NEW HORIZON
COLLEGE OF ENGINEERING

Module 5

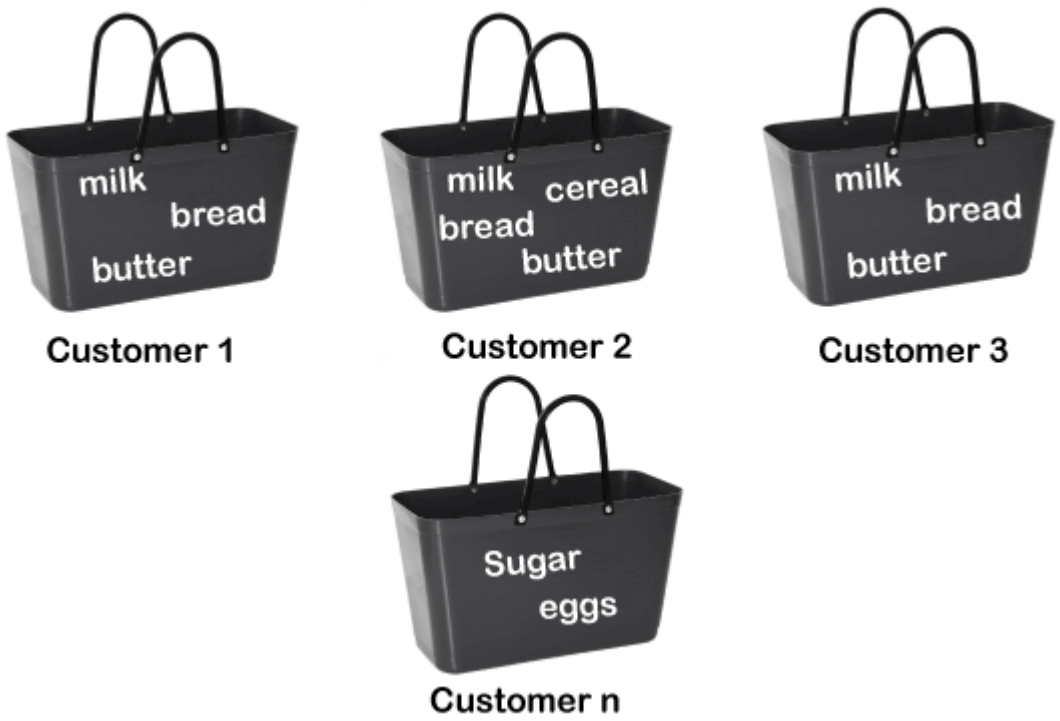
Unsupervised Algorithms

- Introduction to Rule Based learning and Association rules- Apriori Algorithm, FP-Growth. Introduction to unsupervised Algorithms- types of clustering algorithms- k-means clustering algorithms-DBSCAN clustering algorithm.

Introduction to Rule Based learning and Association rules

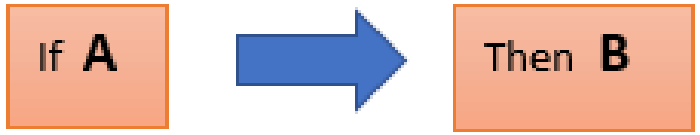
- Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable.
- It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.
- Association rule mining finds interesting associations and relationships among large sets of data items.
- This rule shows how frequently a itemset occurs in a transaction. A typical example is a ***Market Based Analysis.***

- For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby.



Association Rule Learning

- Association rule learning works on the concept of If and Else Statement, such as if A then B.



Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known as single cardinality.

Association Rule Learning

To measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- ***Support***
- ***Confidence***
- ***Lift***

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as

$$\text{Supp}(X) = \frac{Freq(X)}{T}$$

- Confidence
- Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

- Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

- It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:
- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

- Association rule learning can be divided into three types of algorithms:

1.Apriori

2.Eclat

3.F-P Growth Algorithm

- **The Apriori Algorithm: Basics**

- The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

-

- Key Concepts :

- Frequent Itemsets: The sets of item which has minimumsupport (denoted by Li for ith-Itemset).
- Apriori Property: Any subset of frequent itemset must befrequent.
- Join Operation: To find Lk , a set of candidate k-itemsetsis generated by joining Lk-1 with itself.

The Apriori Algorithm in aNutshell

- Find the *frequent itemsets*: the sets of items that haveminimum support
  - **A subset of a frequent itemset must also be afrequent itemset**
    - i.e., if {*AB*} is a frequent itemset, both {*A*} and {*B*}should be a frequent itemset
  - **Iteratively find frequent itemsets with cardinalityfrom 1 to *k* (*k*-itemset)**
- Use the frequent itemsets to generate association rules

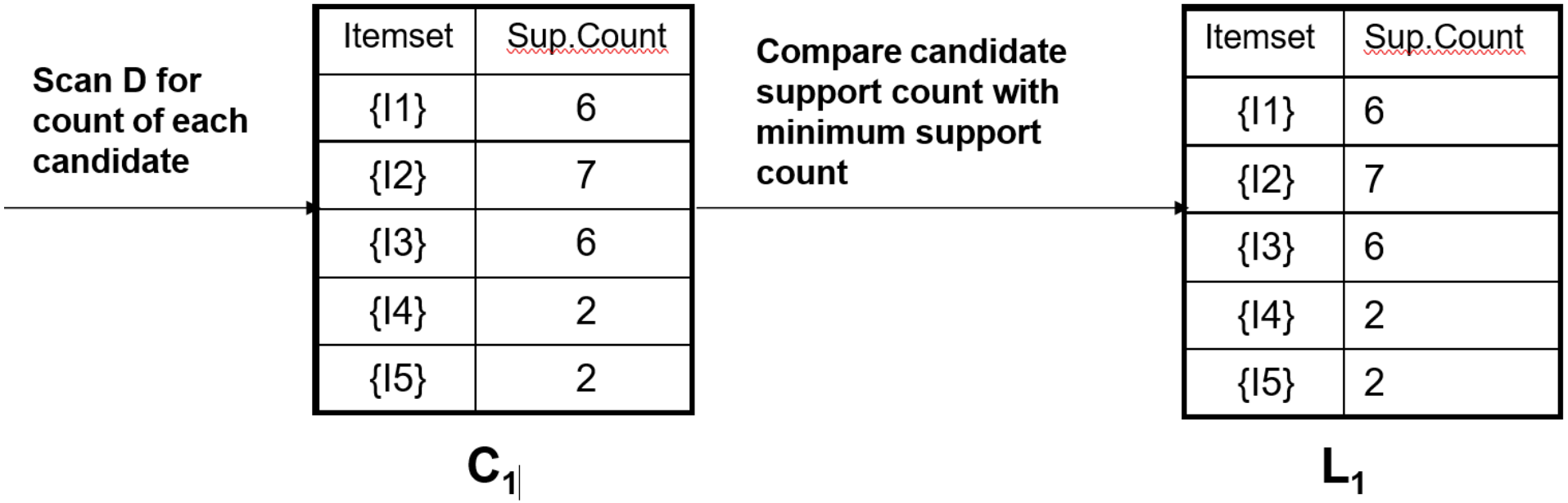
The Apriori Algorithm : Pseudocode

- Join Step: Ck is generated by joining Lk-1with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be asubset of a frequent k-itemset
- Pseudo-code:
    - *Ck*: Candidate itemset of size k
    - *Lk* : frequent itemset of size k
    - L_1 = {frequent items};
    - **for** (*k* = 1; *Lk* !=Æ; *k*++) **do begin**
        - *Ck+1* = candidates generated from *Lk*;
        - **for each** transaction *t* in database do
            - increment the count of all candidates in *Ck+1*
            - that are contained in *t*
        - *Lk+1* = candidates in *Ck+1* with min_support
        - **end return** È*k* *Lk*;

The Apriori Algorithm: Example

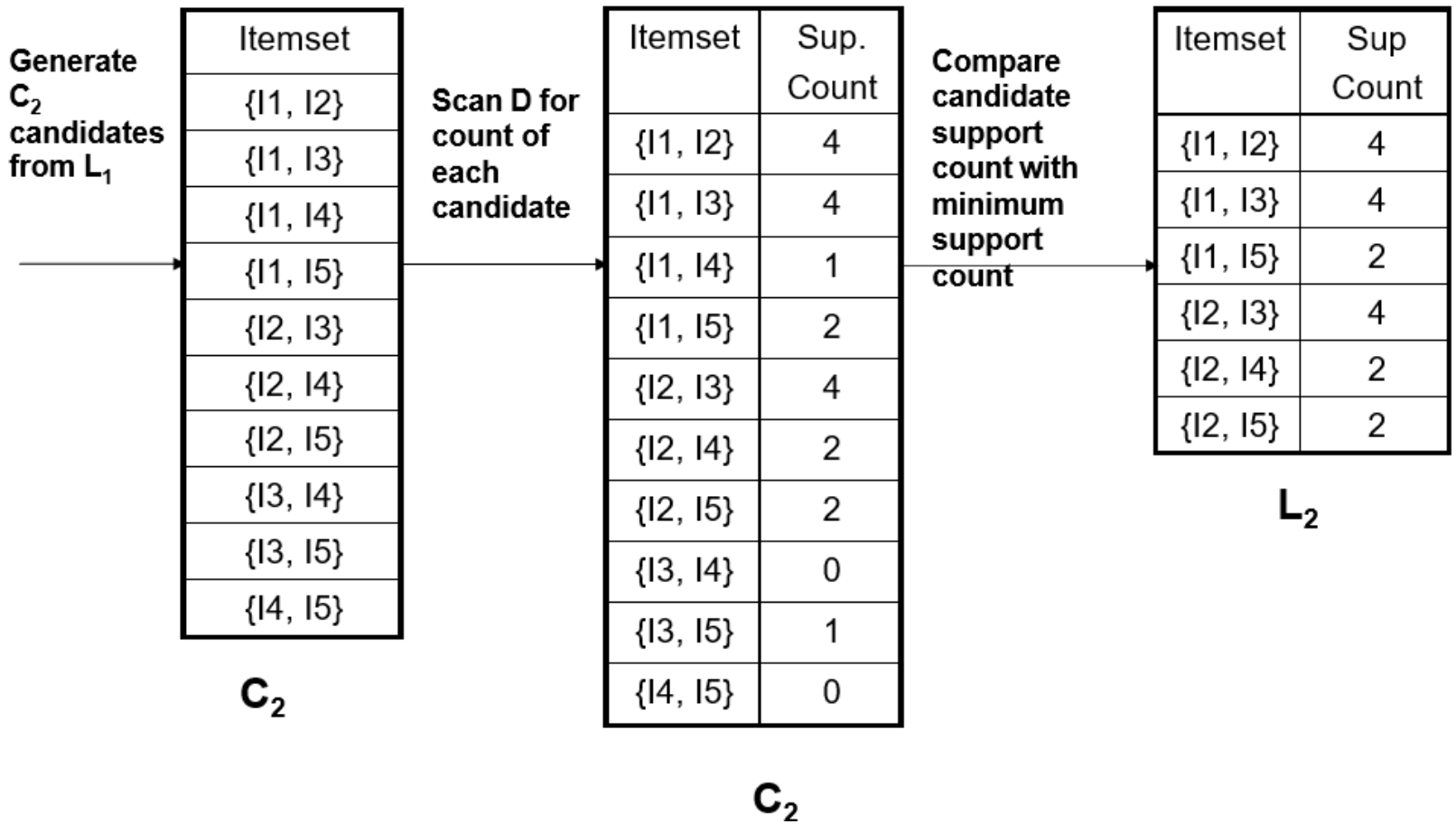
- Consider a database, D , consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3



- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the setof candidate.

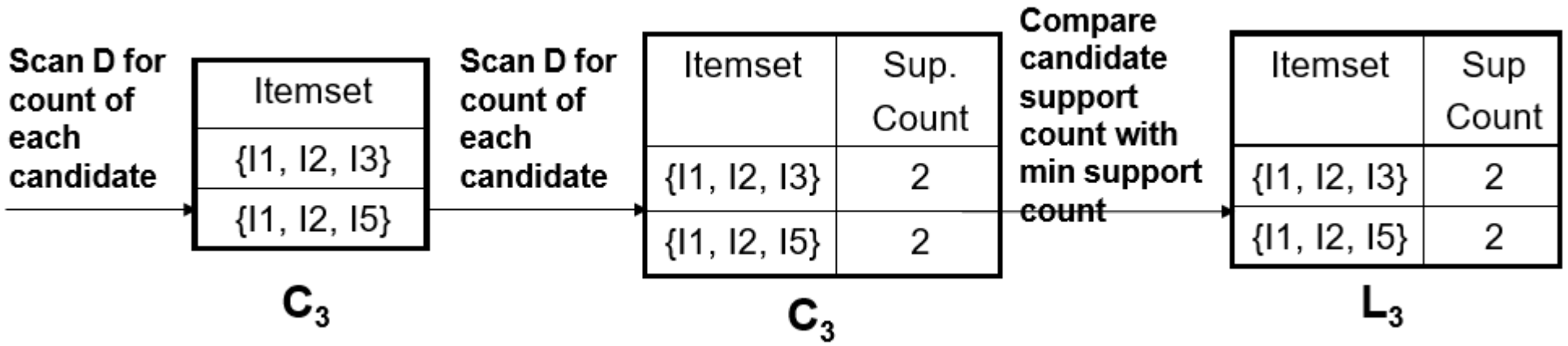
Step 2: Generating 2-itemset Frequent Pattern



Step 2: Generating 2-itemset Frequent Pattern

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses L_1 *Join* L_1 to generate a candidate set of2-itemsets, C_2.
- Next, the transactions in D are scanned and the supportcount for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined,consisting of those candidate 2-itemsets in C_2 having minimum support.
- Note: We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern



- The generation of the set of candidate 3-itemsets, C_3 , involves use ofthe Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3,I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 3: Generating 3-itemset Frequent Pattern

- Based on the Apriori property that all subsets of a frequent itemset mustalso be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example , lets take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, Wewill keep {I1, I2, I3} in C_3.
- Lets take another example of {I2, I3, I5} which shows how the pruning isperformed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L_2 and hence it is not frequent violatingApriori Property. Thus We will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members ofresult of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L_3, consistingof those candidates 3-itemsets in C_3 having minimum support.

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}}is not frequent.
- Thus, $C_4 = \varphi$, and algorithm terminates, having foundall of the frequent items. This completes our Apriori Algorithm.
- What's Next ?
  - These frequent itemsets will be used to generate strong association rules (where strong association rules satisfyboth minimum support & minimum confidence).

Step 5: Generating Association Rules from FrequentItemsets

- Procedure:
  - For each frequent itemset ***"l",*** generate all nonempty subsetsof ***l.***
  - For every nonempty subset ***s*** of ***l***, output the rule **"s → (l-s)"** if
    - **support_count(l) / support_count(s) >= min_conf** wheremin_conf is minimum confidence threshold.
-
- Back To Example:
  - We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3},
  - {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

– Lets take ***l*** = {I1,I2,I5}.

– Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules fromFrequent Itemsets

- **Let minimum confidence threshold is , say 70%.**
- The resulting association rules are shown below,each listed with its confidence.
  - – R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - – R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - – R3: I2 ^ I5 → I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

Step 5: Generating Association Rules fromFrequent Itemsets

- **R4: I1 → I2 ^ I5**
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- **– R5: I2 → I1 ^ I5**
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- **– R6: I5 → I1 ^ I2**
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.
    - In this way, We have found three strong association rules.

Methods to Improve Apriori's Efficiency

- Hash-based itemset counting: A *k*-itemset whose correspondinghashing bucket count is below the threshold cannot be frequent.
- Transaction reduction: A transaction that does not contain anyfrequent k-itemset is useless in subsequent scans.
- Partitioning: Any itemset that is potentially frequent in DB must befrequent in at least one of the partitions of DB.
- Sampling: mining on a subset of given data, lower support threshold
  - + a method to determine the completeness.
- Dynamic itemset counting: add new candidate itemsets only whenall of their subsets are estimated to be frequent.

Mining Frequent Patterns Without CandidateGeneration

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure

– highly condensed, but complete for frequent patternmining
– avoid costly database scans

- Develop an efficient, FP-tree-based frequent patternmining method

– **A divide-and-conquer methodology: decomposemining tasks into smaller ones**
– Avoid candidate generation: sub-database testonly!

FP-Growth Method : An Example

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

Consider the same previous example of a database, D , consisting of 9 transactions.

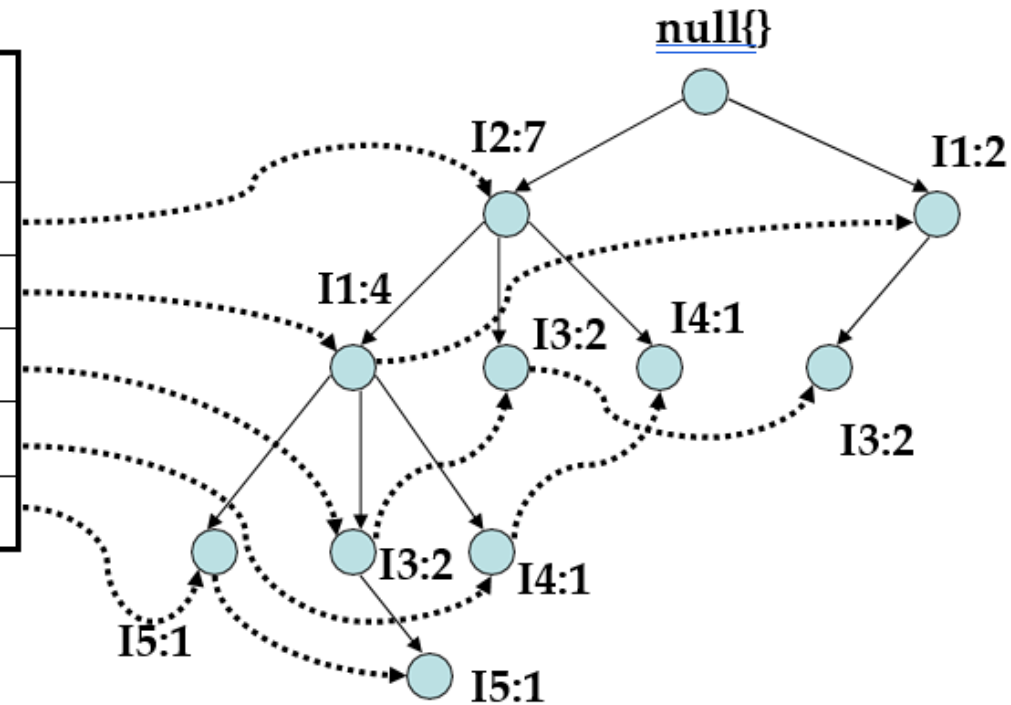
- –Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- –The first scan of database is same as Apriori, which derives the set of 1-itemsets & their support counts.
- –The set of frequent items is sorted in the order of descending support count.
- –The resulting set is denoted as L = {I2:7, I1:6, I3:6, I4:2, I5:2}

FP-Growth Method: Construction of FP-Tree

- First, create the root of the tree, labeled with “null”.
- Scan the database D a second time. (First time we scanned it tocreate 1-itemset and then L).
- The items in each transaction are processed in L order (i.e. sortedorder).
- A branch is created for each transaction with items having theirsupport count separated by colon.
- Whenever the same node is encountered in another transaction, wejust increment the support count of the common node or Prefix.
- To facilitate tree traversal, an item header table is built so that eachitem points to its occurrences in the tree via a chain of node-links.
- Now, The problem of mining frequent patterns in database istransformed to that of mining the FP-Tree.

FP-Growth Method: Construction of FP-Tree

Item Id	Sup Count	Node-link
I2	7	
I1	6	
I3	6	
I4	2	
I5	2	



An FP-Tree that registers compressed, frequent pattern information

Mining the FP-Tree by Creating Conditional (sub)pattern bases

- Steps:

1. Start from each frequent length-1 pattern (as an initial suffixpattern).
2. Construct its conditional pattern base which consists of theset of prefix paths in the FP-Tree co-occurring with suffix pattern.
3. Then, Construct its conditional FP-Tree & perform miningon such a tree.
4. The pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from aconditional FP-Tree.

- The union of all frequent patterns (generated by step 4)gives the required frequent itemset

FP-Tree Example Continued

- Now, Following the above mentioned steps:
  - Lets start from I5. The I5 is involved in 2 branches namely {I2 I1 I5: 1} and {I2I1 I3 I5: 1}.
  - Therefore considering I5 as suffix, its 2 corresponding prefix paths would be
    - {I2 I1: 1} and {I2 I1 I3: 1}, which forms its conditional pattern base.

Item	Conditional patternbase	Conditional FP-Tree	Frequent pattern generated
I5	{(I2 I1: 1),(I2 I1 I3: 1)}	<I2:2 , I1:2>	I2 I5:2, I1 I5:2, I2 I1 I5: 2
I4	{(I2 I1: 1),(I2: 1)}	<I2: 2>	I2 I4: 2
I3	{(I2 I1: 1),(I2: 2), (I1: 2)}	<I2: 4, I1: 2>,<I 1:2>	I2 I3:4, I1, I3: 2 , I2 I1 I3:2
I2	{(I2: 4)}	<I2: 4>	I2 I1: 4

Mining the FP-Tree by creating conditional (sub) pattern bases

FP-Tree Example Continued

- Out of these, Only I1 & I2 is selected in the conditional FP-Tree because I3 is not satisfying the minimum support count.
  - For I1 , support count in conditional pattern base = 1 + 1 = 2 For I2 , support count in conditional pattern base = 1 + 1 = 2 For I3, support count in conditional pattern base = 1
  - Thus support count for I3 is less than required min_sup which is 2here.
- Now , We have conditional FP-Tree with us.
- All frequent pattern corresponding to suffix I5 are generated by considering all possible combinations of I5 and conditional FP-Tree.
- The same procedure is applied to suffixes I4, I3 and I1.
- Note: I2 is not taken into consideration for suffix because it doesn’thave any prefix at all.

Why Frequent Pattern Growth Fast ?

- Performance study shows
  - – FP-growth is an order of magnitude faster than Apriori,and is also faster than tree-projection
- Reasoning

– No candidate generation, no candidate test

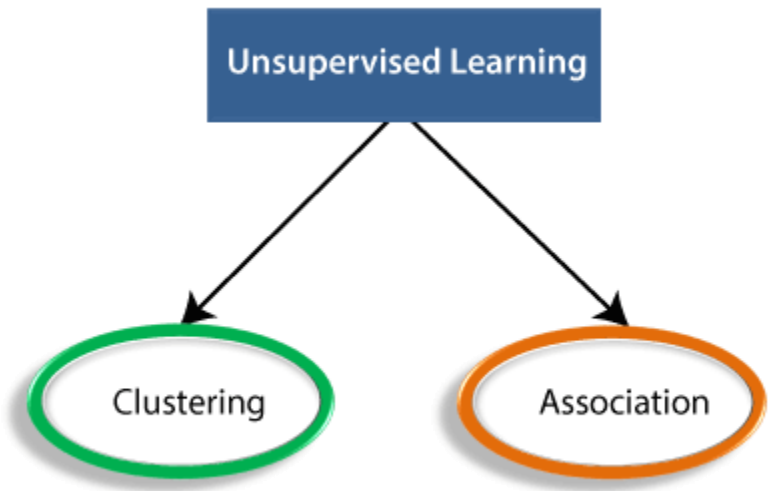
– Use compact data structure

– Eliminate repeated database scan

– Basic operation is counting and FP-tree building

Introduction to unsupervised Algorithms

- Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to ***analyze and cluster unlabeled datasets***.
- These algorithms *discover hidden patterns or data groupings* without the need for human intervention.
- Its ability to *discover similarities and differences in information* make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.


Unsupervised Learning
Clustering
Association

K Maeans Clustering

NEW HORIZON
COLLEGE OF ENGINEERING

K-Means Clustering Algorithm

- K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning.
- It groups the *unlabeled dataset into different clusters*.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It allows us to *cluster the data into different groups* and a convenient way to discover the categories of groups in the unlabeled dataset on its own *without the need for any training*.

- The k-means clustering algorithm mainly performs two tasks:
  - Determines the best value for K center points or centroids by an iterative process.
  - Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

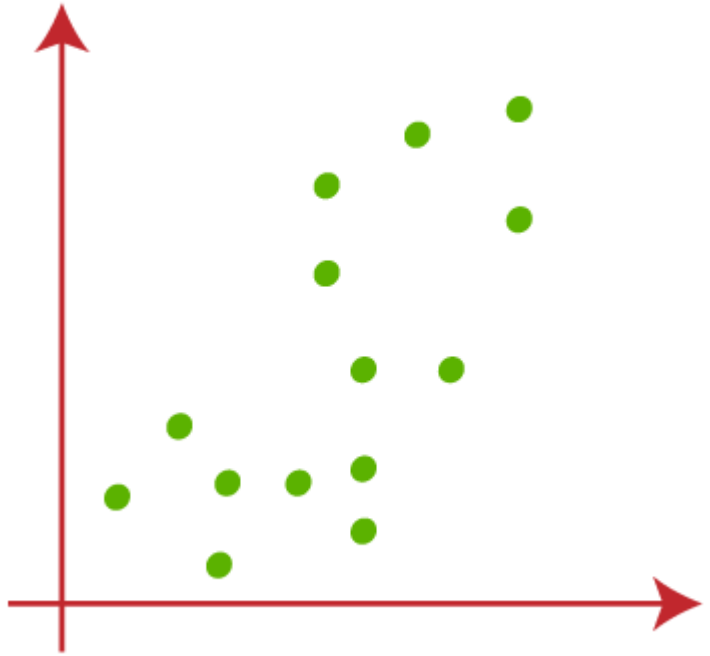
How does the K-Means Algorithm Work?

- The working of the K-Means algorithm is explained in the below steps:
- **Step-01:**
- Choose the number of clusters K.
- **Step-02:**
- Randomly select any K data points as cluster centers.
- Select cluster centers in such a way that they are as farther as possible from each other.
- **Step-03:**
- Calculate the distance between each data point and each cluster center.
- The distance may be calculated either by using given distance function or by using Euclidean distance formula.
-

K-Means Algorithm

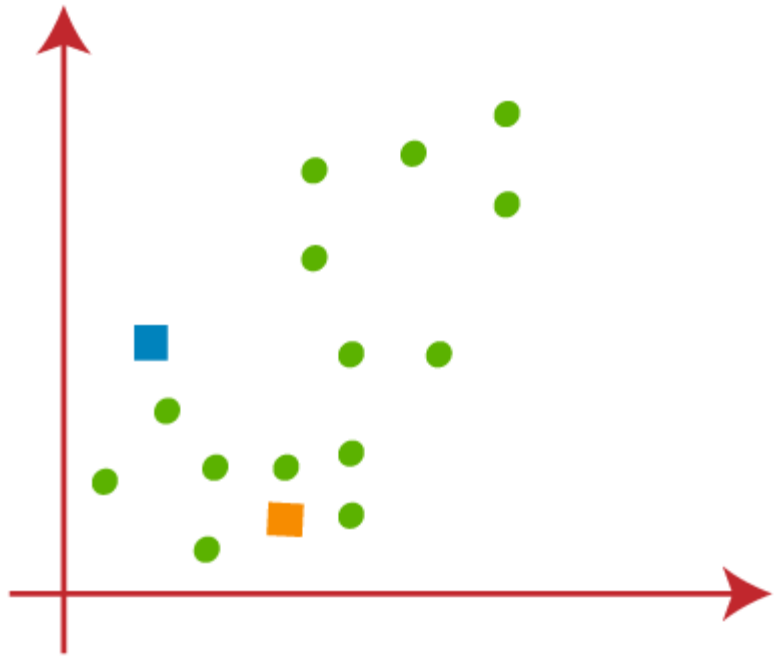
- **Step-04:**
- Assign each data point to some cluster.
- A data point is assigned to that cluster whose center is nearest to that data point.
- **Step-05:**
- Re-compute the center of newly formed clusters.
- The center of a cluster is computed by taking mean of all the data points contained in that cluster.
- **Step-06:**
- Keep repeating the procedure from Step-03 to Step-05 until any of the following stopping criteria is met-
  - Center of newly formed clusters do not change
  - Data points remain present in the same cluster
  - Maximum number of iterations are reached

- Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

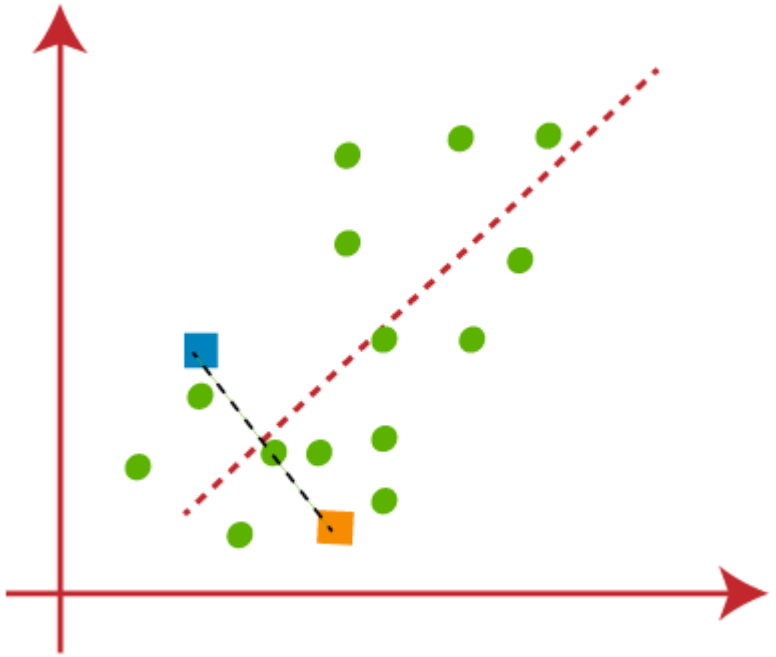


•Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.

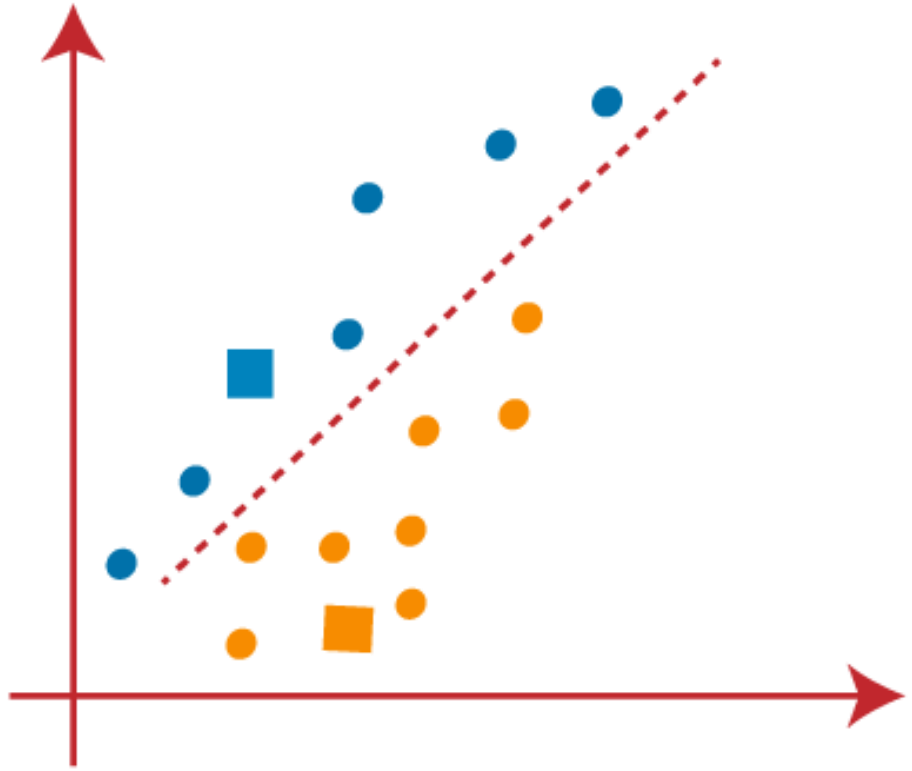
- We need to choose some *random k points or centroid* to form the cluster.
- These points can be either the points from the dataset or any other point.
- So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:



- Now we will assign each data point of the scatter plot to its closest K-point or centroid.
- We will compute it by applying some mathematics that we have studied to calculate the ***distance between two points.***
- So, we will draw a median between both the centroids. Consider the below image:

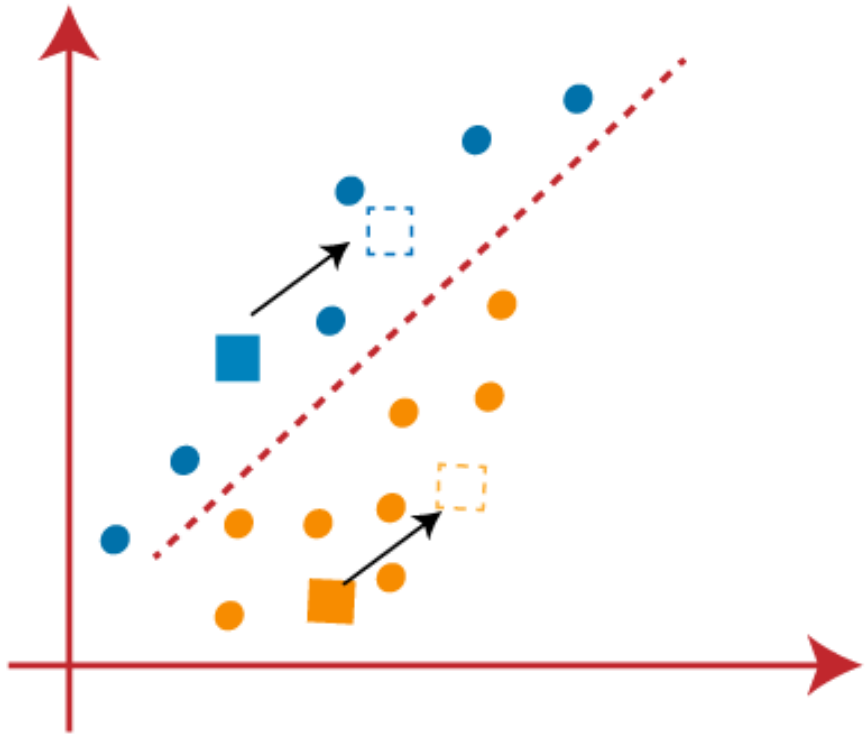


- From the image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid.
- Let's color them as blue and yellow for clear visualization.

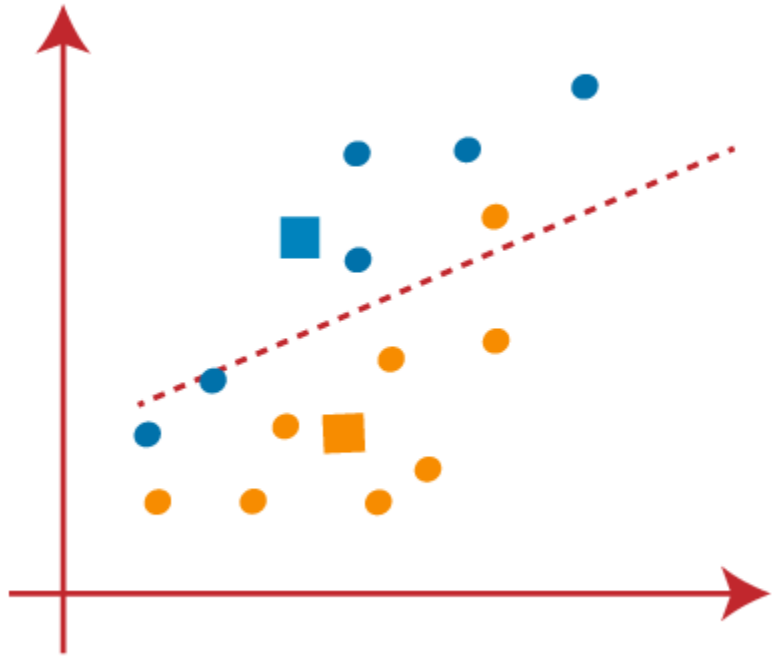


- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**.

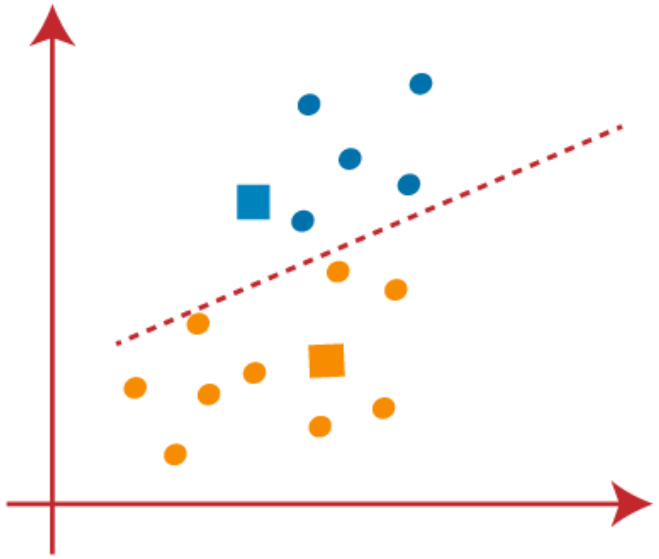
- To choose the new centroids, we will compute the center of gravity of these centroids(*mean* or average point.)



- Next, we will reassign each datapoint to the new centroid.
- For this, we will repeat the same process of finding a median line. The median will be like below image:

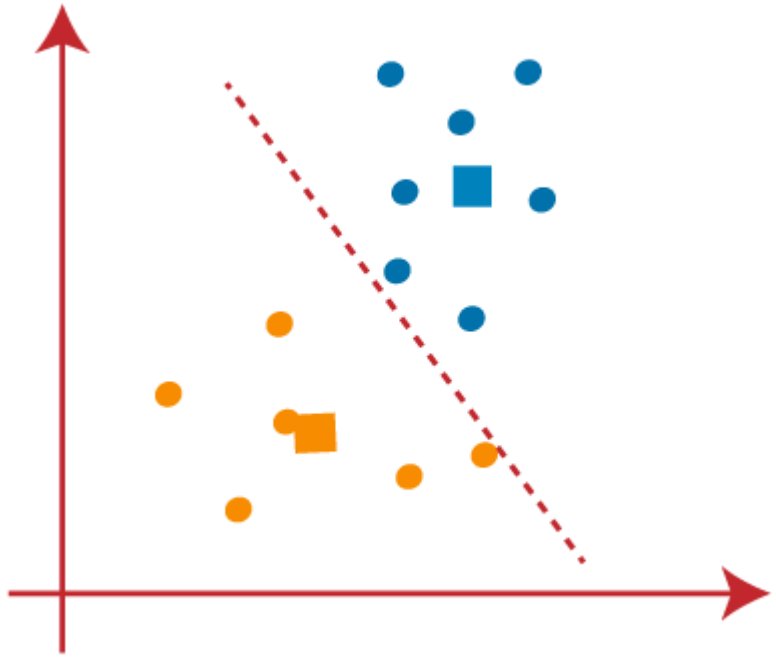


- From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

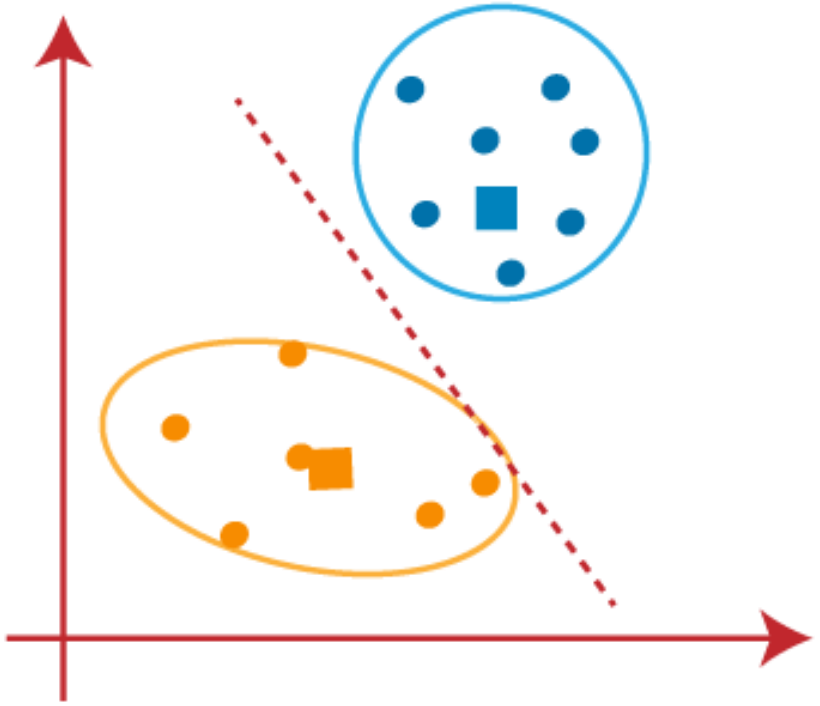


- As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:

- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:

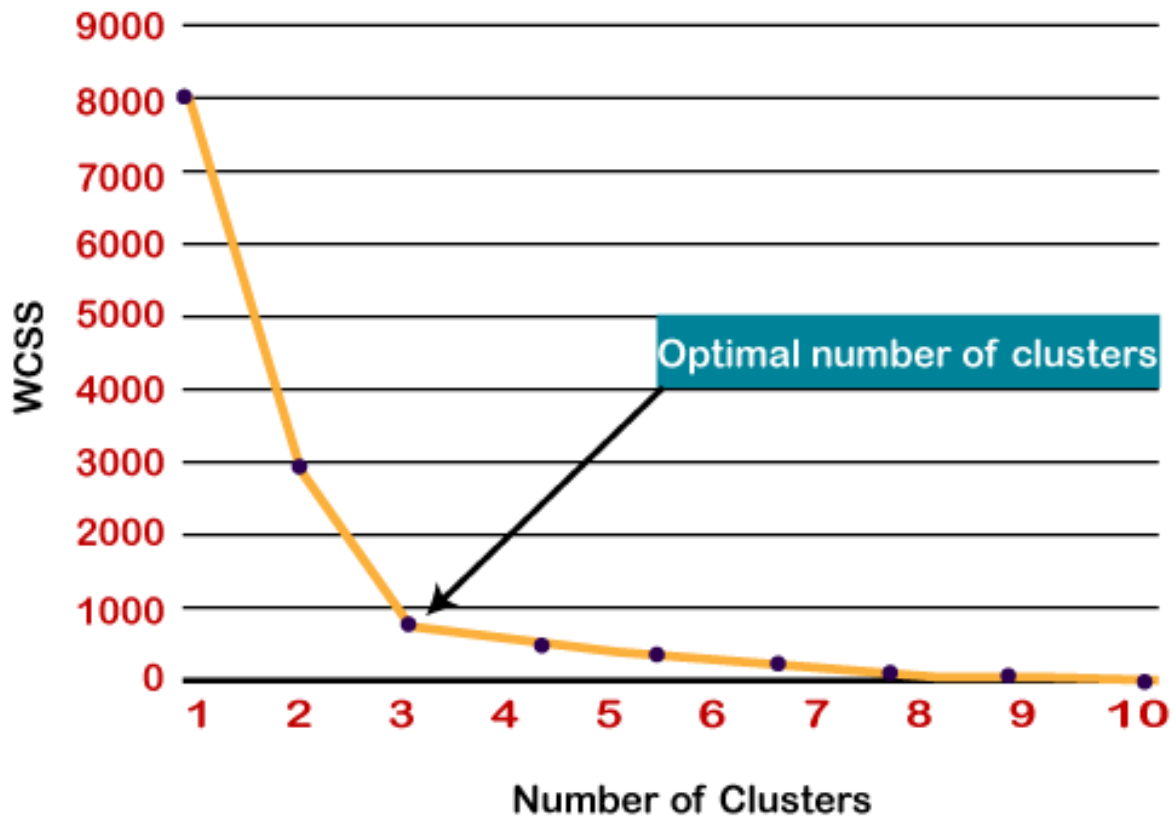


How to choose the value of "K number of clusters" in K-means Clustering?

- Elbow Method
- The Elbow method is one of the most popular ways to find the optimal number of clusters.
- This method uses the concept of WCSS value.
- **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.
- The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS= \sum_{P_i \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{P_i \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{P_i \text{ in CLuster3}} distance(P_i\ C_3)^2$$

- $\sum_{Pi \text{ in Cluster1}}$ distance$(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.
- To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.
- To find the optimal value of clusters, the elbow method follows the below steps:
- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
WCSS
Optimal number of clusters
1
2
3
4
5
6
7
8
9
10
Number of Clusters

- The steps to be followed for the implementation are given below:
- **Data Pre-processing**
- **Finding the optimal number of clusters using the elbow method**
- **Training the K-means algorithm on the training dataset**
- **Visualizing the clusters**

Problem on K-Means Clustering

- Cluster the following eight points (with (x, y) representing locations) into three clusters:
- A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)
- Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
- The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as
- $P(a, b) = |x2 – x1| + |y2 – y1|$
- Use K-Means Algorithm to find the three cluster centers after the second iteration.

- **Iteration-01:**
-
- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.
-
- The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-
-
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
-
- P(A1, C1)
- = |x2 – x1| + |y2 – y1|
- = |2 – 2| + |10 – 10|
- = 0

- **Calculating Distance Between A1(2, 10) and C2(5, 8)-**
-
- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |5 – 2| + |8 – 10|
- = 3 + 2
- = 5
-
- **Calculating Distance Between A1(2, 10) and C3(1, 2)-**
-
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1 – 2| + |2 – 10|
- = 1 + 8
- = 9
-
- In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

- From here, New clusters are-
- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)
- **Cluster-03:**
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)

- **For Cluster-01:**
-
- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.
-
- **For Cluster-02:**
-
- Center of Cluster-02
- = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
- = (6, 6)
-
- **For Cluster-03:**
-
- Center of Cluster-03
- = ((2 + 1)/2, (5 + 2)/2)
- = (1.5, 3.5)
-

- **Iteration-02:**
- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.
- The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
- P(A1, C1)
- = |x2 – x1| + |y2 – y1|
- = |2 – 2| + |10 – 10|
- = 0

- **Calculating Distance Between A1(2, 10) and C2(6, 6)-**

- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |6 – 2| + |6 – 10|
- = 4 + 4
- = 8
-
- **Calculating Distance Between A1(2, 10) and C3(1.5, 3.5)-**
-
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1.5 – 2| + |3.5 – 10|
- = 0.5 + 6.5
- = 7
- In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

- From here, New clusters are-
- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- A8(4, 9)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- **Cluster-03:**
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)

- **For Cluster-01:**
-
- Center of Cluster-01
- = ((2 + 4)/2, (10 + 9)/2)
- = (3, 9.5)
-
- **For Cluster-02:**
-
- Center of Cluster-02
- = ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)
- = (6.5, 5.25)
-
- **For Cluster-03:**
-
- Center of Cluster-03
- = ((2 + 1)/2, (5 + 2)/2)
- = (1.5, 3.5)

- After second iteration, the center of the three clusters are-
- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)

- Using K means clustering algorithm form two clusters for given data.- Problem 1

Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

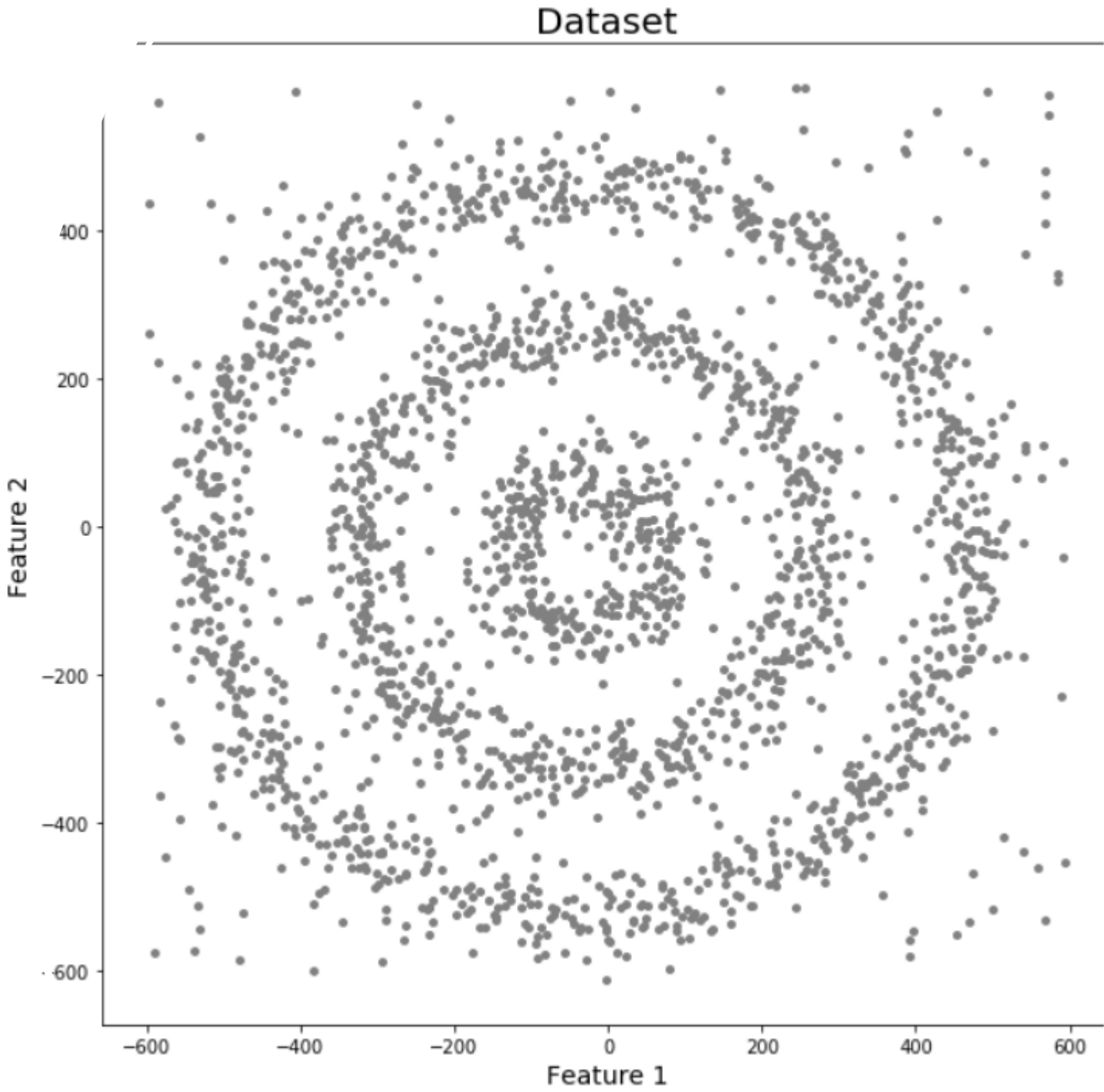
consider first two data points of our data and assign them as a centroid

Problem 2

- Lets consider we have cluster points P1(1,3) , P2(2,2) , P3(5,8) , P4(8,5) , P5(3,9) , P6(10,7) , P7(3,3) , P8(9,4) , P9(3,7).
- First, we take our K value as 3 and we assume that our Initial cluster centers are P7(3,3), P9(3,7), P8(9,4) as C1, C2, C3. We will find out the new centroids after 2 iterations for the above data points.

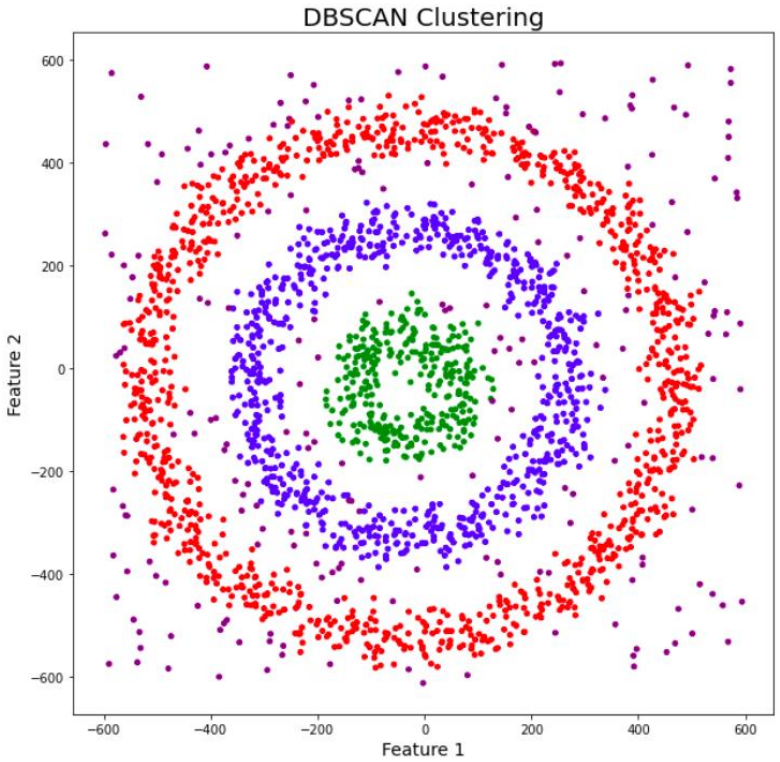
DBSCAN Clustering

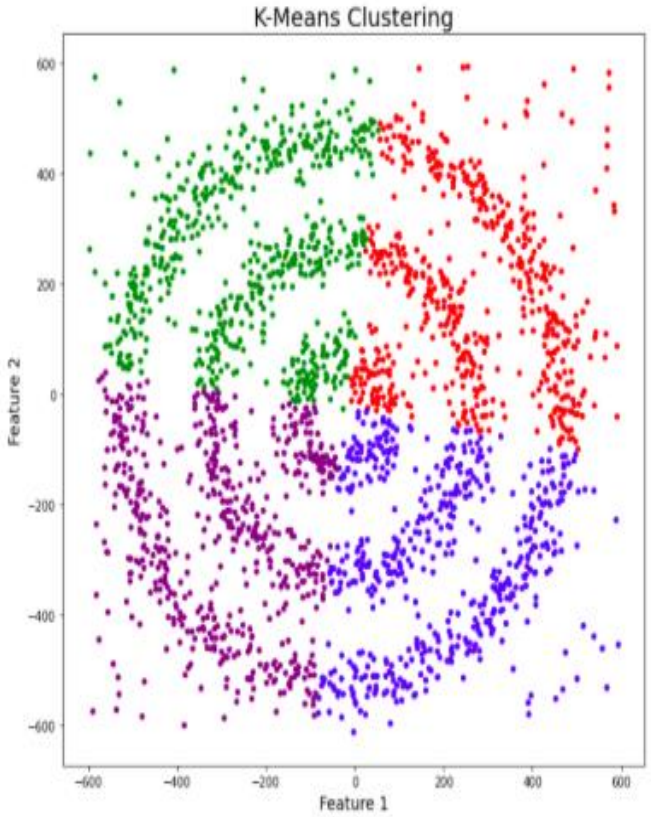
- K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering.
- we have data points densely present in the form of concentric circles:

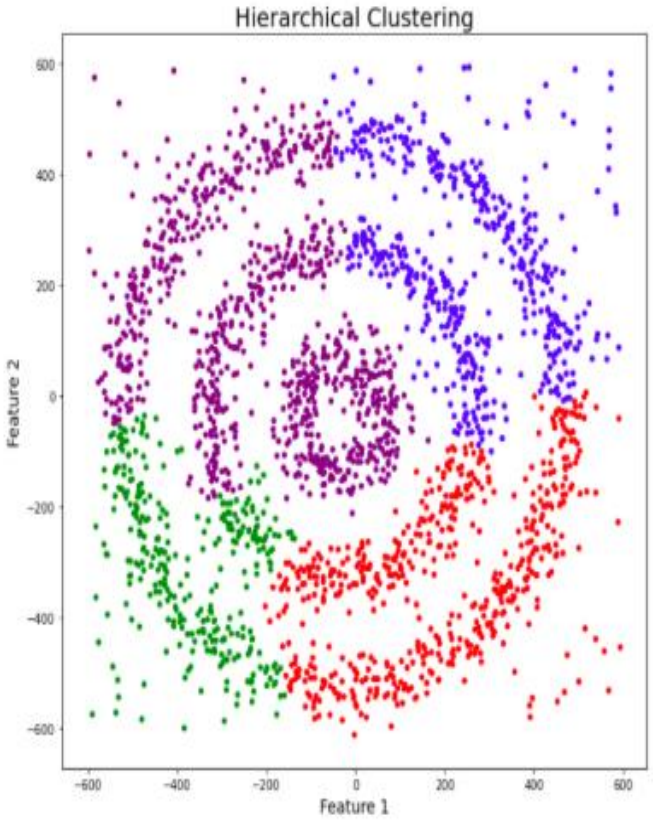


DBSCAN Clustering

- This data contains noise too, therefore, noise is taken as a different cluster which is represented by the purple color. Both of K means, Hierarchical failed to cluster the data points. Also, they were not able to properly detect the noise present







DBSCAN Clustering

- **DBSCAN** stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.
- It was proposed by Martin Ester et al. in 1996. DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density.
- It groups 'densely grouped' data points into a single cluster. It can identify clusters in large spatial datasets by looking at the local density of the data points. **The most exciting feature of DBSCAN clustering is that it is robust to outliers**. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

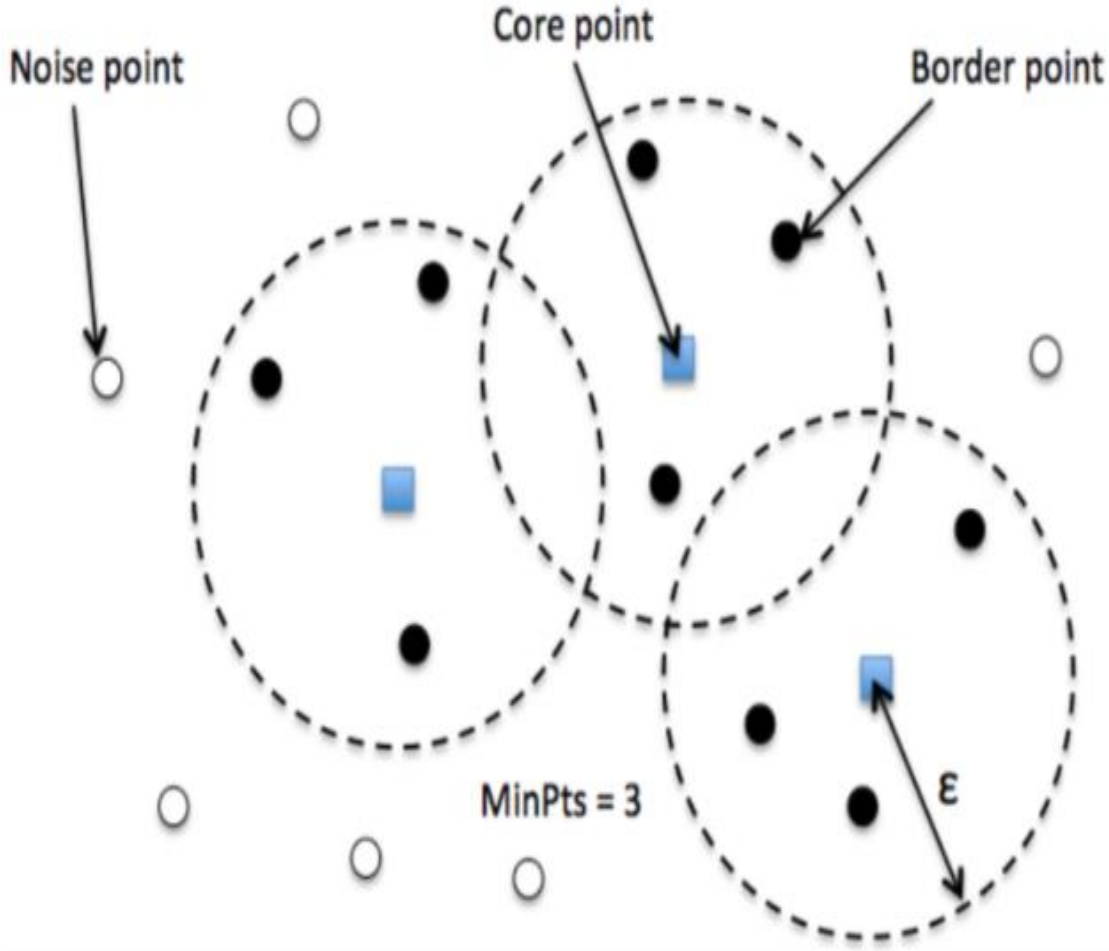
parameters required

1. **eps (ε):** This parameter defines the radius of a neighborhood around a data point. Points within this distance are considered neighbors of the central point.
2. **minPts:** This parameter represents the minimum number of points required within the ε-neighborhood of a point to classify it as a core point. A core point is considered to be dense enough to be part of a cluster.

- These parameters work together to identify high-density regions in your data. Points with enough neighbors within the specified distance (core points) are grouped together as clusters, while points with insufficient neighbors are classified as noise

DBSCAN

- DBSCAN creates a circle of *epsilon* radius around every data point and classifies them into **Core** point, **Border** point, and **Noise**.
- A data point is a **Core** point if the circle around it contains at least '*minPoints'* number of points.
- If the number of points is less than *minPoints*, then it is classified as **Border** Point, and if there are no other data points around any data point within *epsilon* radius, then it treated as **Noise**.



DBCAN

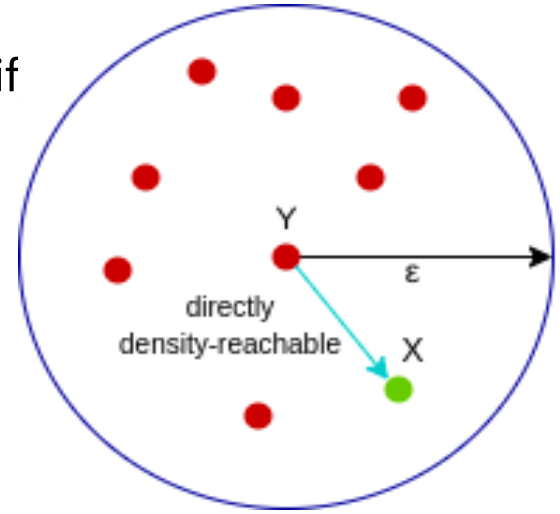
- The above figure shows us a cluster created by DBCAN with *minPoints = 3*. Here, we draw a circle of equal radius *epsilon* around every data point. These two parameters help in creating spatial clusters.
- All the data points with at least 3 points in the circle including itself are considered as **Core** points represented by red color. All the data points with less than 3 but greater than 1 point in the circle including itself are considered as **Border** points. They are represented by yellow color. Finally, data points with no point other than itself present inside the circle are considered as **Noise** represented by the purple color.
- For locating data points in space, DBSCAN uses **Euclidean distance**, although other methods can also be used (like great circle distance for geographical data). It also needs to scan through the entire dataset once, whereas in other algorithms we have to do it multiple times.

Reachability and Connectivity

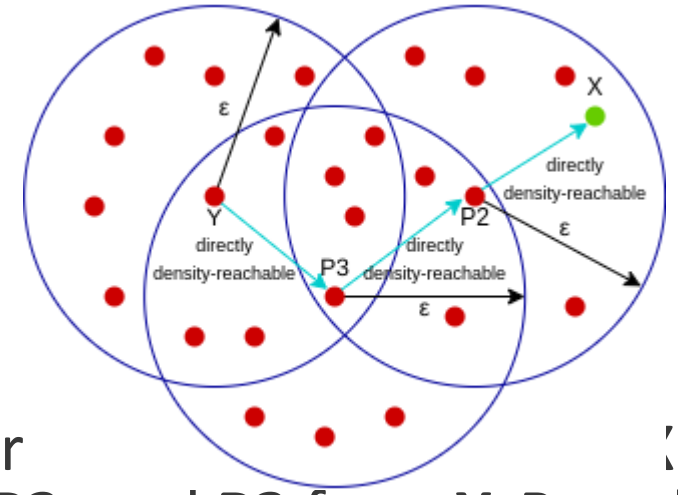
- Reachability states if a data point can be accessed from another data point directly or indirectly, whereas Connectivity states whether two data points belong to the same cluster or not. In terms of reachability and connectivity, two points in DBSCAN can be referred to as:
- **Directly Density-Reachable**
- **Density-Reachable**
- **Density-Connected**
- A point **X** is **directly density-reachable** from point **Y** w.r.t *epsilon, minPoints* if

1. **X** belongs to the neighborhood of **Y**, i.e, *dist(X, Y) <= epsilon*
2. **Y** is a core point

Here, **X** is directly density-reachable from Y**,** but vice versa is not valid.

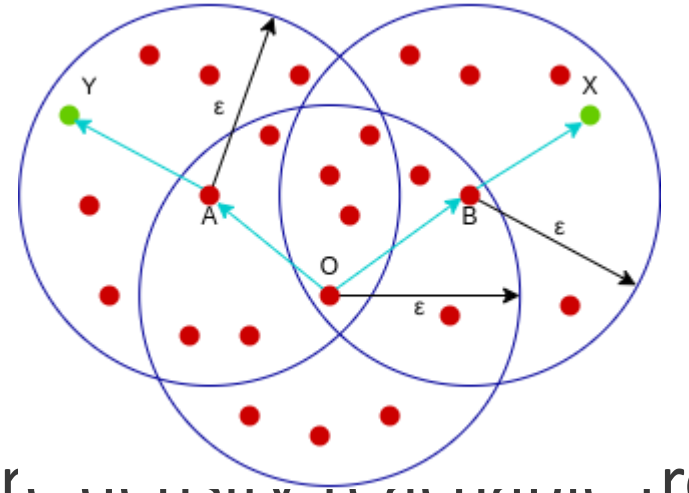


- A point **X** is **density-reachable** from point **Y** w.r.t *epsilon, minPoints* if there is a chain of points p1, p2, p3, …, pn and p1=**X** and pn=**Y** such that pi+1 is directly density-reachable from pi.



- Here, **X** is density-r being directly density-reachable from **P2**, **P2** from **P3,** and **P3** from **Y.** But, the inverse of this is not valid.

- A point **X** is **density-connected** from point **Y** w.r.t *epsilon and minPoints* if there exists a point **O** such that both **X** and **Y** are density-reachable from **O** w.r.t to *epsilon and minPoints.*



- Here, both **X** and **Y** are density-reachable from **O**, therefore, we can say that **X** is density-connected from **Y**.

Working of Density-Based Clustering

- Suppose a set of objects is denoted by D', we can say that an object I is directly density reachable form the object j only if it is located within the ε neighborhood of j, and j is a core object.
- An object i is density reachable form the object j with respect to ε and MinPts in a given set of objects, D' only if there is a sequence of object chains point i1,…., in, i1 = j, pn = i such that i_i + 1 is directly density reachable from i_i with respect to ε and MinPts.
- An object i is density connected object j with respect to ε and MinPts in a given set of objects, D' only if there is an object o belongs to D such that both point i and j are density reachable from o with respect to ε and MinPts.

- Cluster the following eight points (with (x, y) representing locations) into three clusters:
- A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)
- Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
- The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-
- $P(a, b) = |x2 – x1| + |y2 – y1|$

- **Solution-**
- **Iteration-01:**
- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.
- The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
- P(A1, C1)
- = |x2 – x1| + |y2 – y1|
- = |2 – 2| + |10 – 10|
- = 0

- **Calculating Distance Between A1(2, 10) and C2(5, 8)-**
- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |5 – 2| + |8 – 10|
- = 3 + 2
- = 5
- **Calculating Distance Between A1(2, 10) and C3(1, 2)-**
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1 – 2| + |2 – 10|
- = 1 + 8
- = 9

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)
- **Cluster-03:**
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)
-

- Now,
- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.
- **For Cluster-01:**
-
- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.
- **For Cluster-02:**
- Center of Cluster-02
- = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
- = (6, 6)
- **For Cluster-03:**
- Center of Cluster-03
- = ((2 + 1)/2, (5 + 2)/2)
- = (1.5, 3.5)

-

- **Iteration-02:**
- **Calculating Distance Between A1(2, 10) and C1(2, 10)-**
- P(A1, C1)
- = |x2 – x1| + |y2 – y1|
- = |2 – 2| + |10 – 10|
- = 0
- **Calculating Distance Between A1(2, 10) and C2(6, 6)-**
- P(A1, C2)
- = |x2 – x1| + |y2 – y1|
- = |6 – 2| + |6 – 10|
- = 4 + 4
- = 8
- **Calculating Distance Between A1(2, 10) and C3(1.5, 3.5)-**
- P(A1, C3)
- = |x2 – x1| + |y2 – y1|
- = |1.5 – 2| + |3.5 – 10|
- = 0.5 + 6.5
- = 7

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

- **Cluster-01:**
- First cluster contains points-
- A1(2, 10)
- A8(4, 9)
- **Cluster-02:**
- Second cluster contains points-
- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)

- **Cluster-03:**
-
- Third cluster contains points-
- A2(2, 5)
- A7(1, 2)
- Now,
- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01:

Center of Cluster-01

= ((2 + 4)/2, (10 + 9)/2)

= (3, 9.5)

For Cluster-02:

Center of Cluster-02

= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)

= (6.5, 5.25)

For Cluster-03:

Center of Cluster-03

= ((2 + 1)/2, (5 + 2)/2)

= (1.5, 3.5)

- his is completion of Iteration-02.
-
- After second iteration, the center of the three clusters are-
- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)