Assignment No : 3

Aim : Apply a-Priori algorithm to find frequently occurring items from given data and generate strong association rules using support and confidence thresholds.

Objective :

1. Implementation of the Problem Statement using weka tool.
2. Finding frequently occurring items from given data.

Theory

A-Priori algorithm :

1. Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases
2. 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.
3. The frequent item sets determined by apriori can be used to determine association rules which highlight general trends in the database : this has applications in domains such as market basket analysis

1 Finding itemsets with high support

Using the apriori principle, the number of itemsets that have to be examined can be pruned, and the list of popular itemsets can be obtained in these steps:

Step 0 :- Start with itemsets containing just a single items, such as {apple} and {Pear}

Step 1 :- Determine the support for itemsets. Keep the itemsets that meet your minimum support threshold, and remove itemsets that do not.

Step 2 :- Using the itemsets you have kept from step 1, generate all the possible itemset configurations

Step 3 :- Repeat step 1 & 2 until there are no more new itemsets.

2] Finding item rules with high confidence or lift

1 We have seen how the apriori algorithm can be used to identify itemsets with high support.

2 The same principle can also be used to identify item associations with high confidence or lift.

3 Finding rules with high confidence or lift is less computationally taxing once high-support itemsets have been identified, because confidence and lift values are calculated using support values.

Take for example the task of finding high-confidence rules. if the rule

{Manyo, chips -> apple}

has low confidence, all other rules with the same constituent items and with apple on the right hand side would have low confidence too. Specifically, the rules

{Manyo -> apple, chips}
{Manyo -> apple, Manyo}

would have low confidence as well. as before, lower level candidate item rules can be pruned using the apriori algorithm, so that fewer candidate rules need to be examined.

Limitations

1 Computationally Expensive. Even through the apriori algorithms reduces the number of candidate itemsets to consider, this number could still be huge when store inventories are large

or when the support threshold is low. however, an alternative solution would be to reduce the number of comparisions by using advanced data structures, such as hash tables, to sort candidate itemsets more efficiently.

2 Spurious Associations :- Analysis of large inventories would involve more itemset configurations, and the support threshold might have to be lowered to detect certain associations. however, lowering the support threshold might also increase the number of spurious associations detected. To ensure that identified associations are generalizable, they could first be distilled from a training dataset, before having their support and confidence assessed in a separate test dataset.

Example :-

Assume that a large supermarket tracks sales data by stock-keeping unit for each item.
each item, such as "butter" or "bread", is identified by a numerical SKU. The supermarket has a database of transactions where each transaction is a set of SKUs that were bought together.

Let the database of transactions consist of following itemsets:

Itemsets
{1, 2, 3, 4}
{1, 2, 4}
{1, 2}
{2, 3, 4}
{2, 3}
{3, 4}
{2, 4}

We will use Apriori to determine the frequent item sets of this database.
To do this, we will say that an item set is frequent if it appears in at least 3 transactions of the database: the value 3 is the support threshold.

The first step of Apriori is to count up the number of occurrences, called the support, of each member item separately. By scanning the database for the first time, we obtain the following result.

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

All the itemsets of size 1 have a support of at least 3, so they are all frequent.

The next step is to generate a list of all pairs of the frequent items.

For example, regarding the pair {1,2}: the first table of example 2 shows items 1 and 2 appearing together in three of the itemsets; therefore, we say item {1,2} has support of three.

Item	Support
{1,2}	3
{1,3}	1
{1,4}	2
{2,3}	3
{2,4}	4
{3,4}	3

The Pairs {1,2}, {2,3}, {2,4} and {3,4} all meet or exceed the minimum support of 3, so they are frequent. The pairs {1,3} and {1,4} are not.

Now, because {1,3} and {1,4} are not frequent, any larger set which contains {1,3} or {1,4} cannot be frequent. In this way, we can prune sets: we will now look for frequent triples in the database, but we can already exclude all the triples that contain one of these two pairs:

Item	Support
{2,3,4}	2

In the example, there are no frequent triplets. {2,3,4} is below the minimal threshold, and the other triplets were excluded because they were super sets of pairs that were already below the threshold.

We have thus determined the frequent sets of items in the database, and illustrated how some items were not counted because one of their subsets was already known to be below the threshold.

Conclusion :- Thus we have applied a-priori algorithm to find frequently occurring items from given data and generated strong association rules using support and confidence thresholds.