Unsupervised ML
- No labels / output given
- Model finds pattern by itself
- Main task clustering

Clustering
↳ grouping similar data points together.
Same group → more similar data points.
different group → ~~ett~~ less similar.

K-means Clustering Algorithm

It is an unsupervised ML algo that divided dataset into K distinct cluster.
Each cluster is represented by centroid (mean).
Each data point belongs to nearest cluster.

Goal → group similar data together without labels.

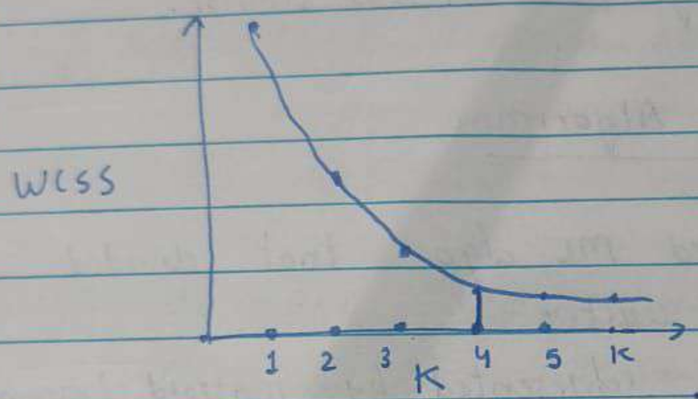
Working cycle

Step 1 → choose K, you have to tell algo don't find it
But to know what will be the best K for your cluster algorithm.

★ Elbow method is used to find best K.
→ Plot K vs WCSS, Look for elbow point.
WCSS (within cluster sum of squares).
WCSS needs to be minimum
Small WCSS → points are closed to centroid (Good cluster)
Large WCSS → points far from centroid (Bad cluster)

Core idea of Elbow Method
↳ As K increase WCSS decreases, but after a point improvement becomes very small. that point is elbow. best K value to use.

Why Elbow → algo don't finds K, we have to tell. if we guess K and it will be wrong K (bad clustering) so finds optimal K by elbow method & use.



K = 4.

Step 2 → Initialize centroids
Randomly select K points as initial centroid

Many times K-Means select centroid randomly so chances are there it can initialize centroids very closely. which lead to form bad clusters.

So (K++) algo comes in picture, it is same as K-means, slight change, it initialize centroids not randomly but based on distance & Probabilistic which helps in best centroid selection.

Step 3 Compute distance of points to all centroid using euclidian Distance formula.
Then assign point to nearest centroid.

Step 4 → Update centroid, calculate mean of all points & update centroid.

Step 5 → Repeat (Convergence).

ex → Points, (1,1) (2,1) (4,3) (5,4), K = 2

initial centroid
C1 (1,1) C2 (5,4)

Assignment after distance calculate
[C1 (1,1), (2,1)] [C2 [5,4), (4,3)]

↓ updation ↓

C1 (1.5, 1) C2 (4.5, 3.5)

---

② Hierarchical clustering

It is an unsupervised clustering algo that builds a hierarchy of clusters in the form of tree structure (Dendogram).

↳ Types
- ↳ Agglomerative (Bottom up) (Commonly used).
- ↳ Divisive (Top-down) (Rarely used). Computationally expensive.

Agglomerative HC → intuition

Step 1 → Start with individual clusters
if there n points, initially n clusters.

Step 2 → Compute distance Matrix
calculate distance b/w every pair of clusters/points.
with euclidean distance.

Step 3 → Merge Closest clusters
Find two nearest clusters, merge them into one.

Step 4 → Update Distance Matrix
recalculate distances

Step 5 → Repeat until one cluster
~~on~~ continue till points are in single cluster.

~~St~~ Final clusters
Draw a ~~Hozi~~ Horizontal line at max distance height
no. of vertical lines cut = no. of clusters.

Adv
No need to specify K.
Dendogram given full picture
Works well with small datasets

Disadv
Computationally expensive
Not for large datasets
Sensitive to outliers

Date: ___

ex → to visualize

A (1,1) B (2,1) C (4,3) D (5,4).

	A	B	C	D
A	0	1	3.61	5
B	1	0	2.83	4.24
C	3.61	2.83	0	1.41
D	5	4.24	1.41	0

First merge, min distance = 1 b/w A & B

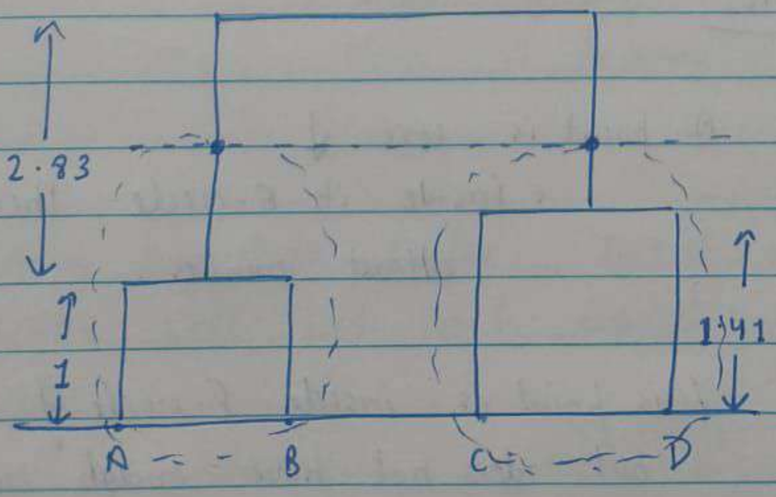
	AB	C	D
AB	0	2.83	4.24
C	2.83	0	1.41
D	4.24	1.41	0

Second merge, min distance = 1.41 b/w C & D

{AB} , {C, D}

Distance b/w AB & CD = min (2.83, 4.24) = 2.83)

Final merge → {ABCD}



Two point vertical line cut, so two clusters.

(Just to visualize)
Dendogram

Date: __.__.__

③ DBSCAN clustering algo

Full Form → Density Based spatial clustering Algorithm with Noise

It is an unsupervised, density based clustering algorithm that groups based on density of data points, and automatically detect noise / outliers.

Places where many points are close together → cluster
Lonely points far away → noise (outliers)

→ ① ε (epsilon)
A radius / circle size
Radius of neighbourhood
Too-small → many outliers / noise - no cluster formed
Too-large → all point merge in one cluster.

→ ② minPts
Minimum no. of points required to form a dense region includes the point itself.
or
min no. of points ~~ind~~ inside circle to form a cluster.

★ Types of Points

① Core Point — A point is core if
- inside its ε-circle there are atleast minPts.

② Border Point — This point is inside ε-circle of core point but does not have enough neighbour by itself.

(3) Noise Point

↳ Not Core, Not Border

not in any core ε-circle area and also self no minPts neighbour available in ε-circle.

* Working DBSCAN

Step1 - Choose ε & MinPts (hyperparameter tuning).

Step2 - Pick any point

└ Draw ε-circle around it & count point inside including that point also.

Step3 - if points ≥ MinPts → Core

Else → Noise (For now).

Step4 → If point is Core,

Make a new cluster, add all nearby point, if nearby points is also core expand it.

Step5 → Repeat for all Points.

- Every point is visited once
- Noise stays noise unless it touch a ε-circle of any core point if it does then it became border point.

Adv

- Finding any shape cluster, Detect outliers, no K needed.
- works well with real world noisy data

Dis adv

Hard to choose ε

High dimensional data problem.