UNIT14 ASSOCIATION RULES

Structure

14.1 INTRODUCTION

Imagine a salesperson at a shop. If you buy a product from the shop, the salespersonrecommends you more items related to the product you have purchased. He may also suggest you about the items that are frequently bought with the product you have purchased. The salesperson may also try to figure out your choices about the product you observe at the shop and may recommend you further. One of the common examples of the situation is **market basket analysis** as illustrated in two cases in Figure 1. If you add bread and butter to your basked at the store, the salesperson may recommend you add cookies, eggs, milk to your shopping cart too. Similarly, if customer puts vegetables like onion and potato in their cart, the salesman at shop may suggest adding other vegetables like tomato to the basket. If salesman notices a male customer, then he may suggest adding beer too from his historical analysis of male customer preferring to buy beer as well.

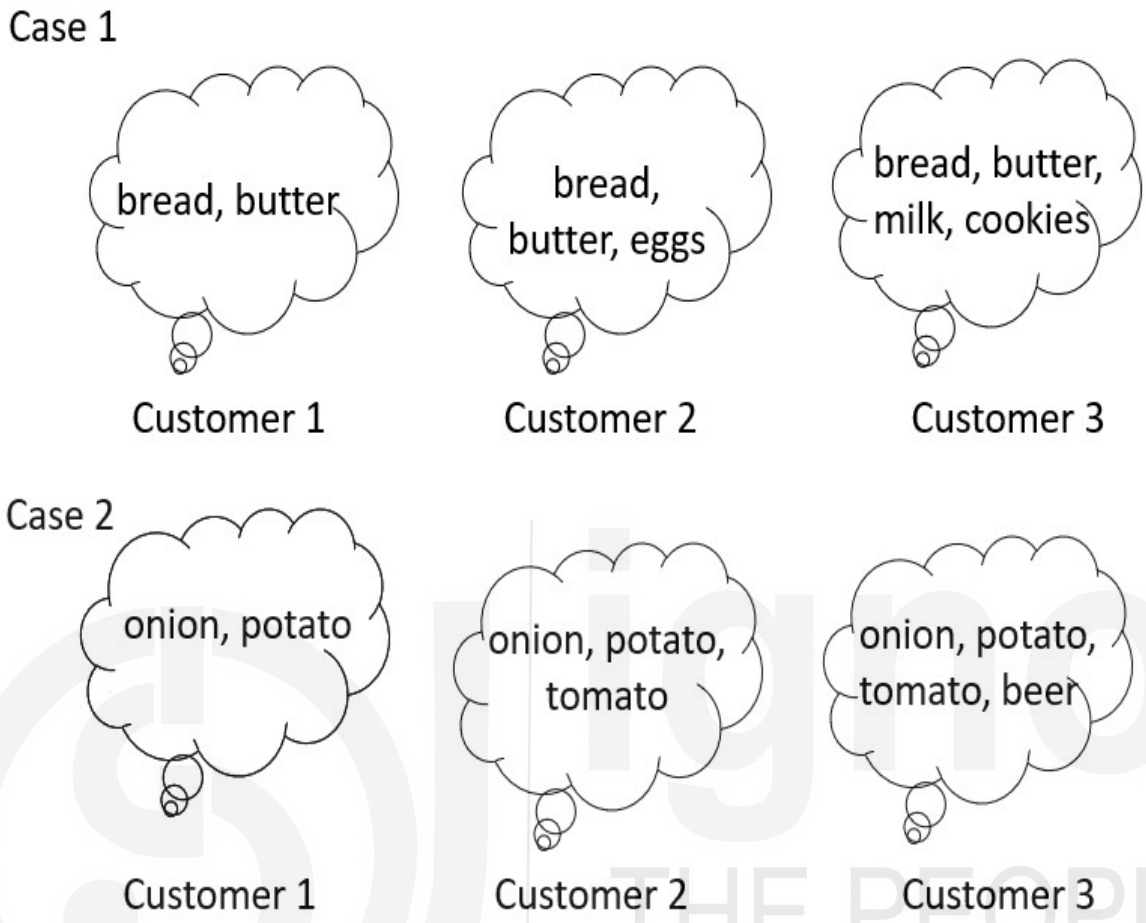


Figure 1: Two different cases of Market Basket Analysis

The salesperson thus analyses the purchasing habits of the customers and tries to analyze the correlations among the items/products that are frequently bought together by them. The analysis of such correlationshelps the retailers to develop marketing strategies to increase sale of products in the shop. Discovering the correlations among all the items sold in a shop help businessman in making decisions regarding designing catalogue, organizing store, and customer shopping analysis.

OBJECTIVES

After going through this unit, you should be able to:

- Understand the purpose of association rules.
- Understand the purpose of pattern search and,
- Describe various algorithms for pattern search

14.3 WHAT ARE ASSOCIATION RULES?

In machine learning, association rules are one of the important concepts that is widely applied in problems like market basket analysis. Consider a supermarket, where all the related items such as grocery items, dairy items, cosmetics, stationary items etc are kept together in same aisle. This helps the customers to find their required items timely. This further helps them to remember the items to purchase they might have forgotten or to they may like to purchase if suggested. Association rules thus enable one to corelate among various products from a huge set of available items. Analysing the items customer buy together also helps the retailers to identify the items they can offer on discount. For example, retailer selling baby lotion and baby shampoo on MRP, but offering a discount on their combination. Customer who wished to buy only shampoo or only lotion, may now think of buying the combination. Other factors too can contribute to the purchase of combination of products. Another strategy can keep related products on the opposite ends of the shelf to prompt the customer to scan through the entire shelf hoping that he might add a few more items to his cart.

It is important to note that the association rules do not extract the customer's preference about the items but find the relations among the items that are generally bought together by them. The rules only identify the frequent associations between the items. The rules work with an antecedent (if) and a consequent (then), both connecting to the set of items. For example, if a person buys pizza, then he may buy a cold drink too. This is because there is a strong relation between pizza and cold drink. Association rules help to find the dependency of one item on other by consider the history of customer's transaction patterns.

14.3.1 Basic Concepts

There are few terms that one should understand before understanding the algorithm.

a. **k-Itemset:** It is a set of *k*items. For example, 2-itemset can be {pencil, eraser} or {bread, butter} etc., 3-itemset can be {bread, butter, milk}.

b. **Support:** Frequency of appearance of an item appears in all the considered transactions is called as the support of an item. Mathematically, support of an item *x* is defined as:

$$support(x) = \frac{Number\ of\ transactions\ containing\ x}{Total\ number\ of\ considered\ transctions}$$

c. **Confidence:** Confidence is defined as the likelihood of obtaining item *y* along with an item *x*. Mathematically, it is defined as the ratio of frequency of transactions containing items x and *y* to the frequency of transactions that contained item *x*.

$$confidence(y) = \frac{Number\ of\ transactions\ containing\ x\ and\ y}{Number\ of\ considered\ containing\ x}$$

Confidence can also be defined as probability of occurrence of y, given probability of occurrence of x.

$$confidence(x => y) = P(y/x)$$

where x is antecedent, and y is a consequent. In terms of support, confidence can be described as:

$$confidence(y) = \frac{support\ (x \cup y)}{support(x)}$$

d. **Frequent Itemset:** An item whose support is at least the minimum support threshold is known as a *frequent itemset*. For example, let minimum support threshold is 10, then an item set with support score 11 is a frequent itemset but an item set with support score 9 is not.

14.3.2 Association Rules: Binary Representation

Let I = {I_1, I_2, I_3……..I_n} be a set of n items and T = {T_1, T_2, T_3……..T_t} be a set of t transactions, where each transaction T_i contains a not null set of items purchased by a customer such that $T_i \subseteq I$. Let each item I_i in the store be represented by a binary variable, B. The variable takes up the value 0 or 1, representing the absence or presence of item at the store.

$$B(i) = 1, \text{if an item } I_i \text{ is available at the store}$$
$$0, \text{otherwise}$$

For example, consider a set of four transactions T1, T2, T3 and T4 with the following items:

T1 = {milk, cookies, bread},
T2 = {milk, bread, egg, butter},
T3 = {milk, cookies, bread, butter}, and
T4 = {cookies, bread, butter}.

The binary representations for the transaction set are shown in Table 1.

Table1: Binary representation of the transactions

Transaction id	Milk	Cookies	Bread	Butter	Egg
T1	1	1	1	0	0
T2	1	0	1	1	1
T3	1	1	1	1	0
T4	0	1	1	1	0

The binary variable can also be used to analyze the purchasing patterns of the customers. One can analyze a basket in terms of binary values of the items that customer has purchased. Such items are said to frequently bought together or associated to each other. These binary patterns are represented in the form of association rules. For example, the customer who buys milk, also tend to buy bread at the same time. This can be represented using support and confidence as:

Milk => Bread with support as 75% (or value as 0.75) and confidence as 100% (or value as 1).

Support(milk) = $\frac{transactions\ containing\ milk}{total\ number\ of\ transactions} = \frac{3}{4} = 0.75$

Confidence (bread) = $\frac{support(milk,bread)}{support(milk)} = \frac{\frac{3}{4}}{0{\cdot}75} = \frac{0.75}{0.75} = 1$

This means that out of all the purchases made at the store, 10% of the times, whenever milk is purchased, bread is also purchased together. Confidence 100% implies that out of all the customers who bought milk, all of them also bought bread. Thus, support and confidence are the two important rules to show the interest in items. Thus, in this case association rules are important to consider if they satisfy the minimum threshold of support and confidence. These thresholds can be set by the experts.

Let A and B be the two set of transactions such that A ⊆ T and B ⊆ T, such that A ≠Ø, B ≠Ø and A∩ B=Ø. For a given transaction T_i, the rule A => B holds with support *s* and confidence *c* as defined in section 1.3.1. Support *s* is defined as the percentage of transactions that contains the items contained in set A as well as in set B i.e, union of set A and set B represented as A∪B. Confidence *c* is defined as the percentage of transactions containing A also containing B. When writing support and confidence in terms of percentage, their value range between 0 and 100, and when expressed as ratios, their values range between 0 and 1.

Association rules can further be defined as *strong* association rules if they satisfy the minimum threshold support called as *min_sup* and minimum threshold confidence called as *min_conf.*

14.3.3 Association Rules Discovery

The problem of discovery of association rules can be stated as: Given a set of transactions *T,* find the rules whose support and confidence are greater than equal to the minimum support and confidence threshold.

Traditional approach to generate association rules is to compute the support and confidence for every possible combination of items. Butt his approach is computationally not possible as the number of combinations of items can be exponentially large. To avoid such large number of computations, basic approach should be to ignore the needless computations without computing their support and confidence scores. For example, we can observe from Table 1 that the combination {milk, egg} can be ignored as the combination if infrequent. Hence, we prune the rule

Milk => Egg without computing the support and confidence for the items.

Therefore, the steps for obtaining association rules can be summarized as:

1. *Find all frequent item sets:* By definition, obtain the frequent itemset as set of items whose support score is at least the min_sup.
2. *Generate strong association rules:* By definition, obtain rules for the item sets obtained in step 1, with support score as min_sup and confidence score as min_conf. These rules are called as *strong rules*.

Challenge in obtaining association rules:

Low threshold value: If minimum threshold support (min_sup) is set quite low, then many items can belong to the frequent itemset.

Solution to the problem is to define a *closed frequent itemset* and *maximal frequent itemset*.

1. *Closed Frequent Itemset:* A frequent itemset A is said to be a closed frequent itemset if it is closed and has support score at least equal to min_sup. An itemset is said to be closed in a data set T if there does not exist any superset B such that support(B) equals support (A).
2. *Maximal Frequent Itemset:* A frequent itemset A is said to be a maximal frequent itemset in T if A is frequent and there exists no super set B such that A⊂B and B is frequent in T.

We use Venn diagram to understand the concept as shown in Figure 2.

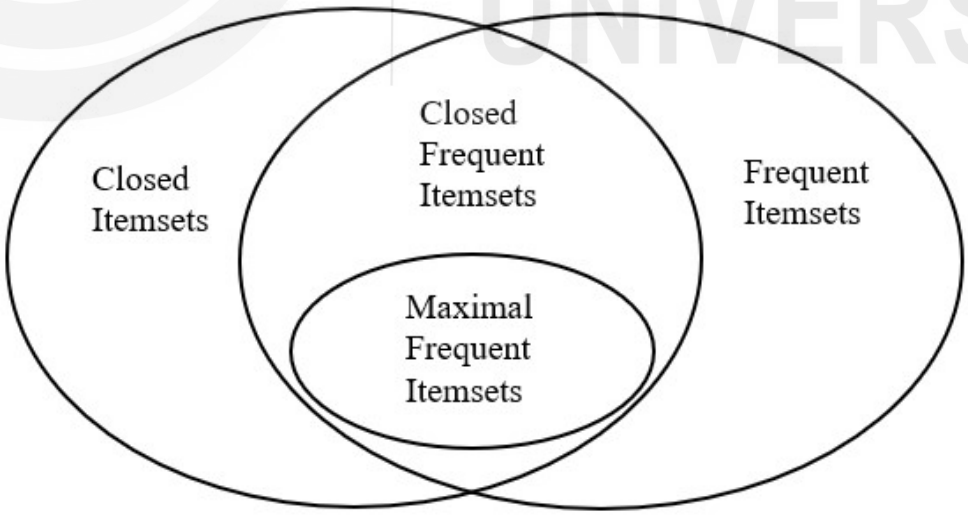


Figure 2: Venn Diagram to demonstrate the relationship between Closed item sets, frequent item sets, closed frequent item sets and maximal frequent item sets.

This illustrates that all the maximal frequent item sets are subsets of closed items sets and frequent item sets.

Check Your Progress 1

Ques 1: Given the following set of transactions for three items X, Y and Z as {{X, Y}, {X,Y, Z}, {X, Z}, {X, Z}, {Z}, {Z}, {Z}, {Z}}.

a) Give the binary representation of the transaction set.

b) Find the support and confidence for the following rules:

i) X =>Y

ii) X =>Z

iii) Y =>Z

Ques 2: Find the maximal frequent itemsets and closed frequent itemsets from the following set of transactions:

T1:A,B,C,E
T2:A,C,D,E
T3: B,C,E
T4:A,C,D,E
T5:C,D,E
T6:A,D,E

14.4 APRIORI ALGORITHM

R Aggarwal and R.Srikant proposed Apriori algorithm in the year 1994. The algorithm is used to obtain the frequent item sets for association rules. The algorithm is names so, as it needs the prior knowledge of the frequent item sets. This section discusses about the generation of frequent patterns as observed in the analysis of market basket problem. Section 1.4.1 presents Apriori algorithm, used to obtain the frequent item sets. Section 1.4.2 talks about generating strong association rules from the frequent item sets generated. Finally, section 1.4.3 presents variations of Apriori algorithm.

14.4.1 Frequent Item sets Generation

For a given set of *n* items, there are 2^n-1 possible combination of items. Consider an itemset I = {A, B, C, D} with four items, there are 15 combinations of items such as {{A}, {B}, {C}, {D}, {A, B}, {A, C}, {A, D}, {B, C}, {B, D}, {C, D}, {A, B, C}, {A, B, D}, {A, C, D}, {B, C, D}, {A, B, C, D}}. This can be represented by a lattice diagram as shown in Figure 3.

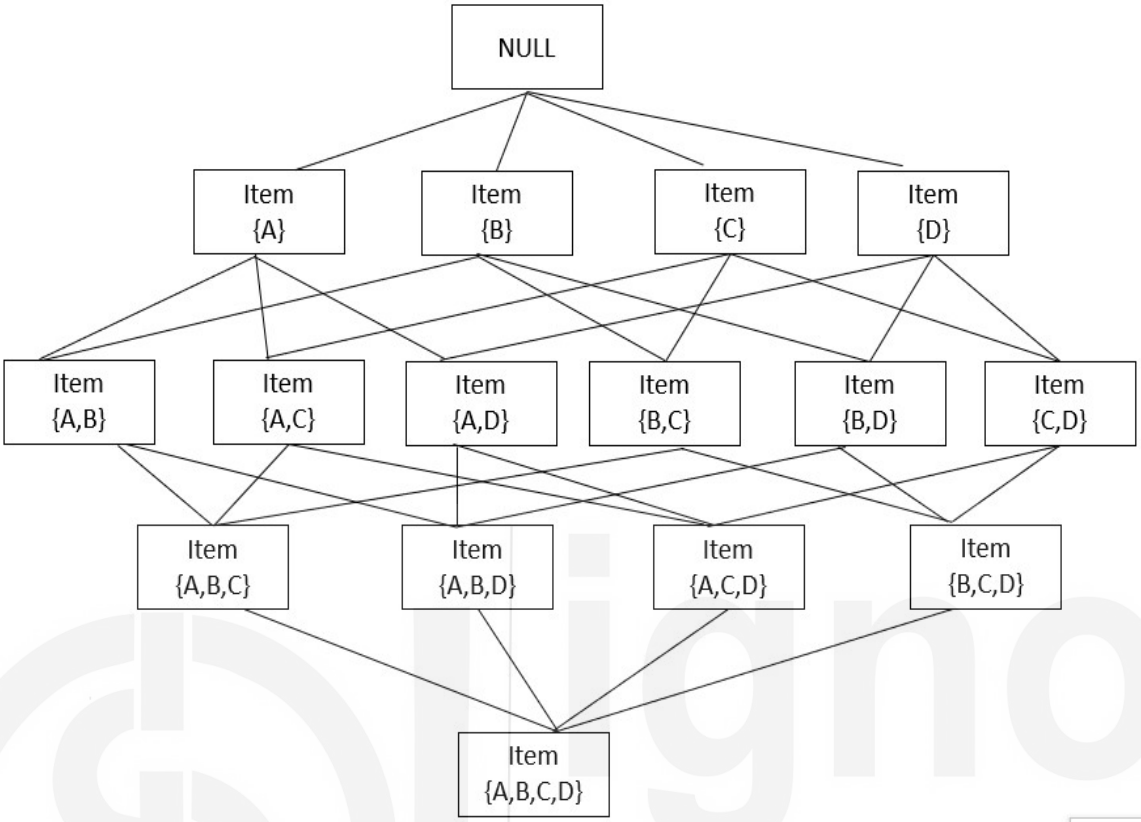


Figure 3: Lattice representing 15 combinations of items

Apriori algorithm searches the items level by level; to find the *(k+1)*item sets, it uses the *k*item sets. To determine the frequent item sets, the algorithm first finds the *candidate* item sets from the lattice representation. But as explained, for a given item sets of size *n*, maximum number of item sets in the lattice can be 2^n -1, one needs to control the search space in exponentially growing item sets and increase the efficiency of the algorithm. For this, two important principles are given below.

Definition 1: Apriori Principle: If an itemset is frequent, then all of its subsets must be frequent.

To understand the principle, let's consider the itemset {A, B, C} is the frequent itemset, then its *subsets* {A, B}, {A, C}, {B, C}, {A}, {B} and {C} are also the frequent item sets (marked in yellow) as shown in Figure 4. This is because if a transaction contains the itemset {A, B, C}, then the transaction will also contain all the subsets of this itemset.

On the other hand, if an itemset say {B, D} is not a frequent itemset, then all the *supersets* containing {B, D} i.e {A, B, D}, {B, C, D} and {A, B, C, D} are also not the frequent item sets (marked in blue) as shown in Figure 4. This is because if an itemset *X* is not a frequent itemset, then any other itemset containing *X* will also not be a frequent itemset. Recalling the definition that an itemset is frequent if and only if its support is greater than or equals to the minimum support threshold. Conversely, an itemset who support is less than the minimum support threshold, then the itemset is infrequent.

This *Apriori property* helps in pruning the item sets, thus reducing the search space and also increases the efficiency of the algorithm. This type of pruning technique is known as *support-based pruning*.

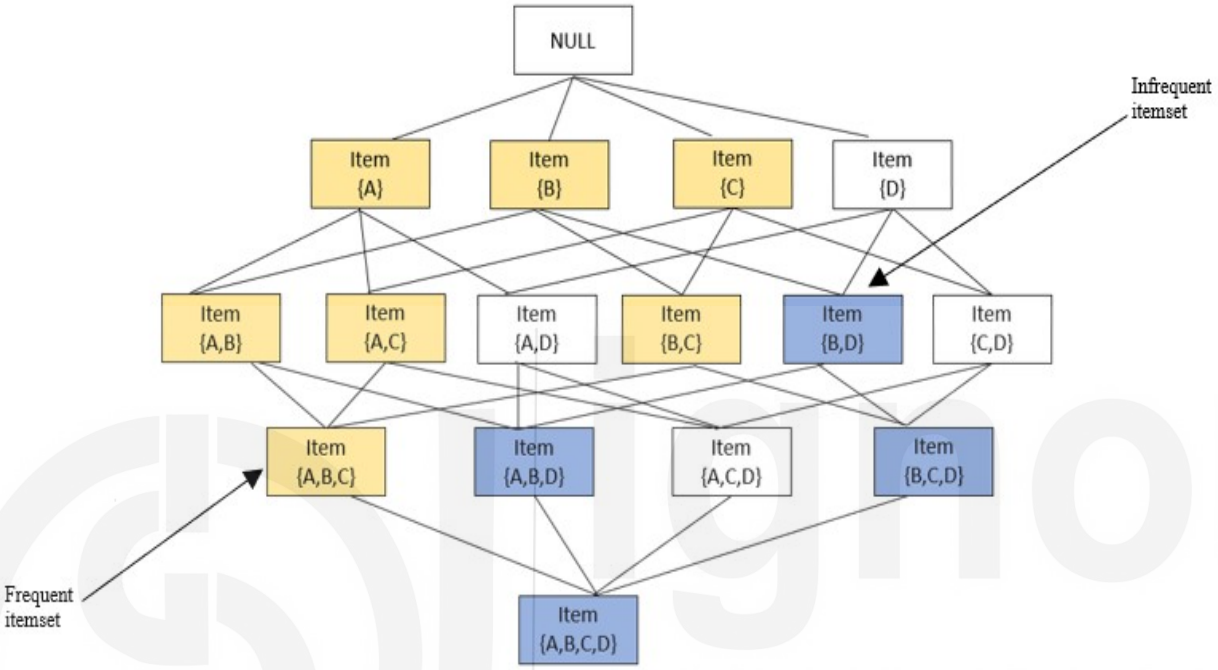


Figure 4: Frequent (yellow nodes) and Infrequent Item sets (blue nodes)

Definition 2: Antimonotone Property: This property states that if an itemset *X* does not satisfy a function, then any proper set containing *X* will also not satisfy the function. This property is called as antimonotone property as it is monotonic in terms of not satisfying a function. This property further helps in reducing the search space.

Given the set of transactions with the minimum support and confidence scores, perform the following steps to generate the frequent item sets using for Apriori algorithm.
Let *C(k)* be the set of *k* candidate item, *F(k)* be the set of *k* frequent items.

Step 1: Find F (1) i.e., frequent 1- item sets by computing the support for each item in the set of transactions.

Step 2: This is a twofold step as described below:

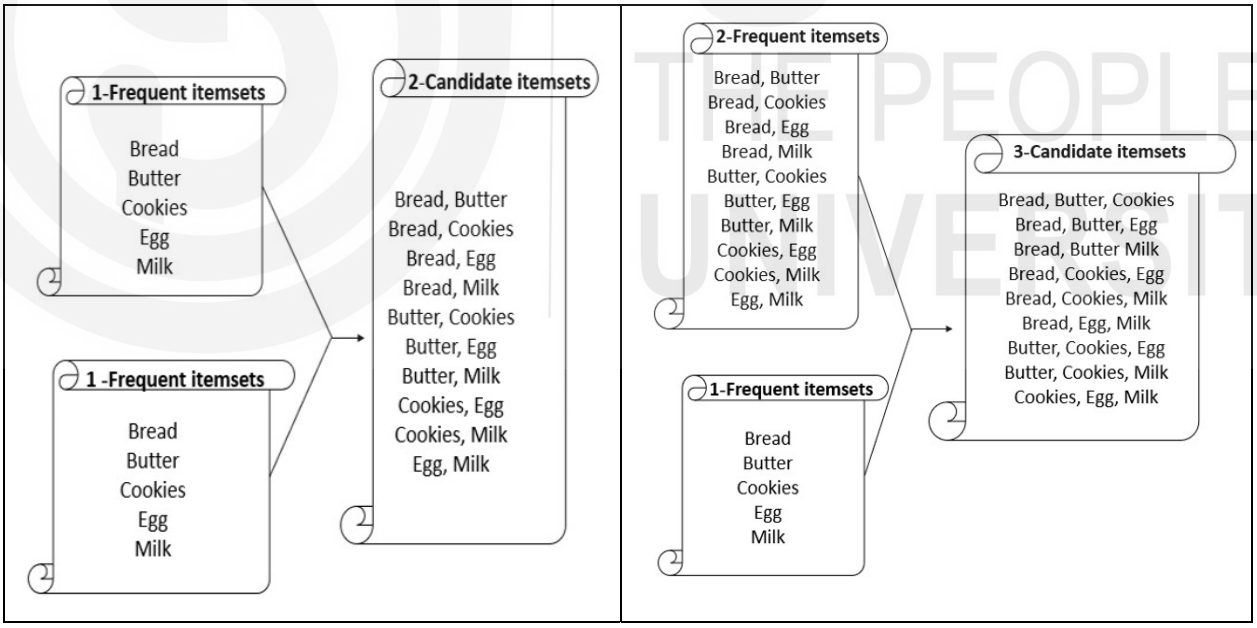
i) **Join Operation:** For $k \geq 2$, it generates C(k), new candidate k- itemset based on frequent item sets obtained in the previous step. This can be done in one of the two

ways:

1. Compute F(k-1) * F(1) to extend each F(k-1) itemset by joining it with each F(1) itemset. This join operation results in C(k) candidate item sets.

   For e.g.: Let k=2, and three 1 item setsF (1) = {A}, F (1) = {B} and F (1) = {C}. The two 1-itemsets can be augmented together to obtain 2-itemset such as F (2) = {A, B}, F (2)= {A, C} and F(2) = {B, C}. Further, to obtain 3-itemsets, we augment 2-itemsets with 1-itemsets such as joining {A, B} with {C} to obtain {A, B, C}.

   However, this method may generate duplicate F(k) item sets. For example, augmenting F(2) = {A, B} and F(1) = {C} generates C(3) = {A, B, C}. Also augmenting F(2) = {A, C} with F(1) = {B} generate C(3) = {A, B, C}. One way to avoid the generation of duplicate item sets is to sort the items in the item sets in lexicographic order. For example, the item sets {A, B}, {B, C}, {A, C}, {A, B, C} are sorted in their lexicographic order but {C, B}, {A, C, B} are not. Thus, items {A, B} , {C, D} can be augmented but the item sets {A, B}, {A, C} cannot be augmented as it will violate the lexicographic order. A working example to generate candidate F(k) itemset using this approach is shown in Figure 5.



a) F(1) joins F(1) to give C(2) *(b) F(2) joins F(1) to give C(3)*

Figure 5: Example of Augmenting F(k−1) and F(1) to generate candidate F(k) itemset

2. F(k-1) * F(k-1): A distinct pair of (k-1) item sets X = $\{A_1, A_2, A_3 \ldots A_{k-1}\}$ and Y = $\{B_1, B_2, \ldots B_{k-1}$, sorted in lexicographic order, are augmented iff:

$X_i = Y_i$, i= 1, 2,….(k-2) but $A_{k-1} \neq B_{k-1}$.

The item sets X and Y would be augmented only if the items A(k-1) and B(k-1) are sorted in lexicographic order. For example, let frequent itemset X(3)= {A, B, C} and Y(3) = {A, B, E}. X is joined with Y to generate the candidate itemset C(4) ={A, B, C, E}. On the other hand, consider another frequent itemset Z = {A, B, D}. Itemset X and Z can be joined to generate candidate set C(4) ={A, B, C, E} but itemset Y and Z cannot be joined as their last items, E and D are not arranged in lexicographic order. As a result, the F(*k*−1) × F(*k*−1) candidate \generation method merges a frequent itemset onlywith ones that contains the items in the sorted list, thus saving some computations. Working example demonstrating candidate generation using F(*k*−1) × F(*k*−1) is demonstrated in Figure 6.

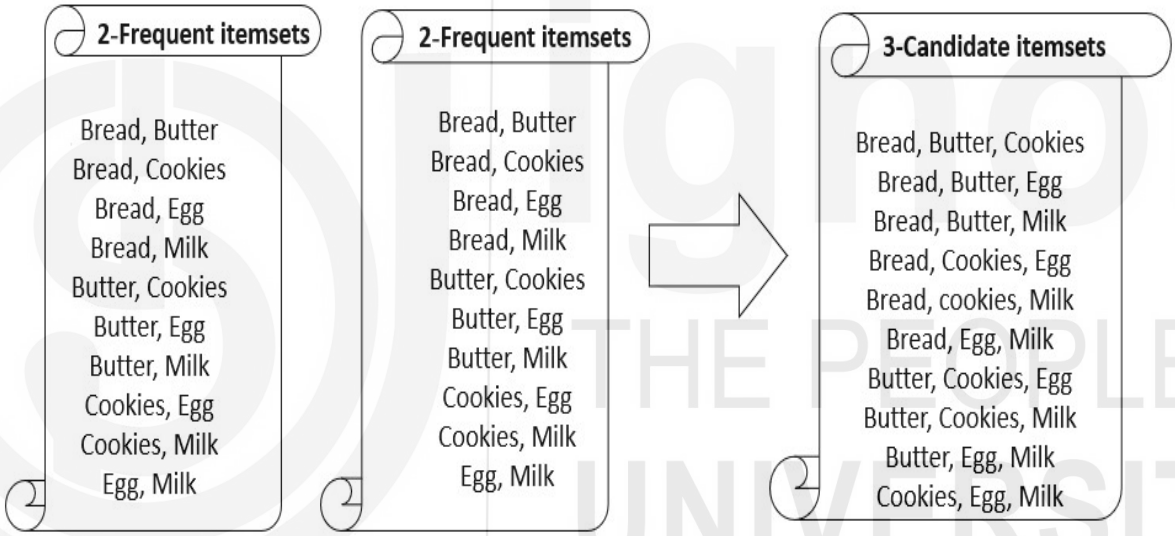


Figure 6: Example of joining F(k−1), F(k−1) to generate candidate F(k) itemset

ii) **Prune Operation:** Scan the generated entire candidate itemsets C(k) to compute the support score of each item in the set. Any itemset with support score less than minimum support threshold is pruned from the candidate itemset. Thus, the itemsets left after pruning would form the Frequent itemset F(K) such that F(k) ⊆ C(k). To prune the itemsets from the dataset, a priori property is used. According to the property, an infrequent (*k*-1)-itemset is not a subset of a frequent *k*-itemset. Thus, if any (*k* -1)-subset of a candidate itemset C(k) is not in *F(k*-1*)*, then the candidate cannot be frequent and so is removed from *C(k)*.

14.4.2 Case Study: *Find the frequent itemset from the given set of items*

Consider the set of transactions represented in binary form in Table 1 as given below. Assume minimum support threshold to be 2.

Transaction Id	List of items
T1	Milk, Cookies, Bread
T2	Milk, Bread, Egg, Butter
T3	Milk, Cookies, Bread, Butter
T4	Cookies, Bread, Butter

Step 1: Arrange the items in lexicographic order. Call this candidate set C(1)

Transaction Id	List of items
T1	Bread, Cookies, Milk
T2	Bread, Butter, Egg, Milk
T3	Bread, Butter, Cookies, Milk
T4	Bread, Butter, Cookies

Step 2: Obtain support score for each item in candidate itemset C(1) as:

S.No	Item	Support
1	Bread	4
2	Butter	3
3	Cookies	3
4	Egg	1
5	Milk	3

Step 3: Prune the items whose support score is less than the minimum support threshold. This results in 1-frequent itemset, F(1).

S.No	Item	Support
1	Bread	4
2	Butter	3
3	Cookies	3
4	Milk	3

Step 4: Generate 2-candidate itemsets from F(1) obtained in previous step and obtain support score of each itemset i.e frequency of each itemset in the original transaction set.

As support score of each itemset is at least 2, hence, none of the itemset is pruned. So, the table above represents 2-candidate itemsets, F(2).

S.No	Itemset	Support
1	Bread, Butter	3
2	Bread, Cookies	3
3	Bread, Milk	3
4	Butter, Cookies	2
5	Butter, Milk	2
6	Cookies, Milk	2

Step 5: Generate 3-candidate itemsets by joining F(2) and F(1) to obtain C(3). Compute the support score of each itemset.

S.No	Itemsets	Count
1	Bread, Butter, Cookies	2
2	Bread, Butter, Milk	2
3	Bread, Cookies, Milk	2
4	Butter, Cookies, Milk	1

Step 6: Prune the itemsets in C(3) if their support is less than the minimum support threshold. We then obtain F(3).

S.No	Itemsets	Count
1	Bread, Butter, Cookies	2
2	Bread, Butter, Milk	2
3	Bread, Cookies, Milk	2

Step 7: Similarly we obtain C(4) using F(3) and prune the candidate itemset to obtain frequent itemset F(4).

S.No	Itemset	Support
1	Bread, Butter, Cookies, Milk	1

As the only itemset in C(4) has support count 1, so it is pruned and F(4) =Ø.

Now, the iteration stops.

Detail of the algorithm is given in Figure 7.

Input: *T:* a set of transactions; *minsup:* minimum support threshold
Output: *F(k):* set of k-frequent itemsets (result)

```
1.  F1 ← {large 1 - itemsets}
2.     k ← 2
3.  while F(k−1) is not empty
4.        C(k)← Apriori_gen(F(k−1), k)
5.  for transactions t in T
6.           Dt ← {c in C(k) : c ⊆ t}
7.  for candidates c in Dt
8.                count[c] ← count[c] + 1
9.
10.       F(k) ← {c in C(k): count[c] ≥ minsup}
11.       k ← k + 1
12.
13. return Union(F(k))
14.
15. Apriori_gen(F, k)
16. for all X ⊆ F, Y ⊆ F where X1 = Y1, X2 = Y2, ..., Xk-2 = Yk-2 and Xk-1,
              Yk-1 are in lexicographic order
17.        c = X ∪ {Yk-1}
18. if u ⊆ c for all u in C
19.           result ←append (result, c)
20. return result
```

Figure 7: Apriori Algorithm

14.4.3 Generating Association Rules using Frequent Itemset

Once the frequent itemsets F(k) are generated from set of transactions T, next step is to generate strong association rules from them. Recall that *strong* association rules satisfy both minimum support as well as minimum confidence. Theoretically, confidence is defined as the ratio of frequency of transactions containing items x and *y* to the frequency of transactions that contained item *x*.

Based on the definition. Follow the given rules to generate strong association rules:

1. For each itemset f ϵ F(k), generate subsets of f.

2. For every non-empty subset $x \epsilon f$, generate the rule $x => f - x$

$$\frac{support(f)}{support(x)} \geq min\_conf$$

Consider the set of transactions and the generated frequent itemsets described in section 1.3.2. One of the frequent itemset *f* belonging to set F(3) is:

f = {Bread, Butter, Cookies}.

Following the steps given above, the non-empty subsets *x ⊆F(3)*,

x= {{Bread}, {Butter}, {Cookies}, {Bread, Butter}, {Bread, Cookies}, {Butter, Cookies}}

where all itemsets are sorted in lexicographic order. The generated association rules with their confidence scores are stated:

Rule	Confidence
{Bread, Butter} => {Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread,\ Butter)} = \frac{2}{3} = 66.67\%$
{Bread, Cookies} => {Butter}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread,\ Cookies)} = \frac{2}{3} = 66.67\%$
{Butter, Cookies} => {Bread}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Butter,\ Cookies)} = \frac{2}{2} = 100\%$
{Butter} => {Bread, Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Butter)} = \frac{2}{3} = 66.67\%$
{Bread} => {Butter, Cookies}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Bread)} = \frac{2}{4} = 50\%$
{Cookies} => {Bread, Butter}	$\frac{support(Bread,\ Butter,\ Cookies)}{support(Cookies)} = \frac{2}{3} = 66.67\%$

Depending on the minimum confidence scores, the obtained association rules are preserved, and rest are pruned.

Check Your Progress 2

Ques 1: For the following given Transaction Data-set, Generate Rules using Apriori Algorithm. Consider the values as Support=50% and Confidence=75%

Transaction Id	Set of Items
T1	Pen, Notebook, Pencil, Colours
T2	Pen, Notebook, Colours
T3	Pen, Eraser, Scale
T4	Pen, Colours, Eraser
T5	Notebook, Colours, Eraser

Ques 2: For the following given Transaction Data-set, Generate Rules using Apriori Algorithm. Consider the values as Support=15% and Confidence=45%

Transaction Id	Set of Items
T1	A, B, C
T2	B, D
T3	B, E
T4	A, B, D
T5	A, E
T6	B, E
T7	A, E
T8	A, B, C, E
T9	A, B, E

14.5 FP TREE GROWTH

Although there had been significant reduction in the numb er of candidate itemsets, yet Apriori algorithm can be slow due to scanning of entire transaction set iteratively. So, *Frequent Pattern growth,* also called as *FP growth* method employs divide and conquer strategy to generate frequent itemsets.

Working of FP growth algorithm:

1. Create **Frequent Pattern Tree,** or **FP-tree** by compressing the transaction database. Along with preserving the information about the itemsets, the tree structure also retains the association among the itemsets.

2. Divide the transaction database into a set of *conditional databases*. where each associated with one frequent item or "pattern fragment," and examines each database separately.

3. For each "pattern fragment," examine its associated itemsets only. Therefore, this approach may substantially reduce the size of the itemsets to be searched, along with examining the "growth" of patterns.

Advantages of FP growth over Apriori algorithm:

1. Efficient than Apriori algorithm
2. No candidate itemset generation
3. Only two passes over the transaction set

14.5.1 FP Tree Construction

1. Obtain the 1−itemsets and their support score as computed in the first step of Apriori algorithm.
2. Sort the itemsets in descending order of their support score.
3. Create the root of tree as NULL.
4. Examine the first transaction in the set T, create the nodes of tree as the itemset with the maximum support score at the top, the next itemset with lower count and so on. At the end, the leaves of the tree will have the itemsets with the least support score.
5. Examine every transaction in the set, whose itemsets are also sorted in descending order of their support score. If any itemset of this transaction is already existing in the tree, then they share the same node, but count is incremented by 1. Else, a new node and branch will be created for a new itemset.
6. Divide the FP tree obtained at the end of step 5 into conditional FP trees. For this, examine the leaf (lowest) node of FP tree. The lowest node represents the frequency pattern of length 1. For each item, traverse the path in the FP Tree to create *conditional pattern base*. It is the path labels of al paths to any node of the given item in FP tree.
7. Conditional pattern base generates the frequent patterns. Thus, one conditional tree is generated for one frequent pattern. Perform this step recursively to generate all the conditional FP trees.

Consider the following transaction set T given in section 1.3.2

Transaction `Id`	**List of items**

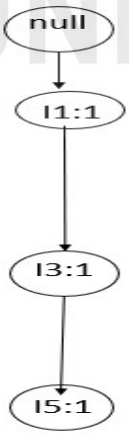
T1	Bread, Cookies, Milk
T2	Bread, Butter, Egg, Milk
T3	Bread, Butter, Cookies, Milk
T4	Bread, Butter, Cookies

We shall be representing each item with its S.No. in the description below. The items are sorted in descending order of their support score as given below.

S.No	Item	Support
I1	Bread	4
I2	Butter	3
I3	Cookies	3
I4	Milk	3
I5	Egg	1

i) Create the null node.

ii) For transaction T_1, item in descending order of their support score is I1 (4), I3 (3) and I4 (3). Create a branch connecting these three nodes in the tree as shown in Figure 8.



Examine T_1

Figure 8: Creating FP Tree from transaction T_1

iii) Examine the transaction T_2 andT_3, and further expand the FP tree as shown in Figure 9

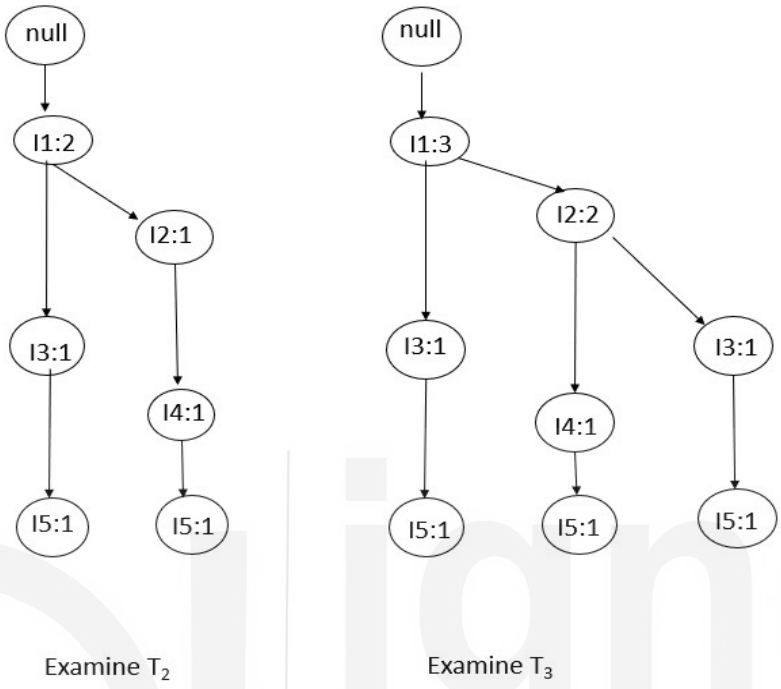


Figure 9: FP Tree creation from transactions T_2 and T_3

iv) Examining the last transaction T_4 generates the final FP tree as shown in Figure 10.

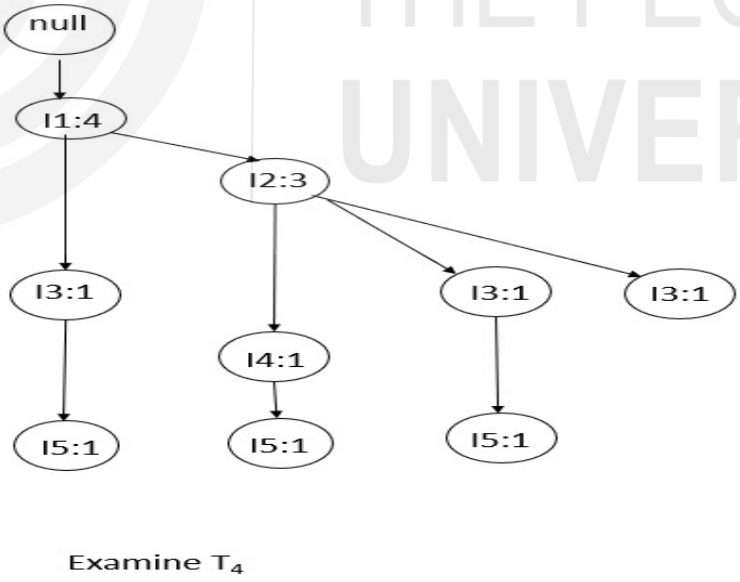


Figure 10: Final FP tree obtained after scanning transaction T_4

v) Now, obtain the conditional FP tree for each item I_1.....$I_5$by scanning the path from the lowest leaf node as shown in the table below:

Item	Conditional Pattern Base
I5	{I1, I3}:1, {I1, I2, I4}:1, {I1, I2, I3}:1
I4	{I1, I2}:1
I3	{I1):1, {I1, I2}:2
I2	{I1}:3
I1	

vi) For each item, build the conditional Pattern Tree by taking the set of elements common in all paths in the Conditional Pattern Base of that item. Further calculate its support score by summing the support scores of all the paths in the Conditional Pattern Base.

Item	Conditional Pattern Base	Conditional FP Tree
I5	{I1, I3}:1, {I1, I2, I4}:1, {I1, I2, I3}:1	{I1: 3, I2: 2}
I4	{I1, I2}:1	
I3	{I1):1, {I1, I2}:2	{I1}:3
I2	{I1}:3	{I1}:3
I1		

vii) Frequent patterns generated are: {I1, I2, I5} i.e {Bread, Butter, Milk}, {I1, I3} i.e. {Bread, Cookies} and {I1, I2}: {Bread Butter}.

Check Your Progress 3

Ques 1: Generate FP Tree for the following Transaction Dataset, with minimum support 40%.

Transaction Id	Set of Items
T1	I5, I1, I4, I2
T2	I4, I1, I3, I5, I2
T3	I3, I1, I2, I5

T4	I2, I1, I4
T5	I4
T6	I4, I2
T7	I1, I4, I5
T8	I2, I3

Ques 2: Find the frequent itemset using FP Tree, from the given set of Transaction Dataset with minimum support score as 2.

Transaction Id	Set of Items
T1	Banana, Apple, Tea
T2	Apple, Carrot
T3	Apple, Peas
T4	Banana, Apple, Carrot
T5	Banana, Peas
T6	Apple, Peas
T7	Banana, Peas
T8	Banana, Apple, Peas, Tea
T9	Banana, Apple, Peas

14.6 Pincer Search

In Apriori algorithm, the computation begins from the smallest set of frequent itemsets and proceeds till it reaches the maximum frequent itemset. But the algorithm's efficiency decreases as the algorithm passes through many iterations. Solution to the problem is to generate the frequent itemsets by using bottom-up and top-down approach together. Thus, the Pincer algorithm works on bidirectional approach. It attempts to find the frequent item sets in a bottom – up manner but, at the same time, it maintains a list of maximal frequent item sets. While iterating the transactions set, it computes the support score of the candidate maximal frequent item sets to record the frequent itemsets. This way, the algorithm outputs the subsets of the

frequent itemsets and, hence, they need not maintain support score in the next iteration. Thus, Pincer–search is advantageous over Apriori algorithm when the frequent item set is long.

In each iteration, while computing the support count of each iteration in bottom-up direction, the algorithm also computes the support score of some itemsets in top-down direction. These itemsets are called as *Maximal Frequent Candidate Set (MFCS)*. Consider an itemset x ϵ MFCS, such that cardinality of x is greater than k, in the k^{th} iteration. Then all the subsets of x must also be frequent. Thus, all these subset itemsets can be pruned from the candidate sets in bottom-up direction. Thus, increasing the algorithm efficiency. Similarly, when the algorithm finds an infrequent itemset in bottom–up direction, then MFCS is updated I order to remove these infrequent itemsets.

The MFCS is initialized to a singleton, which is the item set of cardinality *n* that contains all the elements of the transaction set. If some 'm' infrequent 1– itemsets are observed after the first iteration, then these infrequent itemsets will be pruned from the set, thereby reducing the cardinality of singleton to n –m. This is an efficient algorithm as one the algorithm may result a maximal frequent set in few iterations only.

Check Your Progress 4

Ques1: What is the advantage of using Princer algorithm over Apriori algorithm?

Ques 2: What is Maximal Frequent Candidate set?

Ques 3: What happens if an item is a maximal frequent item?

Ques 4: What is the bi–directional approach of Princer algorithm?

14.7 Summary

This unit presented the concepts of association rules. In this unit, we briefly described about what are these association rules and how various models help to analyse data for patterns, or co-occurrences, in a transaction database. These rules are created by finding the frequent itemsets in the database using the support and the confidence. Support indicates the frequency of occurrence of an item given the entire transaction database. High support score indicates a popular item among the users. Confidence of a rule indicates the rule's reliability. Higher confidence indicates more occurrence of item(s) in a set.
A typical example that has been used to study association rules is market basket analysis.

The unit presented different concepts like frequent itemsets, closed items and maximal itemsets.

Various algorithms viz, Apriori algorithm, FP tree and Pincer algorithm are discussed in detail in this unit, to generate the association rules by identifying the relationship between the items that are often purchased or occurred together.

14.8 Solutions to Check your progress

Check Your Progress 1

Ques 1

a) Binary Representation of the Transaction set:

Transaction	X	Y	Z
{X, Y}	1	1	0
{X, Y, Z}	1	1	1
{X, Z}	1	0	1
{Z}	0	0	1

b) Finding support and confidence of each rule:

Support = frequency of item/ number of transactions

Confidence(A->B) = support(A∪B)/support(B)

Support(X) = 4/8 = 50%
Support(Y) = 2/8 = 25%
Support(Z) = 7/8 = 87.5%
Support (X, Y) = 2/8 = 25%
Support (X, Z) = 3/8 = 37.5%
Support (Y, Z) = 1/8 = 12.5%

i) Confidence(X-> Y) = support (X, Y)/support(X) = (2/8)/(4/8) = 2/4 = 50%
ii) Confidence(X-> Z) = support (X, Z)/support(X) = (3/8)/(4/8) = 3/4 = 75%
iii) Confidence(Y-> Z) = support (Y, Z)/support(Y) = (1/8)/(2/8) = 1/2 = 50%

Ques 2 For finding the maximal frequent itemsets and the closed frequent itemsets:

a) Frequent itemset X ϵ F is maximal if it does not have any frequent supersets.
b) Frequent itemset X ϵ F is closed if it has no superset with the same frequency.

Step i) Count the occurrence of each item set:

Itemset	Frequency	Itemset	Frequency
{A}	4	{D, E}	3
{B}	2	{A, B, C}	1
{C}	5	{A, B, D}	0
{D}	4	{A, B, E}	1

{E}	6	{A, C, D}	2
{A, B}	1	{A, C, E}	3
{A, C}	3	{A, D, E}	3
{A, D}	3	{B, C, D}	0
{A, E}	4	{B, C, E}	2
{B, C}	2	{C, D, E}	3
{B, D}	0	{A, B, C, D}	0
{B, E}	2	{A, B, C, E}	1
{C, D}	3	{B, C, D, E}	0
{C, E}	5		

Let minimum support = 3.

a) {B}, {A, B}, {B, C}, {B, D}, {B, E}, {A, B, C}, {A, B, D}, {A, B, E}, {A, C, D}, {B, C, D}, {B, C, E}, {A, B, C, D}, {A, B, C, E} and {B, C, D, E} are not frequent itemsets as their support count is less than the minimum support score. Hence, we prune these itemsets.

b) {A} is not closed as frequency of {A, E} (superset of {A}) is same as frequency of {A} = 4.

c) {C} is not closed as frequency of {C, E} (superset of {C}) is same as frequency of {C} = 5.

d) {D} and {E} are closed as they do not have any superset with same frequency, but {D} and {E} are not maximal as they both have frequent supersets as {C, D} and {C, E} respectively.

e) {A, C} is not closed as frequency of its superset {A, C, E} is same as that of {A, C}.

f) {A, D} is not closed as frequency of its superset {A, D, E} is same as that of {A, D}.

g) {A, E} is closed as it has no superset with same frequency. But {A, E} is not maximal as it has a frequent superset i.e {A, D, E}.

h) {C, D} is not closed due to its superset {C, D, E}.

i) {C, E} is closed but not maximal due to its frequent superset {C, D, E}.

j) {D, E} is not closed due to its superset {C, D, E}.

k) {A, C, E} is a maximal itemset as it has no frequent super itemset. It is also closed as it has no superset with same frequency.

l) {A, D, E} is also a maximal and closed itemset.

Check Your Progress 2

Ques 1:

Step i) Find Frequent itemset and its support, where

support = frequency of item/number of transactions

Item	Frequency	Support %
Pen	4	4/5 = 80%
Notebook	3	3/5 = 60%
Pencil	1	1/5 = 20%
Colours	4	4/5 = 80%
Eraser	3	3/5 = 60%
Scale	1	1/5 = 20%

Step ii) Remove the items whose support is less than the minimum support (=50%)

Item	Frequency	Support %
Pen	4	4/5 = 80%
Notebook	3	3/5 = 60%
Colours	4	4/5 = 80%
Eraser	3	3/5 = 60%

Step iii) Form the two-item candidate set and find their frequency and support.

Item	Frequency	Support %
Pen, Notebook	2	2/5 = 40%
Pen, Colours	3	3/5 = 60%
Pen, Eraser	2	2/5 = 40%
Notebook, Colours	3	3/5 = 60%
Notebook, Eraser	1	1/5 = 20%
Colours, Eraser	2	2/5 = 40%

Step (iv) Remove the pair of items whose support is less than the minimum support

Item	Frequency	Support %
Pen, Colours	3	3/5 = 60%
Notebook, Colours	3	3/5 = 60%

Step (v) Generate the rule:

For rules, consider the following item pairs:

1. (Pen, Colours): Pen -> Colours and Colours -> Pen
2. (Notebook, colours): Notebook -> Colours and Colours -> Notebook

Confidence(A->B) = support (A∪ B)/support(A)

Hence,

Rule 1: Confidence (Pen -> Colours) = support (Pen, Colours)/support (Pen)
= (3/5) / (4/5) = 3/4 = 75%

Rule 2: Confidence (Colours -> Pen) = support (Colours, Pen)/support (Colours)
= (3/5) / (4/5) = 3/4 = 75%

Rule 3: Confidence (Notebook->Colours)
= support (Notebook, Colours) / support (Notebook)
= (3/5) / (3/5) = 1 = 100%

Rule 4: Confidence (Colours -> Notebook)
= support (Notebook, Colours) / support (Colours)
= (3/5) / (4/5) = 3/4 = 75%

All the 4 generated rules are accepted as the confidence score of each rule is greater than or equal to the minimum confidence given in the question.

Ques 2

Given set of transactions as:

Transaction Id	Set of Items
T1	A, B, C
T2	B, D
T3	B, E
T4	A, B, D
T5	A, E
T6	B, E
T7	A, E
T8	A, B, C, E
T9	A, B, E

Step i) Find the frequency and support of each item in the transaction set, where
support = frequency of item / number of transactions

Item	Frequency	Support
A	6	6/9 = 66.67%
B	7	7/9 = 77.78%
C	2	2/9 = 22.22%

D	2	2/9 = 22.22%
E	6	6/9 = 66.67%

Step ii) Prune the items whose support is less than the minimum support: As support of each item is greater than the minimum support (15%), hence, no item is pruned from the set. Now, form the two-item candidate set and find their support score.

Item	Frequency	Support
A, B	4	4/9 = 44.44%
A, C	2	2/9 = 22.22%
A, D	1	1/9 = 11.11%
A, E	3	3/9 = 33.33%
B, C	6	6/9 = 66.67%
B, D	2	2/9 = 22.22%
B, E	4	4/9 = 44.44%
C, D	0	0
C, E	1	1/9 = 11.11%
D, E	0	0

Step iii) Prune the two items whose support is less than the minimum support(15%)

Item	Frequency	Support
A, B	4	4/9 = 44.44%
A, C	2	2/9 = 22.22%
A, E	3	3/9 = 33.33%
B, C	6	6/9 = 66.67%
B, D	2	2/9 = 22.22%
B, E	4	4/9 = 44.44%

Step iv) Now, find three item frequent set and their support score

Item	Frequency	Support
A, B, C	2	2/9 = 22.22%
A, B, D	1	1/9 = 11.11%
A, B, E	1	1/9 = 11.11%
A, C, E	1	1/9 = 11.11%
B, C, D	0	0
B, C, E	1	1/9 = 11.11%
B, D, E	0	0

Pruning the three item sets whose support is less than the minimum support, we get:

Item	Frequency	Support
A, B, C	2	2/9 = 22.22%
A, B, E	2	2/9 = 22.22%

Step v) Generate the rules from each three itemset and compute the confidence of each rule as:

Confidence(x->y) = support(x∪y)/support(x)

For three itemset (A, B, C):

Rule	Support
(A, B)->C	(2/9) / (4/9) = 1/2 = 50%
(A, C) -> B	(2/9) / (2/9) = 1 = 100%
(B, C) -> A	(2/9) / (2/9) = 1 = 100%
A -> (B, C)	(2/9) / (6/9) = 2/6 = 33.33%
B -> (A, C)	(2/9) / (7/9) = 2/7 = 28.57%
C -> (A, B)	(2/9) / (2/9) = 1 = 100%

From the generated set of rules, the rules with confidence greater than or equal to the minimum confidence (45%) are the valid rules.

Thus, the valid rules are:

i) (A, B) -> C
ii) (B, C) -> A
iii) (A, C) -> B
iv) C -> (A, B)

Check Your Progress 3

Ques 1

Step i) From the given transaction set, find the frequency of each item and arrange them in decreasing order of their frequencies.

Item	Frequency
I1	5
I2	6
I3	3
I4	6
I5	4

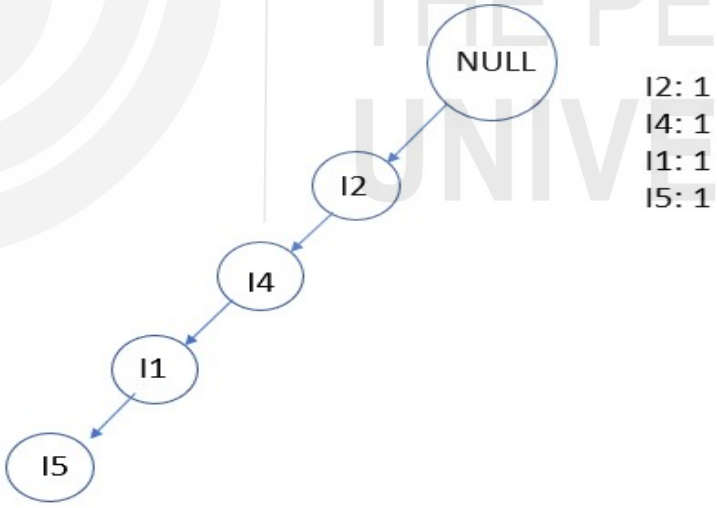
Item	Frequency
I2	6
I4	6
I1	5
I5	4
I3	3

Step ii) For every transaction set, arrange the item in the decreasing order of their frequency

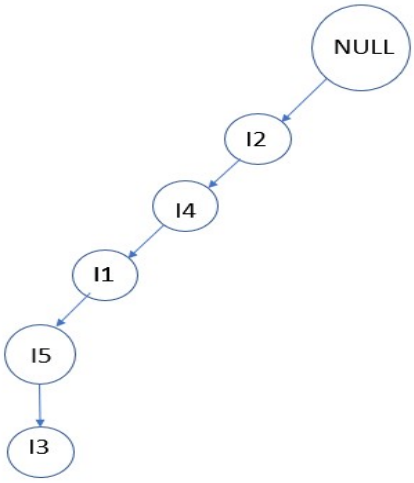
Transaction ID	Items	Ordered Items
T1	I5, I1, I4, I2	I2, I4, I1, I3
T2	I4, I1, I3, I5, I2	I2, I4, I1, I5, I3
T3	I3, I1, I2, I5	I2, I1, I5, I3
T4	I2, I1, I4	I2, I4, I1
T5	I4	I4
T6	I4, I2	I2, I4
T7	I1, I4, I5	I4, I1, I5
T8	I2, I3	I2, I3

Step iii) Create the FP tree now.

- Create a NULL node.
- From ordered items in T1, we get FP tree as:

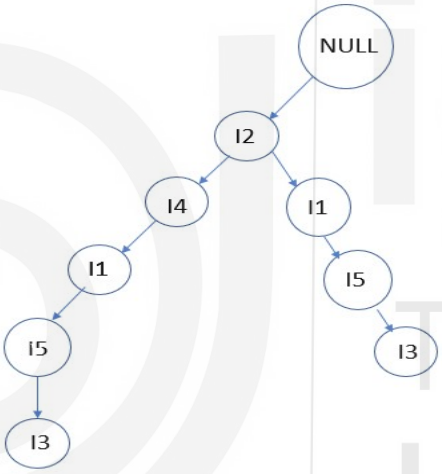


- Follow transaction T2 and increase the count of existing items and add the new items to the tree



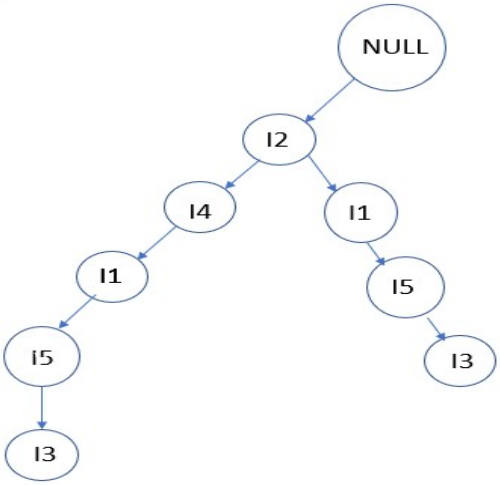
I2: 3
I4: 2
I1: 2
I5: 2
I3: 1

- Follow transaction T3 and repeat the above process



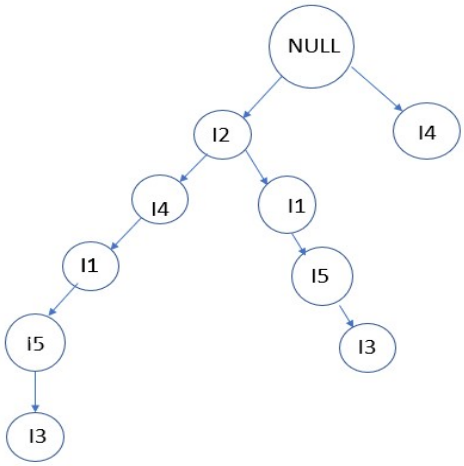
I2: 3
I4: 2
I1: 2
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1

- Follow Transaction 4, this leads to increase in count of I2, I4, I1.



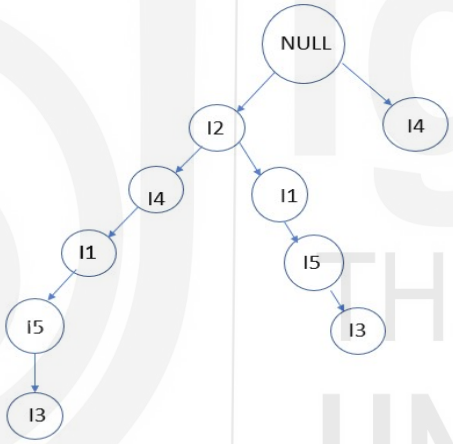
I2: 4
I4: 4
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1

- Follow transaction T5



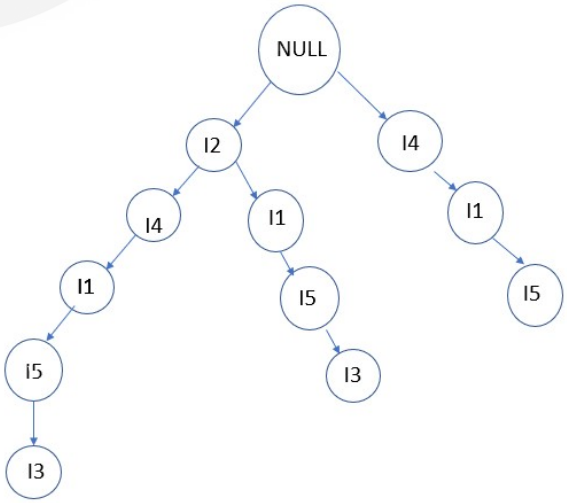
I2: 4
I4: 4
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 1

- Follow transaction T6. This leads to increase in count of I2 and I4 as the path in tree already exists.



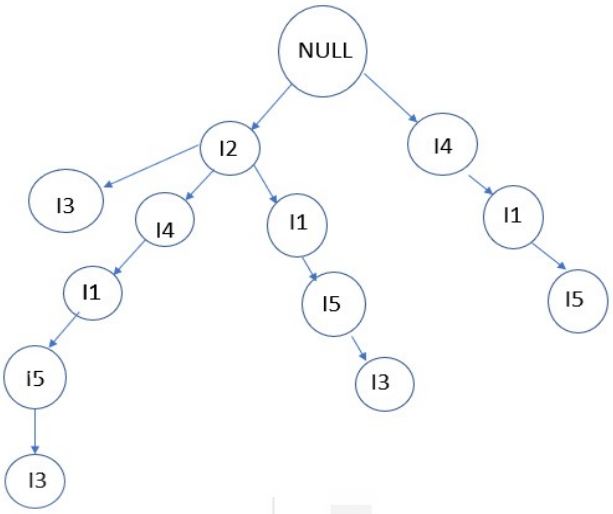
I2: 5
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 1

- Follow Transaction T7



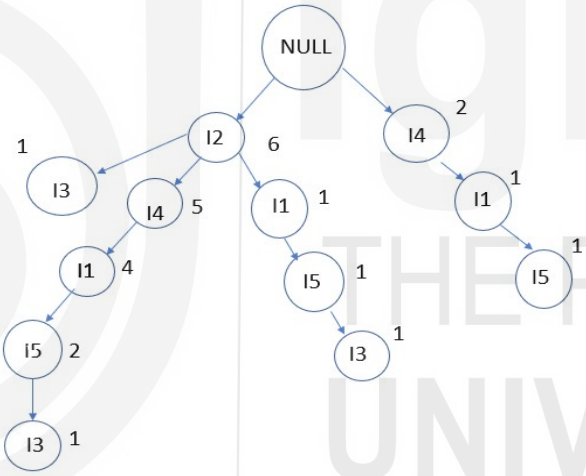
I2: 5
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1

- Finally follow transaction T8



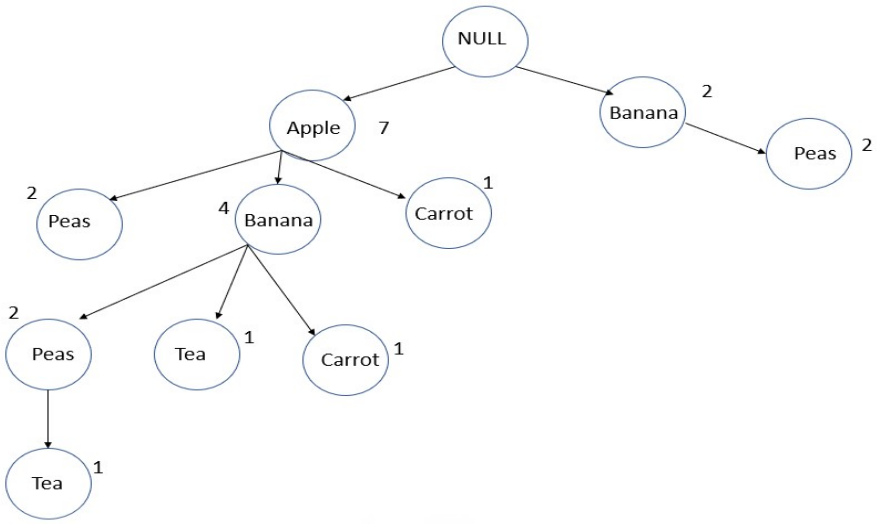
I2: 6
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1
I3: 1

The final FP tree is represented as:



I2: 6
I4: 5
I1: 4
I5: 2
I3: 1
I1: 1
I5: 1
I3: 1
I4: 2
I1: 1
I5: 1
I3: 1

Ques 2 Step i) FP Tree for the given Transaction set is:



Step ii) Create the conditional Pattern base as:

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Generation
Tea	{{Apple, Banana:1}, {Apple, Banana, Peas: 1}}	{Apple: 2, Banana: 2}	{Apple,Tea:2},{Banana,Tea:2}, {Apple,Banana,Tea:2}
Carrot	{{Apple,Banana:1},{Apple:1}	{Apple:2}	{Apple,Carrot:2}
Peas	{{Apple,Banana:2}, {Apple:2},{Banana:2}}	{Apple:4,Banana:2}, {Banana:2}	{Apple, Peas:4},{Banana, Peas:4},{Apple,Banana,Peas:2}
Banana	{{Apple:4}}	{Apple:4}	{Apple,Banana:4}

As the minimum support given 3. Hence all the itemsets with support score greater than equal to 3 form the frequent itemset as:
{Apple, Banana}, {Apple, Peas}, and {Banana, Peas}

14.9 FURTHER READINGS

1. Machine learning an algorithm perspective, Stephen Marshland, 2nd Edition, CRC Press, 2015.
2. Machine Learning, Tom Mitchell, 1st Edition, McGraw- Hill, 1997.
3. Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Peter Flach, 1st Edition, Cambridge University Press, 2012.
4. Data Mining – Concepts and Techniques(3rd Edition) , Kamber and Han, Morgan Kaufman