Name:	Sunit Sunil Khaire
Roll No:	19
Class/Sem:	TE/V
Experiment No.:	5
Title:	Using open-source tools Implement Association Mining Algorithms.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: To implement Apriori Algorithm on a large dataset using Open-source tool WEKA.

Objective: To make students well versed with open-source tools like WEKA to implement Apriori algorithm.

Theory:

- Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction.
- A typical example is a Market Based Analysis. Market Based Analysis is one of the key techniques used by large relations to show associations between items.
- It allows retailers to identify relationships between the items that people buy together frequently.
- Given a set of transactions, we can find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Support Count () – Frequency of occurrence of a itemset.
Here ({Milk, Bread, Diaper})= 2

Frequent Itemset – An itemset whose support is greater than or equal to minsup threshold.

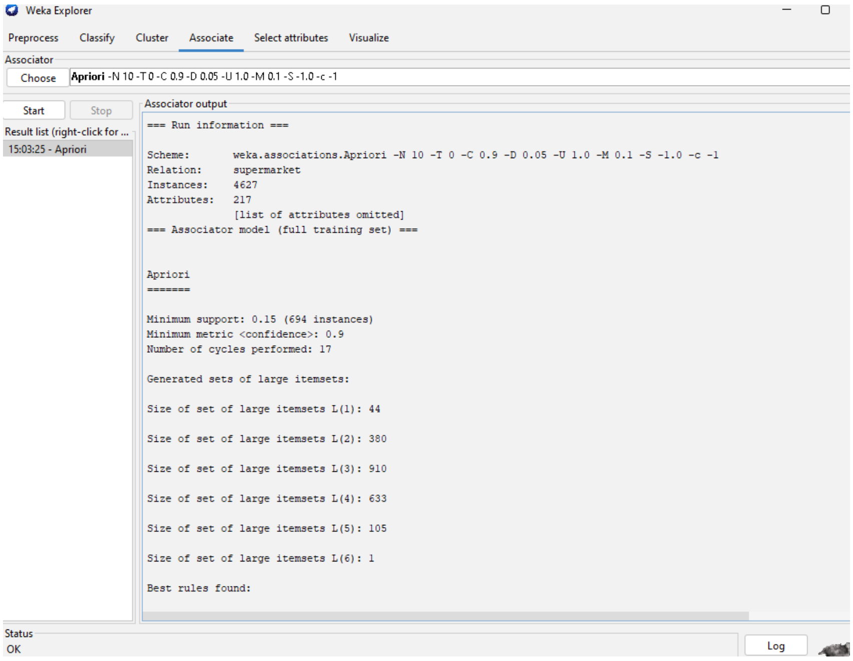
Association Rule – An implication expression of the form X Y, where X and Y are any 2 itemsets.

Example: {Milk, Diaper}{Beer}

- WEKA contains an implementation of the Apriori algorithm. The algorithm works only with discrete data.
- It can identify statistical dependencies between groups of attributes.
- Apriori algorithm can compute all rules that have a given minimum support and exceed a given confidence.
- Clicking on the "Associate" tab will bring up the interface for the association rule algorithms.

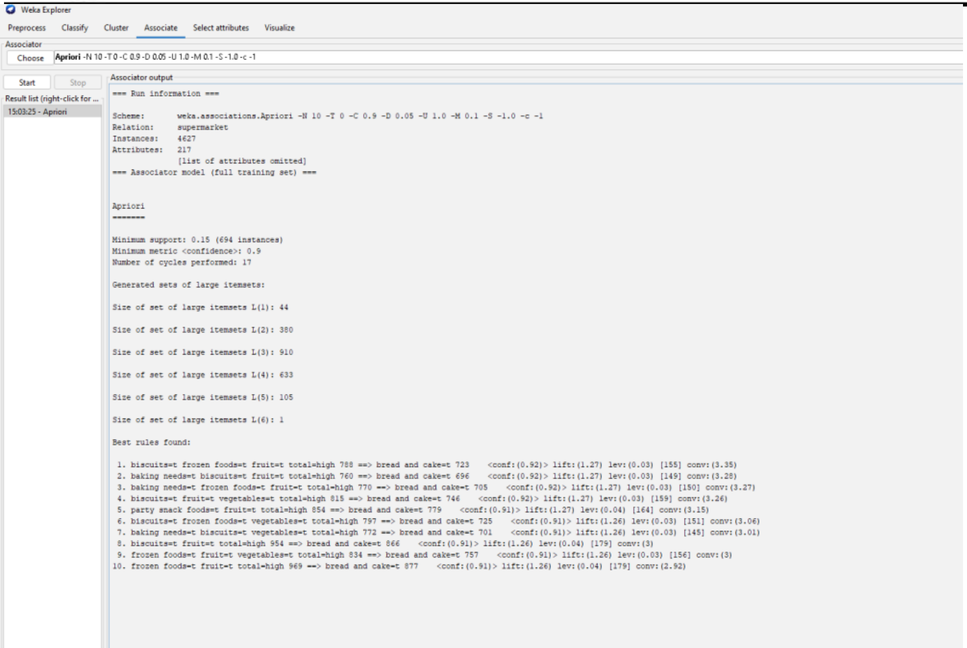
- The Apriori algorithm which we will use is the default algorithm selected. However, in order to change the parameters for this run (e.g., support, confidence, etc.) we click on the text box immediately to the right of the "Choose" button. Note that this box, at any given time, shows the specific command line arguments that are to be used for the algorithm.
- WEKA allows the resulting rules to be sorted according to different metrics such as confidence, leverage, and lift. We can also change the default value of rules (10) to be 20; this indicates that the program will report no more than the top 20 rules. The upper bound for minimum support is set to 1.0 (100%) and the lower bound to 0.1 (10%).
- Apriori in WEKA starts with the upper bound support and incrementally decreases support (by delta increments which by default is set to 0.05 or 5%). The algorithm halts when either the specified number of rules are generated, or the lower bound for min. support is reached. Once the parameters have been set, the command line text box will show the new command line. We now click on start to run the program. This results in a set of rules. The panel on the left ("Result list") now shows an item indicating the algorithm that was run and the time of the run. You can perform multiple runs in the same session each time with different parameters. Each run will appear as an item in the Result list panel. Clicking on one of the results in this list will bring up the details of the run, including the discovered rules in the right panel. In addition, right-clicking on the result set allows us to save the result buffer into a separate file. Note that the rules were discovered based on the specified threshold values for support and lift. For each rule, the frequency counts for the LHS and RHS of each rule is given, as well as the values for confidence, lift, leverage, and conviction. In most cases, it is sufficient to focus on a combination of support, confidence, and either lift or leverage to quantitatively measure the "quality" of the rule. However, the real value of a rule, in terms of usefulness and action ability, is subjective and depends heavily on the particular domain and business objectives.

OUTPUT:



Apriori -N 10 -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 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:
```



Apriori -N 10 -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 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     supermarket
Instances:    4627
Attributes:   217
              [list of attributes omitted]
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (694 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:

 1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723    <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
 2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696    <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
 3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705    <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
 4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746    <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)
 5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779    <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
 6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725    <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
 7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701    <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
 8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
 9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757    <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877    <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)
```

Conclusion:

1) Explain the main steps involved in the Apriori algorithm.

Ans. The Apriori algorithm works by:

1. Identifying frequent itemsets: Find items that meet the minimum support.
2. Pruning: Remove infrequent itemsets.
3. Generating association rules: Create rules from frequent itemsets using confidence and lift.
4. Repeating: Continue until no more frequent itemsets exist.

2) What are the key parameters in the Apriori algorithm and how do they affect its performance?

Ans. Key parameters in the Apriori algorithm and their impact:

1. Minimum support: Determines the threshold for frequent itemsets. Lower values find more itemsets but increase computational complexity; higher values reduce itemsets and processing time.

2. Minimum confidence: Sets the threshold for association rule strength. Lower values generate more rules, but may include weak ones; higher values give fewer but stronger rules.

3. Dataset size: Larger datasets increase time and memory requirements, affecting performance.

These parameters balance between computational efficiency and the quality of results.