St. Francis Institute of Technology, Mumbai-400 103

Department Of Information Technology

A.Y. 2024-2025
Class: TE-ITA/B, Semester: VI

Subject: **Business Intelligence Lab**

Experiment – 9 To implement Apriori association mining using any one language (JAVA/Python)

1. **Aim: :** Implementation of Apriori Association in Data Mining using any one Language

0. **Objectives:** After study of this experiment, the students will be able to implement Apriori Algorithm in (JAVA/R/Python)

0. **Outcomes:** After study of this experiment, the students will be able to
CO 5: Design and Implement various frequent data mining techniques and formulate association rules on large data sets

0. **Prerequisite:** Introduction to algorithms of Associativity

0. **Requirements:** Personal Computer, Windows XP operating system/Windows 7, Internet Connection, Microsoft Word, WEKA tool.

0. **Theory:**
   a. What are Association Rules in Data Mining?
   a. How Association helps in Boosting the Business Profit

0. **Laboratory Exercise:** Implementation of Apriori Algorithm in Java/Python, printout of implementation along with coding and snapshot.

0. **Post-Experiments Exercise**
   a. Questions:
   - MCQ type test
   - Compare Apriori and FP Tree

   a. Conclusion:
   - Summary of Experiment
   - Importance of Experiment
   - Application of Experiment

0. **Reference:** Data Mining: Concept & Techniques, 3rd Edition, Jiawei Han, Micheline Kamber, Jian Pei, Elsevier.

Theory:

What are Association Rules in Data Mining?

Definition

Association rules are "if-then" statements that help identify relationships between variables in large datasets. In the context of market basket analysis, for example, an association rule might be formulated as: “If a customer buys item A, then they are likely to buy item B.” These rules are not deterministic but are based on probabilities derived from the data.

Key Metrics

To evaluate the strength and usefulness of association rules, several key metrics are used:

- **Support:**
  Support measures how frequently an itemset appears in the dataset. It is the proportion of transactions that contain the itemset. A rule with higher support is generally more reliable as it is based on a larger subset of data.
- **Confidence:**
  Confidence indicates the likelihood that the consequent (the "then" part) is present in transactions that contain the antecedent (the "if" part). It is calculated as the ratio of the number of transactions with both the antecedent and consequent to the number of transactions with just the antecedent.
- **Lift:**
  Lift measures how much more likely the consequent is to occur when the antecedent is present, compared to its occurrence by chance. A lift value greater than 1 suggests a positive association between the items, indicating that the presence of the antecedent increases the probability of the consequent.

How Association helps in Boosting the Business Profit

Enhanced Market Basket Analysis

- Understanding Customer Behavior:

By analyzing association rules, businesses can determine which products are often purchased together. For instance, if analysis reveals that customers who buy bread are likely to purchase butter as well, retailers can arrange these items in proximity or create bundled offers. This strategic placement can drive incremental sales by encouraging add-on purchases.

Optimized Promotions and Discounts:

With clear insights into product associations, businesses can design targeted promotions. For example, a discount on complementary products purchased together can incentivize customers to buy more items, thereby increasing the average transaction value.

Inventory and Supply Chain Management

- Demand Forecasting:

Knowing which products are commonly bought together helps in forecasting demand more accurately. Businesses can ensure that the associated items are stocked in the right quantities, reducing the risk of stockouts or overstock situations. This leads to more efficient inventory management and lower holding costs.

Cross-Selling and Up-Selling Opportunities:

Association rules can be used to implement effective cross-selling and up-selling strategies. For instance, an online retailer can recommend related products based on a customer’s current selection, enhancing the shopping experience and driving additional revenue.

Customer Relationship and Loyalty

- Personalized Recommendations:

Leveraging association rules enables businesses to create personalized recommendation systems. When customers receive suggestions based on their previous purchases or browsing behavior, they are more likely to engage and make additional purchases. This personalization improves customer satisfaction and loyalty.

Improved Customer Insights:

By understanding the associations between different products, companies can segment their customers more effectively and tailor their marketing messages. This segmentation allows for more targeted communication and fosters long-term relationships with customers.

Strategic Decision-Making

- Pricing Strategies:

Insights from association rules can inform pricing strategies by highlighting which products have a strong complementary relationship. Businesses might bundle these items together at a competitive price, increasing the perceived value for the customer while driving higher overall sales.

Store Layout and Product Placement:

Retailers can use association rules to design more effective store layouts. By positioning frequently associated items near each other, businesses can improve the shopping experience and stimulate impulse purchases.

Laboratory Exercise: Implementation of Apriori Algorithm in Python

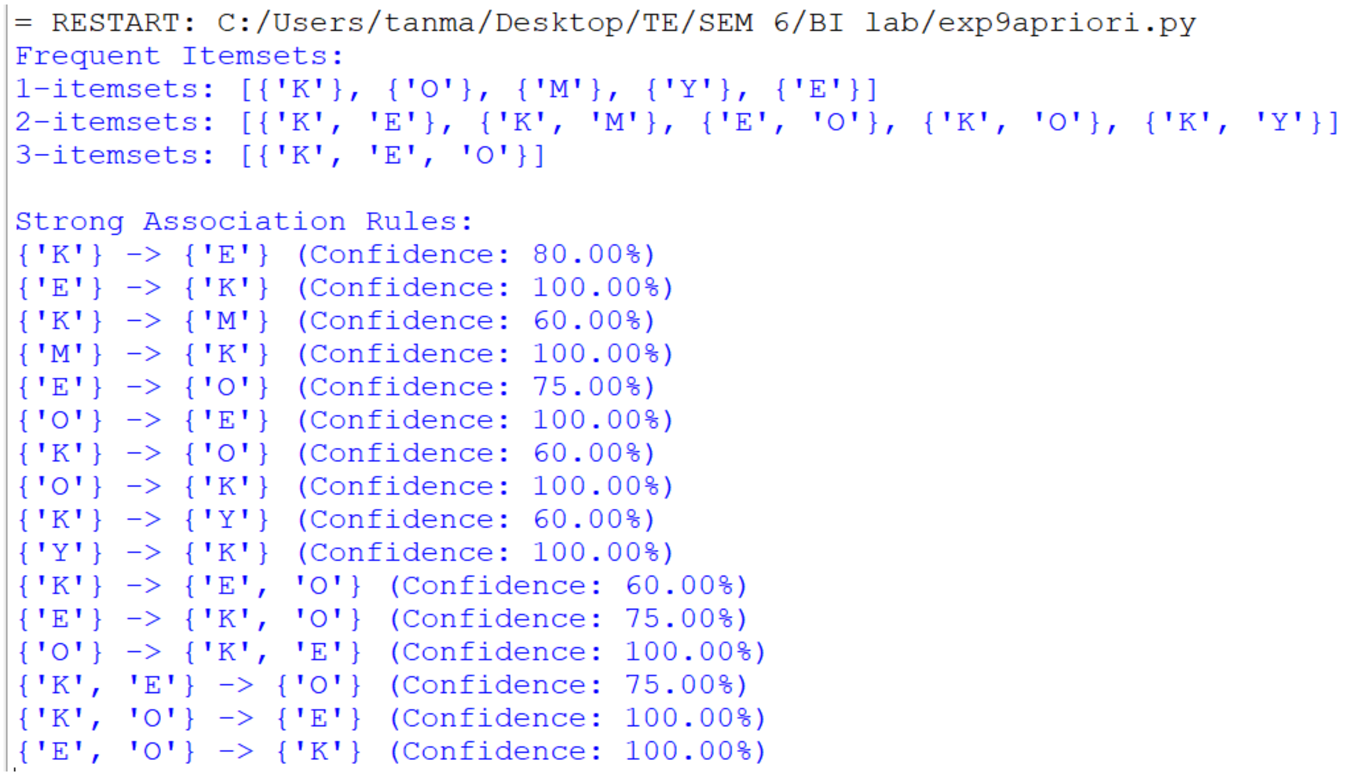
Consider the following database with minimum support count=60% Find all frequent itemsets using Apriori Algorithm. And also generate strong association rules if the minimum confidence is 50%.

T_Id	**ITEMS BOUGHT**
T1	**M, O, N, K, E, Y**
T2	**D, O, N, K, E, Y**
T3	**M, A, K, E**
T4	**M, U, C, K, Y**
T5	**C, O, O, K, I, E**

```
from itertools import combinations

transactions = [
    {'M', 'O', 'N', 'K', 'E', 'Y'},
    {'D', 'O', 'N', 'K', 'E', 'Y'},
    {'M', 'A', 'K', 'E'},
    {'M', 'U', 'C', 'K', 'Y'},
    {'C', 'O', 'K', 'I', 'E'}
]

min_support_count = 3  # 60% of 5 transactions
min_confidence = 50
# Apriori Algorithm
def apriori(transactions, min_support):
    items = set.union(*transactions)
    freq_itemsets = {}

    # Generate L1
    L1 = {}
    for item in items:
        count = sum(1 for t in transactions if item in t)
        if count >= min_support:
            L1[frozenset({item})] = count
    freq_itemsets[1] = L1
     k = 2
    while True:
        Ck = set()
        Lk_prev = freq_itemsets[k-1]
        for itemset1 in Lk_prev:
            for itemset2 in Lk_prev:
                union = itemset1 | itemset2
                if len(union) == k and union not in Ck:
                    Ck.add(union)

        # Prune Ck
        Lk = {}
        for candidate in Ck:
            count = sum(1 for t in transactions if candidate.issubset(t))
            if count >= min_support:
                Lk[candidate] = count
        if not Lk:
            break
        freq_itemsets[k] = Lk
        k += 1
    return freq_itemsets

freq_itemsets = apriori(transactions, min_support_count)

# Generate Association Rules
def generate_rules(freq_itemsets, min_conf):
    rules = []
    for k in freq_itemsets.keys():
        if k == 1:
            continue
        for itemset in freq_itemsets[k]:
            subsets = [frozenset(s) for l in range(1, k) for s in combinations(itemset, l)]
            for antecedent in subsets:
                consequent = itemset - antecedent
                support_itemset = freq_itemsets[len(itemset)][itemset]
                support_antecedent = freq_itemsets[len(antecedent)][antecedent]
                confidence = (support_itemset / support_antecedent)*100
                if confidence >= min_conf:
                    rules.append((antecedent, consequent, confidence))
    return rules
```

```
association_rules = generate_rules(freq_itemsets, min_confidence)

# Output Results
print("Frequent Itemsets:")
for k in freq_itemsets:
   print(f"{k}-itemsets:", [set(itemset) for itemset in freq_itemsets[k].keys()])

print("\nStrong Association Rules:")
for rule in association_rules:
   antecedent, consequent, confidence = rule
   print(f"{set(antecedent)} -> {set(consequent)} (Confidence: {confidence:.2f}%)")
```

OUTPUT:



```
= RESTART: C:/Users/tanma/Desktop/TE/SEM 6/BI lab/exp9apriori.py
Frequent Itemsets:
1-itemsets: [{'K'}, {'O'}, {'M'}, {'Y'}, {'E'}]
2-itemsets: [{'K', 'E'}, {'K', 'M'}, {'E', 'O'}, {'K', 'O'}, {'K', 'Y'}]
3-itemsets: [{'K', 'E', 'O'}]

Strong Association Rules:
{'K'} -> {'E'} (Confidence: 80.00%)
{'E'} -> {'K'} (Confidence: 100.00%)
{'K'} -> {'M'} (Confidence: 60.00%)
{'M'} -> {'K'} (Confidence: 100.00%)
{'E'} -> {'O'} (Confidence: 75.00%)
{'O'} -> {'E'} (Confidence: 100.00%)
{'K'} -> {'O'} (Confidence: 60.00%)
{'O'} -> {'K'} (Confidence: 100.00%)
{'K'} -> {'Y'} (Confidence: 60.00%)
{'Y'} -> {'K'} (Confidence: 100.00%)
{'K'} -> {'E', 'O'} (Confidence: 60.00%)
{'E'} -> {'K', 'O'} (Confidence: 75.00%)
{'O'} -> {'K', 'E'} (Confidence: 100.00%)
{'K', 'E'} -> {'O'} (Confidence: 75.00%)
{'K', 'O'} -> {'E'} (Confidence: 100.00%)
{'E', 'O'} -> {'K'} (Confidence: 100.00%)
```