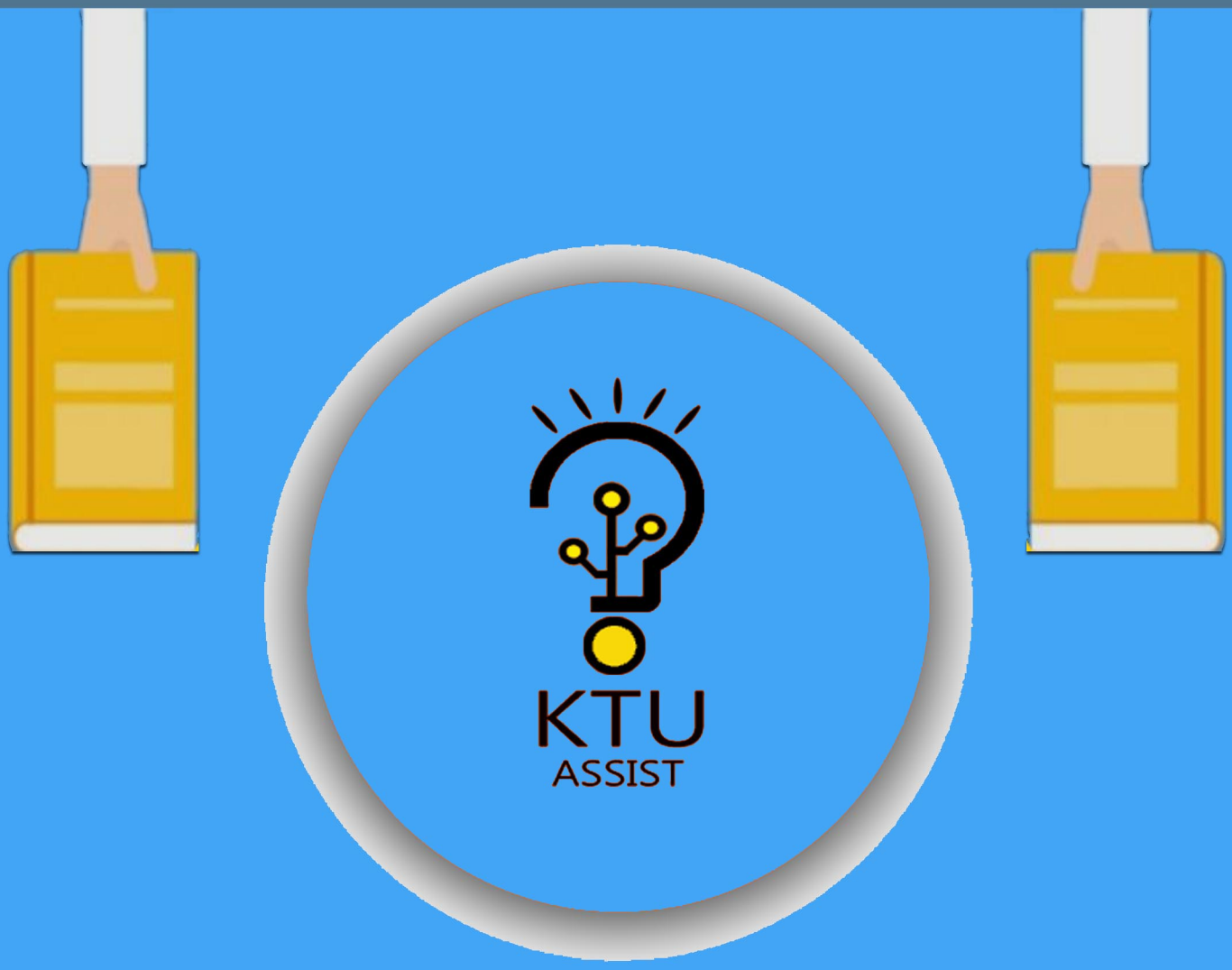
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS




**a complete app for ktu students**

Get it on Google Play

www.ktuassist.in

# Module 5: Syllabus

**Association Rules Mining**: Concepts, Apriori and FP-Growth Algorithm.
**Cluster Analysis**: Introduction, Concepts, Types of data in cluster analysis, Categorization of clustering methods.
**Partitioning method:** K-Means and K-Medoid Clustering

# Association Rules Mining: Concepts

***Association rule mining finds interesting association or correlation relationships among a large set of data items***. With massive amounts of data continuously being collected and stored in databases, many industries are becoming interested in mining association rules from their databases. For example, the discovery of interesting association relationships among huge amounts of business transaction records can help catalog design, cross-marketing, lossleader analysis, and other business decision making processes.

A typical example of association rule mining **is market basket analysis**. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets" (Figure 6.1). The discovery of such 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 to also buy bread (and what kind of bread) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers to do selective marketing and plan their shelf space. For instance, placing milk and bread within close proximity may further encourage the sale of these items together within single visits to the store.

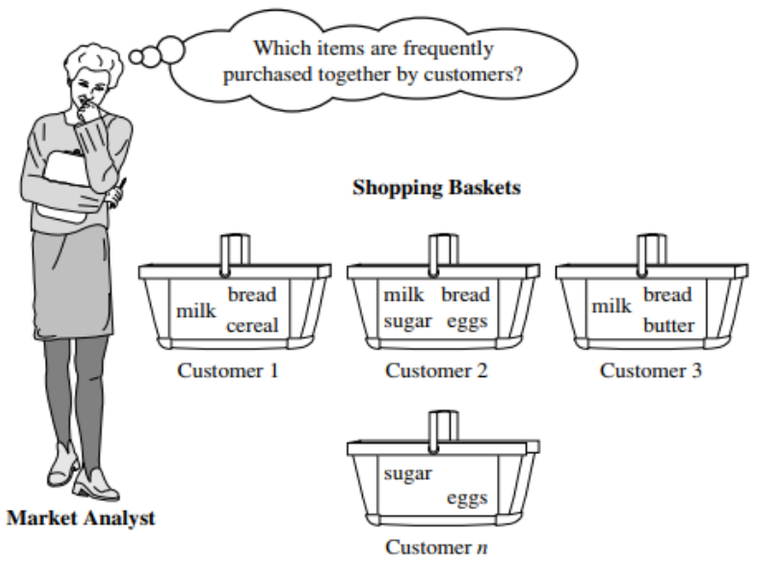



**Figure 6.1** Market basket analysis.

## Basic concepts

## 1. Transaction database

DOWNLOADED FROM KTUASSIST.IN

- Let I = {I1, I2,..., Im} be an itemset {SET OF ITEMS} .
- Let D, the task-relevant data, be a set of database transactions T where each transaction $t_i$ is a nonempty item set such that $t_i \subseteq I$.
- Transaction database, T={$t_1$, $t_2$,.....,$t_n$}
- A transaction database T contains "n" transaction.
- Each transaction is associated with an identifier, called a TID.

**Table 6.1** 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 |

*Fig: Sample Transaction database*

## 2. Association rule

***An association rule is an implication of the form X → Y, where X⊆ I, Y ⊆ I, and X∩Y = φ.***

- Meaning that whenever X appears in the Transaction Y also tends to appear.
- X and Y are single item or set of Items.
- Example: {Milk} → {Bread},{Computer, CD Drive}→ {CD}
- Usually we prefer association rules that satisfies required support and confidence.

## 3. Support, Confidence and Lift

- Support and confidence are used to measure ***interestingness of the rule***
- **Support(X)** is the number of times X appears in the transaction divided by total number of transaction.

**Support(X)** =(No. of times X appears)/( total number of transaction)=**P(X)**

**Support(X,Y)** =(No. of times X and Y appears together)/( total number of transaction)=**P(X∩*Y*)**

- **Confidence** of the association rule ***X → Y*** *is defined as the ratio of support of X and Y together to the support of X*

**Confidence** ( ***X → Y)=*** *( support of X and Y together)/( support of X)=* **P(X∩*Y*)/ P(X)= P(*Y*|*X*)**

- **Lift** is used to measure the power of association between items that are purchased together.

**Lift( *X → Y)=* P(X∩*Y*)/ P(X)= P(*Y*|*X*)/ P(Y)**

## 4. Frequent Items

DOWNLOADED FROM KTUASSIST.IN

Items that frequently appeared in the transactions are called Frequent Items. Usually we select frequent items based on minimum support count.

## Association Rules Mining: Algorithms

In general, association rule mining can be viewed as a two-step process:

Step 1. **Find all frequent itemsets**: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count.

2. **Generate strong association rules from the frequent itemsets**: By definition, these rules must satisfy minimum support and minimum confidence.

## 1. Apriori Algorithm

Apriori algorithm consist of two parts

**Part 1: Find all frequent item sets:** Item sets that exceeds minimum support

**Part 2: Generating strong association rules from frequent** itemsets ( Association rule that meet minimum confidence)

### PART 1: Finding frequent itemsets

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space.

| **The Apriori property**. All non empty subsets of a frequent itemset must also be frequent. |
|---|

This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well.

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

First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to fnd L2, the frequent 2-itemsets, which is used to fnd L3, and so on, until no more frequent k-itemsets can be found. The finding of each Lk requires one full scan of the database.

A two-step process is followed to find frequent items consisting of **join** and **prune** actions.

1. **The join step**: To find $L_k$, a set of **candidate** $k$-itemsets is generated by joining $L_{k-1}$ with itself. This set of candidates is denoted $C_k$. Let $l_1$ and $l_2$ be itemsets in $L_{k-1}$. The notation $l_i[j]$ refers to the $j$th item in $l_i$ (e.g., $l_1[k-2]$ refers to the second to the last item in $l_1$). For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$-itemset, $l_i$, this means that the items are sorted such that $l_i[1] < l_i[2] < \cdots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of $L_{k-1}$ are joinable if their first $(k-2)$ items are in common. That is, members $l_1$ and $l_2$ of $L_{k-1}$ are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \cdots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining $l_1$ and $l_2$ is $\{l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]\}$.

DOWNLOADED FROM KTUASSIST.IN

**The prune step:** Ck is a superset of Lk , that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in Ck . A database scan to determine the count of each candidate in Ck would result in the determination of Lk (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk). Ck , however, can be huge, and so this could involve heavy computation. To reduce the size of Ck , the Apriori property is used as follows. Any (k − 1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. Hence, if any (k − 1)-subset of a candidate k-itemset is not in $L_{k-1}$, then the candidate cannot be frequent either and so can be removed from Ck .

## PART 2: Generating Association Rules from Frequent Itemsets

Once the frequent itemsets from transactions in a database D have been found, it is straightforward to generate strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence). This can be done using Equation for confidence.

**Confidence** $(A \Rightarrow B) = (\text{support of } A \text{ and } B \text{ together}) / (\text{support of } A) = P(A \cap B) / P(A) = P(B|A)$

**Method:**

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                  ∧ ... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

**Figure 6.4** Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

DOWNLOADED FROM KTUASSIST.IN

**Example Problem 1:**

Based on the AllElectronics transaction database, D, of Table 6.1. Find association rule present in the database by considering Minimum support of 20% and confidence of 75%?.

Based on the AllElectronics transaction database, D, of Table 6.1. Find association rule present in the database by considering Minimum support of 20% and confidence of 75%?.

**Table 6.1** 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 |

**Solution:**

There are nine transactions in this database, that is, n = 9.

Set of Items, I={I1,I2,I3,I4,I5}

## Part 1: Frequent Item Set generation

**Step 1-** E item is a member of the set of candidate 1-itemsets, C1. The algorithm simply scans all of the transactions to count the number of occurrences of each item.

Scan *D* for count of each candidate →

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

**Step 2-** The set of frequent 1-itemsets, L1, can then be determined Minimum support count required is 2, that is, min sup = 2. (Here, support is 2/9 = 22%.). All Items in C1 are qualified

Compare candidate support count with minimum support count →

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

DOWNLOADED FROM KTUASSIST.IN

Generate $C_2$ candidates from $L_1$ →

$C_2$

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

**Step 4.** Next, the transactions in D are scanned and the support count of each candidate itemset in C2 is calculated

Scan $D$ for count of each candidate →

$C_2$

| 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 |

Step 5: The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.

Compare candidate support count with minimum support count →

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

| **C3** |
|---|
| Items |
| {I1, I2, I3}, |
| {I1, I2, I5}, |
| {I1, I3, I5}, |
| {I2, I3, I4}, |
| {I2, I3, I5} |
| {I2, I4, I5} |

**Step 7.** The transactions in D are scanned to determine support count of C3

DOWNLOADED FROM KTUASSIST.IN

| C3 | |
|---|---|
| Items | Sup count |
| {I1, I2, I3}, | 2 |
| {I1, I2, I5}, | 2 |
| {I1, I3, I5}, | 1 |
| {I2, I3, I4}, | 0 |
| {I2, I3, I5} | 1 |
| {I2, I4, I5} | 0 |

**Step 8:** Find 3 frequent item set, L3. ( candidate 3-itemsets in C3 having minimum support.)

Compare candidate support count with minimum support count →

$L_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

**Step 9:** Generation of the set of the candidate 4-itemsets, C4. From the join step, we first get C4 = L3 ⋈ L3

| C4 | |
|---|---|
| Items | Sup count |
| {I1, I2, I3,I5 }, | |

**Step 10:** Find 4 frequent item set, L4. ( C3 having minimum support)

NULL list. Stop Frequent Item set generation.

## PART 2: Generating Association Rules from Frequent Item sets

- Minimum confidence required 75 %
- **Confidence** ( $X \rightarrow Y$)= *( support of X and Y together)/( support of X)*= $P(X \cap Y) / P(X) = P(Y/X)$

Consider L3={**{I1, I2, I3},{I1, I2, I5}**}

Association rules formed from frequent Item set **{I1, I2, I3},**

I1^I2→ I3, confidence = $2/4 = 50\%$
I1^I3→ I2, confidence = $2/4 = 50\%$
I2^I3→ I1, confidence = $2/4 = 50\%$
I1→ I2^I3, confidence = $2/6 = 33\%$
I2→ I1^I3, confidence = $2/7 = 29\%$
I3→ I1^I2, confidence = $2/6 = 33\%$

Association rules formed from frequent Item set **{I1, I2, I5},**

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

DOWNLOADED FROM KTUASSIST.IN

**Association rules that satisfy minimum confidence are {I1^I5 I2 ,I2^I5 I1, I5 I1^I2}**

**Major drawbacks of Apriori Algorithm**

1. The number of candidate itemset grows quickly and result in large candidate set. For example, if there are $10^4$ frequent 1-itemsets (L1), the Apriori algorithm will need to generate more than $10^7$ candidate 2-itemsets(C2).
2. The Apriori algorithm requires many scans of the database.

# FP-growth Association Rule Mining Algorithm

adopts a divide-and-conquer strategy as follows.

First, it compresses the database representing frequent items into a frequent pattern tree, or FP-tree, which retains the itemset association information. It then divides the compressed database into a set of conditional databases (a special kind of projected database), each associated with one frequent item or

to be searched, along with the "growth" of patterns being examined.

**Note**: FPgrowth algorithm find set of frequent itemsets without candidate generation process

**Advantages of FP growth algorithm:-**

1. Faster than apriori algorithm

2. No candidate generation

3. Only two passes over dataset

**Disadvantages of FP growth algorithm:-**

1. FP tree may not fit in memory

2. FP tree is expensive to build

Fp growth algorithm example

# FP-growth Association Rule Mining Algorithm

**Algorithm:** FP growth. Mine frequent itemsets using an FP-tree by pattern fragment growth.

**Input:** D- a transaction database; min_sup- the minimum support count threshold.

**Output:** The complete set of frequent patterns.

DOWNLOADED FROM KTUASSIST.IN

**Method**:

1. The FP-tree is constructed in the following steps:
   (a) Scan the transaction database $D$ once. Collect $F$, the set of frequent items, and their support counts. Sort $F$ in support count descending order as $L$, the *list* of frequent items.
   (b) Create the root of an FP-tree, and label it as "null." For each transaction *Trans* in $D$ do the following.
   Select and sort the frequent items in *Trans* according to the order of $L$. Let the sorted frequent item list in *Trans* be $[p|P]$, where $p$ is the first element and $P$ is the remaining list. Call **insert_tree**($[p|P]$, $T$), which is performed as follows. If $T$ has a child $N$ such that $N.item\text{-}name = p.item\text{-}name$, then increment $N$'s count by 1; else create a new node $N$, and let its count be 1, its parent link be linked to $T$, and its node-link to the nodes with the same *item-name* via the node-link structure. If $P$ is nonempty, call **insert_tree**($P$, $N$) recursively.
2. The FP-tree is mined by calling **FP_growth**(*FP_tree*, *null*), which is implemented as follows.

```
procedure FP_growth(Tree, α)
(1)  if Tree contains a single path P then
(2)     for each combination (denoted as β) of the nodes in the path P
(3)        generate pattern β ∪ α with support_count = minimum support count of nodes in β;
(4)  else for each a_i in the header of Tree {
(5)     generate pattern β = a_i ∪ α with support_count = a_i.support_count;
(6)     construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)        call FP_growth(Tree_β, β); }
```

**Figure 6.9** FP-growth algorithm for discovering frequent itemsets without candidate generation.

**Example Problem:**

Based on the AllElectronics transaction database, D, of Table 6.1. Find frequent pattern present in the database using FP Growth Algorithm by considering minimum support of 20% and confidence of 75%?.

**Table 6.1** 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 |

DOWNLOADED FROM KTUASSIST.IN

**Solution:**

**FP-Tree corresponding to given Dataset is shown in below figure**

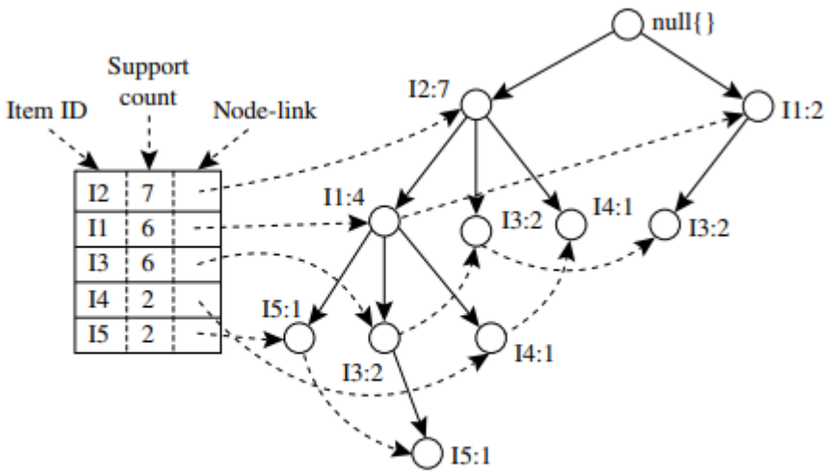



**Figure 6.7** An FP-tree registers compressed, frequent pattern information.

**Frequent patterns generated from the given Dataset is shown in below figure**

**Table 6.2** Mining the FP-Tree by Creating Conditional (Sub-)Pattern Bases

| *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} |

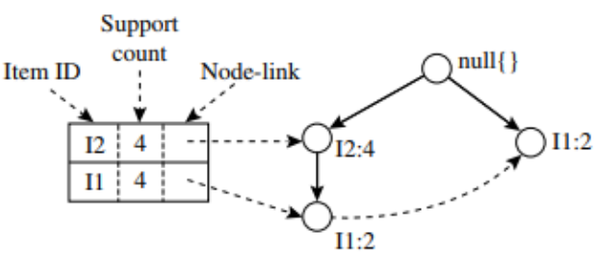



**Figure 6.8** The conditional FP-tree associated with the conditional node I3.

DOWNLOADED FROM KTUASSIST.IN

## What is Cluster Analysis?

- **Cluster:** A collection of data objects
  - **similar** (or related) to one another within the same group
  - **dissimilar** (or unrelated) to the objects in other groups
  -
- **Cluster analysis** (or *clustering*, *data segmentation, …*)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning:** no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)

- Typical applications of Cluster analysis
  - As a **stand-alone tool** to get insight into data distribution
  - As a **preprocessing step** for other algorithms

## Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- Information retrieval: document clustering
- Land use: Identification of areas of similar land use in an earth observation database
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults
- Climate: understanding earth climate, find patterns of atmospheric and ocean
- Economic Science: market research

## Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

## Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - **high intra-class similarity**: cohesive within clusters
  - **low inter-class similarity**: distinctive between clusters

- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and

DOWNLOADED FROM KTUASSIST.IN

- Its ability to discover some or all of the hidden patterns

## Major Clustering Approaches

In general, the major clustering methods can be classified into the following categories.

Partitioning Methods
Hierarchical Methods
Density-Based Methods
Grid-Based Methods

**Partitioning approach:**

- Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
- Given a database of n objects, a partitioning method constructs k partitions of the data object, where each partition represents a cluster .
- That is, it clusters the data into k groups, which together satisfy the following requirements:
  (1) each group must contain at least one object, and
  (2) Each object must belong to exactly one group
- Typical methods: **k-means, k-medoids, CLARANS**

**Hierarchical approach:**

- A hierarchical method creates a hierarchical decomposition of the given set of data objects.
- A hierarchical method can be classified as being either agglomerative or divisive
- **The agglomerative approach**, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds.
- **The divisive approach**, also called the top-down approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.
- Typical methods: **Diana, Agnes, BIRCH, CAMELEON**

**Density-based approach:**

- Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical¯shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.
- Density-based clustering methods have been developed based on the notion of **density.**
- Based on connectivity and density functions
- Typical methods: **DBSACN, OPTICS, DenClue**

**Grid-based approach:**

DOWNLOADED FROM KTUASSIST.IN

- Grid-based methods quantize the ***object space into a finite number of cells that form a grid structure.***
- All of the clustering operations are performed on the grid structure (i.e., on the quantized space).
- The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.
- Typical methods: STING, WaveCluster, CLIQUE

- **Model-based:**
  - Model-based methods hypothesize a ***model for each of the clusters*** and find the best fit of the data to the given model.
  - Typical methods: **Expectation-Maximization (EM), SOM, COBWEB**
- Frequent pattern-based:
  - Based on the analysis of ***frequent patterns***
  - Typical methods**: p-Cluster**
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: **COD (obstacles), constrained clustering**

DOWNLOADED FROM KTUASSIST.IN

*try it now*

A KTU
STUDENTS
PLATFORM

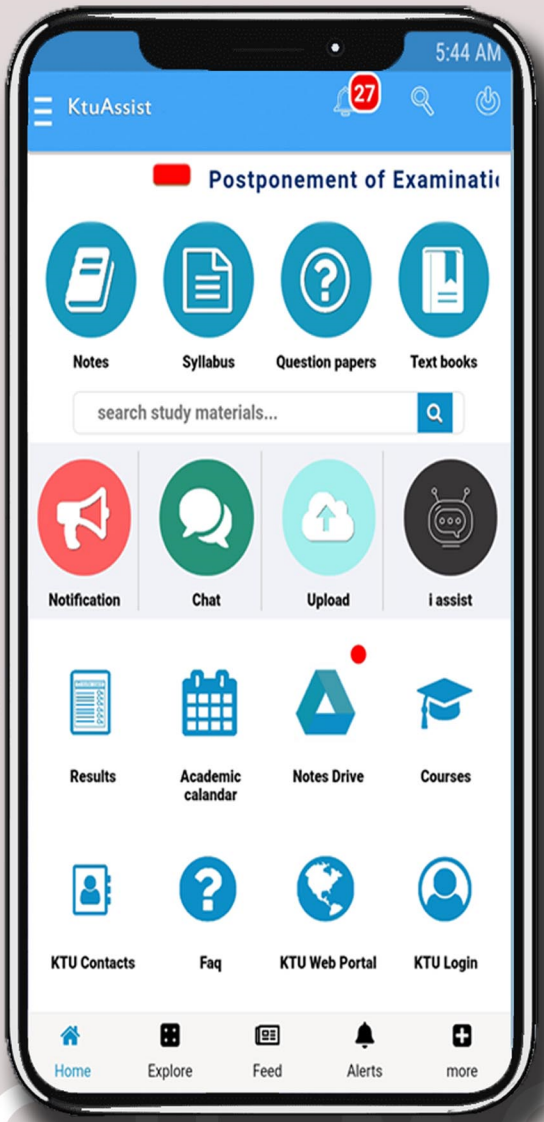
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION




DOWNLOAD
IT
FROM
GOOGLE PLAY

CHAT

A FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist