UNIT-5 [Around 15 marks] (10+5)

Mining Frequent Patterns

Frequent Patterns: [Imp],

Frequent patterns are patterns that appear in a data set frequently. Frequent patterns can be frequent itemsets, frequent subsequences, or frequent substructures. 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 subsequence.

→ A substructure can refer to different structural forms, such as sub-graphs or sub-trees. If a substructure occurs frequently, it is called a frequent structured pattern or frequent substructure.

→ Finding such frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data.

Market Basket Analysis:

It is the earliest form of frequent pattern mining for association rules. It analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets".

The discovery of these associations can help retailers develop marketing strategies by analyzing which items are frequently purchased together by customers. Example: For instance, if customers are buying milk, how likely are they to also buy bread on the same trip? Such information can lead to increased sales by helping retailers do selective marketing and design different store layouts.

Some terms:

Itemset: A set of items is referred to as an itemset. An itemset that contains k items is a k-itemset. The set {computer, antivirus-software} is a 2-itemset.

Support Count (σ/sigma): It is the frequency of occurrence of a itemset.

Frequent Itemset: An itemset whose support is greater than or equal to minimum support threshold. i.e, an itemset that occurs more than minimum specified number.

Closed Itemset: An itemset X is closed in a data set D if there exists no proper super-itemset Y such that Y has the same support count as X in D.

Association Rules: [Imp]

Association rules are if/then statements that help uncover relationships between clearly unrelated data in a relational database or other information repository. An example of an association rule would be "If a customer buys a dozen eggs, he/she is 60% likely to purchase milk."

An association rule has two parts, an antecedent (if part) and a consequent (then part). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Depending on the following two parameters, the important relationships are observed:

Support: It indicates how frequently the if/then relationship appears in the database.

Confidence: It tells about the number of times these relationships have been found to be true. E.g. "If a customer buys a dozen eggs, he/she is 60% likely to purchase milk." Here, 60% is the confidence.

$Support(A \Rightarrow B) = P(A \cup B)$	$Confidence(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)} = \frac{Support}{P(A)}$
Support of association rule $A \Rightarrow B$ is the probability that the database contains both A and B.	Confidence of $A \Rightarrow B$ is conditional probability that the transactions that contains item A also contains item B.

Example 1: Calculate support and confidence of rule bread ⇒ milk considering the example given below:

Transaction ID	Milk	Bread	Butter
1	1	1	0
2	0	0	1
3	0	0	0
4	1	1	1
5	0	1	0

here 1 denotes true
0 denotes false.
Milk र bread दुवैमा
सँगै 1 भएको 2
ठाउमा छ out of
total 5 so 2/5

Solution:

$$\text{Support}(\text{bread} \Rightarrow \text{milk}) = 2/5 = 0.4 = 40\%$$

$$\text{Confidence}(\text{bread} \Rightarrow \text{milk}) = 2/3 = 0.66 = 66\%$$

Milk र bread 2 वटै
ठाउमा सँगै 1 कतिचोटि
छ त्यो / Bread मा
कति चोटि 1 छ त्यो
i.e, 2/3

Example 2: Consider the example below and calculate support and confidence of rule A ⇒ C.

Tid	Items
1	A, B, C
2	A, C
3	A, D
4	B, E, F

A र C कतिवटा
transaction मा आको छ
दुवै त्यो by total no. of Tid
i.e, 2/4

Solution:

$$\text{Support}(A \Rightarrow C) = 2/4 = 0.5 = 50\%$$

$$\text{Confidence}(A \Rightarrow C) = 2/3 = 0.66 = 66\%$$

दुवै A र C कति Tid मा
आयो त्यो by A कति
Tid मा आयो total

OR $\dfrac{\text{Support}}{\text{Probability}(A)}$ can also be done i.e, $0.5/(3/4) = 0.66$.
where, $3/4$ is Probability of A.

#Types of Association Rules: [Imp]

i) **Single Dimensional:** Association rules involving single predicate repeated multiple times are called single dimensional rules. Consider the example below, which is single dimensional association rule:

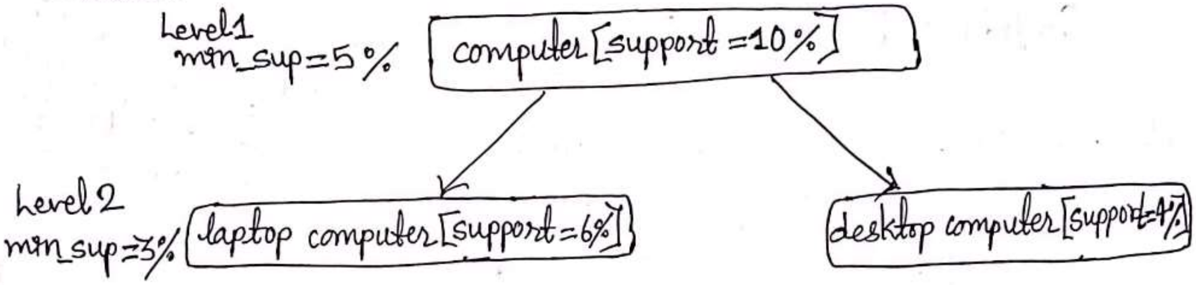
$$\text{buys}(X, \text{"digital camera"}) \Rightarrow \text{buys}(X, \text{"Printer"})$$

i.e, If X buys digital camera then X ~~also buys Printer~~ is likely to buy Printer.

ii) **Multidimensional:** Association rules that involve two or more predicates are referred as multidimensional association rules. Consider the rule given below that contains three predicates (age, occupation, and buys), each of which occurs only once in the rule.

age (X, "20:::29") ^ occupation (X, "student") => buys (X, "laptop").

iii) **Multilevel:** Association rules generated from mining data at multiple levels of abstraction are called multilevel association rules.



iv) **Quantitative:** Database attributes can be ~~categorical~~ quantitative. These attributes have a finite number of possible values, with no ordering among the values (e.g, occupation, brand, color). Quantitative attributes are numeric and have an implicit ordering among values (e.g, age, price).

Finding Frequent Itemset:

Apriori र FP-Growth मध्ये एउटा सोध्छ 10 marks मा [V.V Imp]

1) Apriori Algorithm:

It is a classic algorithm used in data mining for learning association rules. Mining association rules basically means finding the items that are purchased together more frequently than others. The name of the algorithm is based on the fact the the algorithm uses prior knowledge of frequent itemset properties.

→ Apriori employs an iterative approach known as a level-wise search, where frequent k-itemsets are used to explore frequent (k+1) itemsets.

→ First, the set of frequent 1-itemsets that satisfy minimum support is found by scanning the database. The resulting set is denoted by L_1.

→ Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found.

→ The finding of each L_k requires one full scan of database. To improve efficiency of level-wise generation of frequent itemsets, Apriori property is used. Apriori property states that any subset of frequent itemset must be frequent.

Example: Consider the database, consisting of 9 transactions. Suppose min. support count required is 2 (i.e, min-sup = 2/9 = 22%). Let minimum confidence required is 70%. Find out the frequent itemsets using Apriori algorithm. Then generate association rules using min. support & min. confidence.

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

Solution:

Step1: Generate 1-itemset Frequent Pattern:

→ Initially, each item is a member of the set of candidate (C1) itemset. Next compute support count for each candidate itemset in C1. Frequent 1-itemsets (L1) is then determined by using minimum support.

Scan database for count of each candidate →

Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

दिएको database मा I1, I2, I3, ... कति चोटि आको छ त्यो गनेर लेख्ने

Compare candidate support count with minimum support count →

Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L1

min. Support Count 2 छ so ≥2 भाको value लिएको / यसमा सबै छन् so सबै लिएको

Step2: Generate 2-itemset Frequent Pattern:

→ Use L1 Join L1 to generate a candidate set of 2-itemsets (C2). Next, compute support count for each candidate itemset in C2. Frequent 2-itemsets (L2) is then determined by using minimum support.

Generate C_2 Candidates from L1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C2

2 item को set बनाउने 1 itemset को I1 to I5 (i.e, L1) को use गरि / Set भएपछि set भित्र save value repeat नगराइ लेख्ने for e.g. {I1, I1} can not be placed. Similarly {I_1, I2} and {I2, I1} are treated as same so written only once.

(First item मुनि हेर भयो, set मा / 2nd को लागि माथि हेर्नु पर्न)

Scan database for count of each candidate

I_1 I_2 दुवै कति transaction मा आको छ database मा यो हेरेको

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C2

Compare C_2 with minimum Sup. Count

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L2

min. Sup = 2 छ so 2 र 2 भन्दा बढिलाई लिएर बाँकि discard गरेको

Step3: Generate 3-itemset Frequent Pattern.

→ Generate 3-itemset as C3 = L2 Join L2. Then use Apriori property to prune the members of C3. Finally generate frequent 3-itemset using minimum support.

Generate C3 Candidates from L2 then use Apriori to purne C3

Itemset
{I1, I2, I3}
{I1, I2, I5}

C3

Combine गर्दा L2 को सक्पोहिमा 2 set मात्र लिन मिल्यो संगै 3 or more लिन मिल्न

Then आएको candidate list लाई Apriori property use गर्ने. for e.g. {I_1, I_2, I_3} को लागि subsets {I_1, I_2}, {I_1, I_3} र {I_2, I_3} हुन्छन् अब यो सबै subset L2 को Itemset मा भए राख्ने / कुनै एउटा पनि set नभए discard गर्ने / Possible {I_1, I_2, I_4} नि छ but यसरि discard भाको अरु

Scan for count of each candidate

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

C3

Compare with min. Sup. Count

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L3

Step4: Generate 4-itemset Frequent Pattern:

→ The algorithm uses L3 Join L3 to generate a candidate set of 4-itemsets (C4). Although the join results in {I1, I2, I3, I5}, this itemset is pruned since its subset {I2, I3, I5} is not frequent. Thus $C4 = \emptyset$, and algorithm terminates, having found frequent itemsets: {{I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I1, I2, I3}, {I1, I2, I5}}.

Step5: Generating Association Rules from Frequent Itemsets:

→ For each frequent itemset l, generate all nonempty subsets of l. For every nonempty subset s of l, output the rule $s \Rightarrow l-s$ if confidence of the rule is greater or equal to minimum confidence.

→ We had L = {{I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I1, I2, I3}, {I1, I2, I5}}.

सजिलो को लागि सानो लिए नि भयो for e.g. {I1, I5}

→ Let's take l = {I1, I2, I5}. It's all nonempty subsets are {I1, I2}, {I1, I5}, {I2, I5}, {I1}, {I2}, {I5}.

min. 70% given in question

R1: $I1 \wedge I2 \Rightarrow I5$ Confidence = 2/4 = 50%, R1 is Rejected.
R2: $I1 \wedge I5 \Rightarrow I2$ Confidence = 2/2 = 100%, R2 is Selected.
R3: $I2 \wedge I5 \Rightarrow I1$ Confidence = 2/2 = 100%, R3 is Selected.
⋮

In this way selected ones are the strong association rules. We need to repeat same process for every frequent itemset.

Limitations of Apriori Algorithm:

→ It needs to generate a huge number of candidate sets.

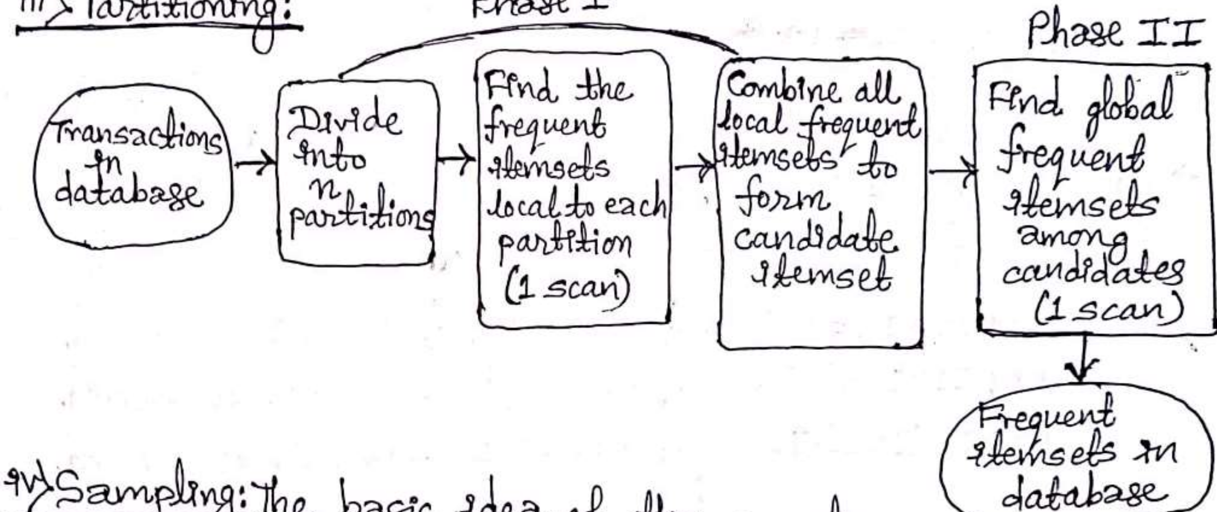
→ It may need to repeatedly scan the whole database and check a large set of candidates by pattern matching.

Improving Efficiency of Apriori Algorithm: [Imp] may be asked for 5 marks

i) **Hash Based Technique:** A hash-based technique can be used to reduce the size of the candidate k-itemsets, for $k > 1$.
For example, when scanning each transaction in the database to generate the frequent 1-itemsets, we can generate all the 2-itemsets for each transaction, hash them into the different buckets of a hash table.

ii) **Transaction reduction:** A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k+1)-itemsets. Therefore, such a transaction can be removed or marked from further consideration.

iii) **Partitioning:**



iv) **Sampling:** The basic idea of the sampling approach is to pick a random sample S of the given database D, and then search for frequent itemsets in S instead of D. In this way, we trade off some degree of accuracy against efficiency.

The S sample size is such that the search for frequent itemsets in S can be done in main memory, and so only one scan of the transactions in S is required overall. Because we are searching for frequent itemsets in S rather than in D, it is possible that we will miss some of the global frequent itemsets.

2) FP: Growth Algorithm:

The FP-Growth Algorithm is an alternative way to find frequent itemsets without using candidate generations, thus improving performance.

It uses a divide-and-conquer strategy. It uses special data structure named frequent-pattern tree (F-P tree), which retains the itemset association information. It uses two step approach:

Step1: Build a compact data structure called the FP-tree. It is built using two passes over the data-set.

Step2: Extracts frequent itemsets directly from the FP-tree by traversing through FP-Tree.

Example: Consider the database, consisting of 9 transactions. Suppose min. support count required is 2 (i.e, min-sup = 2/9 = 22%). Let minimum confidence required is 70%. Find out the frequent itemsets using FP-growth algorithm. Then generate association rules using min. support & min. confidence.

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

Solution:

Step1: Generate 1-itemset frequent pattern and then sort 1-itemset frequent pattern by using minimum support count.

Scan database for count of each item →

Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count →

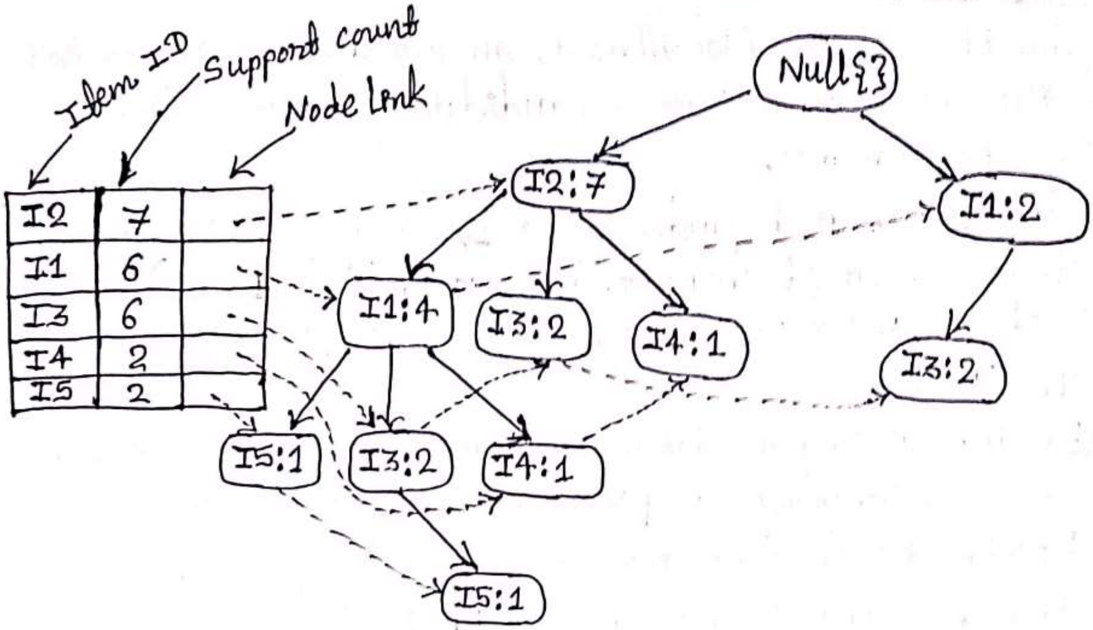
Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

Sort based on Sup. Count →

Itemset	Sup. Count
{I2}	7
{I1}	6
{I3}	6
{I4}	2
{I5}	2

Step2: Construct FP-Tree



Step3: Mine FP-Tree by using Conditional Pattern Bases

Conditional FP-tree लाई Item संग combine गर्दा

Item	Conditional Pattern Base	Conditional FP-tree	Frequent Patterns 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, I1:2}, {I2:2}, {I1:2}}	⟨I2:4, I1:2⟩, ⟨I1:2⟩	{I2, I3:4}, {I1, I3:4}, {I2, I1, I3:2}
I1	{{I2:4}}	⟨I2:4⟩	{I2, I1:4}

Step4: Generate Association Rules

(Same as in Apriori algorithm).

कुनै एउटा लिनो from frequent patterns. For e.g. {I1, I2, I5} as we look in Apriori

From Association Mining to Correlation Analysis:

→ Most association rule mining algorithms employ a support-confidence framework.

→ Although minimum support and confidence thresholds helps to exclude uninteresting rules, many rules so generated are not still interesting to the users.

→ This is especially true when mining at low suppord thresholds.

→ Support-confidence framework can be supplemented with additional interestingness measures based on statistical significance and correlation analysis.

→ Some association rules {A ⇒ B} that satisfy minimum support and threshold may be uninteresting if 'A' and 'B' are negatively correlated with each other. This type of rules can be excluded by using the measure correlation analysis. Lift is the measure that measures correlation.

Lift: The lift between the occurrence of A and B can be measured as below:

$$\text{Lift}(A,B) = \frac{P(A \cup B)}{P(A)\, P(B)} = \frac{\text{Confidence}(A \Rightarrow B)}{\text{Support}(B)}$$

If the resulting value of above equation is less than 1, then the occurrence of A is negatively correlated with the occurrence of B. If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other. If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.