Lab Number: 13

Title
Given minimum support of 2, programmatically generate frequent itemsets using Apriori method

Objective
To implement the Apriori algorithm programmatically in Python to generate frequent itemsets from a transactional dataset using a minimum support of 2, and to interpret the patterns found.

IDE/Tools Used
VS Code

Programming Language Used
Python

Theory
Apriori Algorithm: Apriori is a classic algorithm used to generate frequent itemsets and from them, association rules. The key steps in apriori are:

- Identify all items with support ≥ minimum support (frequent 1-itemsets).
- Generate candidate 2-itemsets from frequent 1-itemsets.
- Prune candidates that do not meet the minimum support threshold.
- Repeat for k-itemsets until no further frequent itemsets are found.

The key concepts related to Apriori are:

- **Support:** Number of transactions containing an itemset.
- **Frequent Itemset:** An itemset whose support ≥ minimum support threshold.
- **Apriori Property:** Any subset of a frequent itemset must also be frequent.

Advantages of Apriori algorithm

- Reduces the search space for candidate itemsets.
- Easy to implement for small to medium datasets.

Limitations of Apriori algorithm

- Can be inefficient for very large datasets due to multiple database scans.
- Apriori is particularly efficient for datasets with boolean or categorical attributes.

Encoding: Encoding is the process of converting categorical (non-numeric) data into numerical values so that machine learning algorithms can process them. Most algorithms cannot directly interpret text labels because they rely on mathematical operations like distance calculation.

Therefore, encoding translates human-readable categories into machine-readable numbers while preserving the meaning of the categories.

One-Hot Encoding: One-Hot Encoding is a method of converting categorical variables into a binary matrix, where each category is represented as a separate column and a value of 1 (or True) indicates the presence of that category, while 0 (or False) indicates its absence.

Why it's needed for Apriori: Apriori in Python (mlxtend) requires input as a boolean matrix, with each column representing an item and each row representing a transaction. This is necessary so the algorithm can efficiently compute supports of individual items and their combinations. One-Hot Encoding ensures the transaction data is in the correct format for Apriori to identify frequent itemsets.

Implementation

The python code to implement the apriori algorithm to generate frequent item set is given below:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

# Small dataset containing list of transaction
dataset = [
    ['Bread', 'Milk'],
    ['Bread', 'Diaper', 'Beer'],
    ['Milk', 'Diaper', 'Beer', 'Cola'],
    ['Bread', 'Milk', 'Diaper', 'Beer'],
    ['Bread', 'Milk', 'Diaper', 'Cola']
]

# Converting the transactions to one-hot encoded DataFrame
te = TransactionEncoder()
te_array = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_array, columns=te.columns_)

# Applying Apriori algo with min support of 2
min_support = 2 / len(dataset)  # 2 transactions out of 5
frequent_itemsets = apriori(df, min_support=min_support, use_colnames=True)
frequent_itemsets['support_count'] = frequent_itemsets['support'] *
len(dataset)

# Display frequent itemsets
print(frequent_itemsets)
```

Output:

```
sayuj@Sayujya MINGW64 ~/OneDrive/Desktop/BIM054/BIM_7th/IT 274_Data Warehousing and Data Mining/Labs/Lab 13 Apriori Program
$ python -u "c:\Users\sayuj\OneDrive\Desktop\BIM054\BIM_7th\IT 274_Data Warehousing and Data Mining\Labs\Lab 13 Apriori Program\apriori.py"
    support              itemsets  support_count
0       0.6                (Beer)            3.0
1       0.8               (Bread)            4.0
2       0.4                (Cola)            2.0
3       0.8              (Diaper)            4.0
4       0.8                (Milk)            4.0
5       0.4         (Beer, Bread)            2.0
6       0.6        (Diaper, Beer)            3.0
7       0.4          (Beer, Milk)            2.0
8       0.6       (Diaper, Bread)            3.0
9       0.6         (Milk, Bread)            3.0
10      0.4        (Cola, Diaper)            2.0
11      0.4          (Cola, Milk)            2.0
12      0.6        (Diaper, Milk)            3.0
13      0.4  (Diaper, Beer, Bread)            2.0
14      0.4   (Diaper, Beer, Milk)            2.0
15      0.4  (Diaper, Milk, Bread)            2.0
16      0.4   (Cola, Diaper, Milk)            2.0
```

Figure 1: Output in the console

Discussion

The Apriori algorithm was applied to a small transaction dataset with a minimum support of 2 transactions (support ≥ 0.4). The output shows both single items and combinations that appear frequently:

- **Single items:** Bread, Diaper, Milk, and Beer are the most frequent, appearing in 3-4 transactions each. Cola appears less frequently, in only 2 transactions.
- **Pairs of items:** Common pairs include (Diaper, Milk), (Milk, Bread), (Diaper, Bread), (Diaper, Beer), and (Beer, Bread) with support counts of 2-3, indicating customers often purchase these items together.
- **Triplets of items:** Frequent triplets like (Diaper, Milk, Bread), (Diaper, Beer, Bread), and (Diaper, Beer, Milk) all appear in 2 transactions, showing more complex co-occurrence patterns.

The results demonstrate that Apriori effectively identifies frequent itemsets in transactional data, from single items to combinations of three, which can be used to generate association rules or inform retail marketing strategies.

Conclusion

In this lab, the Apriori algorithm identified frequent itemsets in a small transaction dataset using a minimum support threshold corresponding to 2 transactions. The analysis revealed both individual popular items (Bread, Diaper, Milk, Beer) and frequently co-purchased combinations. This demonstrates the effectiveness of Apriori in discovering meaningful patterns in transactional datasets, which can inform decision-making for product placement, cross-selling, and recommendation systems.