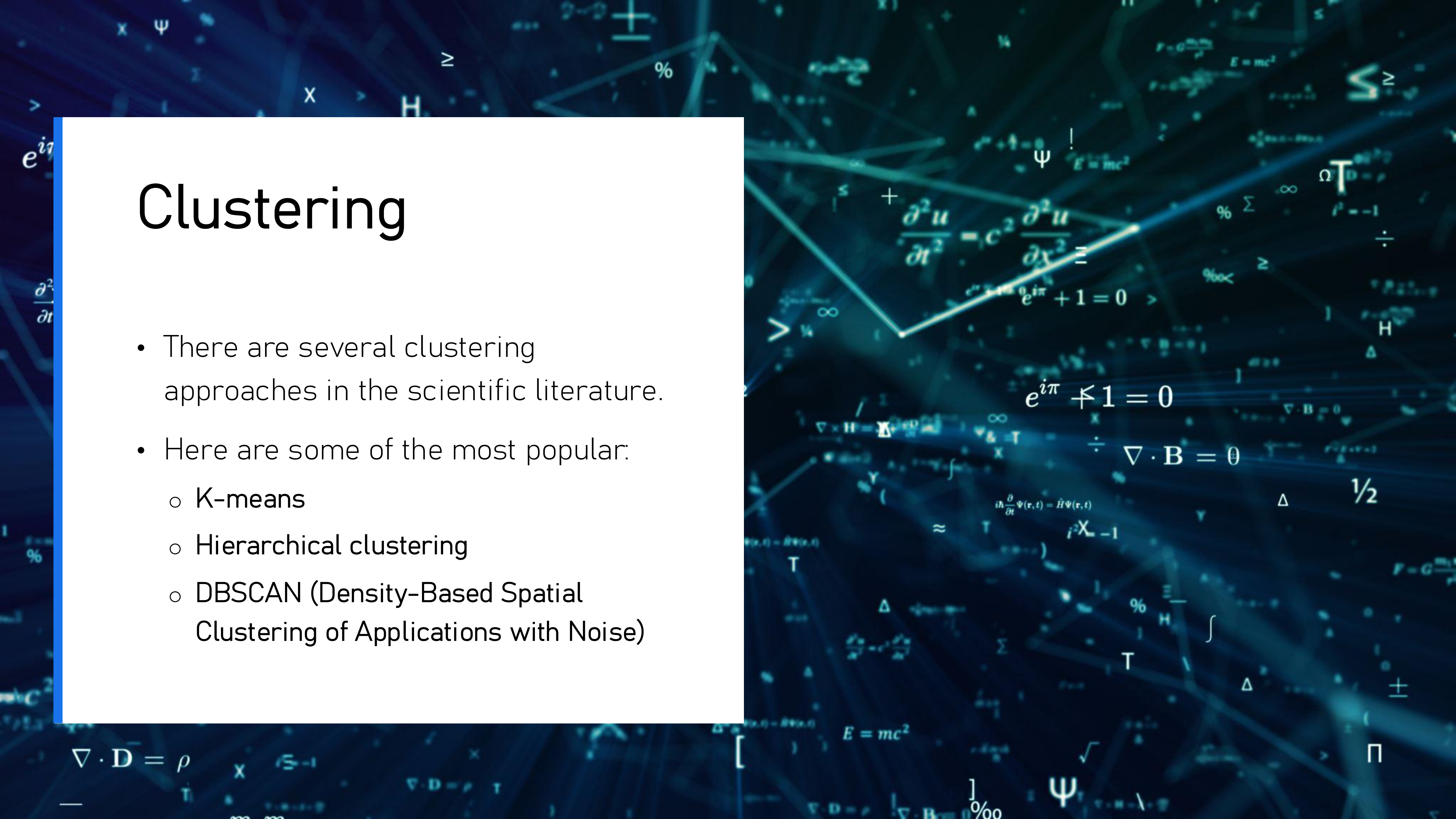
- What is clustering?
- Let's have a defintion for Clustering:
- Clustering is a technique in machine learning and data analysis that groups similar data points together based on their characteristics or features. It's an unsupervised learning method, meaning it doesn't require labelled data to find patterns.

Clustering

- The main goal of clustering is to organize data into meaningful groups where:
  - Items within the same cluster are similar to each other
  - Items in different clusters are dissimilar from each other

Clustering

- The simplest and most fundamental version of cluster analysis is **partitioning**.
- In partitioning clustering, the objects of a given dataset are organised into several exclusive groups or clusters.
- The **number of clusters** in partitioning clustering is given in advance.
- The number of clusters is the starting point for partitioning clustering methods.
- Formally, given a dataset, D, of n objects, and being k the number of clusters to form, a partitioning algorithm organises the objects into k partitions (k lower or equal to n).
- One of the most representative partitioning clustering methods is **K-means**



Clustering

- There are several clustering approaches in the scientific literature.
- Here are some of the most popular:
  - K-means
  - Hierarchical clustering
  - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

K-means

- K-means is a popular and relatively simple algorithm used in various fields like customer segmentation, image analysis, and document clustering.

    - **"K" is the number of clusters you want.** You must decide this beforehand.
    - **It's an iterative process.** It keeps refining the clusters until they stabilize.
    - **It aims to group similar data points together.**

K-means (pseudocode) part 1

1. **You pick a number (that's the "K" in K-means).** This number tells you how many groups (clusters) you want to find. Let's say you pick 3, so you want to find 3 groups.
2. **You randomly place "K" (in our example, 3) "centres" on the map (centroids).** Think of these as initial guesses for the centre of each group. They could be anywhere.
3. **Each dot on the map is analysed according to its distance to the 3 centres.** You then assign each dot to the group of that closest centre. So, all the dots closest to the first centre form group 1, all the dots closest to the second centre form group 2, and so on.

K-means (pseudocode) part 2

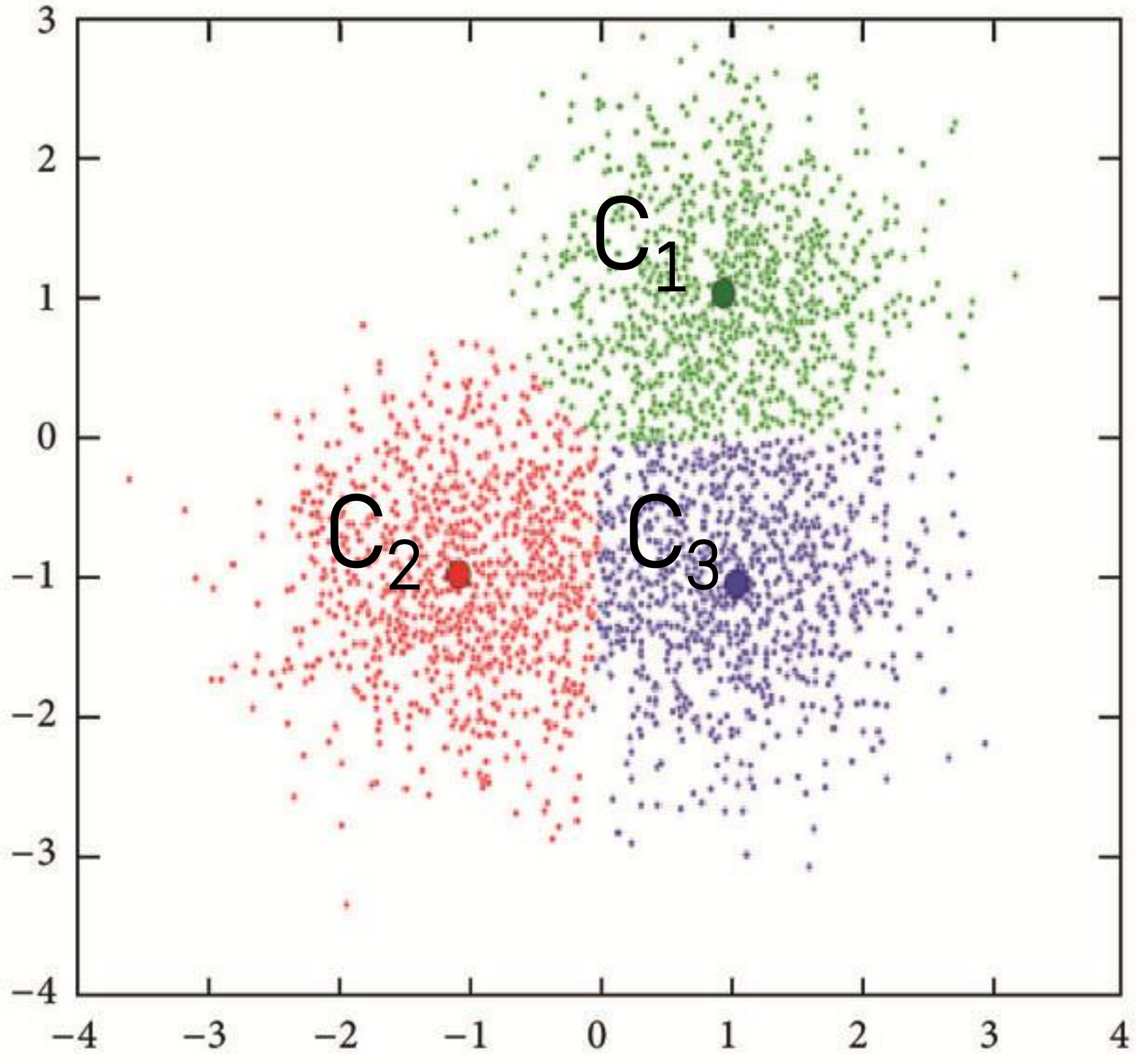
4. **Recalculating the "centre" of each group.**
   For each group, you look at all the dots that belong to it and find the average position (the new centre) of those dots. This new centre is hopefully a better representation of where the group is located.

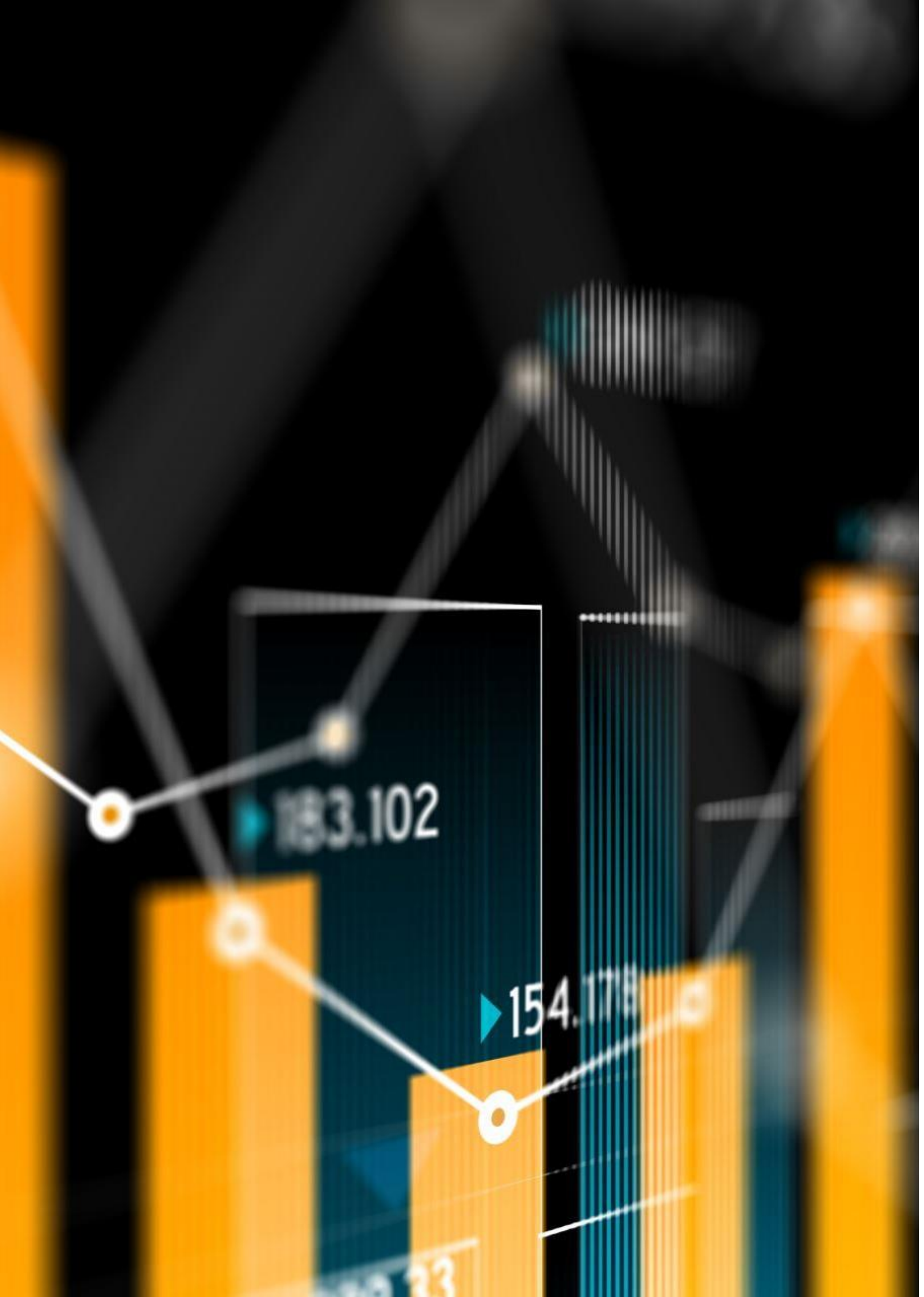
5. **Repeating steps 3 and 4.**
   You re-assign each dot to the closest *new* centre, and then you recalculate the centres of the groups based on the new assignments.

6. **Keep doing this until the centres don't move much anymore, or the groups don't change significantly.** When this happens, it means the algorithm has found a stable set of clusters.

K-means scatter plot

The scatter plot you can see on the right is an example of K-means with K=3.
You can notice the three centroids (C_1, C_2, C_3) within the three clusters.





K-means Pros

- Pros
- **Simplicity**: Conceptually straightforward and easy to implement
- **Efficiency**: Fast with linear time complexity O(n·k·d·i) where n is the number of data points, k is the number of clusters, d is the dimensionality, and i is the number of iterations
- **Scalability**: Works well on large datasets with high dimensions
- **Guaranteed convergence**: Will always converge to a local optimum
- **Adaptability**: Can be modified (k-means++, mini-batch k-means) to improve performance
- **Interpretability**: Results are easy to understand and explain

K-means Cons

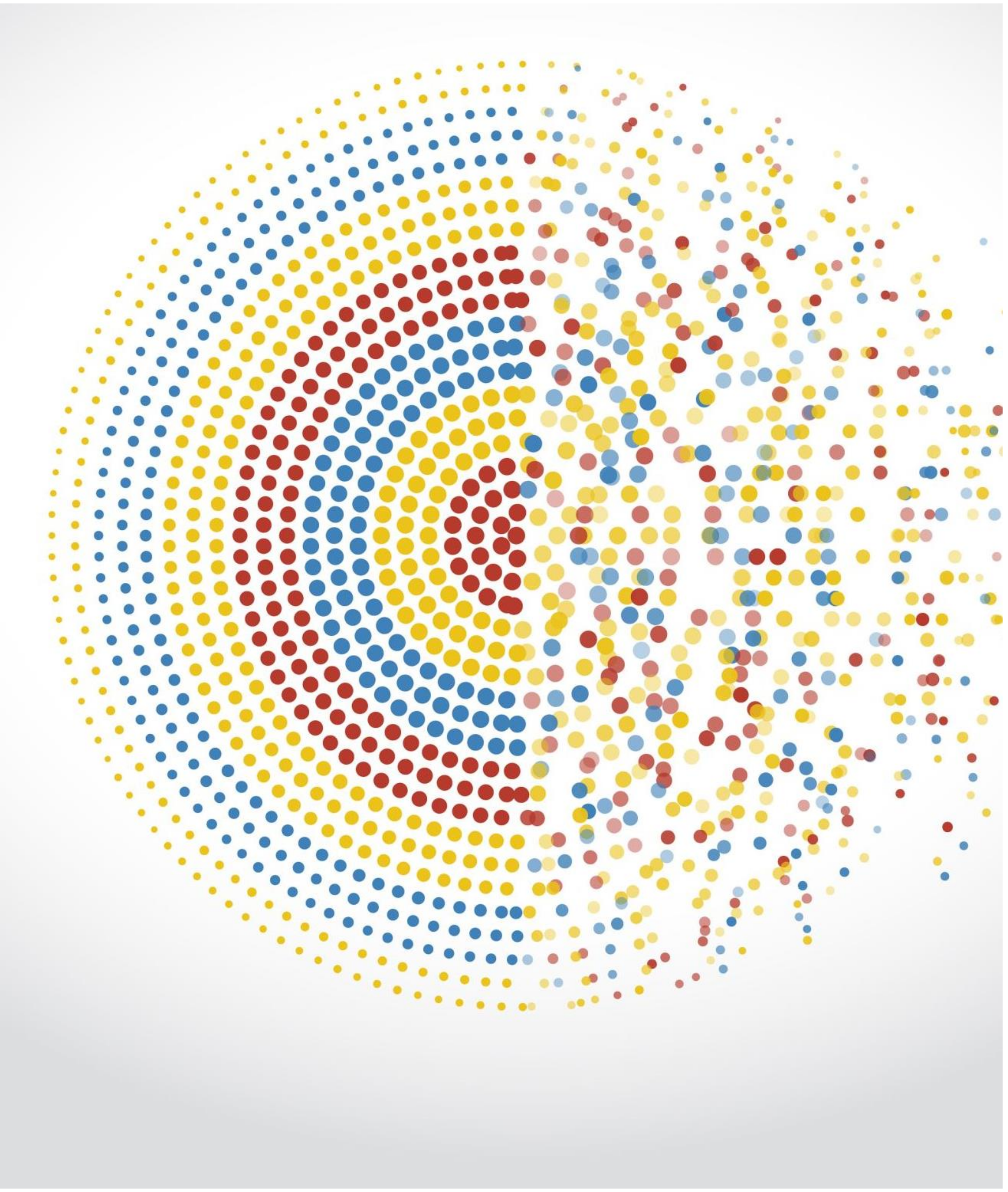
- Cons
- **Requires specifying k**: the number of clusters in advance must be decided in advance
- **Sensitive to initialization**: Final clusters can vary based on initial centroid placement
- **Limited to spherical clusters**: Performs poorly when clusters have complex shapes
- **Sensitive to outliers**: Outliers can significantly distort the mean calculation
- **Euclidean distance limitation**: Standard implementation uses Euclidean distance, which may not be appropriate for all data types
- **Not deterministic**: Different runs can produce different results due to random initialization

Running K-means over a customer purchasing dataset

- Check out the following GitHub repository https://github.com/alessandrobruno10/IULM
- Open your terminal, browse your working folder and type as follows to clone the repo on your folder:
- $ git clone https://github.com/alessandrobruno10/IULM
- You'll have a K-means solution clustering a customer purchasing dataset, which is a .csv file.

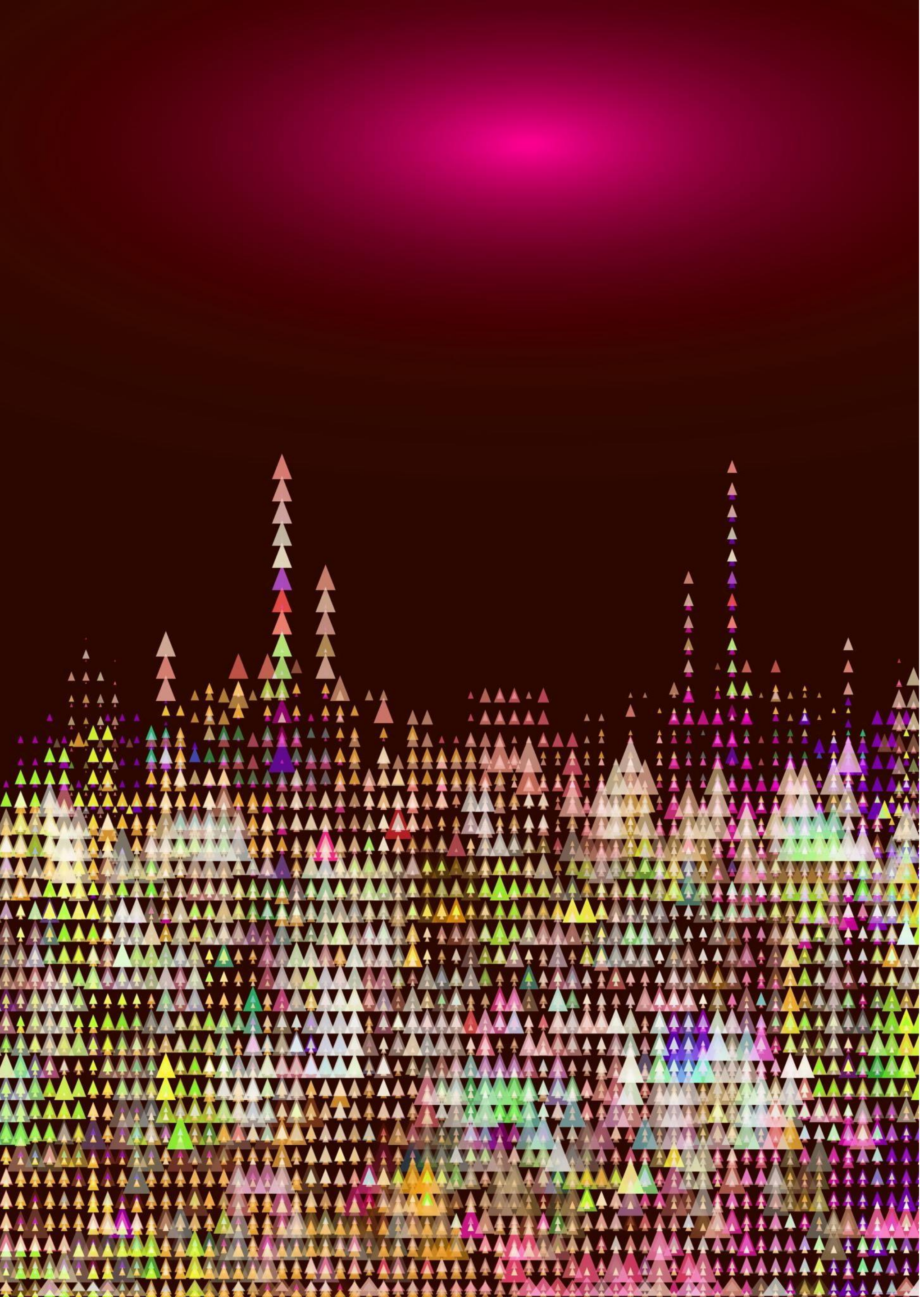
Hierarchical Clustering

- Hierarchical clustering is generally considered the opposite of partitioning clustering methods like K-means.
- What are the most meaningful differences between Partitioning Clustering and Hierarchical Clustering?



Partitioning Clustering

- Creates a single-level, flat partition of data points
- Requires specifying the number of clusters in advance
- Optimizes a global objective function
- Allows data points to move between clusters during optimization
- Typically produces disjoint clusters

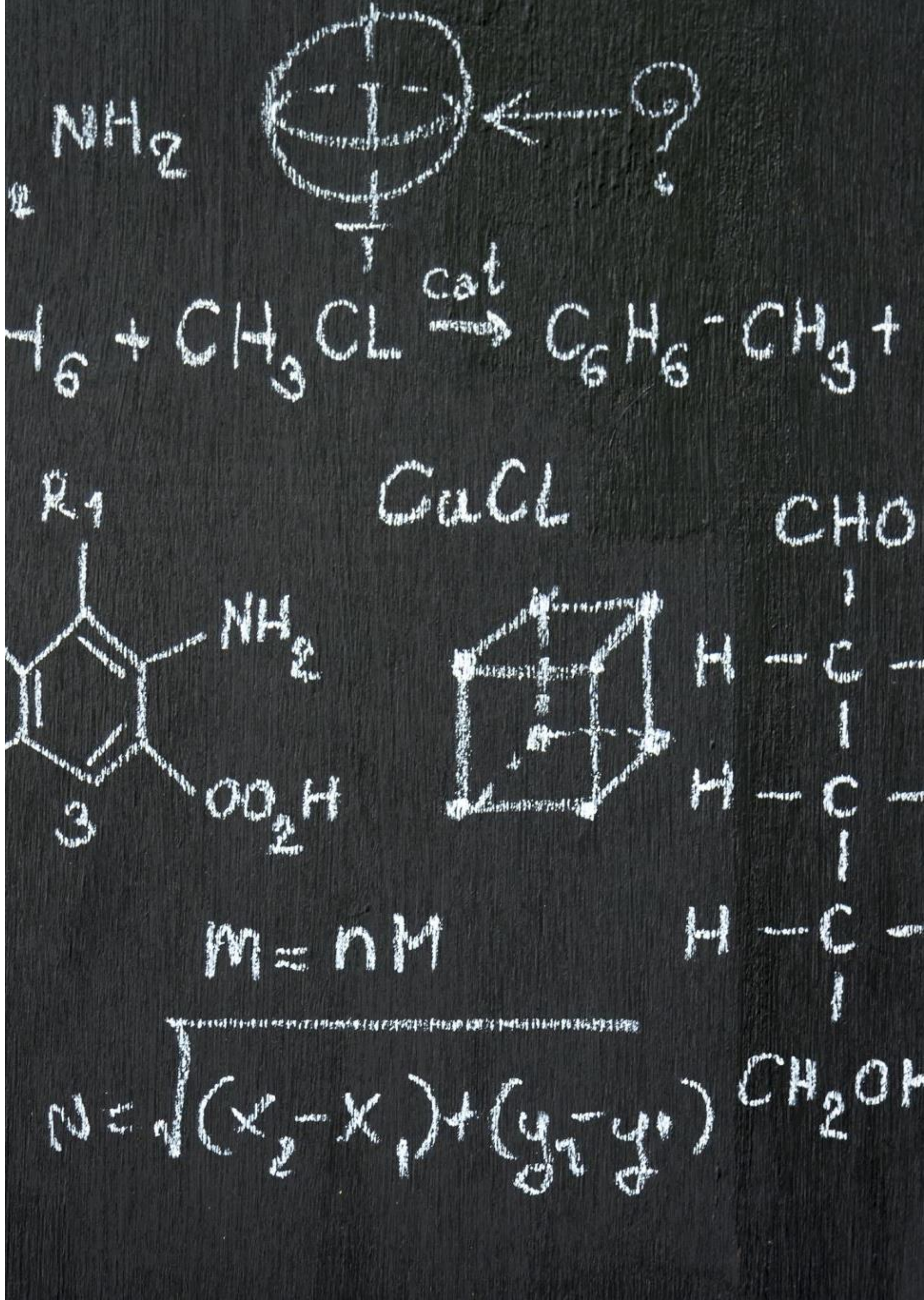


Hierarchical Clustering

- Creates a multi-level hierarchy of nested clusters
- Doesn't require pre-specifying the number of clusters
- Makes local decisions at each step (which clusters to merge/split)
- Once a point is assigned to a cluster, **it cannot be reassigned**
- Produces a complete dendrogram showing relationships between clusters at different levels

Hierarchical Clustering

- **Agglomerative (bottom-up)**:
  - Starts with each data point as its own cluster
  - Progressively merges the closest clusters
  - Continues until all points belong to a single cluster
- **Divisive (top-down)**:
  - Starts with all points in one cluster
  - Recursively splits clusters
  - Continues until each point is in its own cluster

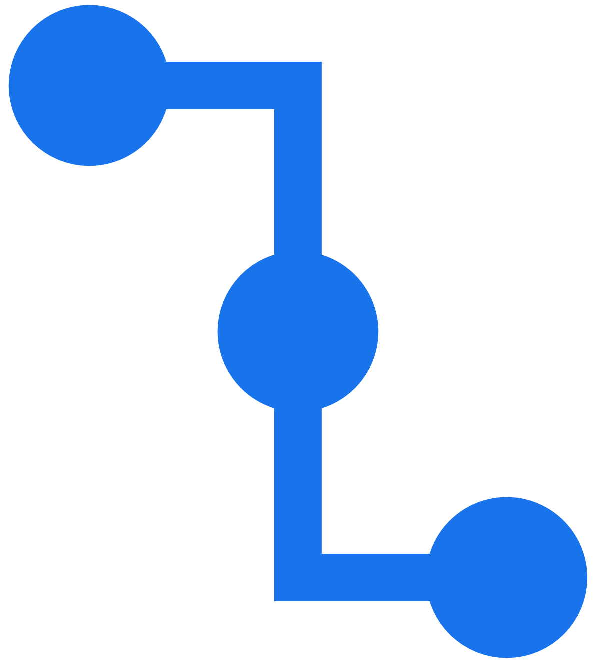


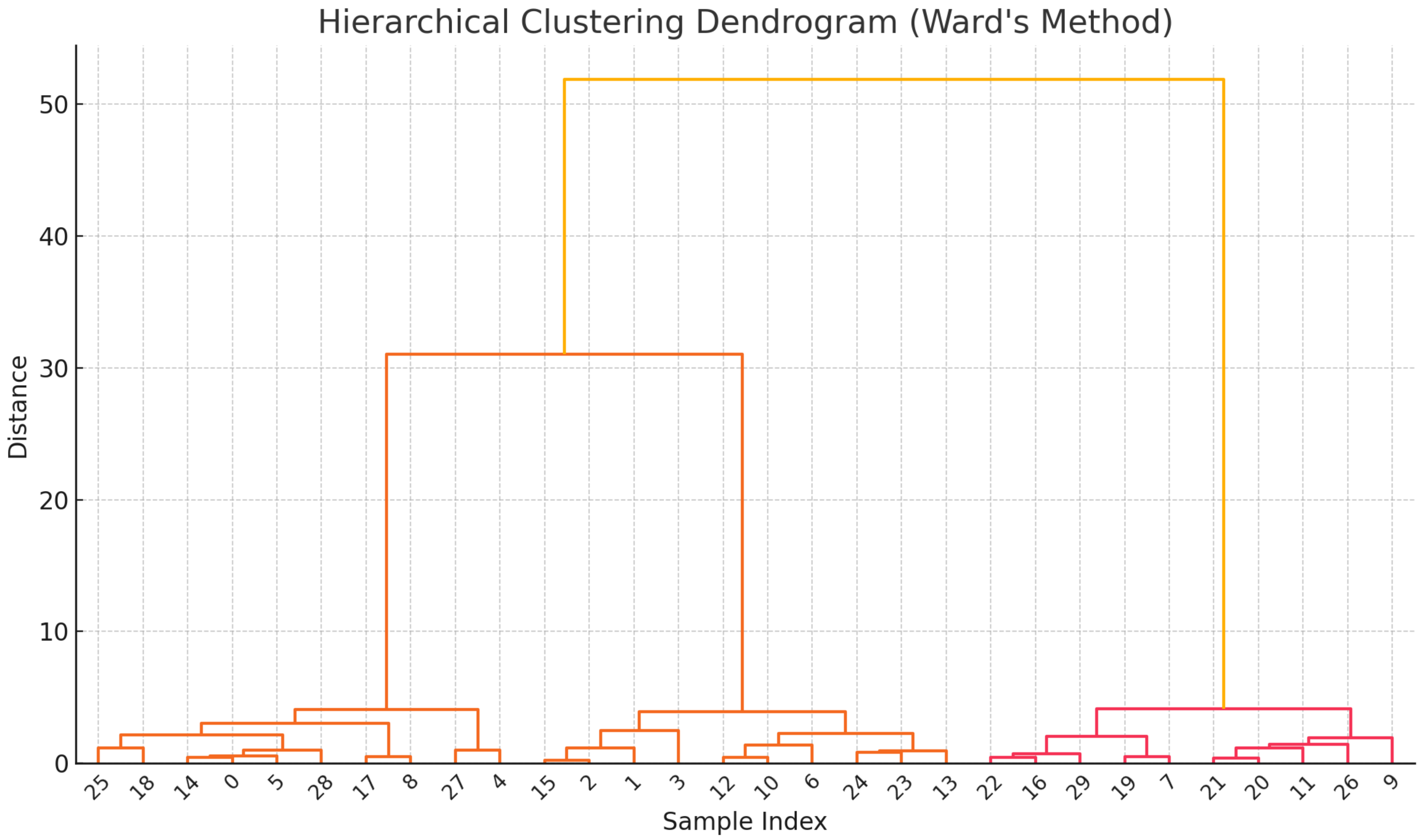
Agglomerative Clustering

1. Assign each point to its own cluster
2. Compute distances between all cluster pairs
3. Merge the two closest clusters
4. Update distances between the new cluster and all others
5. Repeat steps 3-4 until only one cluster remains

Clusters distance

- Distance between clusters can be measured using different approaches:
- **Single linkage**: Distance between the closest points in two clusters
- **Complete linkage**: Distance between the farthest points in two clusters
- **Average linkage**: Average distance between all point pairs across clusters
- **Ward's method**: Minimizes the increase in within-cluster variance




Hierarchical Clustering Dendrogram (Ward's Method)
Distance
0
10
20
30
40
50
25
18
14
0
5
28
17
8
27
4
15
2
1
3
12
10
6
24
23
13
22
16
29
19
7
21
20
11
26
9
Sample Index

Dendrogram description

- **The dendrogram** is a visual representation generated (**Ward's method is used here)** for hierarchical (agglomerative) clustering.
- Each vertical line represents a cluster merge, and the height at which two clusters are joined reflects the **distance (or dissimilarity)** between them.
- You can "cut" the dendrogram at any height to choose the number of clusters you want, for example, cutting at a certain height might give you **3 clusters**.

Hierarchical Clustering (Pros)

Allows for a visual hierarchy (dendrogram)

Does not require pre-specifying cluster count

Versatile with different distance metrics

Can discover clusters of various shapes

Useful for exploring data structure

Hierarchical Clustering (Cons)

- Computationally expensive ($O(n^3)$ time complexity)
- Can't undo merge/split decisions
- Sensitive to noise and outliers
- Challenging to interpret with large datasets

Hierarchical Clustering in Retail Supply Chain Optimization

- The Business Problem
- A national retail chain with over 500 stores was struggling with inventory management challenges:
  - Frequent stockouts at some locations
  - Excess inventory at others
  - Rising logistics costs
  - Declining customer satisfaction
- Traditional approaches using geographic zones or store size weren't effectively capturing the complex patterns in demand and inventory needs.

Hierarchical Clustering in Retail Supply Chain Optimization

- The supply chain analytics team implemented hierarchical clustering to identify natural groupings of stores with similar operational patterns:
- **Data Collection**: They gathered 18 months of data for each store including:
  - Sales velocity by product category
  - Seasonal demand fluctuations
  - Return rates
  - Delivery lead times
  - Storage capacity
  - Local market characteristics

Hierarchical Clustering in Retail Supply Chain Optimization

- **Clustering Approach**: They used agglomerative hierarchical clustering with Ward's linkage to group stores based on operational similarities rather than just geography.
- **Dendrogram Analysis**: By analyzing the dendrogram, they identified seven distinct store clusters that represented natural breakpoints in operational patterns.

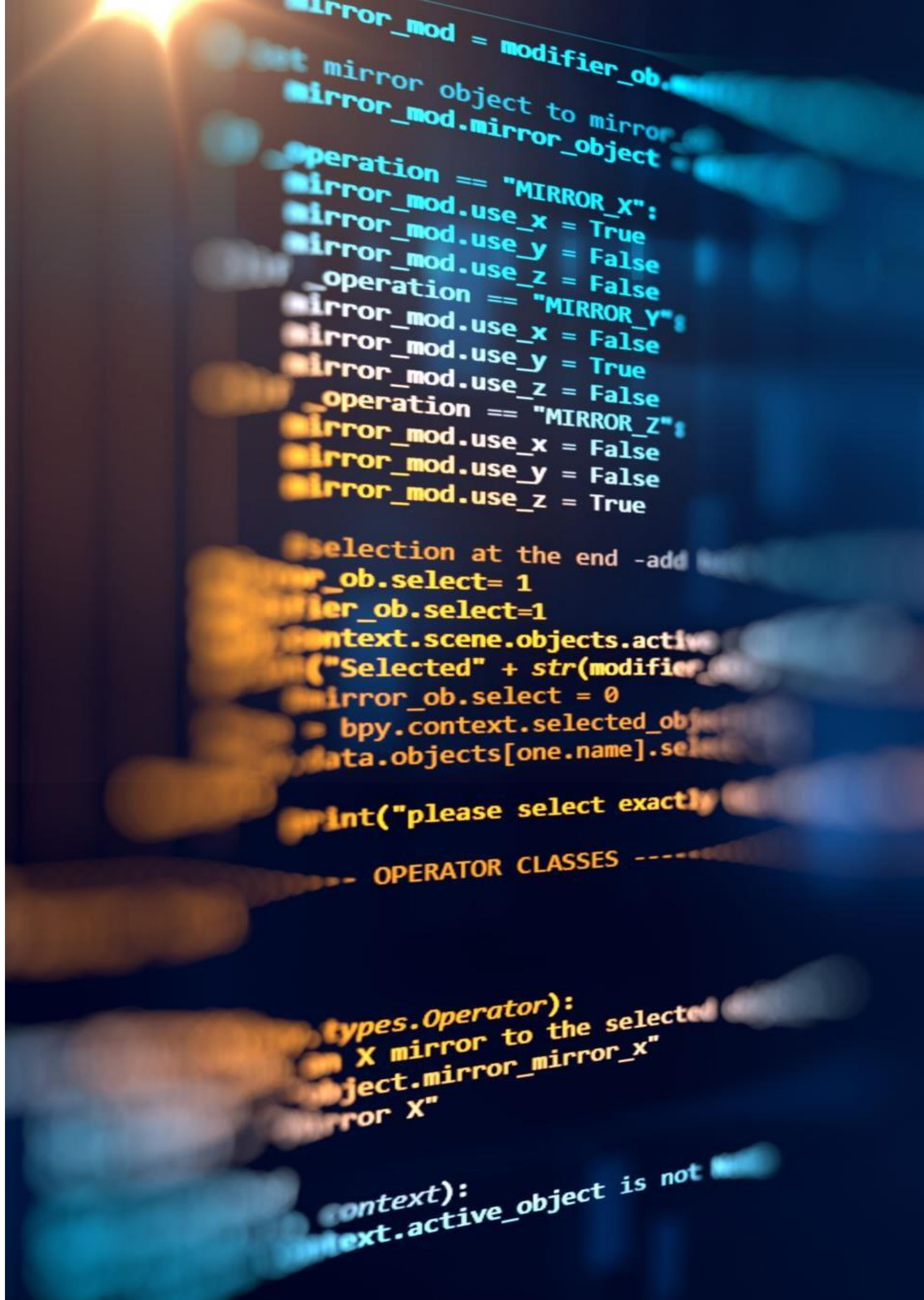
Coding Hierarchical Clustering

Open your terminal, browse your working folder to clone the repository with the code snippets:

$ git clone https://github.com/alessandrobruno10/IULM.git

Look at the csv file containing the dataset

Run the python program to launch Hierarchical Clustering and check out the dendogram

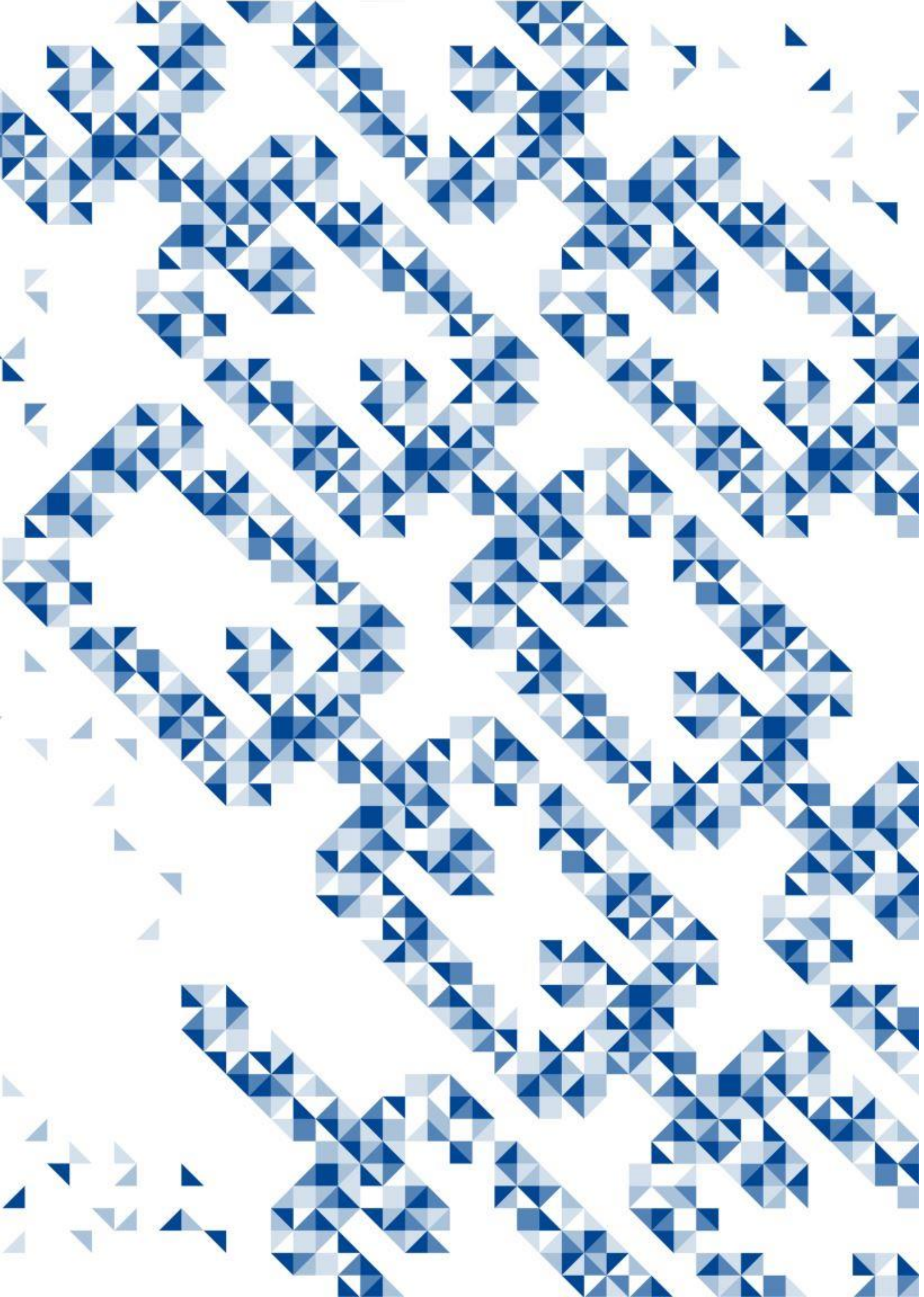


DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- What is DBSCAN for?
- What does it make DBSCAN different from Hierarchical Clustering and K-means?
- Let's dig into it!
- First, DBSCAN is specifically designed to find clusters of arbitrary shapes in data while effectively identifying and isolating noise points.

DBSCAN

- **Finding Arbitrary-Shaped Clusters**: **Unlike k-means** which finds circular/spherical clusters, DBSCAN can identify clusters of **any shape**.
- **Noise Detection**: DBSCAN automatically identifies **outliers** as "noise" points that don't belong to any cluster.
- **No Pre-defined Cluster Count**: You don't need to specify the number of clusters in advance (unlike k-means).



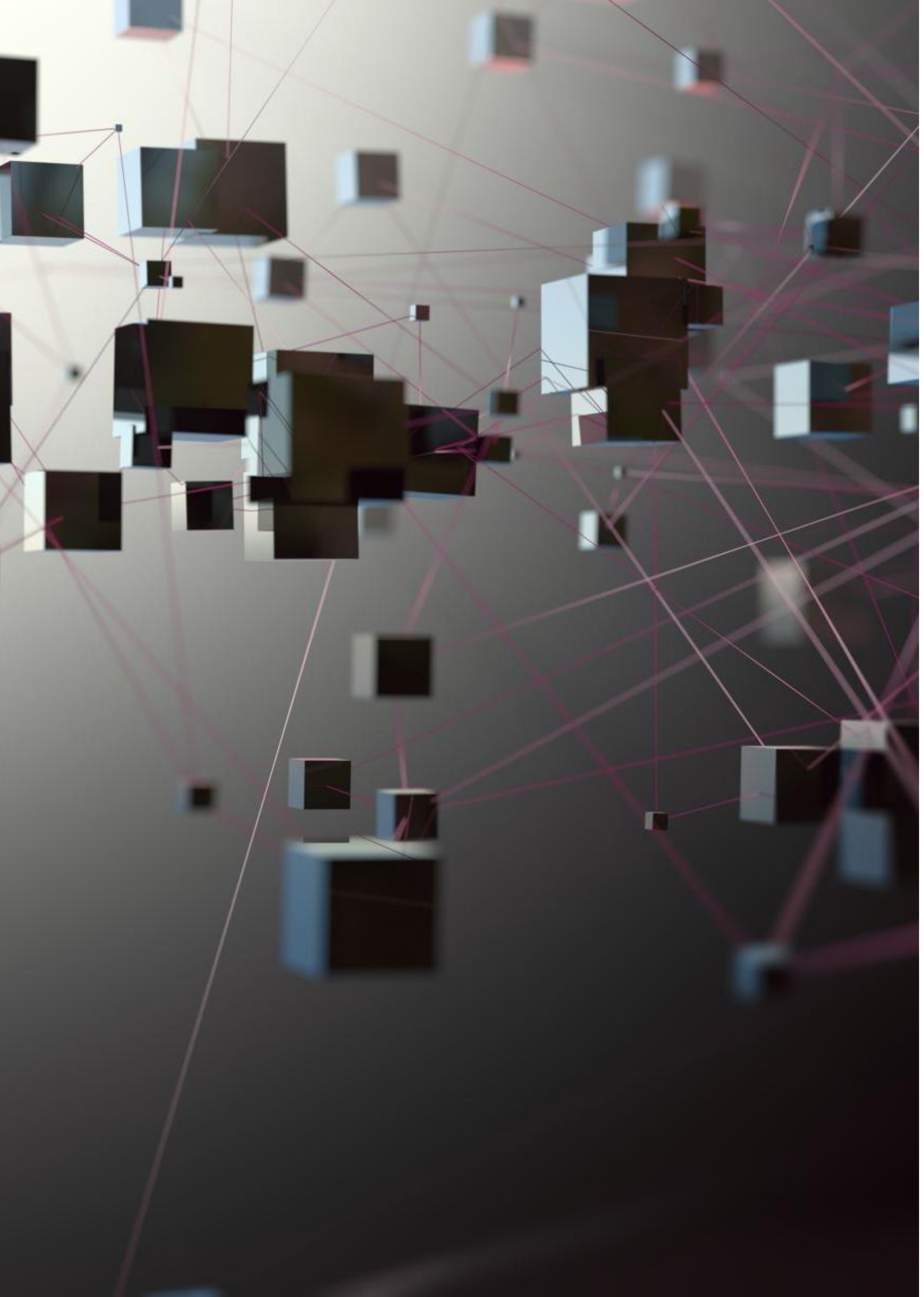
DBSCAN in a nutshell

- DBSCAN functioning relies on two parameters:

- **Epsilon (ε)**: The maximum distance between two points to be considered neighbors
- **MinPts**: Minimum number of points required to form a dense region
- The algorithm classifies points as **Core, Border, and Noise points.**

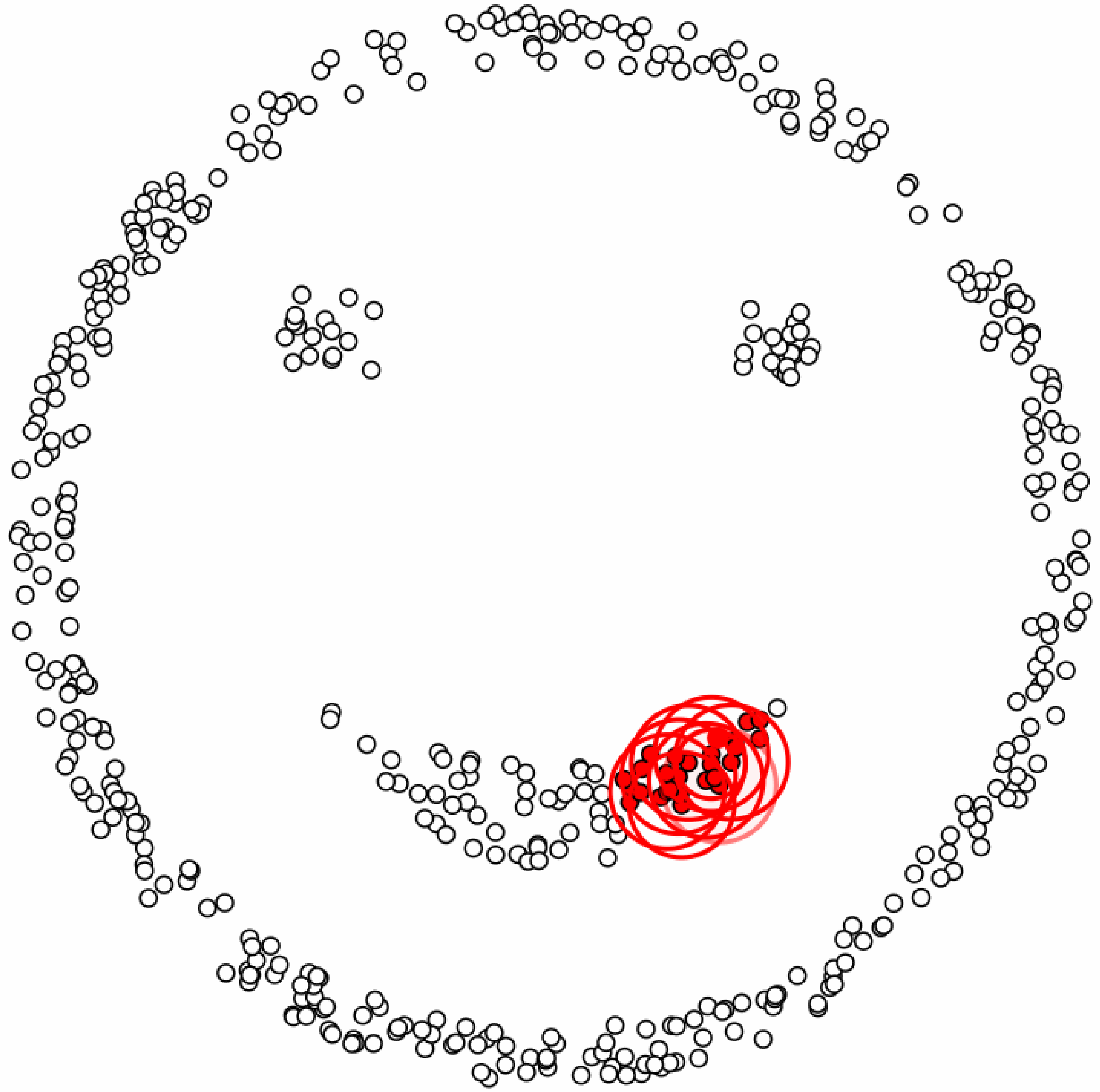
DBSCAN in a nutshell

- **Core points**: Have at least **MinPts** neighbours within **ε** distance
- **Border points**: Within **ε** of a core point but have fewer than **MinPts** neighbours
- **Noise points**: Neither core nor border points
- **DBSCAN** is valuable when you don't know how many clusters exist in your data and when your clusters might have complex, non-convex shapes that algorithms like k-means would struggle with.



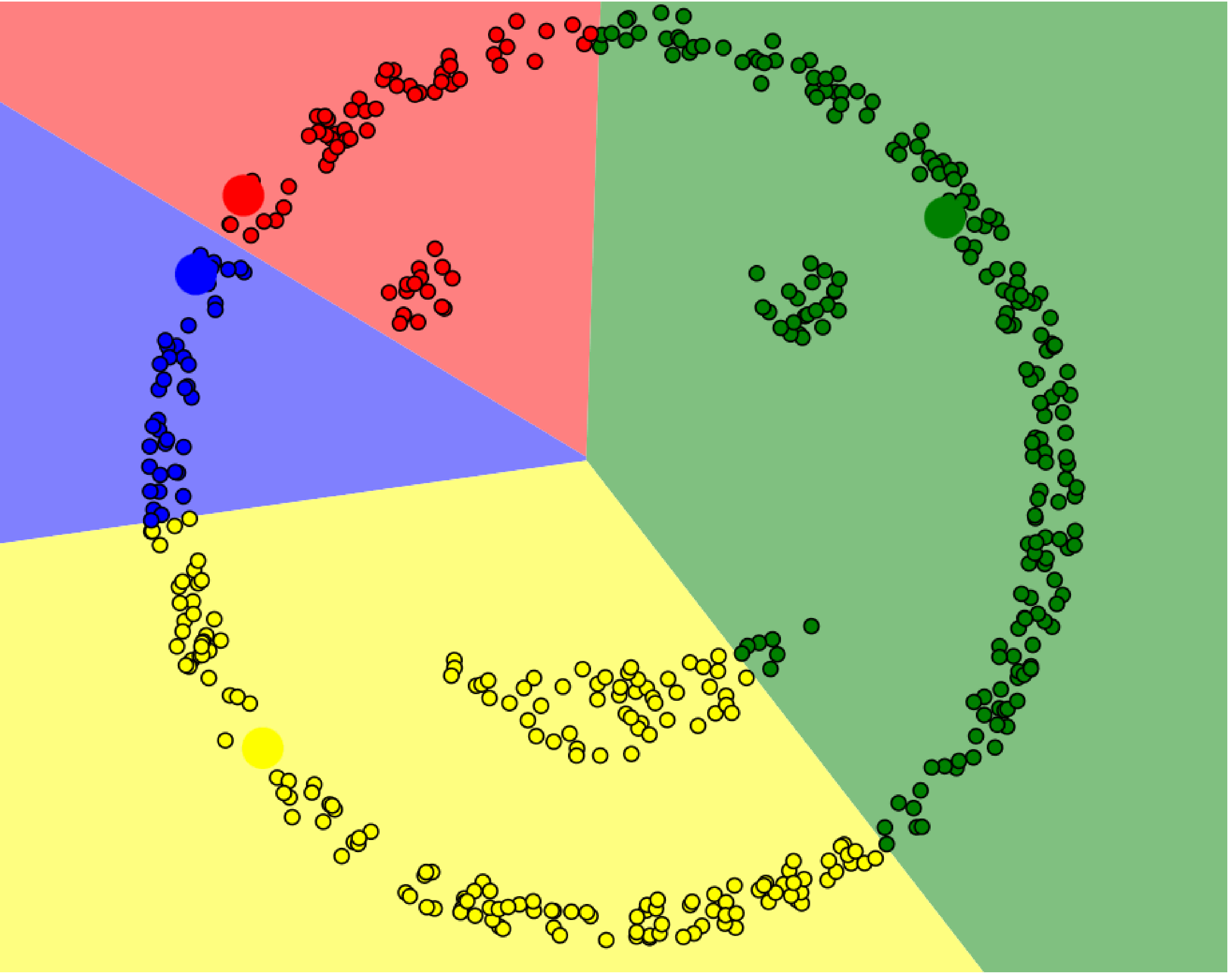
Let's have a look at DBSCAN vs K-means

- Consider a spatial datapoint distribution
- We want to have a better grasp of DBSCAN and K-means working on the data and creating clusters.
- A graphical animation helps us out with the understanding of the different clustering approaches.



White dots represent data points that we want DBSCAN to cluster.

Picture credits:
https://medium.com/@jdseo/understanding-dbscan-a-practical-guide-for-beginners-with-business-applications-9458792d1df8



K-means in action on the same data points. Noticeably, the spatial cluster distribution depends on centroids' coordinates.

Picture credits:
https://medium.com/@jdseo/understanding-dbscan-a-practical-guide-for-beginners-with-business-applications-9458792d1df8

Take-home Messages

- Unsupervised Learning is adopted when samples in our dataset are not labelled
- Clustering is an unsupervised learning approach grouping similar datapoints
- Clustering methods are mainly grouped into partitioning and hierarchical (there is also density-based clustering methods though).
- K-means is a partitioning clustering method (K is to be given in advance)
- Hierarchical Clustering can be agglomerative or divisive
- Hierarchical Clustering is visually represented by a dendrogram.
- DBSCAN is a density-based clustering approach that relies on two parameters ε, and MinPts.