

## Experiment No.08

**Title:** Implementation of Association Rule Mining algorithm (Apriori).

**Theory:**

Apriori Algorithm – Frequent Pattern Algorithms Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

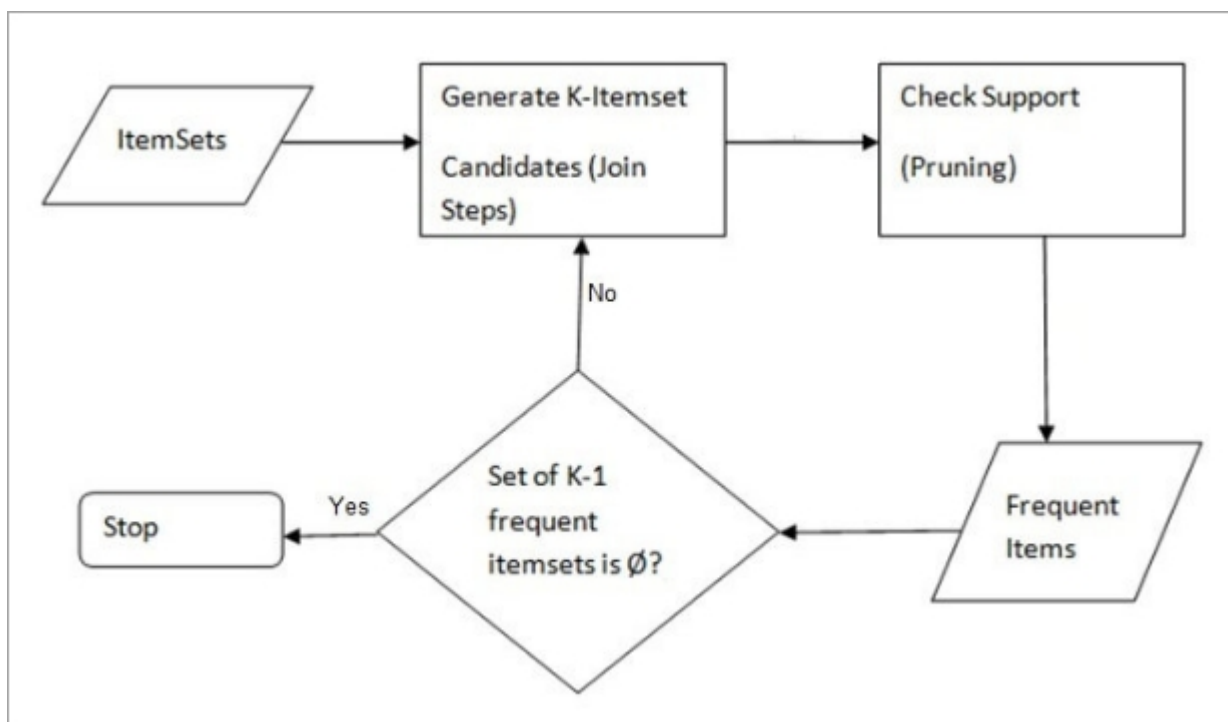
Apriori says: The probability that item I is not frequent is if:

- $P(I) < \text{minimum support threshold}$ , then I is not frequent.
- $P(I+A) < \text{minimum support threshold}$ , 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:

1. Join Step: This step generates (K+1) itemset from K-itemsets by joining each item with itself.
2. 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.
3. Steps In Apriori Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.
  - 1) In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.
  - 2) Let there be some minimum support, min\_sup ( eg 2). The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min\_sup, are taken ahead for the next iteration and the others are pruned.
  - 3) Next, 2-itemset frequent items with min\_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.
  - 4) The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –itemsets with min-sup only.
  - 5) The next iteration will form 3 –itemsets using join and prune step. This iteration will follow antimonotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min\_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.

6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min\_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.



Example of Apriori: Support threshold=50%, Confidence= 60%

TABLE-1

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:

Support threshold=50%  $\Rightarrow 0.5*6 = 3 \Rightarrow \text{min\_sup}=3$

1. Count Of Each Item  
TABLE-2

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

2. **Prune Step:** TABLE -2 shows that I5 item does not meet  $\text{min\_sup}=3$ , thus it is deleted, only I1, I2, I3, I4 meet  $\text{min\_sup}$  count.

TABLE-3

Item	Count
I1	4
I2	5
I3	4
I4	4

3. **Join Step:** Form 2-itemset. From TABLE-1 find out the occurrences of 2-itemset.

TABLE-4

Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2

4. **Prune Step:** TABLE -4 shows that item set {I1, I4} and {I3, I4} does not meet  $\text{min\_sup}$ , thus it is deleted.

TABLE-5

Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

5. *Join and Prune Step: Form 3-itemset.* From the TABLE- 1 find out occurrences of 3-itemset. From TABLE-5, find out the 2-itemset subsets which support min\_sup.

We can see for itemset {I1, I2, I3} subsets, {I1, I2}, {I1, I3}, {I2, I3} are occurring in TABLE-5 thus {I1, I2, I3} is frequent.

We can see for itemset {I1, I2, I4} subsets, {I1, I2}, {I1, I4}, {I2, I4}, {I1, I4} is not frequent, as it is not occurring in TABLE-5 thus {I1, I2, I4} is not frequent, hence it is deleted.

TABLE-6

Item
I1,I2,I3
I1,I2,I4
I1,I3,I4
I2,I3,I4

Only {I1, I2, I3} is frequent.

6. *Generate Association Rules:* From the frequent itemset discovered above the association could be:

{I1, I2}  $\Rightarrow$  {I3}

Confidence = support {I1, I2, I3} / support {I1, I2} = (3/ 4)\* 100 = 75%

{I1, I3}  $\Rightarrow$  {I2}

Confidence = support {I1, I2, I3} / support {I1, I3} = (3/ 3)\* 100 = 100%

{I2, I3}  $\Rightarrow$  {I1}

Confidence = support {I1, I2, I3} / support {I2, I3} = (3/ 4)\* 100 = 75% {I1}  $\Rightarrow$  {I2, I3}

Confidence = support {I1, I2, I3} / support {I1} = (3/ 4)\* 100 = 75% {I2}  $\Rightarrow$  {I1, I3}

Confidence = support {I1, I2, I3} / support {I2} = (3/ 5)\* 100 = 60% {I3}  $\Rightarrow$  {I1, I2}

Confidence = support {I1, I2, I3} / support {I3} = (3/ 4)\* 100 = 75%

This shows that all the above association rules are strong if minimum confidence threshold is 60%. The Apriori Algorithm: Pseudo Code C: Candidate item set of size k L: Frequent itemset of size k

- Join Step:  $C_k$  is generated by joining  $L_{k-1}$  with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code :  $C_k$ : Candidate itemset of size k  
 $L_k$ : frequent itemset of size k

```

 $L_1 = \{\text{frequent items}\};$ 
for ( $k = 1; L_k \neq \emptyset; k++$ ) do begin
     $C_{k+1}$  = candidates generated from  $L_k$ ;
    for each transaction  $t$  in database do
        increment the count of all candidates in  $C_{k+1}$ 
        that are contained in  $t$ 
     $L_{k+1}$  = candidates in  $C_{k+1}$  with min_support
    end
return  $\cup_k L_k$ ;

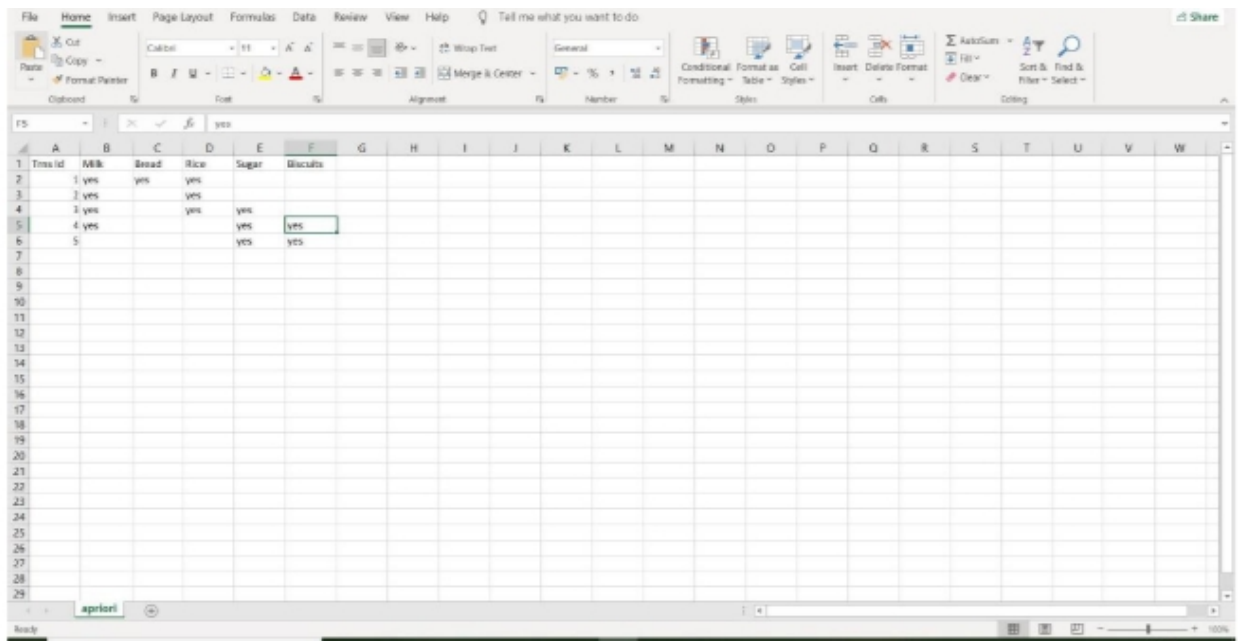
```

### Advantages

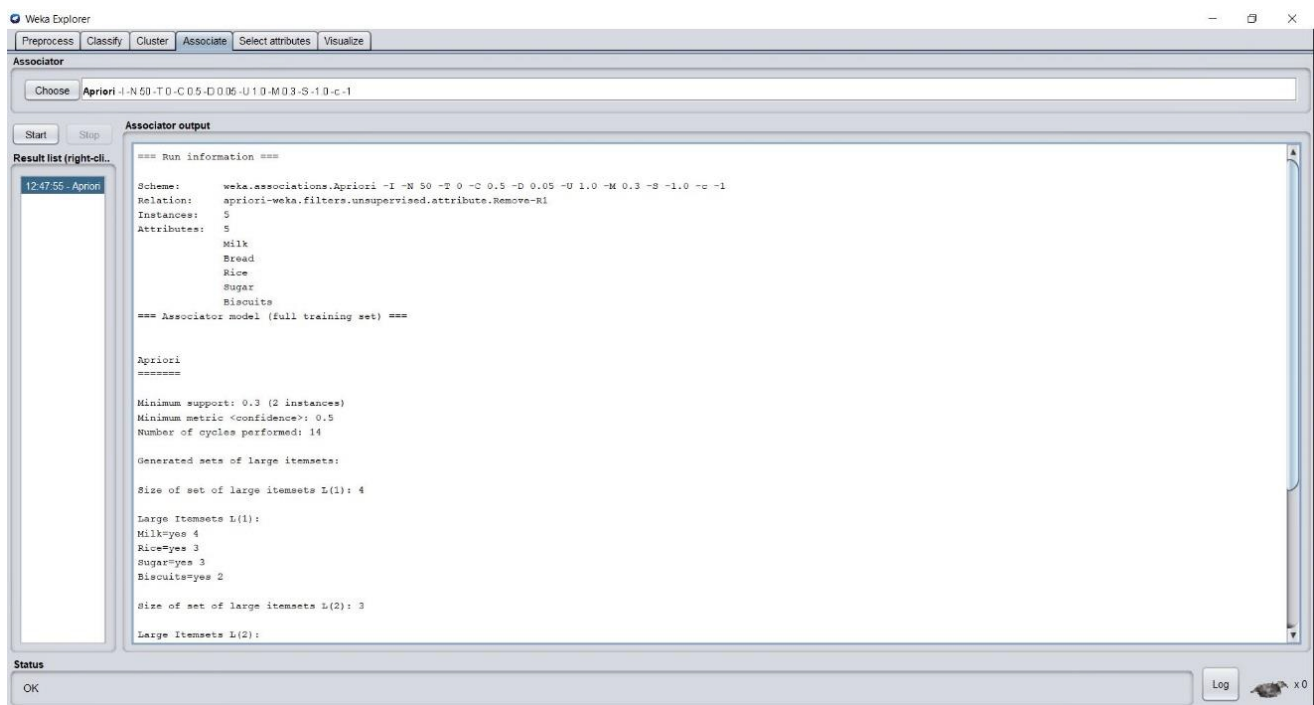
1. Easy to understand algorithm
2. Join and Prune steps are easy to implement on large itemsets in large databases

### Disadvantages

1. It requires high computation if the itemsets are very large and the minimum support is kept very low.
2. The entire database needs to be scanned.



	A	B	C	D	E	F
1	Transid	Milk	Bread	Rice	Sugar	Biscuits
2	1	yes	yes	yes		
3	2	yes		yes		
4	3	yes		yes	yes	
5	4	yes			yes	yes
6	5				yes	yes
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						



```
==== Run information ====
Scheme:      weka.associations.Apriori -I -N 50 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1
Relation:    apriori-weka.filters.unsupervised.attribute.Remove-R1
Instances:   5
Attributes:  5
             Milk
             Bread
             Rice
             Sugar
             Biscuits

==== Associator model (full training set) ====

Apriori
=====
Minimum support: 0.3 (2 instances)
Minimum metric <confidence>: 0.5
Number of cycles performed: 14

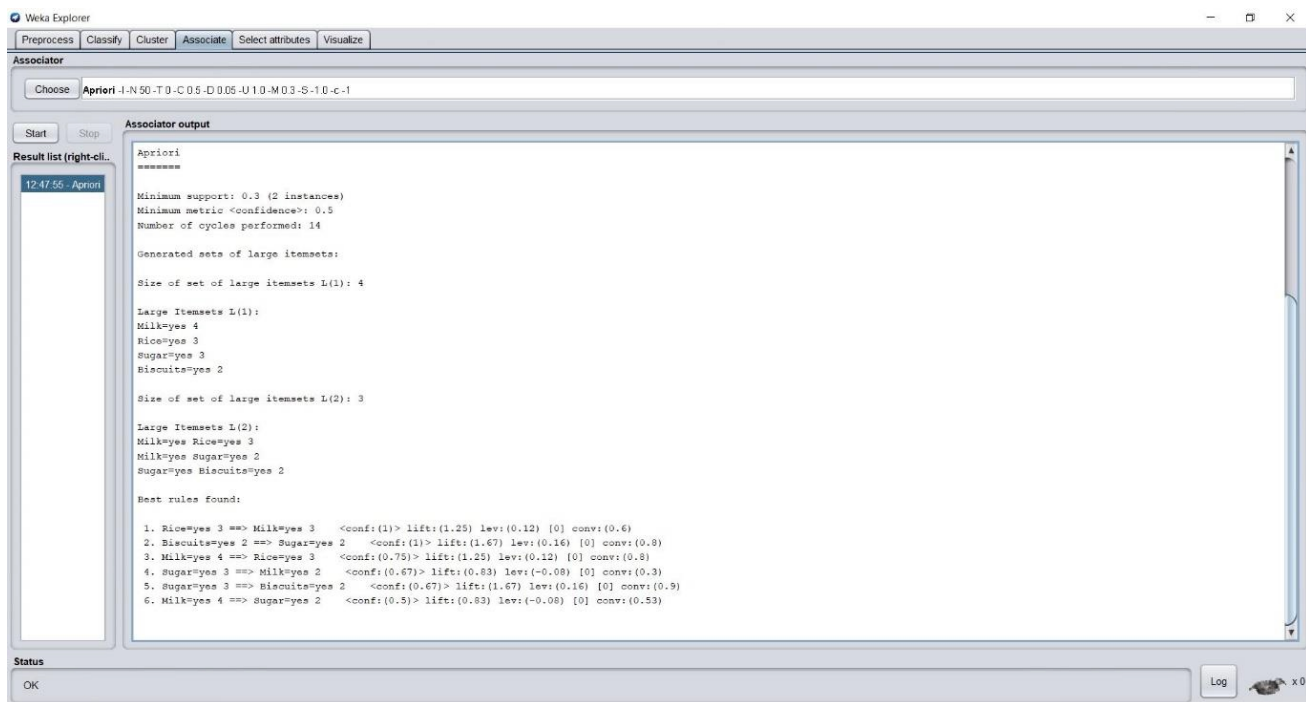
Generated sets of large itemsets:

Size of set of large itemsets L(1): 4

Large Itemsets L(1):
Milk=yes 4
Rice=yes 3
Sugar=yes 3
Biscuits=yes 2

Size of set of large itemsets L(2): 3

Large Itemsets L(2):
```



**Conclusion:** Thus we have studied how to implement Apriori algorithm.