Name : Bhavna More

Div : TY3

Batch : B

Roll no : 38

Subject : Data WareHouse And Mining

Experiment 5 : Implementation of Association Rule Mining algorithm (Apriori)

Aim :

To implement the Apriori Algorithm for Association Rule Mining to find frequent itemsets and generate strong association rules from a given dataset.

Introduction:

Association Rule Mining (ARM) is a technique used in data mining to identify relationships between variables in large datasets. The Apriori Algorithm is a widely used approach for ARM, which operates on the principle that if an itemset is frequent, then its subsets must also be frequent.

It involves two main steps:

Frequent Itemset Generation – Finding all itemsets that occur frequently in the dataset. Rule Generation – Extracting strong association rules using the confidence threshold. Apriori is commonly used in market basket analysis to discover patterns such as "If a customer buys bread, they are likely to buy butter."

Procedure:

Step 1: Load the Dataset The dataset should contain transactional data where each transaction consists of a set of items. Step 2: Set Minimum Support & Confidence Support: The fraction of transactions that contain an itemset. Confidence: The likelihood that an item Y is bought when item X is bought. Step 3: Generate Frequent Itemsets using Apriori Algorithm Find all frequent 1-itemsets (items appearing in transactions above the support threshold). Generate candidate (k+1)-itemsets from frequent k-itemsets using the Apriori Property (if an itemset is frequent, its subsets must be frequent). Prune infrequent itemsets that do not meet the support threshold. Repeat until no more frequent itemsets can be generated. Step 4: Generate Association Rules For each frequent itemset, generate association rules in the form of $X \rightarrow Y$ (where X and Y are subsets of the itemset). Calculate Confidence = (Support of $X \cup Y$) / (Support of X). Retain rules

that meet the confidence threshold. Step 5: Display and Interpret Results Print frequent itemsets and association rules. Analyze patterns for insights (e.g., common purchase behaviors).

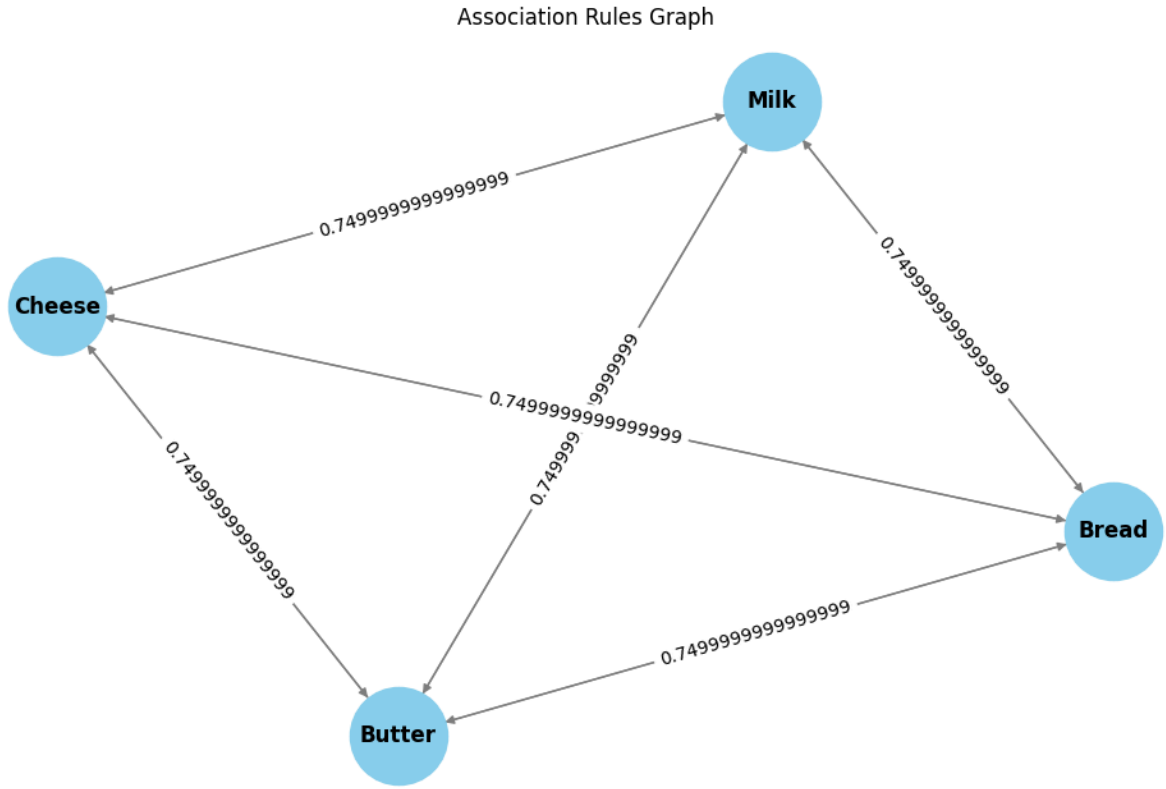
```
# Step 1: Install necessary libraries
!pip install mlxtend networkx matplotlib

# Step 2: Import libraries
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
import networkx as nx
import matplotlib.pyplot as plt

# Step 3: Example transaction data
data = {
    'Bread': [1, 1, 0, 1, 1],
    'Milk': [1, 1, 1, 1, 0],
    'Butter': [0, 1, 1, 1, 1],
    'Cheese': [1, 0, 1, 1, 1]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Display the dataset
print("Transaction Data:")
print(df)

# Step 4: Apply Apriori to find frequent itemsets with min_support=0.5
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
print("\nFrequent Itemsets:")
print(frequent_itemsets)

# Step 5: Generate association rules with min_confidence=0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
print("\nAssociation Rules:")
print(rules)

# Step 6: Visualize association rules using NetworkX
# Create a graph object
G = nx.DiGraph()

# Add edges for each rule in the dataframe
for index, row in rules.iterrows():
    antecedent = list(row['antecedents'])[0]  # Get the antecedent
    consequent = list(row['consequents'])[0]  # Get the consequent
    G.add_edge(antecedent, consequent, weight=row['confidence'])  # Add edge with confide

# Draw the graph
plt.figure(figsize=(10, 6))
pos = nx.spring_layout(G, k=0.5, seed=42)  # Position the nodes using spring layout
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='skyblue', font_size=12, fon
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
plt.title("Association Rules Graph")
```

```
    plt.show()
```

```
Requirement already satisfied: mlxtend in /usr/local/lib/python3.11/dist-packages (0.
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (3
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: scikit-learn>=1.3.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist
Transaction Data:
   Bread  Milk  Butter  Cheese
0      1     1       0       1
1      1     1       1       0
2      0     1       1       1
3      1     1       1       1
4      1     0       1       1

Frequent Itemsets:
   support          itemsets
0      0.8           (Bread)
1      0.8            (Milk)
2      0.8          (Butter)
3      0.8          (Cheese)
4      0.6     (Milk, Bread)
5      0.6   (Butter, Bread)
6      0.6   (Cheese, Bread)
7      0.6    (Milk, Butter)
8      0.6    (Milk, Cheese)
9      0.6  (Butter, Cheese)

Association Rules:
   antecedents consequents  antecedent support  consequent support  support  \
0       (Milk)     (Bread)                 0.8                 0.8      0.6
1      (Bread)      (Milk)                 0.8                 0.8      0.6
2     (Butter)     (Bread)                 0.8                 0.8      0.6
3      (Bread)    (Butter)                 0.8                 0.8      0.6
4     (Cheese)     (Bread)                 0.8                 0.8      0.6
5      (Bread)    (Cheese)                 0.8                 0.8      0.6
6       (Milk)    (Butter)                 0.8                 0.8      0.6
7     (Butter)      (Milk)                 0.8                 0.8      0.6
8       (Milk)    (Cheese)                 0.8                 0.8      0.6
9     (Cheese)      (Milk)                 0.8                 0.8      0.6
10    (Butter)    (Cheese)                 0.8                 0.8      0.6
11    (Cheese)    (Butter)                 0.8                 0.8      0.6

    confidence    lift  representativity  leverage  conviction  zhangs_metric  \
0         0.75  0.9375               1.0     -0.04         0.8          -0.25
1         0.75  0.9375               1.0     -0.04         0.8          -0.25
2         0.75  0.9375               1.0     -0.04         0.8          -0.25
```

```
3         0.75  0.9375                1.0     -0.04         0.8          -0.25
4         0.75  0.9375                1.0     -0.04         0.8          -0.25
5         0.75  0.9375                1.0     -0.04         0.8          -0.25
6         0.75  0.9375                1.0     -0.04         0.8          -0.25
7         0.75  0.9375                1.0     -0.04         0.8          -0.25
8         0.75  0.9375                1.0     -0.04         0.8          -0.25
9         0.75  0.9375                1.0     -0.04         0.8          -0.25
10        0.75  0.9375                1.0     -0.04         0.8          -0.25
11        0.75  0.9375                1.0     -0.04         0.8          -0.25
```

```
    jaccard  certainty  kulczynski
0       0.6      -0.25        0.75
1       0.6      -0.25        0.75
2       0.6      -0.25        0.75
3       0.6      -0.25        0.75
4       0.6      -0.25        0.75
5       0.6      -0.25        0.75
6       0.6      -0.25        0.75
7       0.6      -0.25        0.75
8       0.6      -0.25        0.75
9       0.6      -0.25        0.75
10      0.6      -0.25        0.75
11      0.6      -0.25        0.75
/usr/local/lib/python3.11/dist-packages/mlxtend/frequent_patterns/fpcommon.py:161: De
  warnings.warn(
```



Conclusion:

Review Questions

1.**What is apriori algorithm in association rule mining?**

The Apriori Algorithm is a popular method in Association Rule Mining used to identify frequent itemsets in large datasets and generate strong association rules. It works on the Apriori Property, which states that if an itemset is frequent, all its subsets must also be frequent. The algorithm follows these steps:

Find frequent itemsets (items that appear together often in transactions).

Generate candidate itemsets by joining frequent itemsets.

Prune infrequent itemsets using a minimum support threshold.

Generate strong association rules using confidence and lift.

2.**What is significance of support,confidence and lift in apriori?**

Significance of Support, Confidence, and Lift in Apriori Support

Measures how frequently an itemset appears in the dataset. Formula: Support(X) = (Transactions containing X) / (Total Transactions) Higher support means the itemset is common and significant. Confidence

Measures how often rule X → Y is correct when X occurs. Formula: Confidence(X → Y) = Support(X ∪ Y) / Support(X) Higher confidence means a stronger likelihood that Y occurs when X occurs. Lift

Measures how much more likely Y is to occur when X occurs, compared to when they are independent. Formula: Lift(X → Y) = Confidence(X → Y) / Support(Y) Lift > 1 indicates a strong positive correlation, =1 means no association, <1 means a negative association.