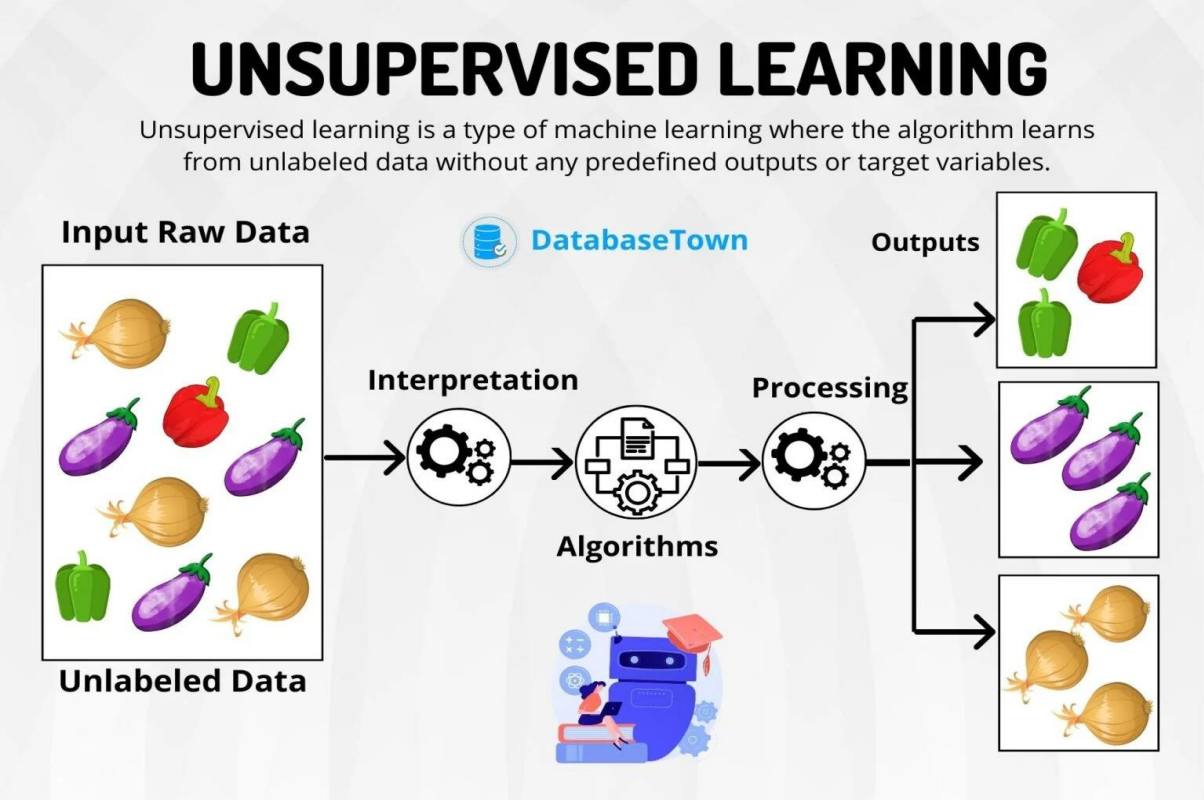
UNIT-5

Clustering and Association Rules

5.1 Unsupervised learning

- Unsupervised learning is a type of machine learning where the algorithm learns from unlabelled data without any predefined outputs or target variables.
- Unsupervised learning finds patterns, similarities, or groupings within the data to get insights and make data-driven decisions.
- It is particularly useful when dealing with large datasets.

5.2 Working of Unsupervised Learning



5.3 Distance Measure

- Distance measure determines the similarity between two elements and it influences the shape of the clusters.
- Some of the ways we can calculate distance measures include:
    - Euclidean distance measure
    - Manhattan distance measure
    - Cosine distance measure

5.3.1 Euclidean distance measure

- The Euclidean distance formula helps to find the distance of a line segment. Let us assume two points, such as (x_1, y_1) and (x_2, y_2) in the two-dimensional coordinate plane.
- Thus, the Euclidean distance formula is given by:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

- **Euclidean Distance Examples**

Find the distance between two points P(0, 4) and Q(6, 2).

Solution:

Given:

P(0, 4) = (x_1, y_1) , Q(6, 2) = (x_2, y_2)

The distance between the point PQ is

PQ = $\sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$

PQ = $\sqrt{[(6 - 0)^2 + (2 - 4)^2]}$

PQ = $\sqrt{[(6)^2 + (-2)^2]}$

PQ = $\sqrt{(36+4)}$

PQ = $\sqrt{40}$ and PQ = $2\sqrt{10}$.

Therefore, the distance between two points P(0,4) and Q(6, 2) is $2\sqrt{10}$.

5.3.2 Manhattan distance

- **Manhattan Distance**: The distance measured by calculating the sum of absolute differences of two points or vectors.
- Suppose we have to find the Manhattan distance between points A (x_1, y_1) and
- B (x_2, y_2). Then, it is given by Distance, $d = |x_1 - x_2| + |y_1 - y_2|$
- The Manhattan distance in a 2-dimensional space is given as:

$$d = |p_1 - q_1| + |p_2 - q_2|$$

- The generalized formula for an n-dimensional space is given as:

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

- **Example:** Calculate the Manhattan distance between Point P1(4,4) and P2(9,9).

- **Solution:** The Manhattan distance between P_1 and P_2 is given by,

  Given: First point, $P_1 = (4, 4)$

  Second point, $P_2 = (9, 9)$

  $P_1P_2 = |x_1 - x_2| + |y_1 - y_2|$

  $P_1P_2 = |4 - 9| + |4 - 9|$

  $= |-5| + |-5|$

  $= 5 + 5$

  $= 10$ units

Hence, the Manhattan distance between point P_1 (4,4) and P_2 (9,9) is 10 units.

5.3.3 Cosine distance measure

- Cosine similarity is a metric that measures the cosine of the angle between two vectors projected in a multi-dimensional space.
- The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.
- Cosine similarity measures how similar two vectors are, and Cosine distance measures how different they are. In real applications, it depends on the task and what function to choose. You can use cosine similarity as a loss function or as a measure for clustering

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$\left\|\vec{a}\right\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\left\|\vec{b}\right\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Where, A and B are vectors in a multidimensional space.

Since the $cos(\theta)$ value is in the range $[-1,1]$:

- -1 value will indicate strongly opposite vectors i.e. no similarity
- 0 indicates independent (or orthogonal) vectors
- 1 indicates a high similarity between the vectors

- **Cosine Distance:** Usually, people use the cosine similarity as a similarity metric between vectors. Now, the cosine distance can be defined as follows:

Cosine Distance = 1 — Cosine Similarity

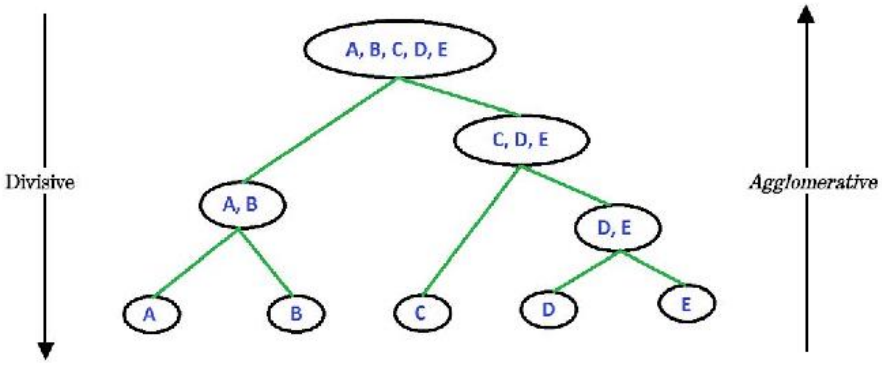
5.4. Types of Unsupervised Learning Algorithm

5.4.1 Clustering Algorithms

- Clustering is a type of unsupervised learning wherein data points are grouped into different sets based on their degree of similarity.
- The various types of clustering are:
  - Hierarchical clustering
  - Partitioning clustering
- Hierarchical clustering is further subdivided into:
  - Agglomerative clustering (bottom-up approach)
  - Divisive clustering (top-down approach)
- Partitioning clustering is further subdivided into:
  - K-Means clustering
  - Fuzzy C-Means clustering

5.4.2 Hierarchical clustering

- Hierarchical clustering is a method of grouping similar objects into clusters in a tree-like structure.
- **Agglomerative clustering:** is a bottom-up approach. We begin with each element as a separate cluster and merge them into successively more massive clusters.
- **Divisive clustering:** is a top-down approach. We begin with the whole set and proceed to divide it into successively smaller clusters.



5.5 Unsupervised Learning Algorithms:

5.5.1 K-means clustering

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.
- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm below shows the simple algorithm of K-means

Step 1: Select K points in the data space and mark them as initial centroids loop

Step 2: Assign each point in the data space to the nearest centroid to form K clusters

Step 3: Measure the distance of each point in the cluster from the centroid

Step 4: Calculate the Sum of Squared Error (SSE) to measure the quality of the clusters.

Step 5: Identify the new centroid of each cluster based on the distance between points

Step 6: Repeat Steps 2 to 5 to refine until the centroids do not change the end loop.

Choosing the Optimal Number of Clusters

The number of clusters that we choose for the algorithm shouldn't be random. Each and every cluster is formed by calculating and comparing the mean distances of each data point within a cluster from its centroid.

We can choose the right number of clusters with the help of the Within-Cluster-Sum-of-Squares (WCSS) method. WCSS stands for the sum of the squares of distances of the data points in each and every cluster from its centroid.

The main idea is to minimize the distance (e.g., euclidean distance) between the data points and the centroid of the clusters. The process is iterated until we reach a minimum value for the sum of distances.

Elbow Method

Here are the steps to follow in order to find the optimal number of clusters using the elbow method:

Step 1: Execute the K-means clustering on a given dataset for different K values (ranging from 1-10).

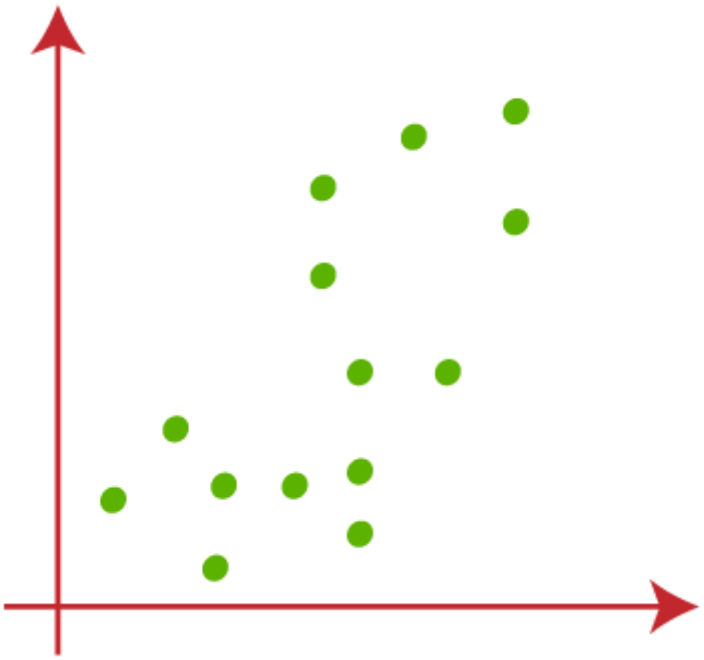
Step 2: For each value of K, calculate the WCSS value.

Step 3: Plot a graph/curve between WCSS values and the respective number of clusters K.

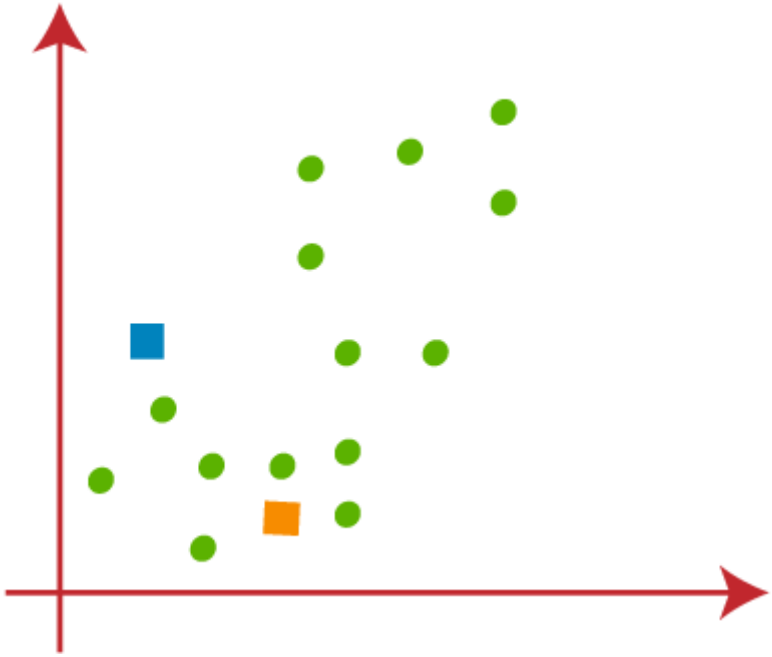
Step 4: The sharp point of bend or a point (looking like an elbow joint) of the plot, like an arm, will be considered as the best/optimal value of K.

Let's understand the above steps by considering the visual plots:

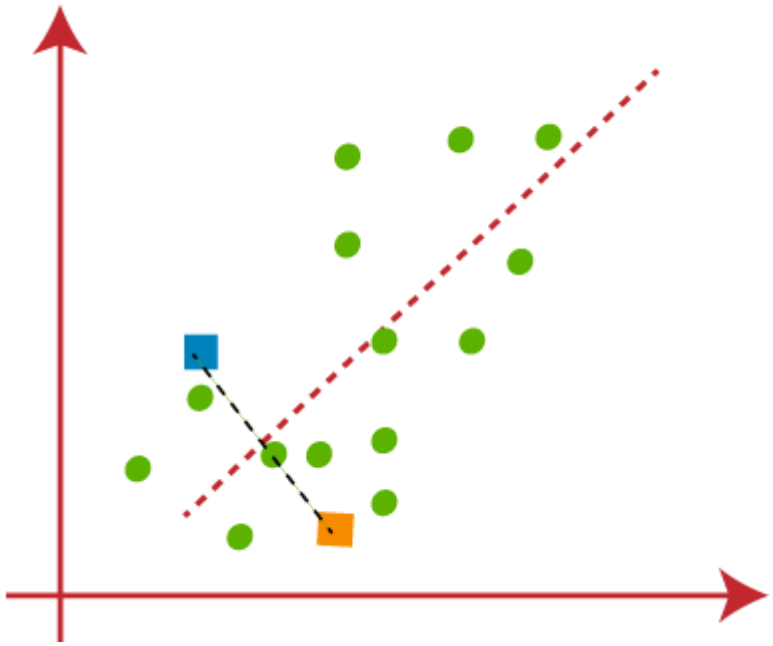
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



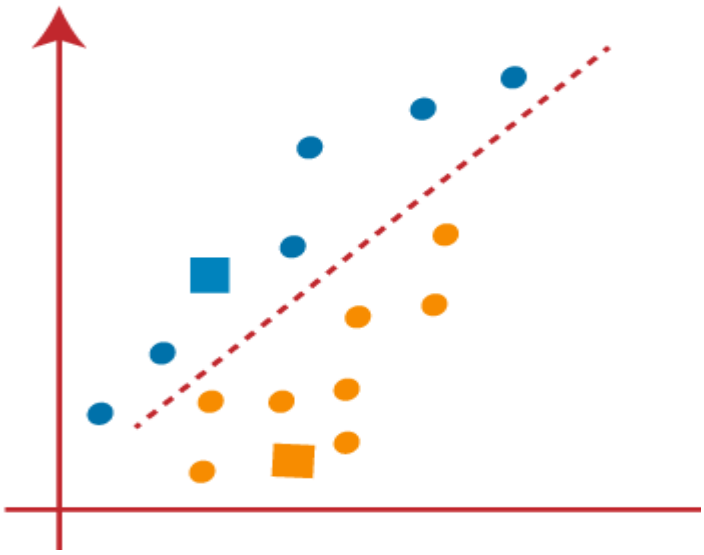
- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
- We need to choose some random k points or centroid to form the cluster. These points can be either the points 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:



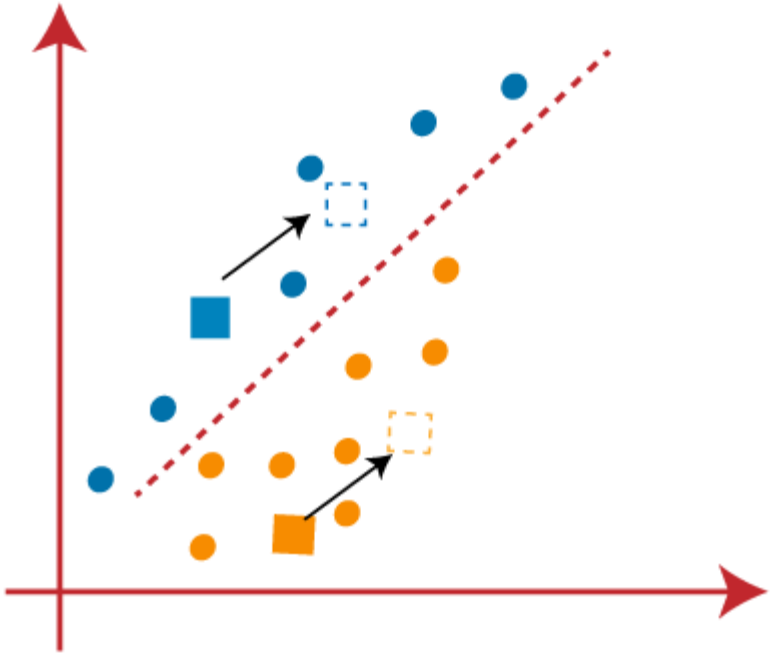
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



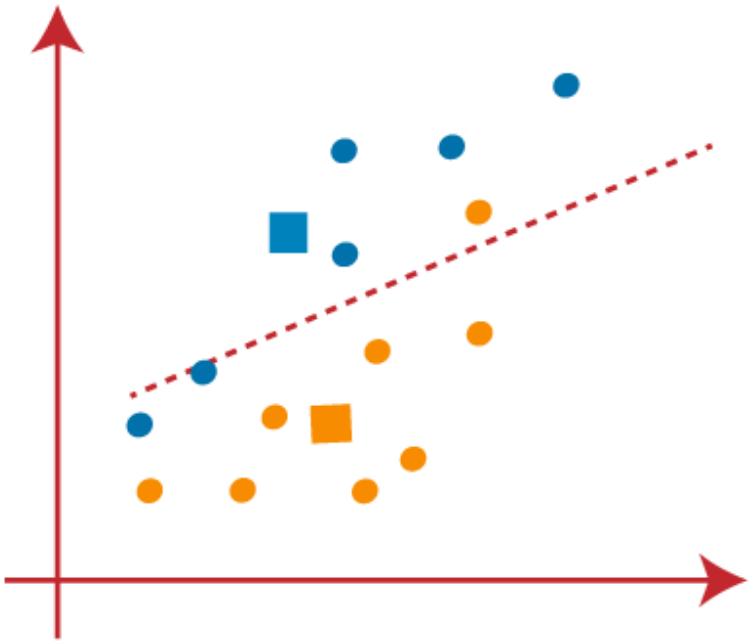
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's colour them as blue and yellow for clear visualization.



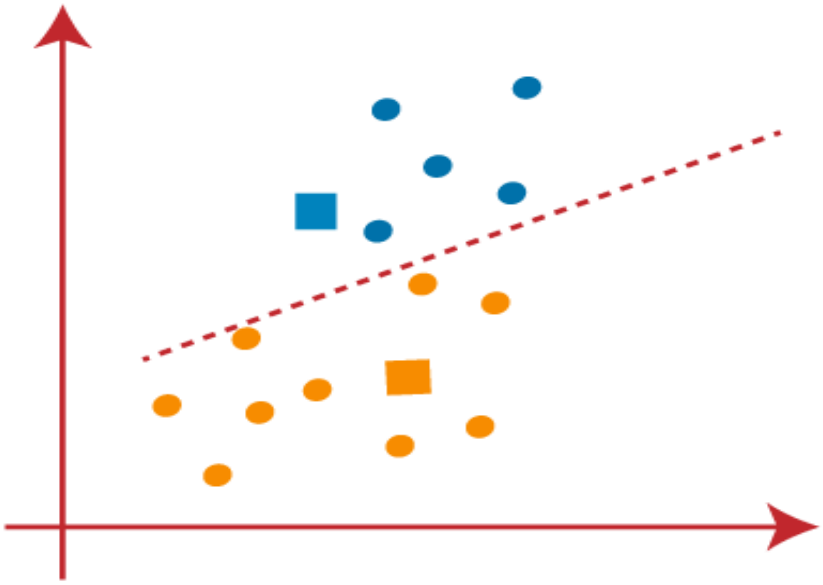
- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids:



- 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

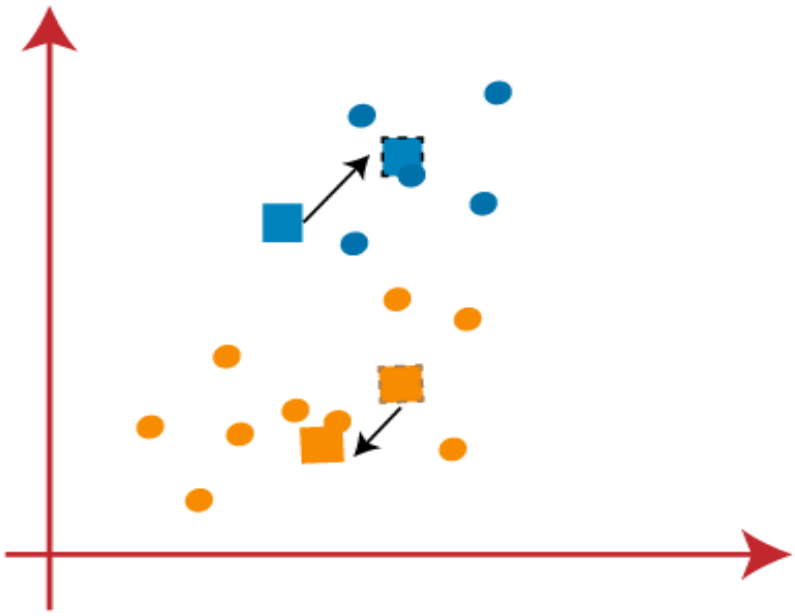


From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

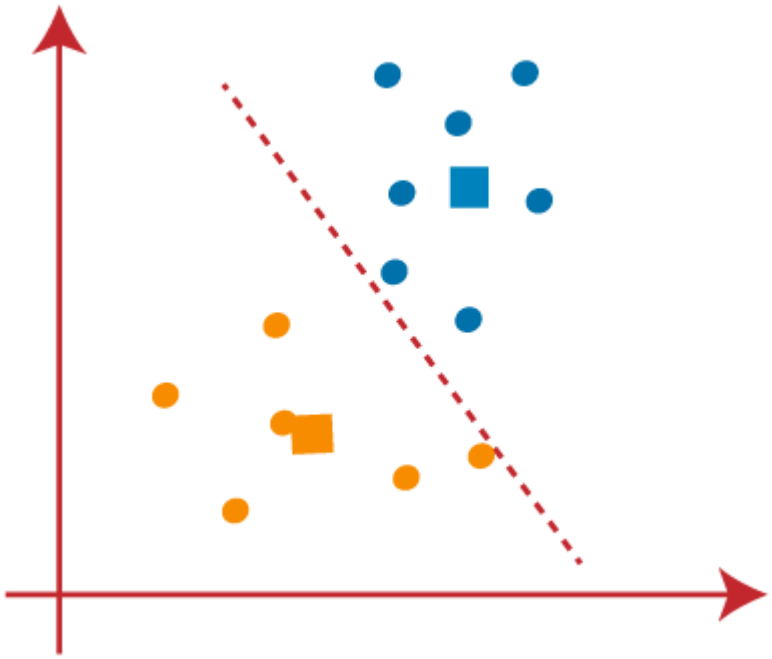


reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

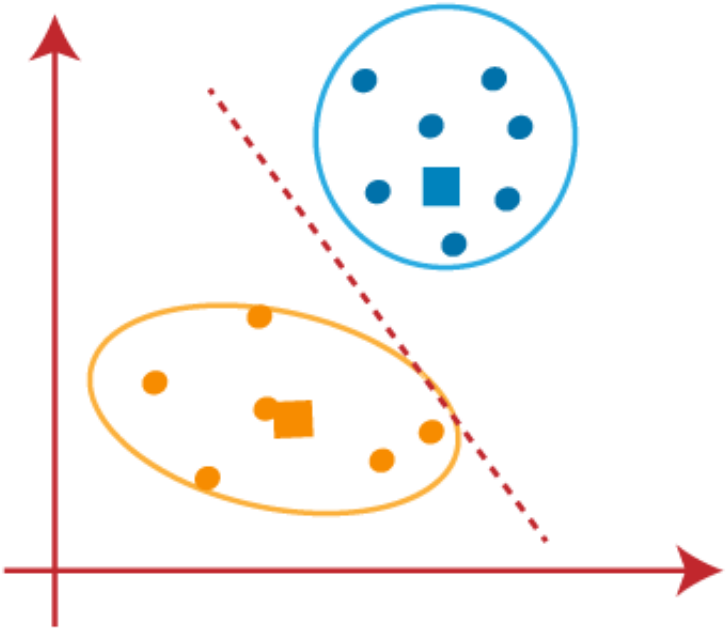
- We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



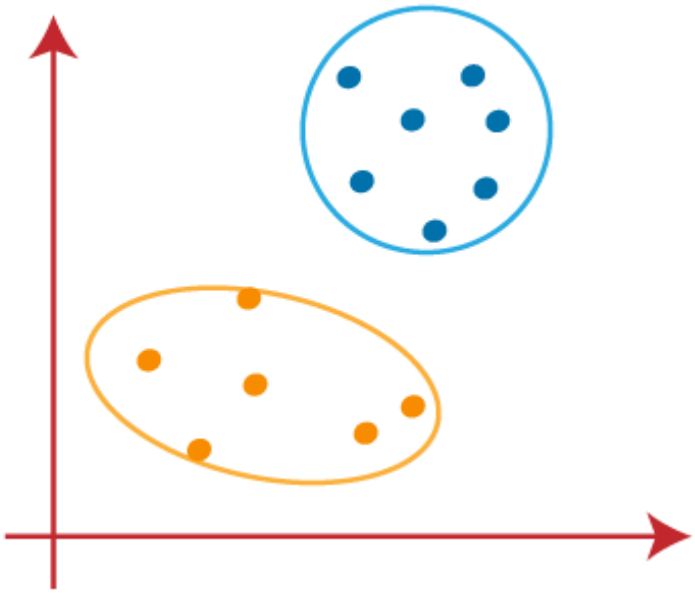
- As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- 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:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



- **Challenges of Unsupervised Learning**

  While unsupervised learning has many benefits, some challenges can occur when it allows machine learning models to execute without any human intervention. Some of these challenges can include:

  - Computational complexity due to a high volume of training data
  - Longer training times
  - Higher risk of inaccurate results
  - Human intervention to validate output variables
  - Lack of transparency into the basis on which data was clustered

5.6. Advantages and Disadvantages Unsupervised Learning

- **Advantages Unsupervised Learning**

- **Use of Unlabelled Data**

  Unsupervised learning helps us to find hidden patterns or structures in data that don't have any labels. It gives us valuable insights and knowledge by uncovering meaningful connections and information that we may not have noticed before.

- **Scalability**

  Unsupervised learning algorithms handle large-scale datasets without manual labelling and make them more scalable than supervised learning in certain scenarios. Unsupervised learning algorithms handle large-scale datasets without manual labelling and make them more scalable than supervised learning in certain scenarios. Unsupervised learning algorithms handle large-scale datasets without manual labeling and make them more scalable than supervised learning in certain scenarios.

- **Anomaly Detection**

  Unsupervised learning can effectively detect anomalies or outliers in data, which is particularly useful for fraud detection, network security, or identifying rare events.

- **Data Preprocessing**

  Unsupervised learning techniques like dimensionality reduction can help preprocess data by reducing noise, removing irrelevant features, and improving efficiency in subsequent supervised learning tasks.

- **Disadvantages of Unsupervised Learning**

  Unsupervised learning has some limitations and challenges:

- **Lack of Ground Truth**

  Since unsupervised learning deals with unlabelled data, there is no definitive measure of correctness or accuracy. Evaluation and interpretation of results become subjective and rely heavily on domain expertise.

- **Interpretability**

  Unsupervised learning algorithms often provide clusters or patterns without explicit labels or explanations. Interpreting and understanding the meaning of these clusters can be challenging and subjective.

- **Overfitting and Model Selection**

  Unsupervised learning models are susceptible to overfitting and choosing the optimal model or parameters can be challenging due to the absence of a labeled validation set.

- **Limited Guidance**

  Unlike supervised learning, where the algorithm learns from explicit feedback, unsupervised learning lacks explicit guidance, which can result in the algorithm discovering irrelevant or noisy patterns.

5.7. Difference between Supervised and Unsupervised learning

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labelled data.	Unsupervised learning algorithms are trained using unlabelled data.
The supervised learning model takes direct feedback to check if it is predicting the correct output or not.	The unsupervised learning model does not take any feedback.

The supervised learning model predicts the output.	The unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized into **Classification** and **Regression** p roblems.	Unsupervised Learning can be classified in **Clustering** and A**ssociation** pr oblems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
A supervised learning model produces an accurate result.	Unsupervised learning models may give less accurate results as compared to supervised learning.

It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Hierarchical Clustering, KNN, and Apriori algorithm.

5.8 Association rule

- The Association rule is a learning technique that helps identify the dependencies between two data items. Based on the dependency, it then maps accordingly so that it can be more profitable. The association rule furthermore looks for interesting associations among the variables of the dataset. It is undoubtedly one of the most important concepts of Machine Learning and has been used in different cases such as association in data mining and continuous production, among others. However, like all other techniques, association in data mining, too, has its own set of disadvantages.
- An Association rule has 2 parts:
    - an antecedent (if) and
    - a consequent (then)
- An antecedent is something that's found in data, and a consequent is an item that is found in combination with the antecedent. Have a look at this rule for instance:

    "If a customer buys bread, he's 70% likely to buy milk."

    In the above association rule, bread is the antecedent and milk is the consequent. Simply put, it can be understood as a retail store's association rule to target their customers better. If the above rule is a result of a

thorough analysis of some data sets, it can be used to not only improve customer service but also improve the company's revenue.

- Association rules are created by thoroughly analyzing data and looking for frequent if/then patterns. Then, depending on the following two parameters, the important relationships are observed:
    - **Support:** Support indicates how frequently the if/then relationship appears in the database.
    - **Confidence:** Confidence tells about the number of times these relationships are true.

5.8.1 Types of Association Rules

There are typically four different types of association rules in data mining. They are

- Multi-relational association rules
- Generalized Association rule
- Quantitative Association Rules

- **Multi-Relational Association Rule**

  Also known as MRAR, the multi-relational association rule is defined as a new class of association rules that are usually derived from different or multi-relational databases. Each rule under this class has one entity with different relationships that represent the indirect relationships between entities.

- **Generalized Association Rule**
    - Moving on to the next type of association rule, the generalized association rule is largely used for getting a rough idea about the interesting patterns that often tend to stay hidden in data.

- **Quantitative Association Rules**
  - This particular type is one of the most unique kinds of all the four association rules available. What sets it apart from the others is the presence of numeric attributes in at least one attribute of quantitative association rules. This is in contrast to the generalized association rule, where the left and right sides consist of categorical attributes.

- **Algorithms Of Associate Rule**

  Apriori Algorithm
  - The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see later.
  - **Support**

    The rule A→B holds in the transaction set D with supports, where s is the percentage of transactions in D that contain A U B (i.e. the union of sets A and B say, or, both A and B).

    Support (A=>B) = P(A ∪ B).
  - **Confidence**

    The rule A→B has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contains B.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

Example Let's look at a concrete example, based on the All Electronics transaction database, D, of the Table below.

Consider the following dataset and we will find frequent itemsets and generate association rules for them.

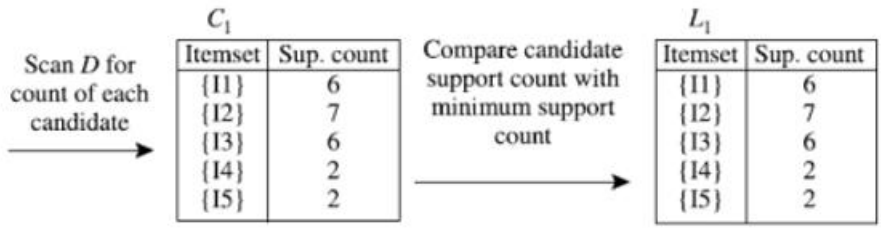
minimum support count is 2

minimum confidence is 60%

Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

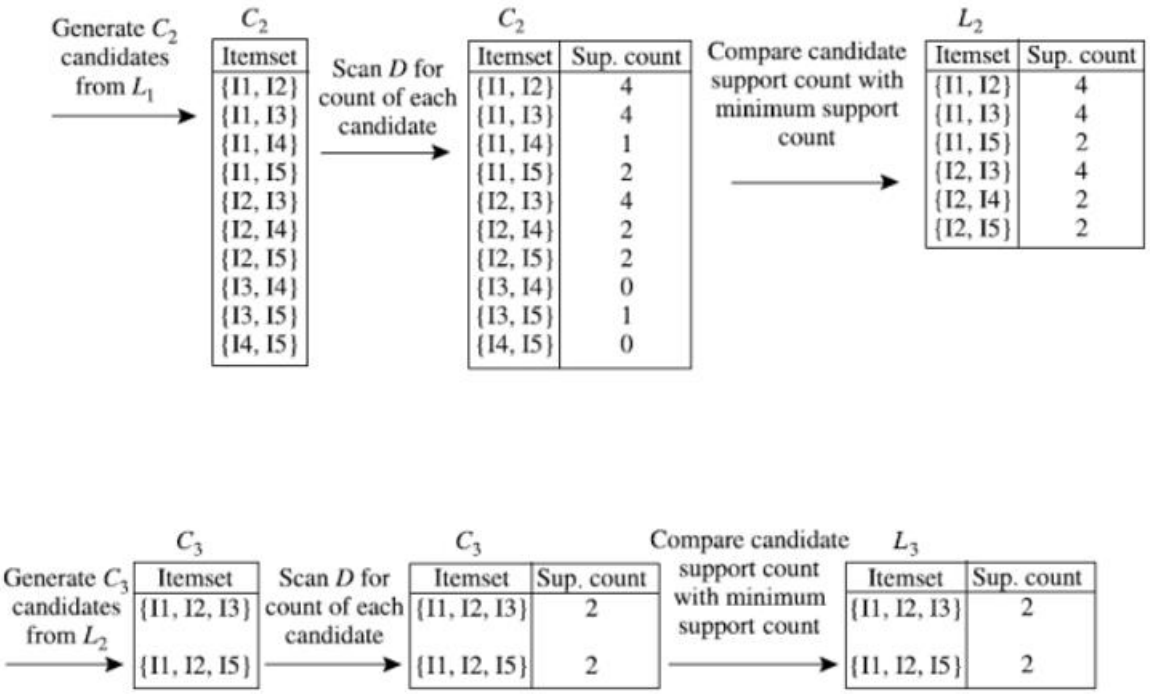


Step-2: K=2

Generate candidate set C2 using L1 (this is called join step). Condition of joining Lk-1 and Lk-1 is that it should have (K-2) elements in common.

Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of{I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)

Now find support count of these itemsets by searching in dataset.



Generation of the candidate itemsets and frequent itemsets, where the minimum support count is 2.

Step-3:

Step 2: Generating Association Rules

$\{I1, I2\} \Rightarrow I5$, $confidence = 2/4 = 50\%$
$\{I1, I5\} \Rightarrow I2$, $confidence = 2/2 = 100\%$
$\{I2, I5\} \Rightarrow I1$, $confidence = 2/2 = 100\%$
$I1 \Rightarrow \{I2, I5\}$, $confidence = 2/6 = 33\%$
$I2 \Rightarrow \{I1, I5\}$, $confidence = 2/7 = 29\%$
$I5 \Rightarrow \{I1, I2\}$, $confidence = 2/2 = 100\%$

If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules are output, because these are the only ones generated that are strong. Note that, unlike conventional classification rules, association rules can contain more than one conjunct in the right side of the rule. ■