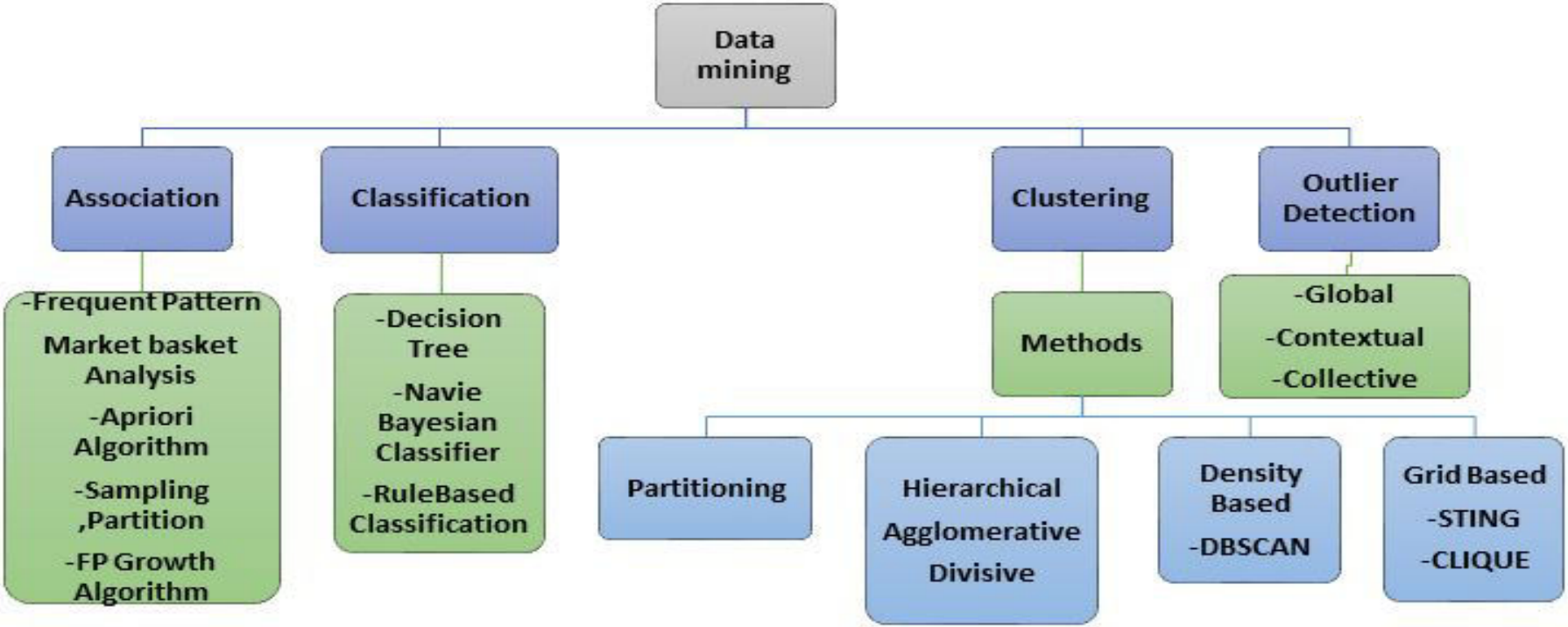
Subject: MCA DMBI 1003

Unit 3
Association , Classification , Clustering

Made By: Priyanka Karale

Subject: MCA DMBI 1003



Made By: Priyanka Karale

Frequent itemset and Frequent Patterns

- **Frequent patterns** are patterns (e.g., itemsets, subsequences, or substructures) that appear frequently in a data set. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a **frequent itemset**.
- A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a **(frequent) sequential pattern.**
- A **substructure** can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a **(frequent) structured pattern**.

Frequent Pattern Mining

Importance:

- Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data. Moreover, it helps in data classification, clustering, and other data mining tasks. Thus, frequent pattern mining has become an important data mining task and a focused theme in data mining research.
- FPM has many applications in the field of data analysis, software bugs, cross-marketing, sale campaign analysis, market basket analysis, etc.
- Frequent itemsets discovered through Apriori have many applications in data mining tasks. Tasks such as finding interesting patterns in the database, finding out sequence and Mining of association rules is the most important of them.

Market Basket Analysis:

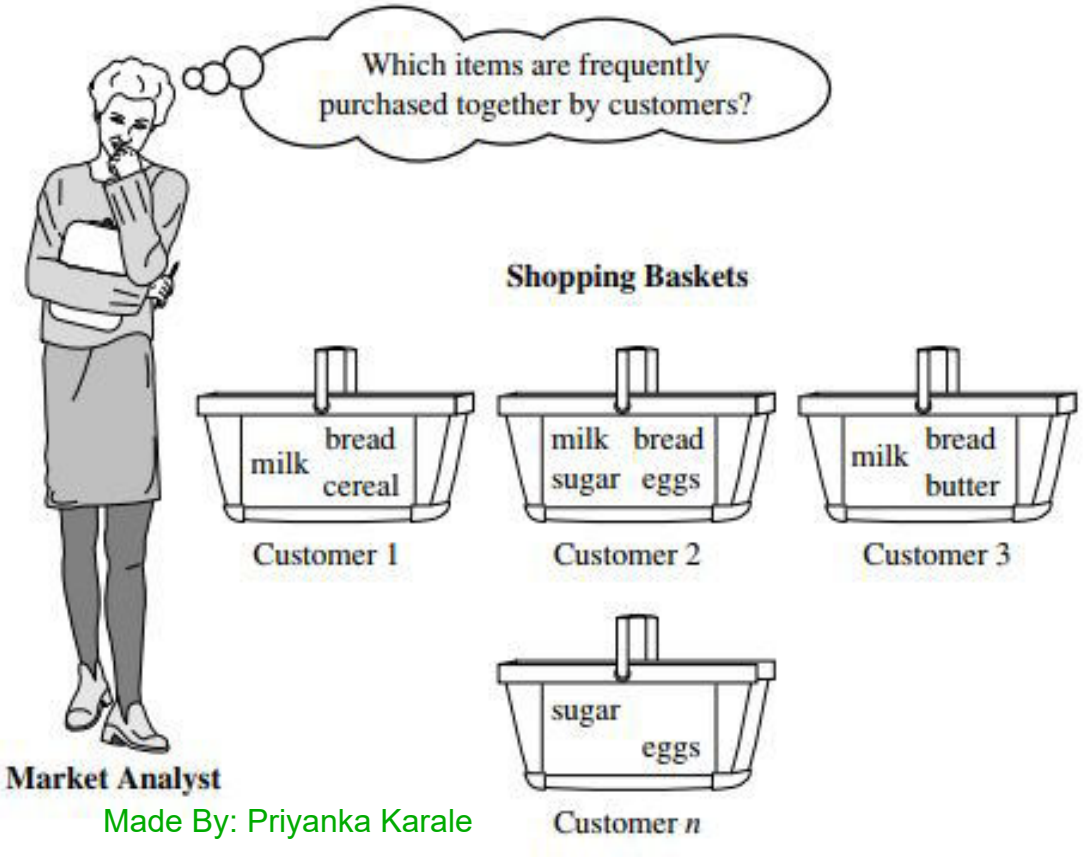
- This process analyzes customer buying habits, by finding associations between different items that customers place in their “shopping baskets”.

- The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.

- For instance, if customers are buying milk, how likely are they also buy bread (and what kind of bread) on the same trip to the supermarket.

- Suppose, as a branch manager of a general store, you would like to learn more about the buying habits of your customers.

- In one strategy, items that are frequently purchased together can be placed in proximity, to further encourage the combined sale of such items.



Frequent ItemSet

A set of items is called frequent if it satisfies a minimum threshold value for support and confidence. Support shows transactions with items purchased together in a single transaction.

Association Rules

• Association Rule Mining is defined as:

"Let I= { ...} be a set of 'n' binary attributes called items. Let D= { ....} be set of transaction called database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of form X->Y where X, Y? I and X?Y=?. The set of items X and Y are called antecedent and consequent of the rule respectively."

• Learning of Association rules is used to find relationships between attributes in large databases. An association rule, A=> B, will be of the form" for a set of transactions, some value of itemset A determines the values of itemset B under the condition in which minimum support and confidence are met".

Support and **Confidence** can be represented by the following example:

Bread=> butter [support=2%, confidence-60%]

The above statement is an example of an association rule. This means that there is a 2% transaction that bought bread and butter together and there are 60% of customers who bought bread as well as butter.

• Support and Confidence for Itemset A and B are represented by formulas:

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

$$\text{Confidence (A}\rightarrow\text{B)} = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

• Association rule mining consists of 2 steps:

1. Find all the frequent itemsets.

2. Generate association rules from the above frequent itemsets.

Apriori Algorithm – Frequent Pattern Algorithms

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules [AS94b].

. This algorithm uses two steps “join” and “prune” to reduce the search space.

Apriori says:

The probability that item I is not frequent is if:

• P(I) < minimum support threshold, then I is not frequent.

• P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to itemset.

• If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property

The steps followed in the Apriori Algorithm of data mining are:

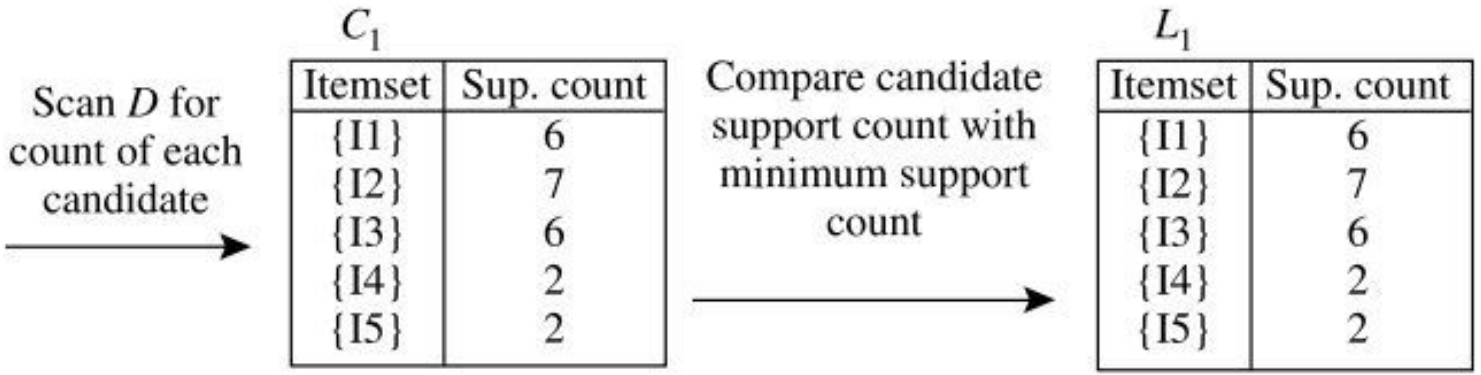
• **Join Step**: This step generates (K+1) itemset from K-itemsets by joining each item with itself.

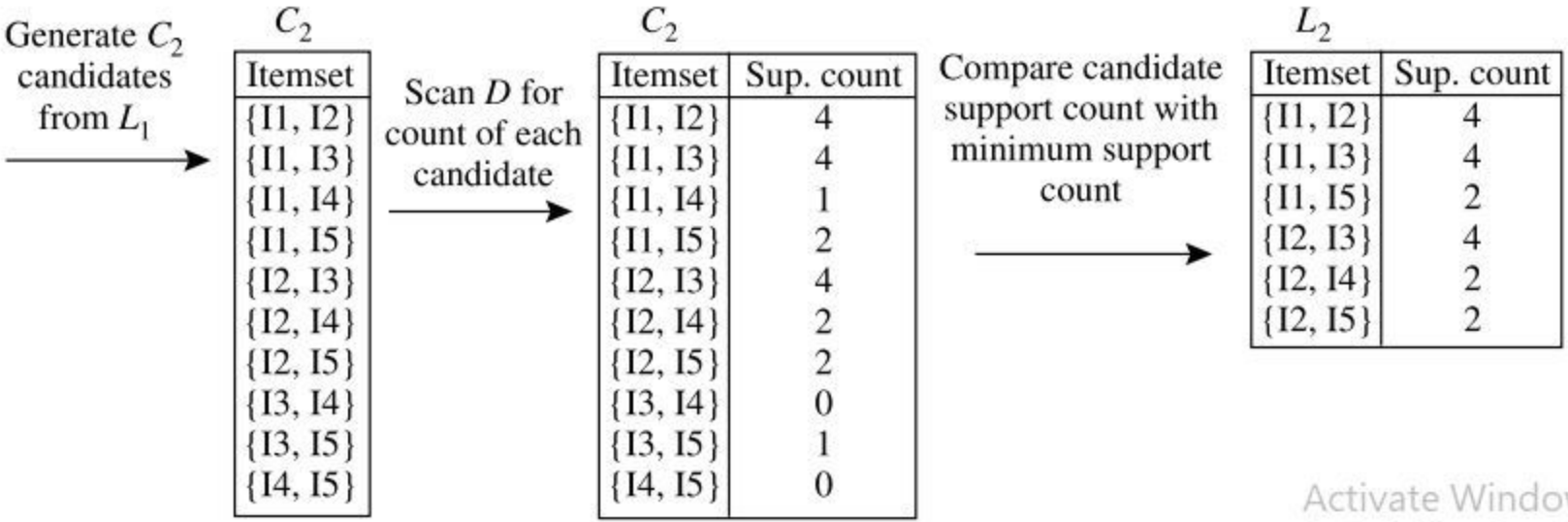
• **Prune Step**: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

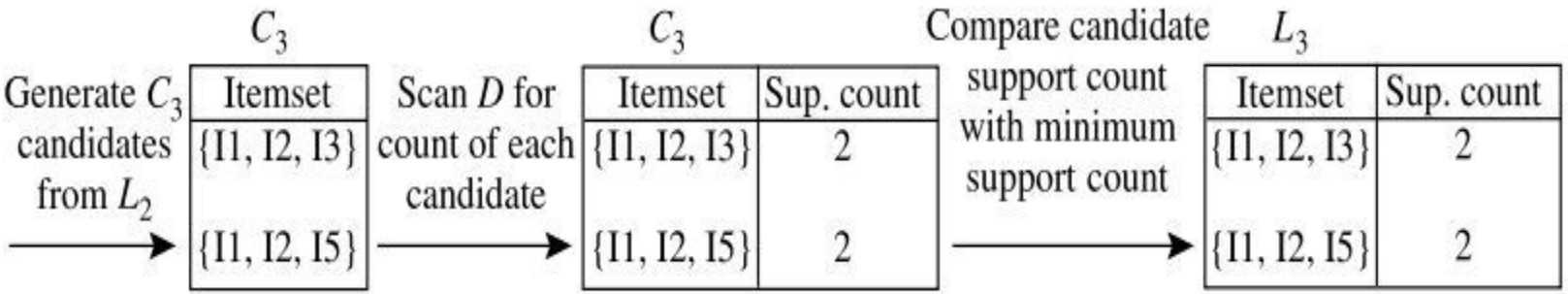
Apriori Example

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


Scan D for count of each candidate
C1
Itemset
Sup. count
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2
Compare candidate support count with minimum support count
L1
Itemset
Sup. count
{I1} 6
{I2} 7
{I3} 6
{I4} 2
{I5} 2




C_3
Generate C_3 candidates from L_2
Itemset
{I1, I2, I3}
{I1, I2, I5}
Scan D for count of each candidate
C_3
Itemset
Sup. count
{I1, I2, I3}
2
{I1, I2, I5}
2
Compare candidate support count with minimum support count
L_3
Itemset
Sup. count
{I1, I2, I3}
2
{I1, I2, I5}
2

Generating association rules

Let's try an example based on the transactional data for AllElectronics shown before in Table.
The data contain frequent itemset X = {I1, I2, I5}.
What are the association rules that can be generated from X?
The nonempty subsets of X are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, and {I5}.
The resulting association rules are as shown below, each listed with its confidence:

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

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

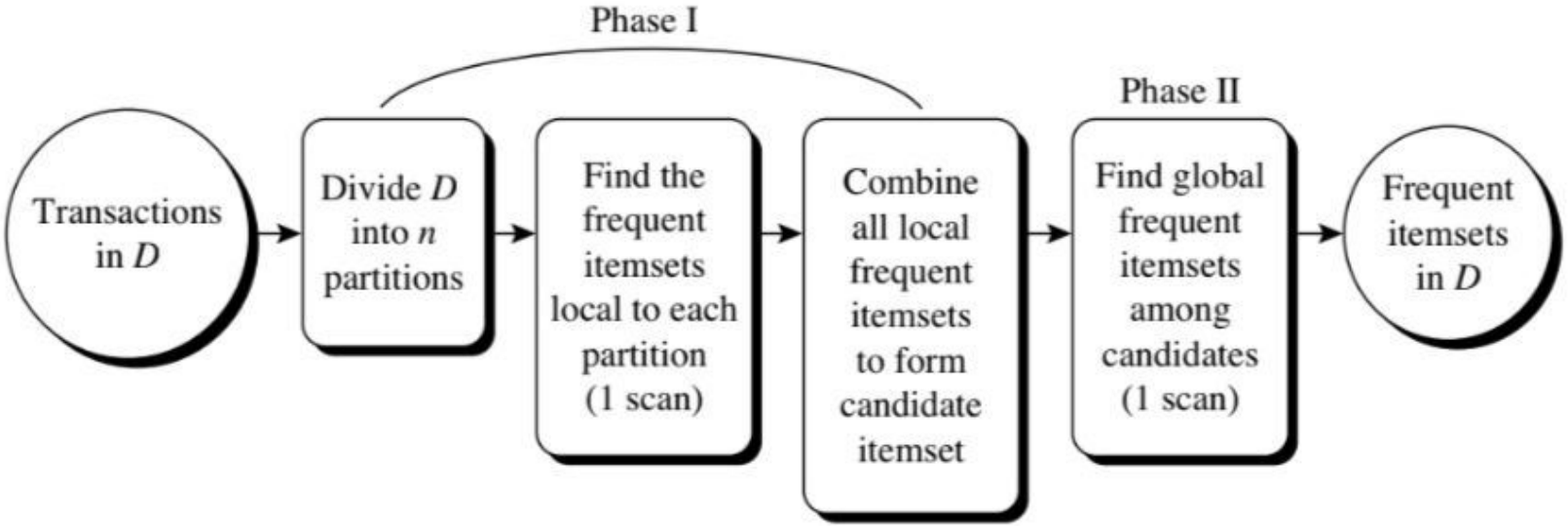
$$\text{Confidence (A}\longrightarrow\text{B)} = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

Improving the Efficiency of Apriori

- Hash-based technique
- Transaction reduction
- Partitioning
- Sampling
- Dynamic itemset counting

Made By: Priyanka Karale

Partitioning



Made By: Priyanka Karale

- Divide the database into non-overlapping subsets.
- Treat each subset as a separate database where each subset fits entirely into main memory.
- Apply the Apriori algorithm to each partition.
- Take the union of all frequent itemsets from each partition.
- These itemsets form the global candidate frequent itemsets for the entire database.
- Verify the global set of itemsets by having their actual support measured for the entire database

Sampling

The **sampling algorithm** selects samples from the database of transactions that individually fit into memory. Frequent itemsets are then formed for each sample

Frequent Pattern Growth Algorithm

The two primary drawbacks of the Apriori Algorithm are:

1. At each step, candidate sets have to be built.
2. To build the candidate sets, the algorithm has to repeatedly scan the database.

These two properties inevitably make the algorithm slower. To overcome these redundant steps, a new association-rule mining algorithm was developed named **Frequent Pattern Growth Algorithm.**

FP Growth Algorithm:Example

Transaction ID	Items
T1	{E, K, M, N, O, Y}
T2	{D, E, K, N, **O**, Y}
T3	{A, E, K, M}
T4	{C, K, M, U, Y}
T5	{C, E, I, K, O, O}

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
0	3
U	1
Y	3

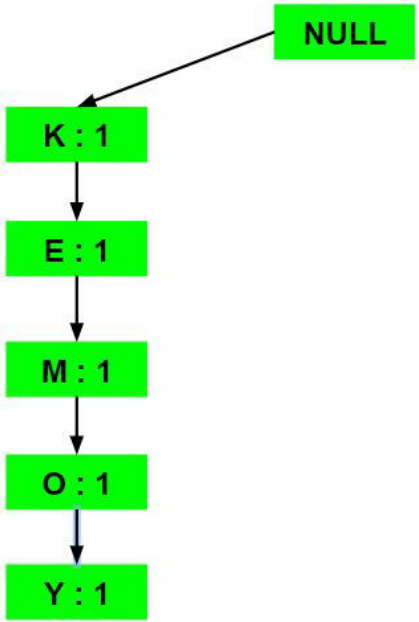
minimum support = 3

L = {K : 5, E : 4, M : 3, O : 3, Y : 3}

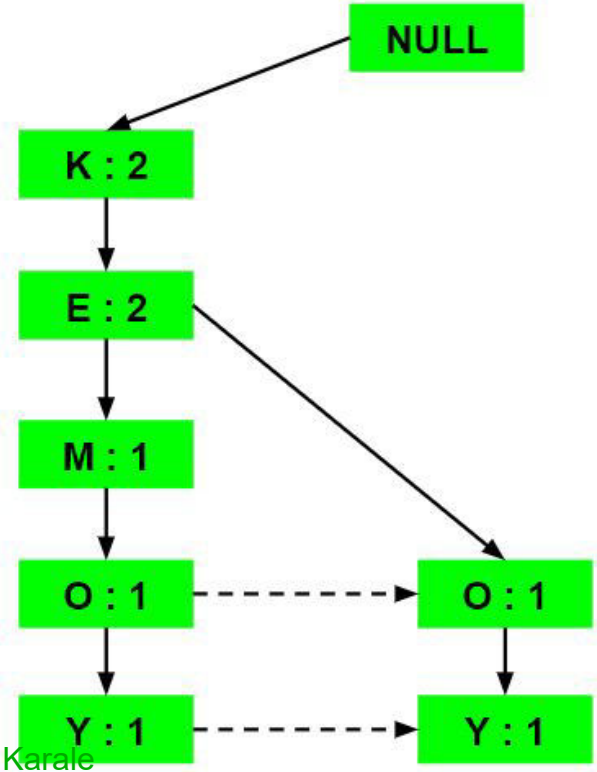
Create Ordered-Item Set

Transaction ID	Items	Ordered-Item Set
T1	{E, K, M, N, O, Y}	{K, E, M, O, Y}
T2	{D, E, K, N, O, Y}	{K, E, O, Y}
T3	{ A, E, K, M}	{K, E, M}
T4	{C, K, M, U, Y}	{K, M, Y}
T5	{C, E, I, K, O, O}	{K, E, O}

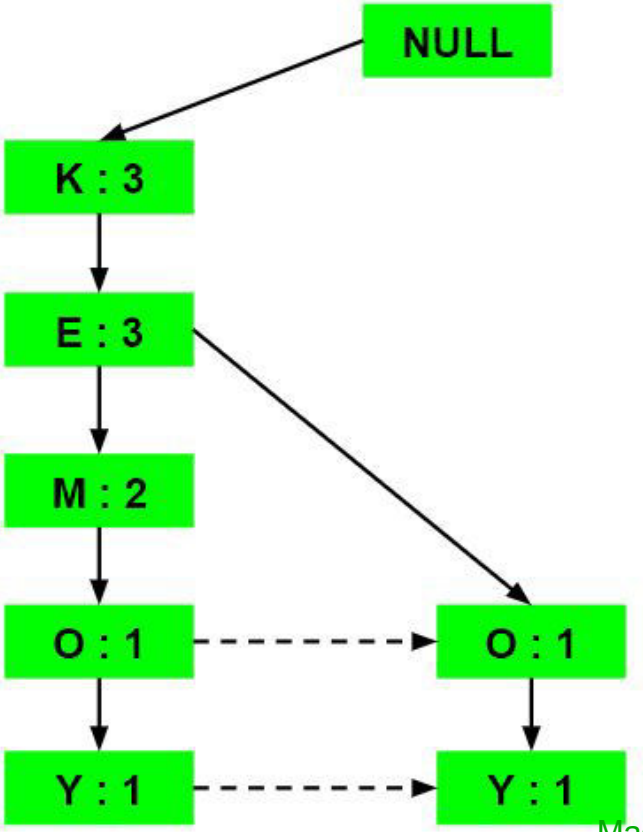
Inserting the set {K, E, M, O, Y}



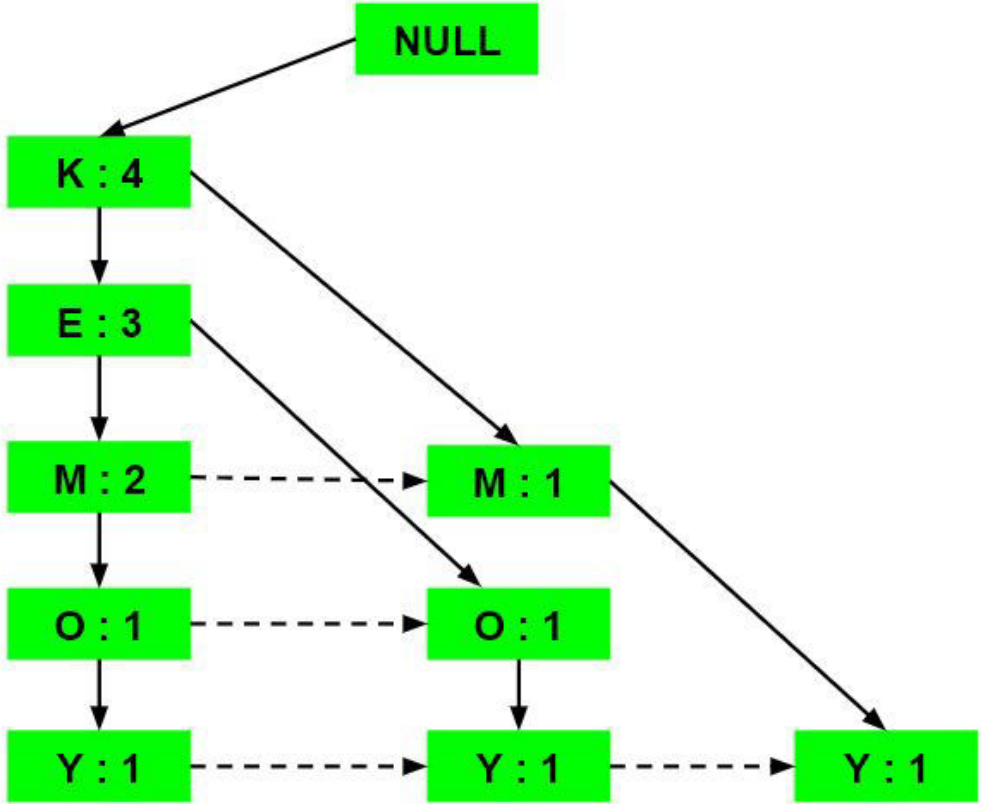
Inserting the set {K, E, O, Y}



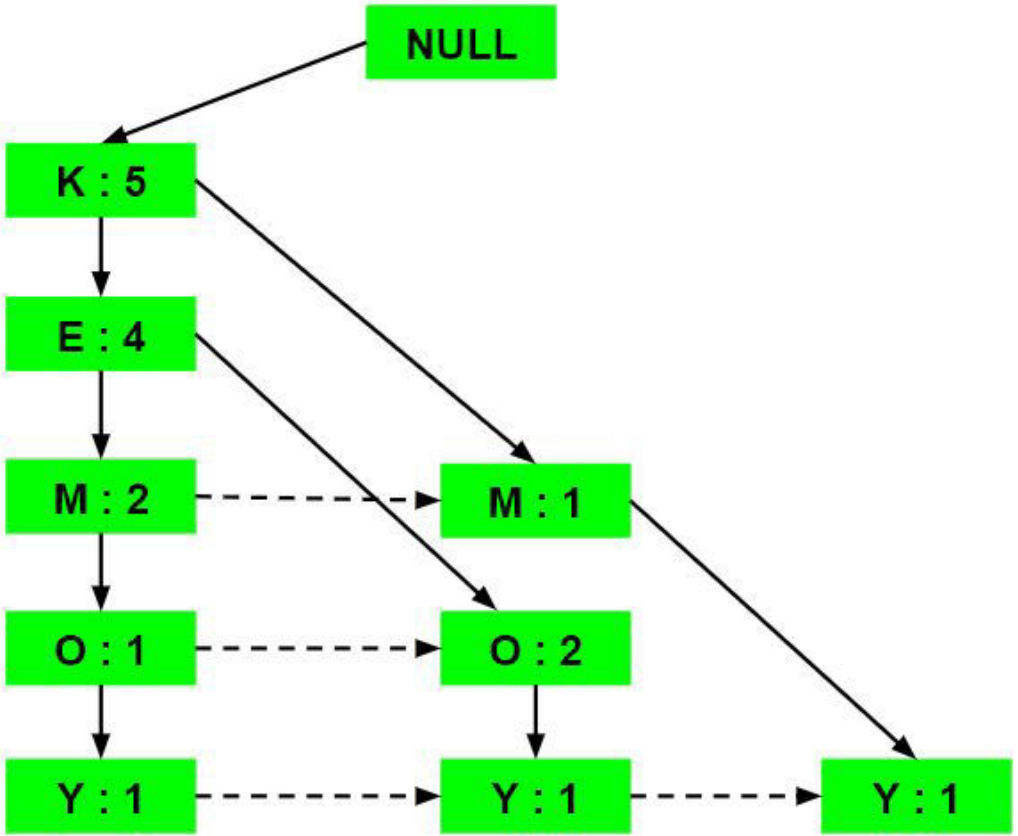
Inserting the set {K, E, M}



Inserting the set {K, M, Y}



Inserting the set {K, E, O}



Made By: Priyanka Karale

Conditional Pattern Base:

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

Conditional Frequent Pattern Tree

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Frequent Pattern rules

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

FP Growth	Apriori
Pattern Generation	
FP growth generates pattern by constructing a FP tree	Apriori generates pattern by pairing the items into singletons, pairs and triplets.
Candidate Generation	
There is no candidate generation	Apriori uses candidate generation
Process	
The process is faster as compared to Apriori. The runtime of process increases linearly with increase in number of itemsets.	The process is comparatively slower than FP Growth, the runtime increases exponentially with increase in number of itemsets
Memory Usage	
A compact version of database is saved	The candidates combinations are saved in memory

Subject: MCA DMBI 1003

Classification

Made By: Priyanka Karale

Classification

- **Data classification** is a two-step process, consisting of a *learning step* (where a classification model is constructed) and a *classification step* (where the model is used to predict class labels for given data).

- In the first step, a classifier is built describing a predetermined set of data classes or concepts.

- This is the **learning step** (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a **training set** made up of database tuples (samples) and their associated class labels.

- The class label attribute is discrete-valued and unordered. It is *categorical* (or nominal) in that each value serves as a category or class.

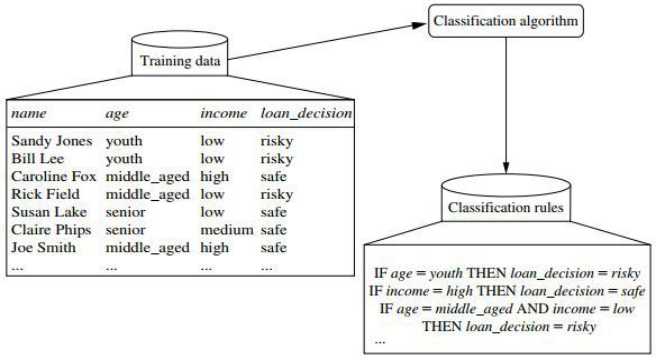
- Because the class label of each training sample *is provided*, this step is also known as **supervised learning** (i.e., the learning of the classifier is "supervised" in that it is told to which class each training tuple belongs).

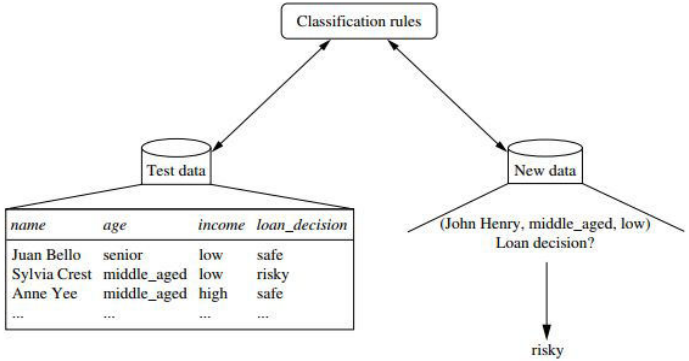
- In contrast with **unsupervised learning** (or **clustering**), in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance.

- In the second step, the model is used for classification.

- First, the predictive accuracy of the classifier is estimated

Case Study: The data classification process





Step 1

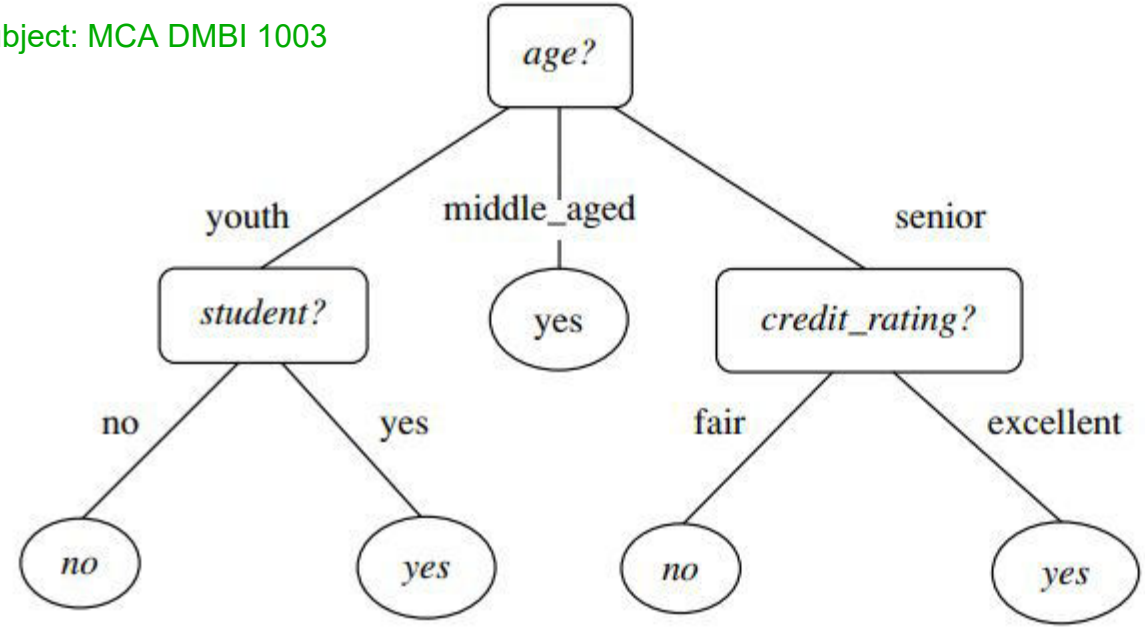
Learning: Training data are analyzed by a classification algorithm. Here, the class label attribute is *loan decision*, and the learned model or classifier is represented in the form of classification rules

Step 2

Classification: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data samples.

Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples/samples.
- A **decision tree** is a flowchart-like tree structure, where each **internal node** (non-leaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or *terminal node*) holds a class label.
- The topmost node in a tree is the **root** node.
- Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals.
- Popular decision tree algorithms include ID3, C4.5, CART, and Random Forest.



A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

Attribute Selection Measures

1. Information Gain: Information gain is a measure of this change in entropy.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

2. Entropy

   Entropy is nothing but the uncertainty in our dataset or measure of disorder. Let me try to explain

$$E(S) = -p_{(+)}\log p_{(+)} - p_{(-)}\log p_{(-)}$$

3. Gain

Here,

- p_+ is the probability of positive class
- p_- is the probability of negative class
- S is the subset of the training example

Exercise: For the following Medical Diagnosis Data, create a decision tree.

Sore Throat	Fever	Swollen Glands	Congestion	Headache	Diagnosis
YES	YES	YES	YES	YES	Strep Throat
NO	NO	NO	YES	YES	Allergy
YES	YES	NO	YES	NO	Cold
YES	NO	YES	NO	NO	Strep Throat
NO	YES	NO	YES	NO	Cold
NO	NO	NO	YES	NO	Allergy
NO	NO	YES	NO	NO	Strep Throat
YES	NO	NO	YES	YES	Allergy
NO	YES	NO	YES	YES	Cold
YES	YES	NO	YES	YES	Cold

$$(i).I(p,n) = -\frac{p}{S}\log_2\frac{p}{S} - \frac{n}{S}\log_2\frac{n}{S},$$

$$where, S = (p+n)$$

$$(ii).E(A) = \sum_{i=1}^{v}\frac{p_i + n_i}{p+n}I(p_i,n_i)$$

$$(iii).Gain(A) = I(p_i,n_i) - E(A)$$

S=Strep Throat (3)+Allergy(3)+Cold(4)=10

$$InfoGain = -\left[\frac{3}{10}\log_2\left(\frac{3}{10}\right)+\frac{3}{10}\log_2\left(\frac{3}{10}\right)+\frac{4}{10}\log_2\left(\frac{4}{10}\right)\right]$$

$$= -\left[0.3\log_2(0.3)\times 2 + 0.4\log_2(0.4)\right]$$

$$= -\left[0.6\frac{\log_{10}(0.3)}{\log_{10}2}+0.4\frac{\log_{10}(0.4)}{\log_{10}2}\right]$$

$$= -\left[0.6\frac{(-0.522)}{0.301}+0.4\frac{(-0.397)}{0.301}\right]$$

$$= \left[0.6(1.73)+0.4(1.318)\right]$$

$$= 1.038 + 0.5272 \Rightarrow 1.562$$

Info(S)=1.562

Made By: Priyanka Karale

Finding Splitting Attribute

- **Select Attribute with highest Gain**

Sore Throat=

	Strep Throat	Allergy	Cold	
YES	2	1	2	Information Gain x P
NO	1	2	2	Information Gain x P

+ = Entropy

Sore Throat=

$$Info(YES) = -\left[\frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{1}{5}\log_2\left(\frac{1}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right]$$

$$Info(YES) = 1.52$$

$$Info(NO) = -\left[\frac{1}{5}\log_2\left(\frac{1}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right]$$

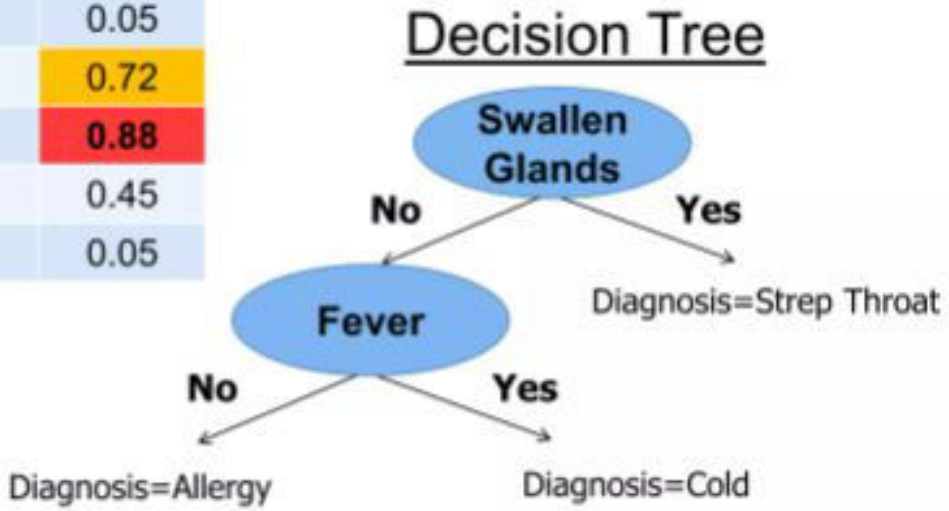
$$Info(NO) = 1.52$$

Entropy (E(Sore Throat)= P(YES)x1.52 + P(NO)x1.52
= (5/10)x1.52 + (5/10)x1.52 = 1.52

Gain (Sore Throat)= **Info(S)-E(Sore Throat)**
= **1.562-1.52 = 0.05**

- Gain for each Attribute

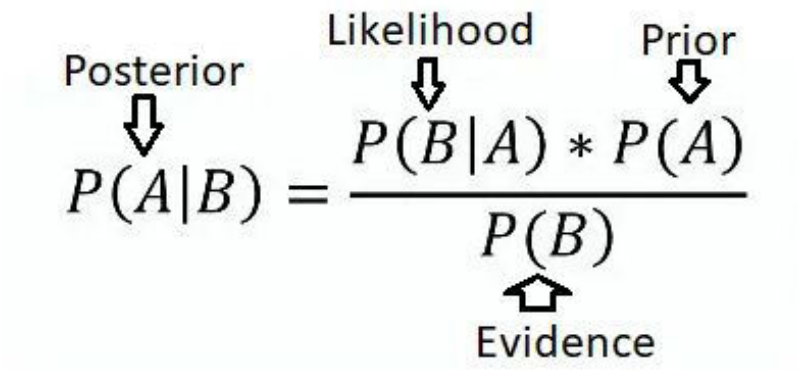
Attribute	Gain
Sore Throat	0.05
Fever	0.72
Swallen Glands	**0.88**
Congestion	0.45
Headache	0.05



IF Swallen Glands = "YES", **THEN** Diagnosis=Strep Throat
IF Swallen Glands = "NO" **AND** Fever = "YES", **THEN** Diagnosis=Cold
IF Swallen Glands = "NO" **AND** Fever = "NO", **THEN** Diagnosis=Allergy

Subject: MCA DMBI 1003

Baye's Theorem



P(A | B) is the conditional probability of event A occurring, given that B is true.

P(B | A) is the conditional probability of event B occurring, given that A is true.

P(A) and P(B) are the probabilities of A and B occurring independently of one another.

Made By: Priyanka Karale

Bayes Theorem Derivation

Bayes Theorem can be derived for events and random variables separately using the definition of conditional probability and density. From the definition of conditional probability, Bayes theorem can be derived for events as given below:

$P(A|B) = P(A \cap B)/ P(B)$, where $P(B) \neq 0$

$P(B|A) = P(B \cap A)/ P(A)$, where $P(A) \neq 0$

Here, the joint probability $P(A \cap B)$ of both events A and B being true such that,

$P(B \cap A) = P(A \cap B)$

$P(A \cap B) = P(A \mid B)\, P(B) = P(B \mid A)\, P(A)$

$P(A|B) = [P(B|A)\, P(A)]/ P(B)$, where $P(B) \neq 0$

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions

How Do Naive Bayes Algorithms Work?

Let’s understand it using an example. Below I have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

1. Convert the data set into a frequency table

   In this first step data set is converted into a frequency table

2. Create Likelihood table by finding the probabilities

   Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Subject: MCA DMBI 1003

Fruit(Yellow,Sweet,Long)

Frequency Table:

Fruit	Yellow	Sweet	Long	Total
Mango	350	450	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

Made By: Priyanka Karale

Mango:

P(X | Mango) = P(Yellow | Mango) * P(Sweet | Mango) * P(Long | Mango)

P(Yellow | Mango) = (P(Yellow | Mango) * P(Yellow))/ P (Mango)
= ((350/650) * (800/1200)) / (650/1200)
P(Yellow | Mango)= 0.53 —-1

P(Sweet | Mango) = (P(Sweet | Mango) * P(Sweet))/ P (Mango)
= ((450/650) * (850/1200)) / (650/1200)

P(Sweet | Mango)= 0.69 —--2

)P(Long | Mango) = (P(Long | Mango) * P(Long))/ P (Mango)
= ((0/650) * (400/1200)) / (800/1200)
P(Long | Mango)= 0 —--3

On multiplying eq 1,2,3 ==> P(X | Mango) = 0.53 * 0.69 * 0

P(X | Mango) = 0

Banana:

P(X | Banana) = P(Yellow | Banana) * P(Sweet | Banana) * P(Long | Banana)

P(Yellow | Banana) = (P(Banana | Yellow) * P(Yellow))/ P (Banana)
= ((400/400) * (800/1200)) / (400/1200)
P(Yellow | Banana) = 2 —-----1

P(Sweet | Banana) = (P(Banana | Sweet) * P(Sweet))/ P (Banana)

= ((300/400) * (850/1200)) / (400/1200)
P(Sweet | Banana) = 1.6 —---2

P(Long | Banana) = (P(Banana | Yellow) * P(Long))/ P (Banana)
= ((350/400) * (400/1200)) / (400/1200)
P(Yellow | Banana) = 0.875 —------3

On multiplying eq 1,2,3 ==> P(X | Banana) = 2 * 1.6 * 0.875
P(X | Banana) = 2.8

Others:

P(X | Others) = P(Yellow | Others) * P(Sweet | Others) * P(Long | Others)

P(Yellow | Others) = (P(Others| Yellow) * P(Yellow))/ P (Others)
= ((50/150) * (800/1200)) / (150/1200)
P(Yellow | Others) = 1.78 —---------1

P(Sweet | Others) = (P(Others| Sweet) * P(Sweet))/ P (Others)
= ((100/150) * (850/1200)) / (150/1200)
P(Sweet | Others) = 3.78 —-----2

P(Long | Others) = (P(Others| Long) * P(Long))/ P (Others)
= ((50/150) * (400/1200)) / (150/1200)
P(Long | Others) = 0.9 —------3

On multiplying eq 7,8,9 ==> P(X | Others) = 1.78 * 3.78* 0.9
P(X | Others) = 6.05

So finally from P(X | Mango) == 0 , P(X | Banana) == 0.65 and P(X| Others) == 0.072.

We can conclude Fruit{Yellow,Sweet,Long} is Banana.

Subject: MCA DMBI 1003

Clustering

Made By: Priyanka Karale

CLUSTERING

- *Clustering* is the process of grouping a set of data objects into multiple groups or *clusters* so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters.

- It is the process of partitioning a set of data objects (or observations) into subsets.

- Each subset is a **cluster**, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters.
- The set of clusters resulting from a cluster analysis can be referred to as a **clustering**.
- In this context, different clustering methods may generate different clustering on the same data set.

- A critical difference in classification and clustering is that clustering can automatically find the groupings.

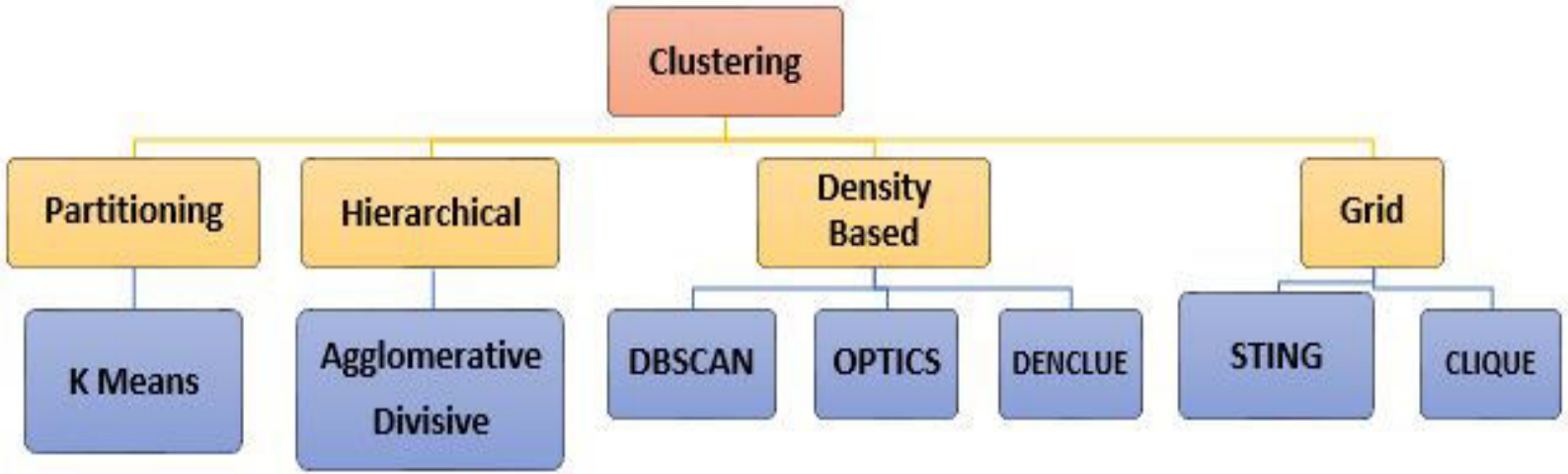
- Clustering is also called **data segmentation** in some applications because clustering partitions large data sets into groups according to their *similarity*.

- Clustering can also be used for **outlier detection**, where outliers (values that are "far away" from any cluster) may be more interesting than common cases.

- Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce.

- For example, exceptional cases in credit card transactions, such as very expensive and infrequent purchases, may be of interest as possible fraudulent activities.

Clustering Methods



Made By: Priyanka Karale

Partitioning methods

- The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters.

- Formally, given a data set, *D*, of *n* objects, and *k*, the number of clusters to form, a **partitioning algorithm** organizes the objects into *k* partitions $.k \leq n/$, where each partition represents a cluster.

- The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters in terms of the data set attributes.

k-means

k-medoids

k-means: A Centroid-Based Technique

- A centroid-based partitioning technique uses the *centroid* of a cluster, *Ci*, to represent that cluster.

- Conceptually, the centroid of a cluster is its center point.

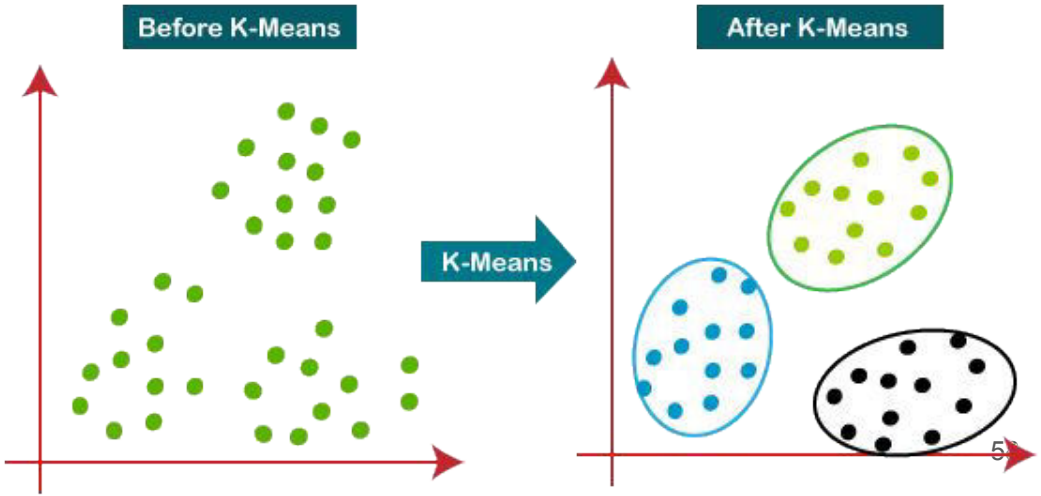
- The *k*-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster.

- The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

- 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.

- Hence each cluster has data points with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



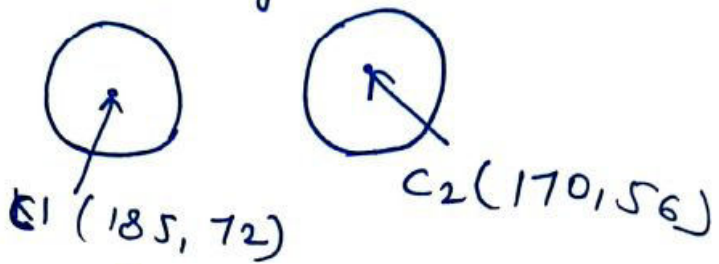
Height	Weight
185	72
170	56
168	60
179	68
182	72
188	77
180	71
180	70
183	84
180	88
180	67
177	76

Height in cm and Weight in Kgs

Euclidean Formula

$$d((x,y),(a,b)) = \sqrt{(x-a)^2 + (y-b)^2}$$

Starting 2 clusters



ED for 3rd

$$k1 = \sqrt{(168-185)^2 + (60-72)^2}$$

$$= 20.80$$

$$k2 = \sqrt{(168-170)^2 + (60-56)^2}$$

$$= 4.48$$

$K_1 = \{1, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

$K_2 = \{2, 3\}$

New Centroid Calculation ED for 4th

for K_2

$= \frac{170+168}{2}, \frac{60+56}{2}$

$= (169, 58)$

$K_1 = \sqrt{(179-185)^2+(68-72)^2}$

$= 6.32$

$K_2 = \sqrt{(179-169)^2+(68-58)^2}$

$= 14.14$

Hierarchical Clustering

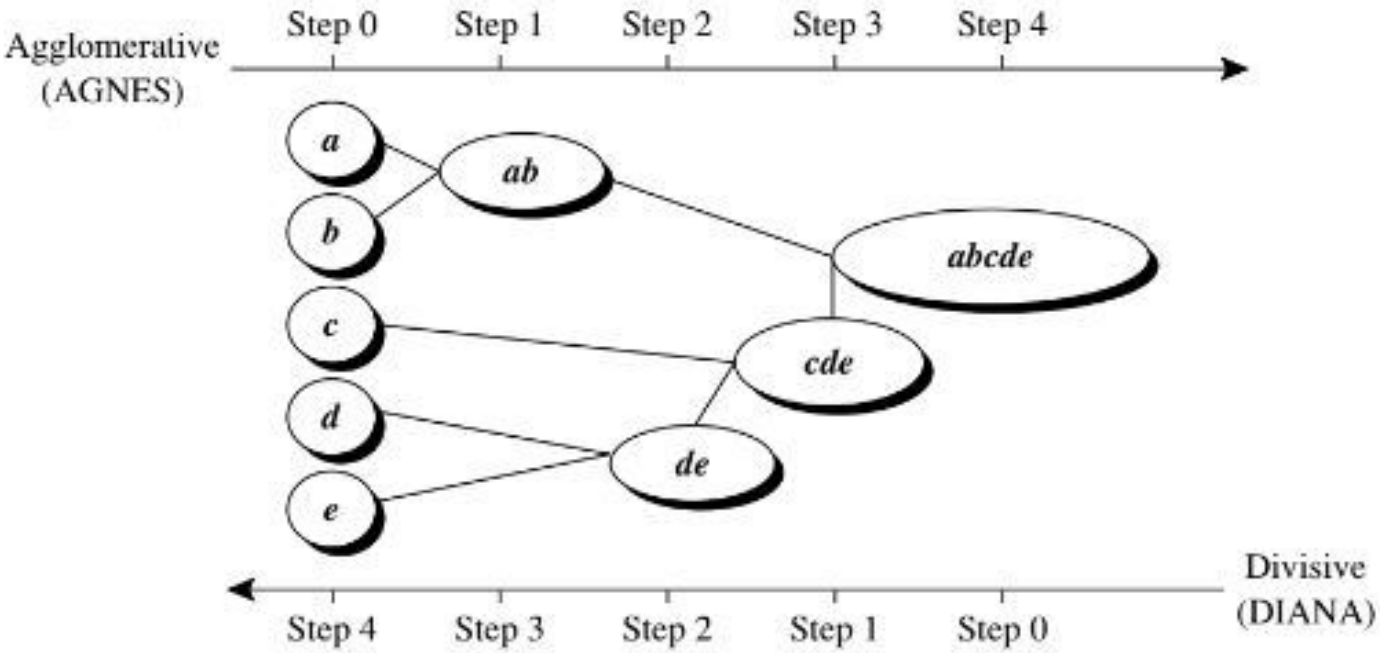
Hierarchical Clustering creates clusters in a hierarchical tree-like structure (also called a Dendrogram). Meaning, a subset of similar data is created in a tree-like structure in which the root node corresponds to the entire data, and branches are created from the root node to form several clusters.

Hierarchical Clustering is of two types.

1. **Divisive**
2. **Agglomerative Hierarchical Clustering**

Divisive Hierarchical Clustering is also termed as a top-down clustering approach. In this technique, entire data or observation is assigned to a single cluster. The cluster is further split until there is one cluster for each data or observation.

Agglomerative Hierarchical Clustering is popularly known as a bottom-up approach, wherein each data or observation is treated as its cluster. A pair of clusters are combined until all clusters are merged into one big cluster that contains all the data.


Agglomerative
(AGNES)
Step 0
Step 1
Step 2
Step 3
Step 4
a
b
c
d
e
ab
de
cde
abcde
Divisive
(DIANA)
Step 4
Step 3
Step 2
Step 1
Step 0

Density-based methods

- The *density* of an object ***o*** can be measured by the number of objects close to ***o.***

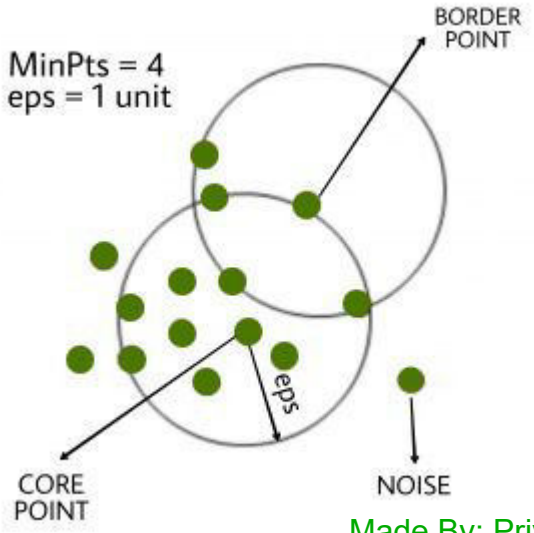
- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

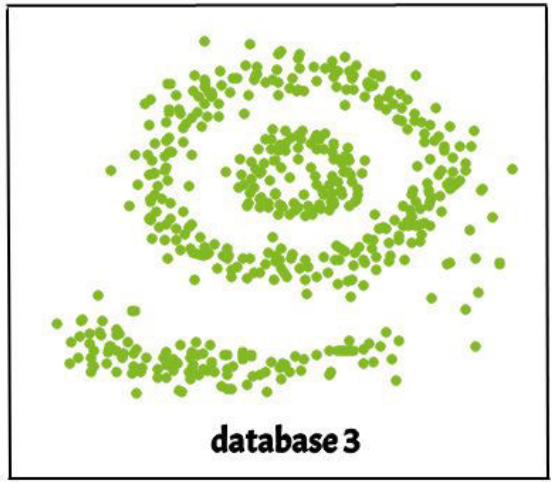
- A user-specified parameter $\in > 0$ is used to specify the radius of a neighborhood we consider for every object.

- The **$\in$-neighborhood** of an object ***o*** is the space within a radius $\in$ centered at ***o***.

Why DBSCAN?

Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.





Parameters Required For DBSCAN Algorithm

1. **eps**: It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to ‘eps’ then they are considered neighbors. If the eps value is chosen too small then a large part of the data will be considered as an outlier. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. The larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

In this algorithm, we have 3 types of data points.

Core Point*: A point is a core point if it has more than MinPts points within eps.*

Border Point*: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.*

Noise or outlier*: A point which is not a core point or border point.*

Grid-Based Clustering

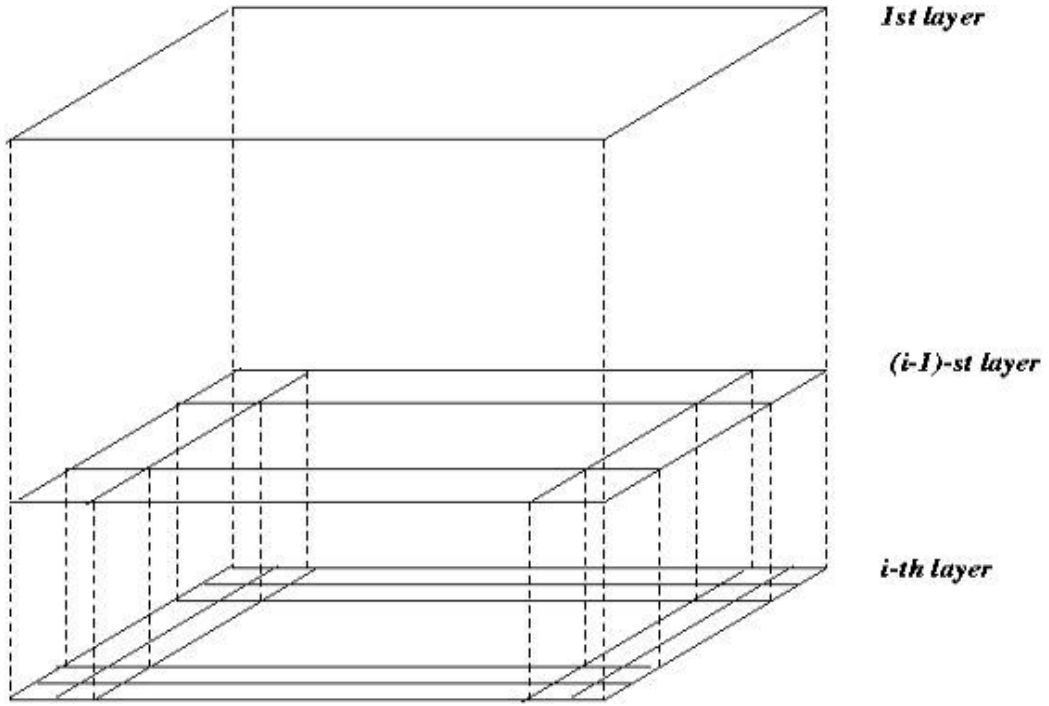
- grid-based clustering method takes a space-driven approach by partitioning the embedding space into cells independent of the distribution of the input objects.
- The grid-based clustering approach uses a multiresolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed.
- The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.

STING: STatistical INformation Grid

STING was proposed by Wang, Yang, and Muntz (VLDB'97).

In this method, the spatial area is divided into rectangular cells.

There are several levels of cells corresponding to different levels of resolution.



For each cell, the high level is partitioned into several smaller cells in the next lower level.

The statistical info of each cell is calculated and stored beforehand and is used to answer queries.

The parameters of higher-level cells can be easily calculated from parameters of lower-level cell

Count, mean, s, min, max

Type of distribution—normal, uniform, etc.

Then using a **top-down** approach we need to answer spatial data queries.

Then start from a pre-selected layer—typically with a small number of cells.

For each cell in the current level compute the confidence interval.

Now remove the irrelevant cells from further consideration.

When finishing examining the current layer, proceed to the next lower level.

Repeat this process until the bottom layer is reached.

Advantages:

It is Query-independent, easy to parallelize, incremental update.

O(K), where K is the number of grid cells at the lowest level.

Disadvantages:

All the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected.

Subject: MCA DMBI 1003

Outlier Detection

Made By: Priyanka Karale

Outliers and Outlier Analysis

- *Outlier detection* (also known as *anomaly detection*) is the process of finding data objects with behaviors that are very different from expectation. Such objects are called **outliers** or **anomalies**.

- An **outlier** is a data object that deviates significantly from the rest of the objects.

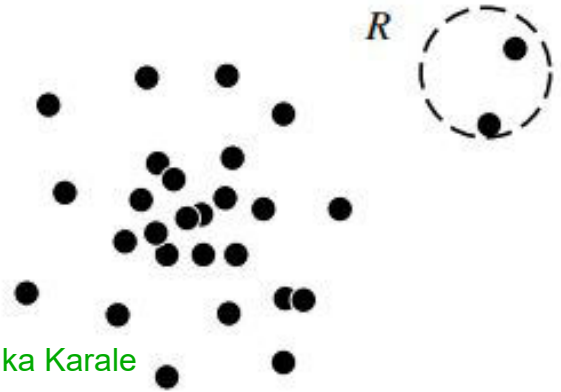
- Outlier detection is important in many applications in addition to fraud detection such as medical care, public safety and security, industry damage detection, image processing, sensor/video network surveillance.

- Clustering finds the majority patterns in a data set and organizes the data accordingly, whereas outlier detection tries to capture those exceptional cases that deviate substantially from the majority patterns.

Contd.

- Outliers are interesting because they are suspected of not being generated by the same mechanisms as the rest of the data.

- For example, by monitoring a social media web site where new content is incoming, novelty detection may identify new topics and trends in a timely manner.



Types of Outliers

Global Outliers

Contextual Outliers

Collective Outliers

- In a given data set, a data object is a **global outlier** if it deviates significantly from the rest of the data set.
- Global outliers are sometimes called *point anomalies*, and are the simplest type of outliers.
- Most outlier detection methods are aimed at finding global outliers.

Contextual Outliers

- In a given data set, a data object is a **contextual outlier** if it deviates significantly with respect to a specific context of the object.

- Contextual outliers are also known as *conditional outliers* because they are conditional on the selected context.

- Therefore, in contextual outlier detection, the context has to be specified as part of the problem definition.

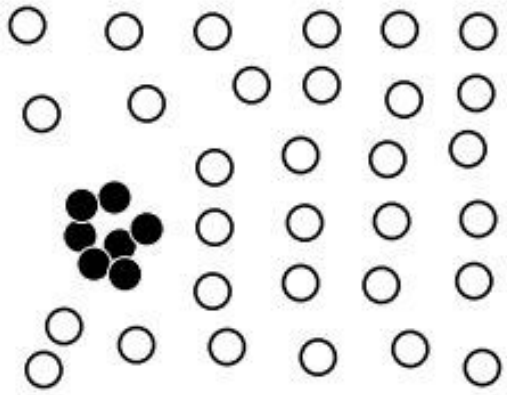
- *For example - "The temperature today is* 28◦*C. Is it exceptional (i.e., an outlier)?"* It depends, for example, on the time and location!

Collective Outliers

- Given a data set, a subset of data objects forms a **collective outlier** if the objects as a whole deviate significantly from the entire data set. Importantly, the individual data objects may not be outliers.

- In the Figure here, the black objects as a whole form a collective outlier because the density of those objects is much higher than the rest in the data set.

- However, every black object individually is not an outlier with respect to the whole data set.



Subject: MCA DMBI 1003

Unit 3 Ends Here

Made By: Priyanka Karale