MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI
Juhu Versova Link Road, Andheri (W) Mumbai 400053
(Permanently Affiliated to University of Mumbai)
Department of Computer Engineering

Experiment No.08

Date:
Roll No: B545

Aim: C) To study and implement Apriori algorithm using WEKA tool.

Theory:

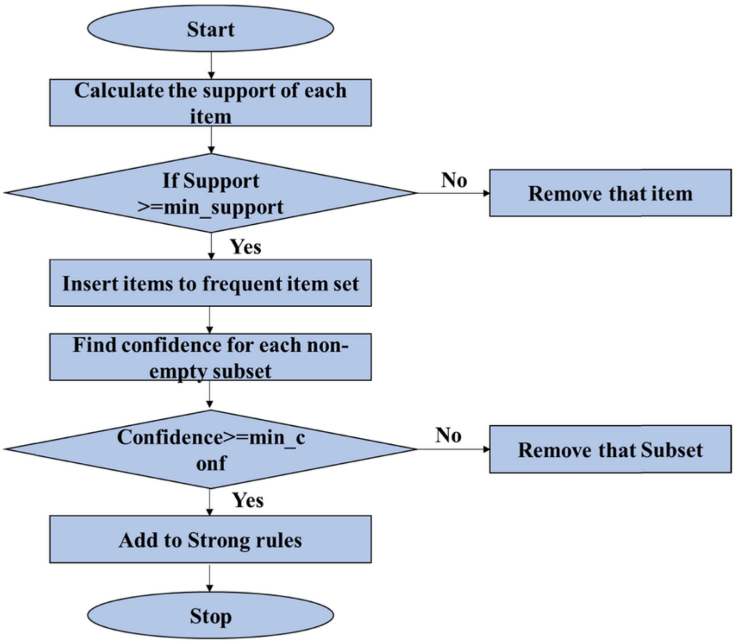
Apriori algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

Apriori Algorithm Working

The Apriori algorithm operates on a straightforward premise. When the support value of an item set exceeds a certain threshold, it is considered a frequent item set. Take into account the following steps. To begin, set the support criterion, meaning that only those things that have more than the support criterion are considered relevant.

- Step 1: Create a list of all the elements that appear in every transaction and create a frequency table.
- Step 2: Set the minimum level of support. Only those elements whose support exceeds or equals the threshold support are significant.
- Step 3: All potential pairings of important elements must be made, bearing in mind that AB and BA are interchangeable.
- Step 4: Tally the number of times each pair appears in a transaction.
- Step 5: Only those sets of data that meet the criterion of support are significant.
- Step 6: Now, suppose you want to find a set of three things that may be bought together. A rule, known as self-join, is needed to build a three-item set. The item pairings OP, OB, PB, and PM state that two combinations with the same initial letter are sought from these sets.
- OPB is the result of OP and OB.
- PBM is the result of PB and PM.
- Step 7: When the threshold criterion is applied again, you'll get the significant itemset.



Support and Confidence

Support and Confidence are the critical metrics guiding the Apriori algorithm. Formally, the formulas for these metrics are:

Support(X) = (Transactions containing (X))/(Total Transactions)

Confidence(X→Y) = (Transactions containing both (X and Y))/(Transactions containing X)

Support measures item sets' frequency in transactions, while confidence gauges the probability of item Y being purchased when item X is purchased.

Advantages of Apriori Algorithm

1. Simplicity and Ease of Implementation

   The Apriori Algorithm is quite straightforward and relatively easy to implement. Its simplicity lies in generating candidate item sets and using support and confidence thresholds to determine associations.

2. Efficient Pruning

   The Apriori principle allows the algorithm to efficiently prune unlikely itemset early on in the process, reducing the number of calculations required and speeding up the overall computation.

3. Scalability

   The Apriori Algorithm is highly scalable and can handle large-scale datasets effectively, making it suitable for various applications and industries.

Disadvantages of Apriori Algorithm

1. Computational Complexity

   Despite the pruning technique, the Apriori Algorithm can still be computationally complex, especially when dealing with numerous items and large datasets, potentially hindering its performance.

2. Multiple Scans of the Dataset

   The Apriori Algorithm requires multiple scans of the dataset to calculate support and generate itemset, which may result in slower processing times for very large datasets.

Applications of Apriori Algorithm

Apriori is used in the following fields:

1. Education:

   Through the use of traits and specializations, data mining of accepted students may be used to extract association rules.

2. Medical:

   Analysing the patient's database, for example, might be appropriate.

3. Forestry:

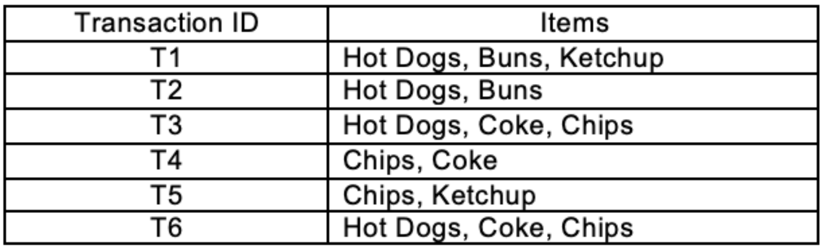
   Frequency and intensity of forest fire analysis using forest fire data.

4. Autocomplete Tool:

   Apriori is employed by a number of firms, including Amazon's recommender system and Google's autocomplete tool.

Example:

Find the frequent item sets and generate association rules on this. Assume that minimum support threshold (s = 33.33%) and minimum confident threshold (c = 60%)



Transaction ID	Items
T1	Hot Dogs, Buns, Ketchup
T2	Hot Dogs, Buns
T3	Hot Dogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	Hot Dogs, Coke, Chips

Sol:-

$$\text{minimum support count} = \frac{33.33}{100} \times 6$$
$$= 2$$

Item set	Sup-count
Hot Dogs	4
Buns	2
Ketchup	2
Coke	3
Chips	4

Item set	Sup-count
Hot Dogs	4
Buns	2
Ketchup	2
Coke	3
Chips	4

Item set	Sup-count
Hot Dogs, Buns	2
Hot Dogs, Ketchup	1
Hot Dogs, Coke	2
Hot Dogs, Chips	2
Buns, Ketchup	1
Buns, Coke	0
Buns, Chips	0
Ketchup, Coke	0
Ketchup, Chips	1
Coke, Chips	3

Item set	Sup-count
Hot Dogs, Buns	2
Hot Dogs, Coke	2
Hot Dogs, Chips	2
Coke, Chips	3

Item set	Sup-count
Hot Dogs, Buns, Coke	0
Hot Dogs, Buns, Chips	0
Hot Dogs, Coke, Chips	2

Item set	Sup-count
Hot Dogs, Coke, Chips	2

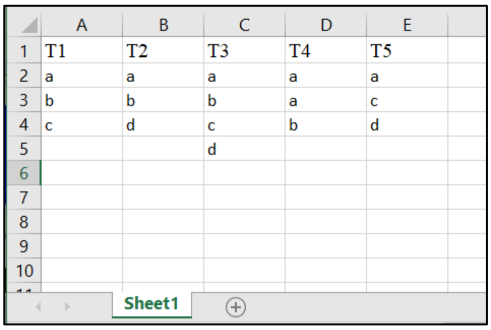
There is only one itemset with minimum support 2. So only one itemset is frequent.

Frequent Itemset (I) = {Hot Dogs, Coke, Chips}

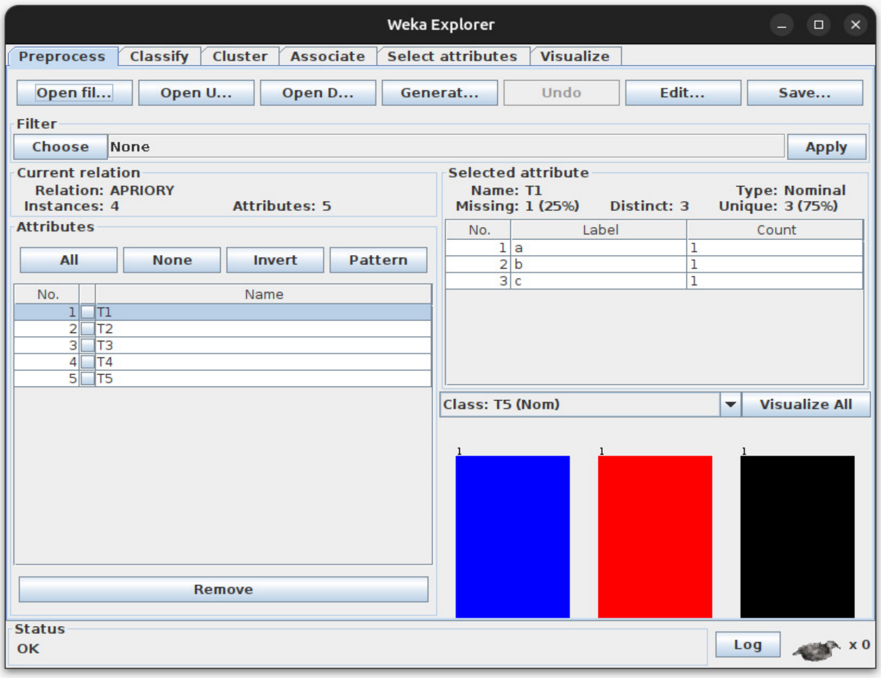
Association rules,

[Hot Dogs^Coke]=>[Chips] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs^Coke) = 2/2*100=100% //Selected
[Hot Dogs^Chips]=>[Coke] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs^Chips) = 2/2*100=100% //Selected
[Coke^Chips]=>[Hot Dogs] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Coke^Chips) = 2/3*100=66.67% //Selected
[Hot Dogs]=>[Coke^Chips] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs) = 2/4*100=50% //Rejected
[Coke]=>[Hot Dogs^Chips] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Coke) = 2/3*100=66.67% //Selected
[Chips]=>[Hot Dogs^Coke] //confidence = sup(Hot Dogs^Coke^Chips)/sup(Chips) = 2/4*100=50% //Rejected

Implementation:



	A	B	C	D	E
1	T1	T2	T3	T4	T5
2	a	a	a	a	a
3	b	b	b	a	c
4	c	d	c	b	d
5			d		
6					
7					
8					
9					
10					



Filter

Choose None

Current relation

Relation: APRIORY

Instances: 4 Attributes: 5

Selected attribute

Name: T1 Type: Nominal

Missing: 1 (25%) Distinct: 3 Unique: 3 (75%)

No.	Label	Count
1	a	1
2	b	1
3	c	1

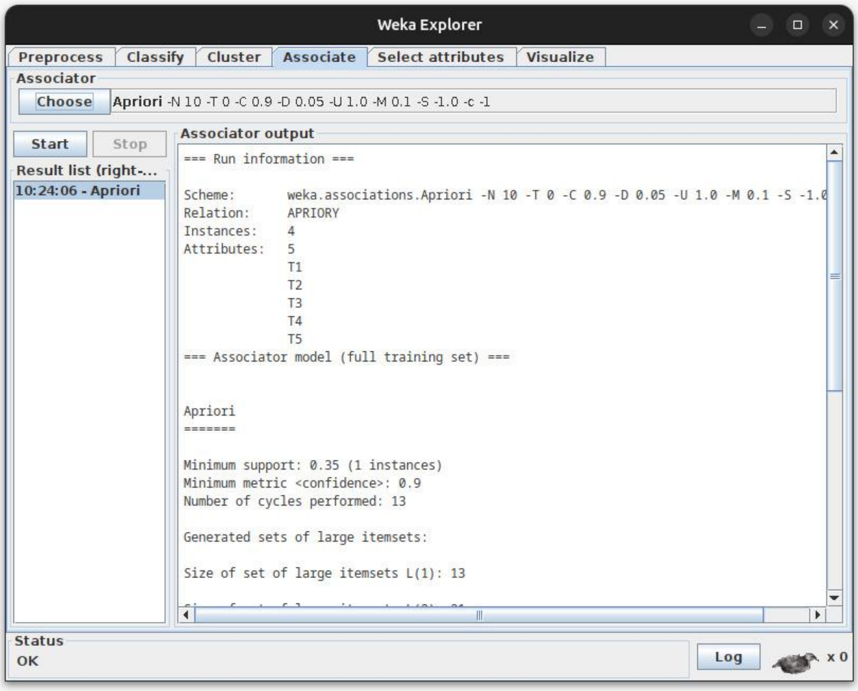
Attributes

No.		Name
1	□	T1
2	□	T2
3	□	T3
4	□	T4
5	□	T5

Class: T5 (Nom)

Status

OK



Associator

Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Result list (right-...

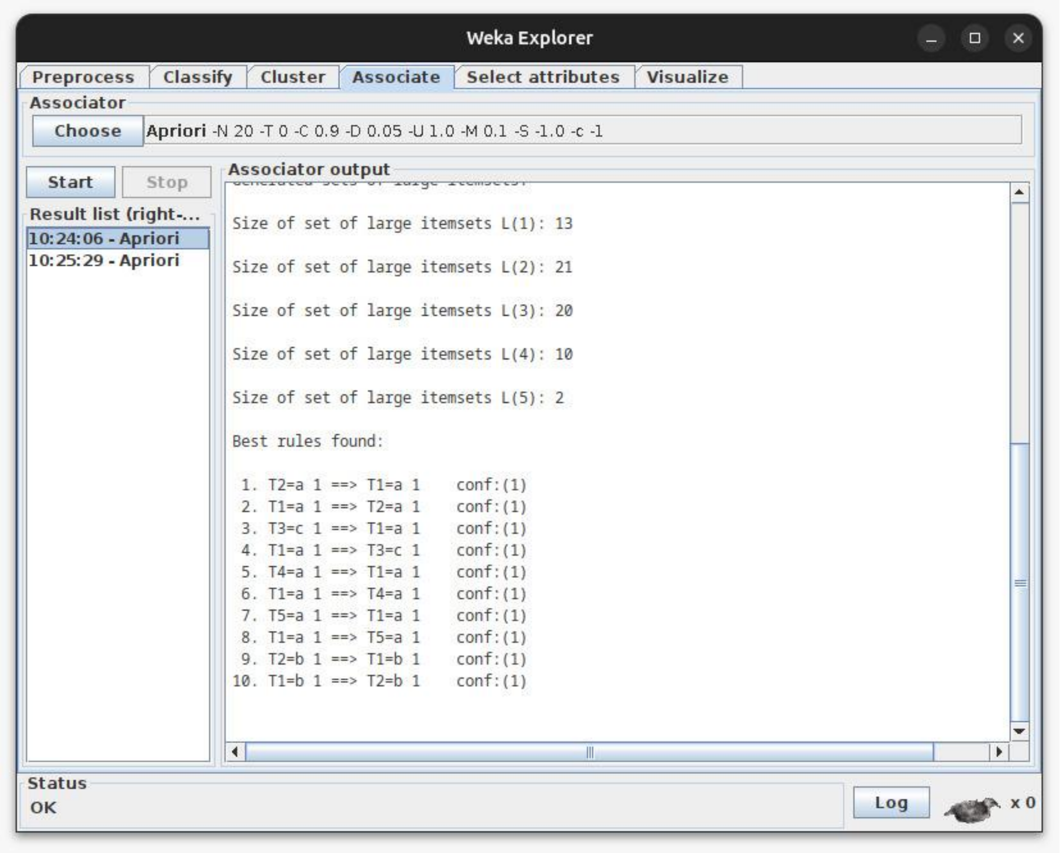
10:24:06 - Apriori

Associator output

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0
Relation:     APRIORY
Instances:    4
Attributes:   5
              T1
              T2
              T3
              T4
              T5
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.35 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 13
```

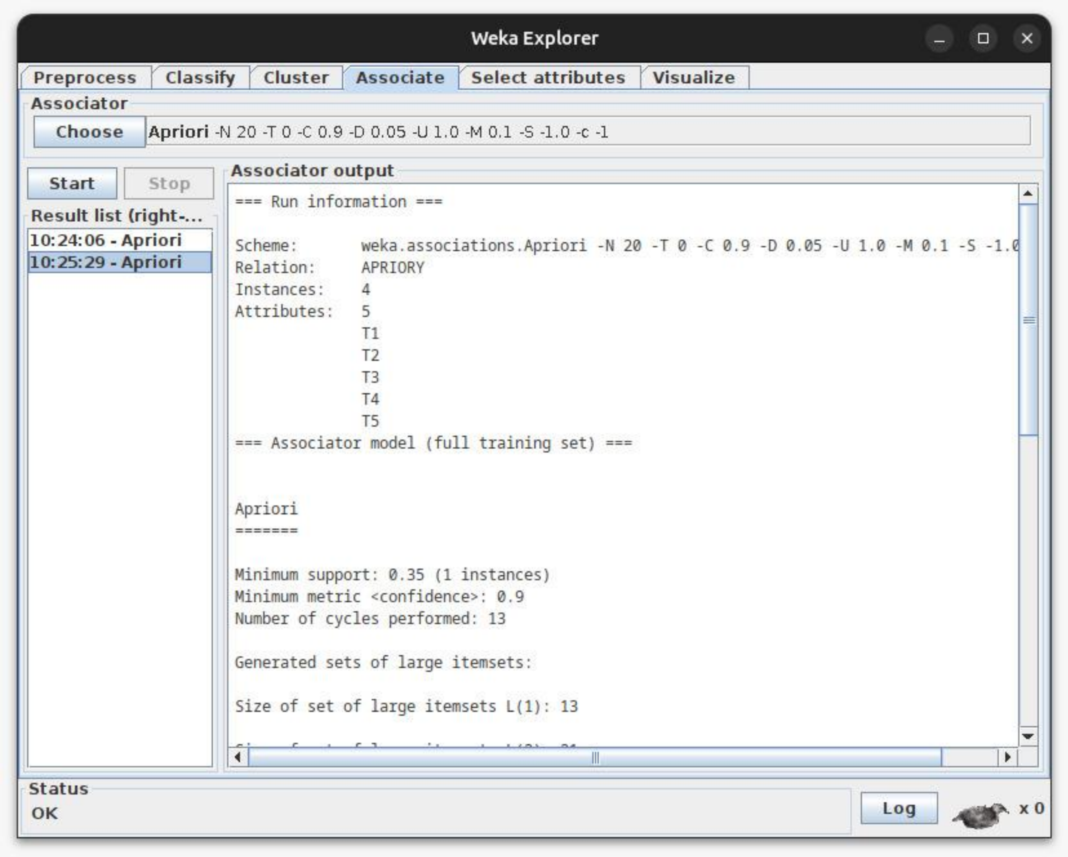
Status

OK



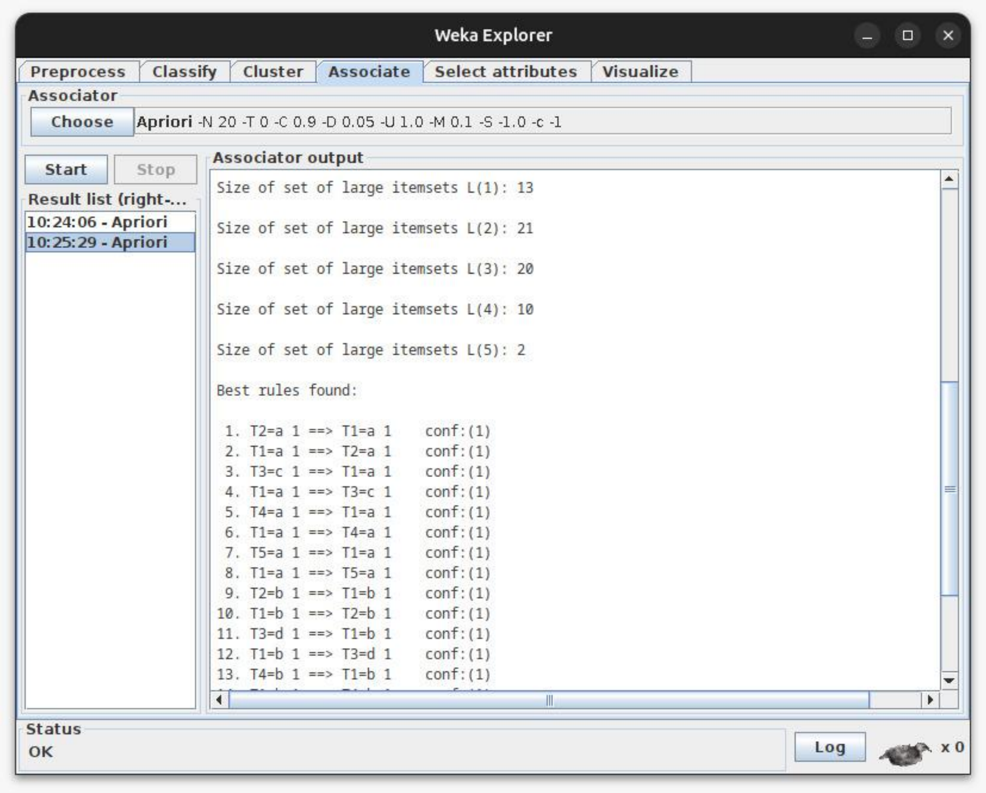
Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

```
Size of set of large itemsets L(1): 13

Size of set of large itemsets L(2): 21

Size of set of large itemsets L(3): 20

Size of set of large itemsets L(4): 10

Size of set of large itemsets L(5): 2

Best rules found:

 1. T2=a 1 ==> T1=a 1    conf:(1)
 2. T1=a 1 ==> T2=a 1    conf:(1)
 3. T3=c 1 ==> T1=a 1    conf:(1)
 4. T1=a 1 ==> T3=c 1    conf:(1)
 5. T4=a 1 ==> T1=a 1    conf:(1)
 6. T1=a 1 ==> T4=a 1    conf:(1)
 7. T5=a 1 ==> T1=a 1    conf:(1)
 8. T1=a 1 ==> T5=a 1    conf:(1)
 9. T2=b 1 ==> T1=b 1    conf:(1)
10. T1=b 1 ==> T2=b 1    conf:(1)
```



Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0
Relation:     APRIORY
Instances:    4
Attributes:   5
              T1
              T2
              T3
              T4
              T5
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.35 (1 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 13

Generated sets of large itemsets:

Size of set of large itemsets L(1): 13
```

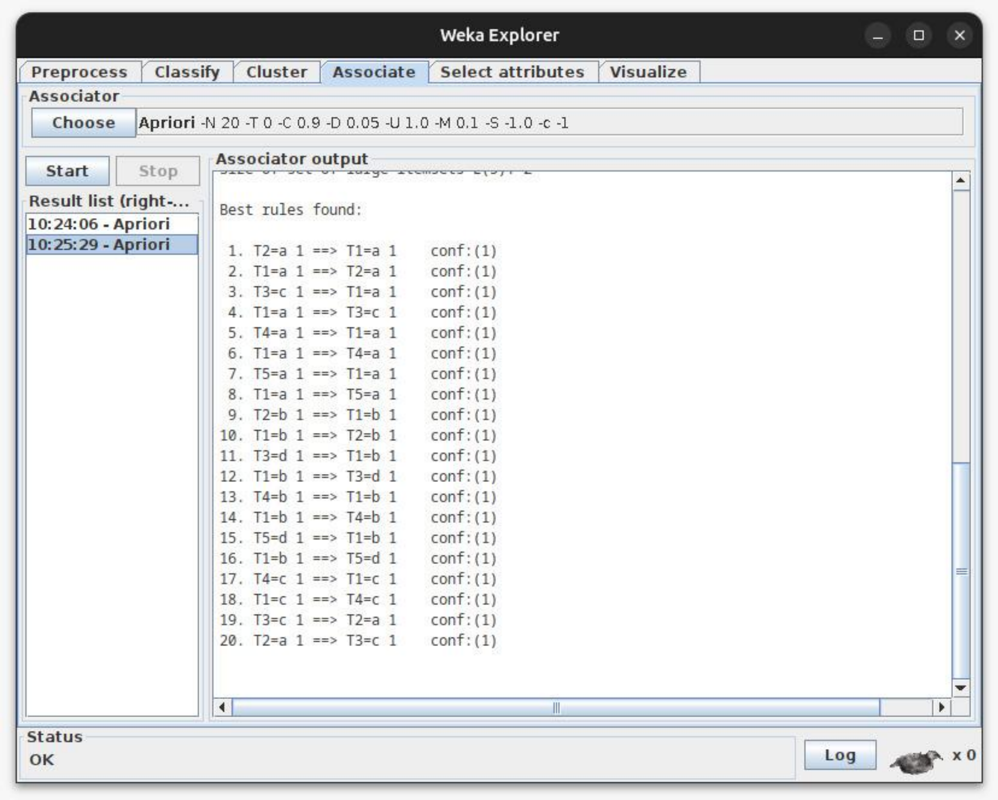


Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

10:24:06 - Apriori
10:25:29 - Apriori

```
Size of set of large itemsets L(1): 13

Size of set of large itemsets L(2): 21

Size of set of large itemsets L(3): 20

Size of set of large itemsets L(4): 10

Size of set of large itemsets L(5): 2

Best rules found:

 1. T2=a 1 ==> T1=a 1    conf:(1)
 2. T1=a 1 ==> T2=a 1    conf:(1)
 3. T3=c 1 ==> T1=a 1    conf:(1)
 4. T1=a 1 ==> T3=c 1    conf:(1)
 5. T4=a 1 ==> T1=a 1    conf:(1)
 6. T1=a 1 ==> T4=a 1    conf:(1)
 7. T5=a 1 ==> T1=a 1    conf:(1)
 8. T1=a 1 ==> T5=a 1    conf:(1)
 9. T2=b 1 ==> T1=b 1    conf:(1)
10. T1=b 1 ==> T2=b 1    conf:(1)
11. T3=d 1 ==> T1=b 1    conf:(1)
12. T1=b 1 ==> T3=d 1    conf:(1)
13. T4=b 1 ==> T1=b 1    conf:(1)
```

Status
OK



Apriori -N 20 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

10:24:06 - Apriori
10:25:29 - Apriori

```
Best rules found:

 1. T2=a 1 ==> T1=a 1    conf:(1)
 2. T1=a 1 ==> T2=a 1    conf:(1)
 3. T3=c 1 ==> T1=a 1    conf:(1)
 4. T1=a 1 ==> T3=c 1    conf:(1)
 5. T4=a 1 ==> T1=a 1    conf:(1)
 6. T1=a 1 ==> T4=a 1    conf:(1)
 7. T5=a 1 ==> T1=a 1    conf:(1)
 8. T1=a 1 ==> T5=a 1    conf:(1)
 9. T2=b 1 ==> T1=b 1    conf:(1)
10. T1=b 1 ==> T2=b 1    conf:(1)
11. T3=d 1 ==> T1=b 1    conf:(1)
12. T1=b 1 ==> T3=d 1    conf:(1)
13. T4=b 1 ==> T1=b 1    conf:(1)
14. T1=b 1 ==> T4=b 1    conf:(1)
15. T5=d 1 ==> T1=b 1    conf:(1)
16. T1=b 1 ==> T5=d 1    conf:(1)
17. T4=c 1 ==> T1=c 1    conf:(1)
18. T1=c 1 ==> T4=c 1    conf:(1)
19. T3=c 1 ==> T2=a 1    conf:(1)
20. T2=a 1 ==> T3=c 1    conf:(1)
```

Status
OK

Conclusion: Hence, we have studied and implemented Apriori algorithm using WEKA tool.