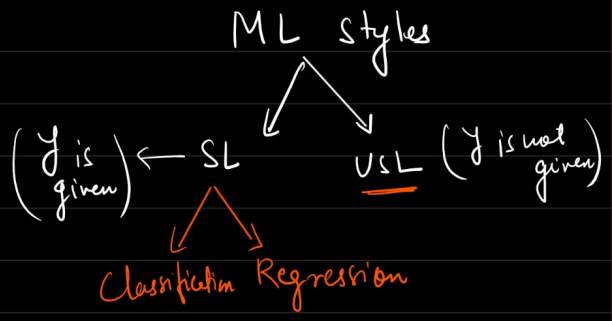
Introduction to Unsupervised learning

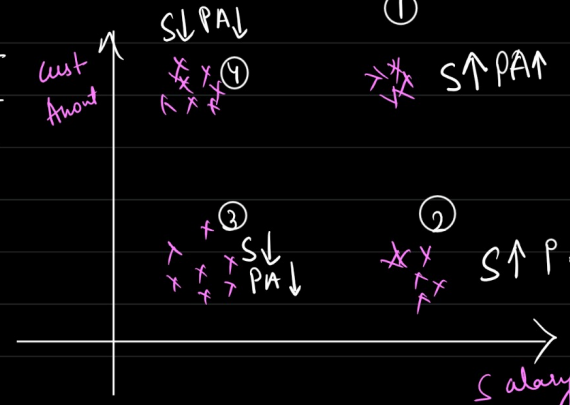
* Target variable (Y) is Not given

* USL groups / segments your data (Finds pattern in the data)



Examples Zara Store

Customer purchase amount	Customer Salary in Lakhs
2500	10
—	—
—	—
—	—
—	—



Clothes mix up

Shop Keeper is blind → will try to touch the cloth and understand the pattern, different rocks

To increase revenue → Group 2 customers should be focussed more.

→ Group 1 → loyalty card & Extra discount.

* depends on you how you want interpret the groups and bring business.

Motivation

→ Need to launch a campaign based on income / sales.
→ Different ways of fraud / money laundering.
→ Grouping of images / patterns of specific image
→ Document / Article Analysis.
→ Cohort Analysis.

* In higher dimensions, you cannot identify groups / patterns manually. Therefor you need USL.

USL

① K-means clustering
② Hierarchical clustering
③ DBSCAN

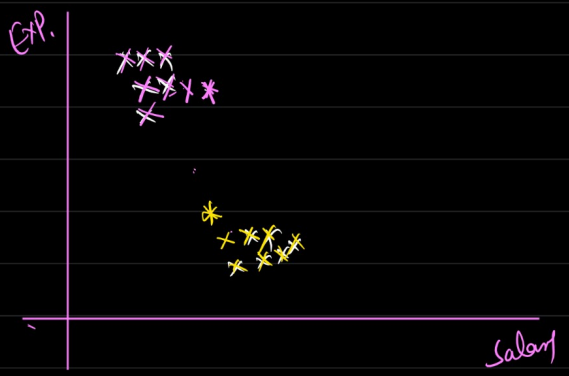
* K means clustering

Steps of K means clustering

Step-1 intialize Centroid (K)
- → can be random dp
- → Centroid of all the dp
- → A rrandom dp

Step-2 → Points nearer to the centroid will be labelled as that group

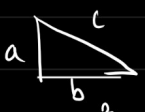
Step-3 Re-Calculate the centroid ↳ Avg.
Again repeat Skp 2 untill Centroid doesnt change.



Exp	Salary
2	5
3	6
5	7

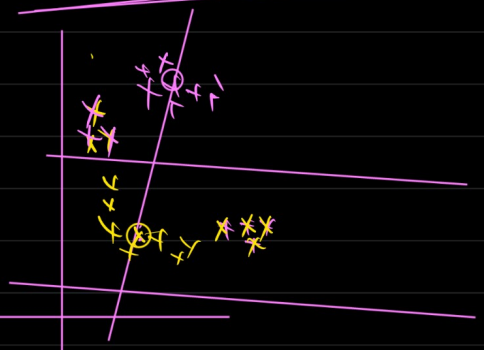
$\left(\frac{2+3+5}{3} \quad \frac{7+6+5}{3}\right)$

( )



$c^2 = a^2 + b^2$

⟹ after recalculating Centroid



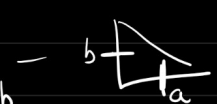
*Distance

① Euclidean distance

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

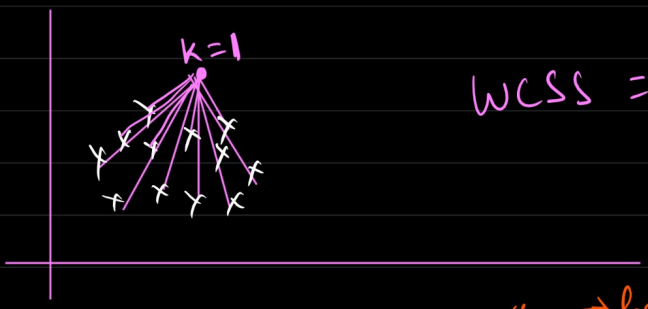
② Manhattan distance

$$\text{Manhattan distance} = |x_2-x_1| + |y_2-y_1|$$



How to Select the k value

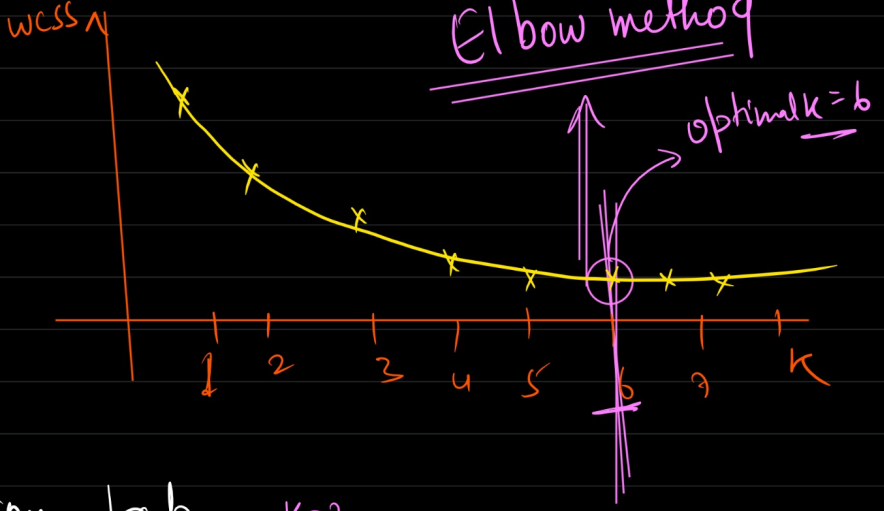
WCSS → Within Cluster Sum of squares distance



$$WCSS = \sum_{i=1}^{n} \left(\text{distance b/w point to nearest Centroid} \right)^2$$

⇒ for k=2 → As you increase k, wcss decreases
→ And at optimal k, wcss stops changes.

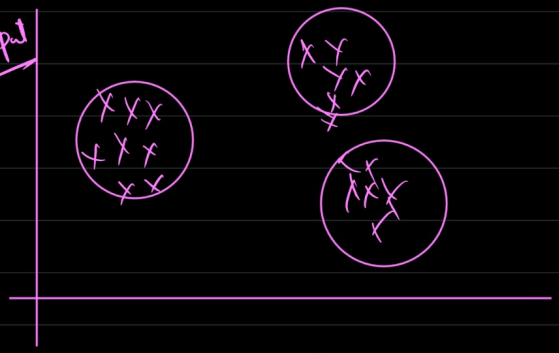
Elbow method



* Random Initialization trap

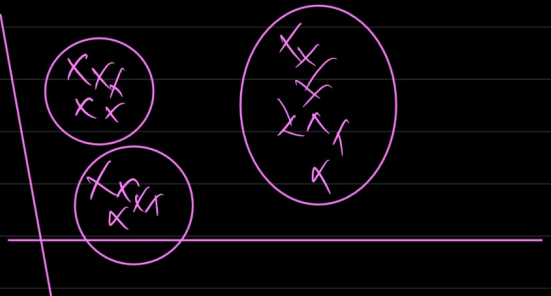
K=3

Kmeans →



Another possibility

Kmeans →



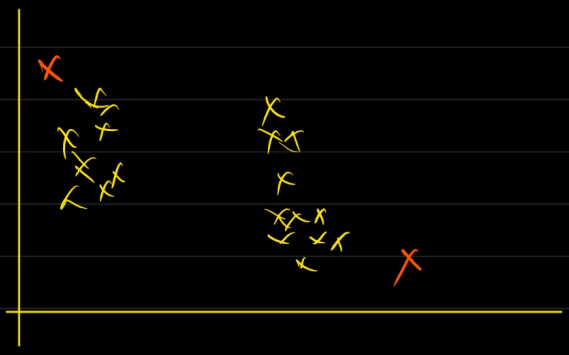
* The final cluster depends on the initialization of K.

⇓

Kmeans++

⇓

initialize the k's as much far as possible



K-means clustering is an unsupervised machine learning algorithm used to partition data into K distinct groups based on similarity. It works by randomly initializing K centroids, then assigning each data point to the nearest centroid to form clusters. The centroids are recalculated as the mean of the points in each cluster, and the process repeats until convergence, meaning the centroids no longer move significantly. K-means is commonly used for tasks like customer segmentation, image compression, and pattern recognition. It requires the number of clusters (K) to be predefined, which can be a limitation in some cases.

Selecting the optimal value of K in K-means clustering can be done using several methods:

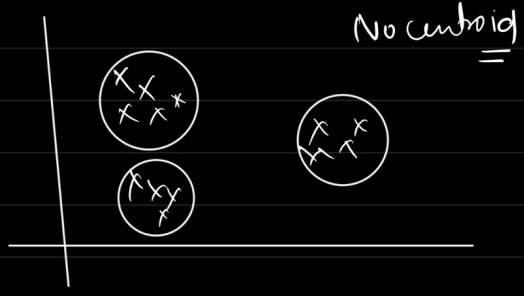
Elbow Method: Plot the sum of squared distances (inertia) between points and their assigned cluster centroids for various values of K. The optimal K is at the "elbow" point, where the reduction in inertia slows down significantly, indicating diminishing returns from adding more clusters.

Silhouette Score: This metric measures how similar points in a cluster are to each other versus points in other clusters. The value of K that maximizes the silhouette score is considered optimal.

The random initialization trap in K-means clustering refers to the problem where the algorithm's outcome depends heavily on the initial random placement of the centroids. Since K-means is sensitive to initial centroid positions, poor initialization can lead to:

Suboptimal Clusters: The algorithm may converge to a local minimum, resulting in inaccurate clustering.
Slow Convergence: Poorly chosen initial centroids may take longer to converge.
Inconsistent Results: Different runs of the algorithm can yield different clusters for the same data set due to random initialization.
Solution:
K-means++ Initialization: This technique strategically selects initial centroids by spreading them out, reducing the chances of poor clustering and improving both convergence speed and quality.

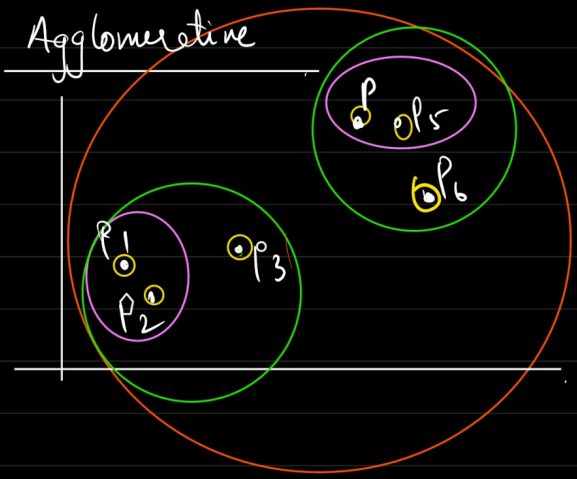
* Hierarchical Clustering



* Hierarchical Clustering.

① Agglomerative Clustering (combining)
② Divisive Clustering (dividing)

* Agglomerative



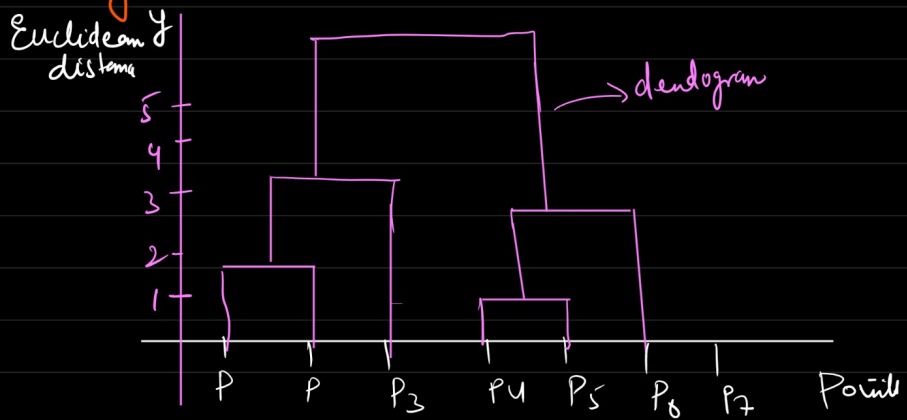
Steps (Agglomerative)

① Each point is a cluster in its own
② Find the nearest point and create a new cluster.
③ keep on doing step 2 untill we get a single cluster.

→ Euclidean dist
→ Manhattan distance
→ Cosine similarity → To calculate distance between two vector (Categorical data | strip variable)

Still how to select K ?

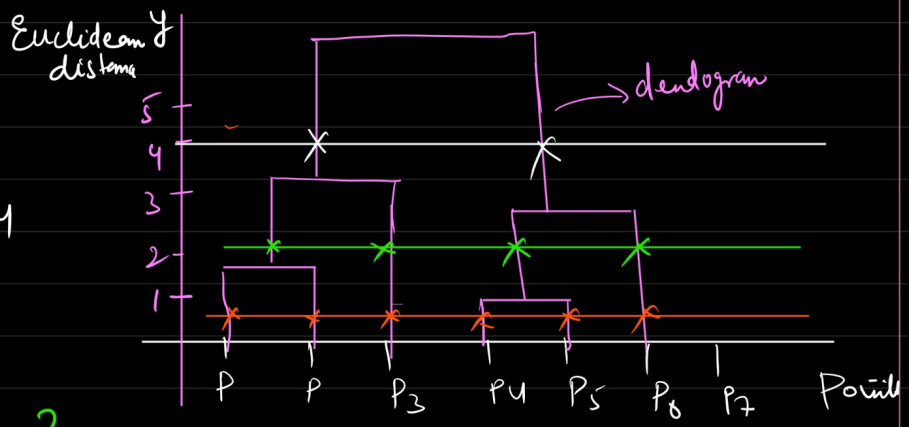
Using dendogram



To determine K

→ threshold on Euclidean distance. (Can be tricky job ⇒ business team)
→ trick → we can take longest vertical line of dendogram where none of the horizontal line of dendogram is passing.

① Using threshold on Euclidean distance



→ Say threshold = 4

⇓

K = 2

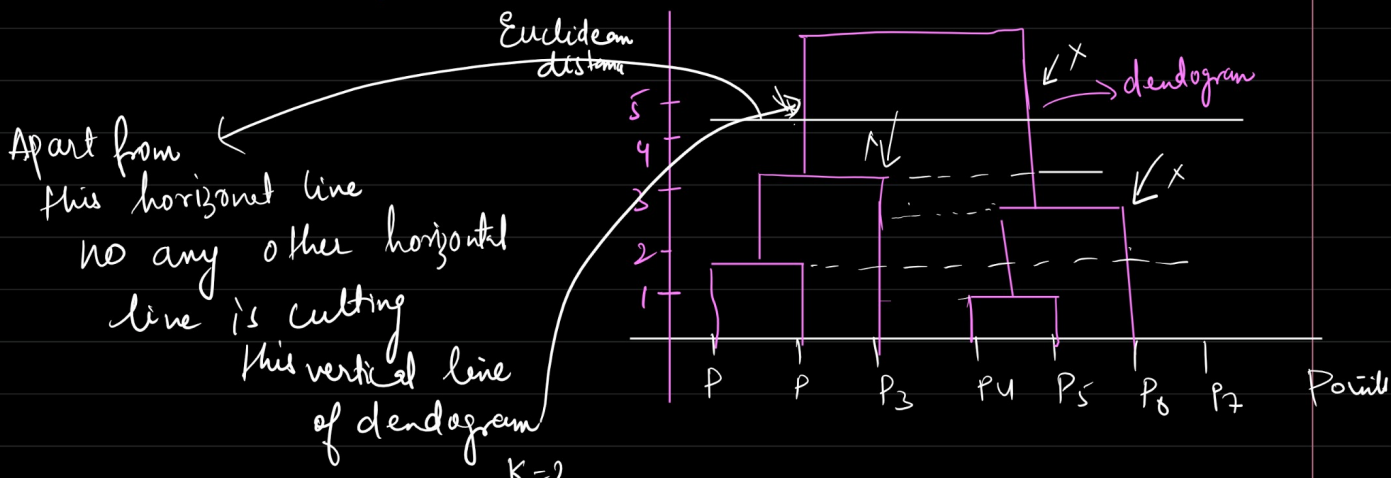
→ Say threshold is 2

K = 2

→ say threshold is 1

K = 6 (all the dP's is cluster in its own)

②

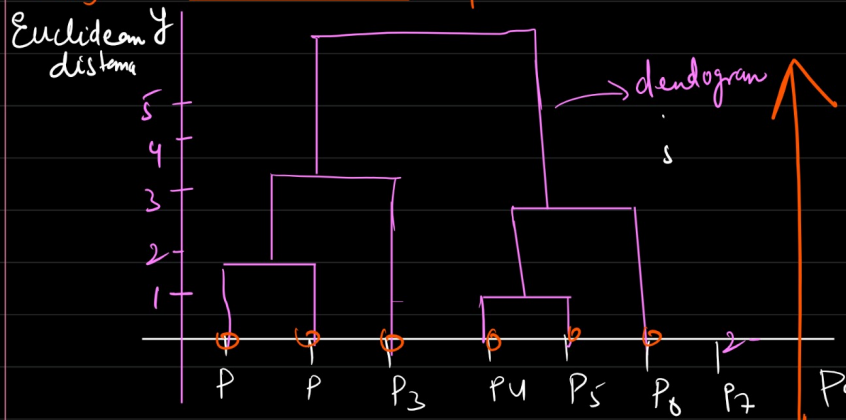


Apart from this horizontal line no any other horizontal line is cutting this vertical line of dendogram

K = 2

Conclusion → Select the longest vertical line such that no horizontal line passed through it.

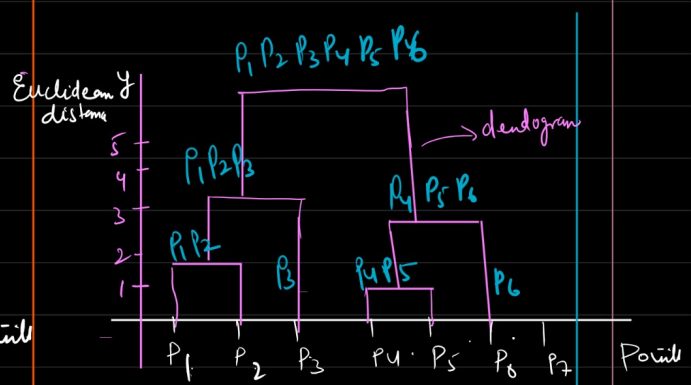
Agglomerative Clustering



→ Here every dP's are getting combined

Combining

Divisive clustering



Keep diving until every individual dp becomes Cluster in its own.

dividing

* K Means Vs Hierarchical (scalability & flexibility)

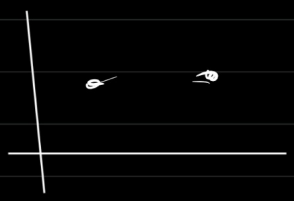
① Size of data → Huge → K Means
→ Small → Hierarchical

② K means — Numerical (only) data (Euclidean | Manhattan distance)

Hierarchical clustering — Variety of data (Euclidean, Manhattan, cosine Similarity)

③ K means → Centroid (k) → Elbow method

Hierarchical — No centroid.

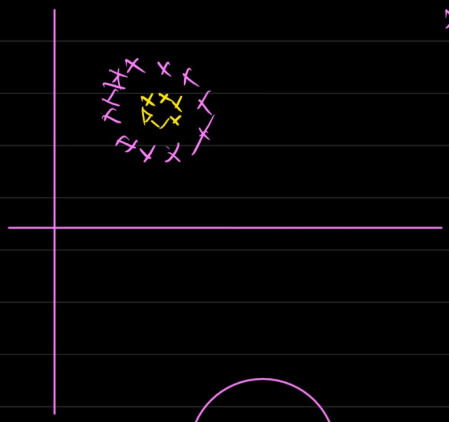


Hierarchical clustering is an unsupervised machine learning algorithm used to build a hierarchy of clusters. It can be performed using two approaches: agglomerative (bottom-up) or divisive (top-down). In agglomerative clustering, each data point starts as its own cluster, and pairs of clusters are merged iteratively based on their similarity until all points are in a single cluster. Divisive clustering works in reverse, starting with one large cluster and splitting it. The results are often visualized using a dendrogram, which shows the cluster hierarchy. It doesn't require a predefined number of clusters, unlike K-means.

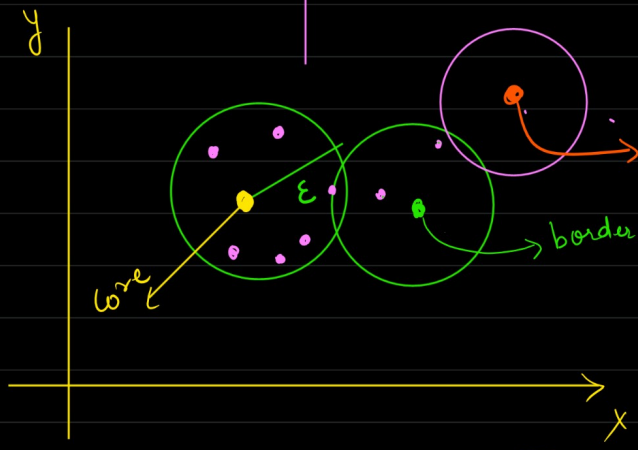
DBSCAN (density based Spatial Clustering of Application with noise)

* Characteristics of DBSCAN
  - → Finds groups / pattern
  - → Finds outliers / noise

if already we had Kmean / hierarchical, then why DBSCAN?



* Distance based Algorithm such as K means & hierarchical can not be used here (non linear data / non linear clustering)



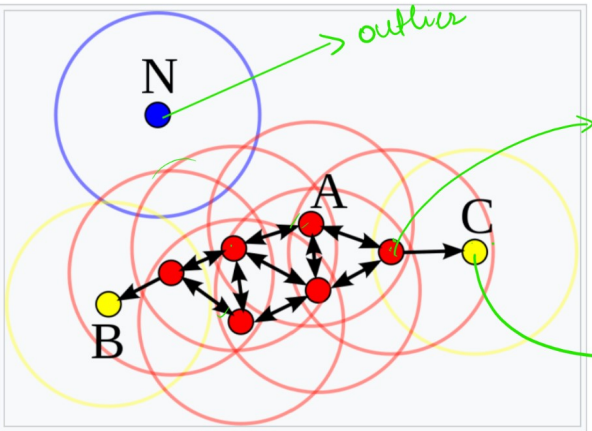
- ● → Core data point
- ● → border data point
- ● → outlier dp.

* minimum no of dp = 4
* radius (ε) = 1 m

* Core dp :-
  The dp's which hv atleast minimum no. of dp in its radius ε (4)

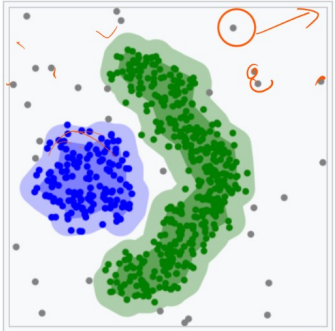
* Border dp → no. of dp within its boundary (ε) will be less than minimum dp

* Outlier (Noise) → No dp in radius ε



In this diagram, $\text{minPts} = 4$. Point A and the other red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.

* Some Examples of DB SCAN working very well with non linear data.



DBSCAN can find non-linearly separable clusters. This dataset cannot be adequately clustered with k-means or Gaussian Mixture EM clustering.

The left image depicts a more traditional clustering method that does not account for multi-dimensionality. Whereas the right image shows how DBSCAN can contort the data into different shapes and dimensions in order to find similar clusters.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups together points that are closely packed, while marking points in low-density regions as outliers. Unlike K-means, DBSCAN doesn't require specifying the number of clusters beforehand. It works by defining a neighborhood around each point using two parameters:

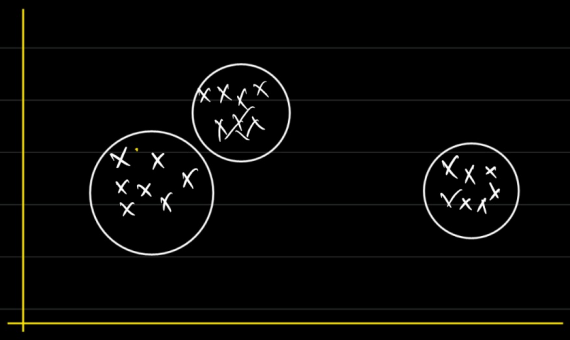
Epsilon : The maximum distance between points to be considered neighbors.
MinPts: The minimum number of points required to form a dense region (cluster).
DBSCAN is effective for identifying clusters of arbitrary shapes and is robust to noise, but its performance depends heavily on appropriate parameter selection.

Silhouette Score / coefficient

→ Metrics for clustering algorithms.

Classification → accuracy
Regression → r square

* When we will say, a given cluster is a good cluster?

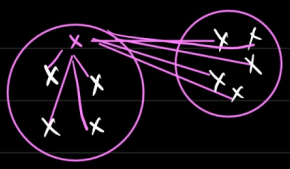


→ within cluster datpoints are compact
↓
tightly packed.

↓
WCSS (within cluster sum of square distance should be minimum)

(intra cluster distance)

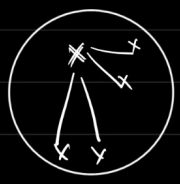
→ Outside cluster sum of square should be as maximum as possible (OCSS) from nearest cluster



(inter cluster distance)

→ separable clusters | good cluster.

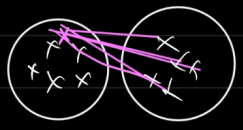
① WCSS



⇒ a(i) = distance within cluster wrt to each dp

for dp(i) belonging a Cluster I is given by

② OCSS → distance from nearest cluster dPs'



b(i)

③ Silhouette Score

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

i is any dP

$s(i)$ → -1 to 1

{More near to 1 better the clustering model is created ⇒}

The silhouette score is a metric used to evaluate the quality of clusters created by a clustering algorithm. It measures how similar a data point is to its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to +1:

+1: The data point is well-clustered and far from neighboring clusters.
0: The data point is on or very close to the boundary between two clusters.
-1: The data point is likely misclassified, being closer to another cluster than its own.
A higher silhouette score indicates better-defined clusters. It is commonly used to assess the optimal number of clusters.