Roll No. 22DEC	**Experiment No.** 05	**Marks :**
BATCH - C		**Sign :**

Aim: Implementation of Association Rule Mining algorithm (Apriori)

Apparatus: Google Colab

Theory:

What is Association Rule Mining?

Association rule mining is a technique used to identify patterns in large data sets. It involves finding relationships between variables in the data and using those relationships to make predictions or decisions. The goal of association rule mining is to uncover rules that describe the relationships between different items in the data set.

For example, consider a dataset of transactions at a grocery store. Association rule mining could be used to identify relationships between items that are frequently purchased together. For example, the rule "If a customer buys bread, they are also likely to buy milk" is an association rule that could be mined from this data set. We can use such rules to inform decisions about store layout, product placement, and marketing efforts.

Association rule mining typically involves using algorithms to analyze the data and identify the relationships. These algorithms can be based on statistical methods or machine learning techniques. The resulting rules are often expressed in the form of "if-then" statements, where the "if" part represents the antecedent (the condition being tested) and the "then" part represents the consequent (the outcome that occurs if the condition is met).

Association rule mining is an important technique in data analysis because it allows users to discover patterns or relationships within data that may not be immediately apparent. By identifying associations between variables, association rule mining can help users understand the relationships between different variables and how those variables may be related to one another.

This can be useful for various purposes, such as identifying market trends, detecting fraudulent activity, or understanding customer behavior. Association rule mining can also be used as a stepping stone for other types of data analysis, such as predicting outcomes or identifying key drivers of certain phenomena. Overall, association rule mining is a valuable tool for extracting insights and understanding the underlying structure of data.

Association Rule Mining Algorithms:

There are several algorithms used for association rule mining. Some common ones are:

1. **ECLAT algorithm**

   The ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) algorithm is a variation of the Apriori algorithm that uses a top-down approach rather than a bottom-up approach. It works by dividing the items into equivalence classes based on their support (the number of transactions in which they appear). The association rules are then generated by combining these equivalence classes in a lattice-like structure. It is a more efficient and scalable version of the Apriori algorithm.

2. **FP-Growth algorithm**

   The FP-Growth (Frequent Pattern Growth) algorithm is another popular algorithm for association rule mining. It works by constructing a tree-like structure called a FP-tree, which encodes the frequent itemsets in the dataset. The FP-tree is then used to generate association rules in a similar manner to the Apriori algorithm. The FP-Growth algorithm is generally faster than the Apriori algorithm, especially for large datasets.

3. **Apriori algorithm**

   The Apriori algorithm is one of the most widely used algorithms for association rule mining. It works by first identifying the frequent itemsets in the dataset (itemsets that appear in a certain number of transactions). It then uses these frequent itemsets to generate association rules, which are statements of the form "if item A is purchased, then item B is also likely to be purchased." The Apriori algorithm uses a bottom-up approach, starting with individual items and gradually building up to more complex itemsets.

   Algorithm Details:

   The apriori algorithm starts by setting the minimum support threshold. This is the minimum number of times an item must occur in the database in order for it to be considered a frequent itemset. The algorithm then filters out any candidate itemsets that do not meet the minimum support threshold.

   The algorithm then generates a list of all possible combinations of frequent itemsets and counts the number of times each combination appears in the database. The algorithm then generates a list of association rules based on the frequent itemset combinations.

   An association rule is a statement of the form "if item A is present in a transaction, then item B is also likely to be present". The strength of the association is measured using the confidence of the rule, which is the probability that item B is present given that item A is present.

The algorithm then filters out any association rules that do not meet a minimum confidence threshold. These rules are referred to as strong association rules. Finally, the algorithm then returns the list of strong association rules as output.

Apriori uses a "bottom-up" approach, starting with individual items and gradually combining them into larger and larger itemsets as it searches for frequent patterns. It also uses a "delete-relabel" approach to efficiently prune the search space by eliminating infrequent itemsets from consideration.

Implementation:

```
ip = ["i1i2i5", "i2i4", "i2i3", "i1i2i4", "i1i3", "i2i3", "i1i3", "i1i2i3i5", "i1i2i3"]
min_supp = 2

# Counting occurrences of individual items
item_counts = {'i1': 0, 'i2': 0, 'i3': 0, 'i4': 0, 'i5': 0}
for transaction in ip:
    for item in item_counts:
        if item in transaction:
            item_counts[item] += 1

print("Items \t\t Count")
for item, count in item_counts.items():
    print(item, "\t\t\t ", count)

# Counting occurrences of pairs of items
pair_counts = {'i1i2': 0, 'i1i3': 0, 'i1i4': 0, 'i1i5': 0, 'i2i3': 0, 'i2i4': 0, 'i2i5': 0, 'i3i4': 0, 'i3i5': 0, 'i4i5': 0}
for transaction in ip:
    for pair in pair_counts:
        if all(item in transaction for item in pair.split('i')):
            pair_counts[pair] += 1
print("\nItems \t\t Count")
for pair, count in pair_counts.items():
    if count >= min_supp:
        print(pair, "\t\t\t ", count)
```

```
triple_counts = {'i1i2i3': 0, 'i1i2i5': 0, 'i1i2i4': 0, 'i1i3i5': 0, 'i2i3i4': 0, 'i2i3i5': 0, 'i2i4i5': 0}
for transaction in ip:
    for triple in triple_counts:
        if all(item in transaction for item in triple.split('i')):
            triple_counts[triple] += 1
print("\nItems \t\t Count")
for triple, count in triple_counts.items():
    if count >= min_supp:
        print(triple, "\t\t\t ", count)
```

OUTPUT:

```
Items            Count
i1                       6
i2                       7
i3                       6
i4                       2
i5                       2

Items            Count
i1i2                     4
i1i3                     4
i1i5                     2
i2i3                     4
i2i4                     2
i2i5                     2

Items            Count
i1i2i3                   2
i1i2i5                   2
```

Conclusion:

Implementing the Apriori algorithm in Python facilitates efficient mining of association rules from transactional data. Through preprocessing, algorithm execution, and interpretation, valuable insights about item relationships are gained. These insights aid in optimizing product recommendations, market basket analysis, and understanding consumer behavior for informed business decisions.