```
Name: Sumeet Kedaare

Class: TY-3

RollNo.: 31

Batch : B
```

# DWM Ex5- Implementation of Association Rule Mining Algorithm (Apriori)

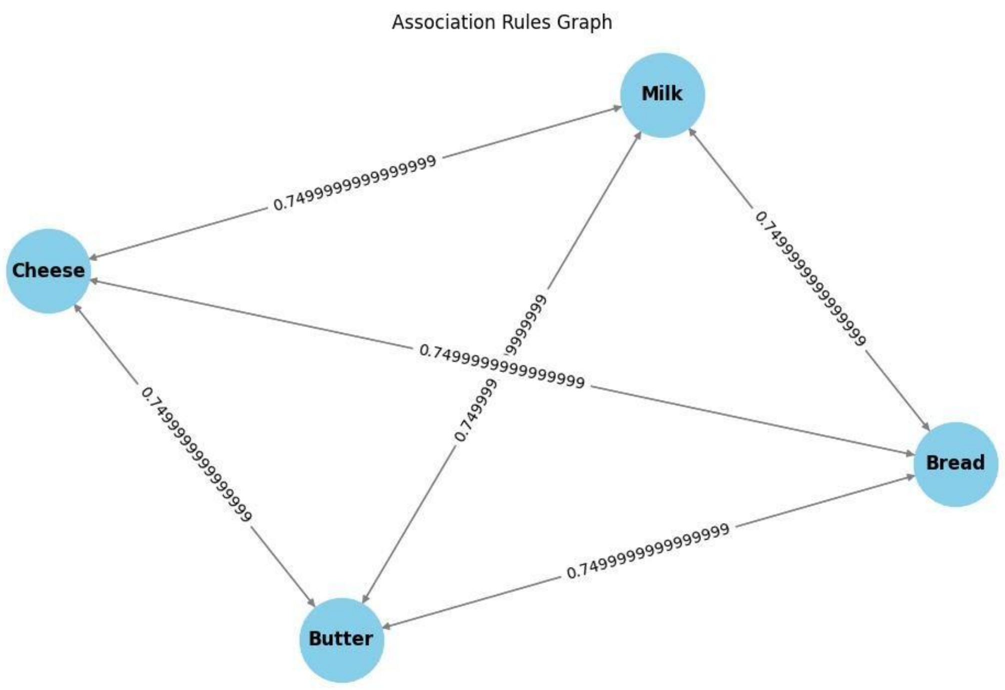
```
!pip install mlxtend networkx matplotlib

import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules import
networkx as nx
import matplotlib.pyplot as plt

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]
}


df = pd.DataFrame(data)

print("Transaction Data:") print(df)

frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
print("\nFrequent Itemsets:") print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.7) print("\nAssociation Rules:") print(rules)

G = nx.DiGraph()

for index, row in rules.iterrows():
    antecedent = list(row['antecedents'])[0]
consequent = list(row['consequents'])[0]
    G.add_edge(antecedent, consequent, weight=row['confidence'])
```

```
plt.figure(figsize=(10, 6)) pos =
nx.spring_layout(G, k=0.5, seed=42)
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='skyblue',
font_size=12, font_weight='bold', edge_color='gray') 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/distpackages (0.23.4)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/distpackages (3.4.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/distpackages (3.10.0)
Requirement already satisfied: scipy>=1.2.1 in
/usr/local/lib/python3.11/dist-packages (from mlxtend) (1.14.1)
Requirement already satisfied: numpy>=1.16.2 in
/usr/local/lib/python3.11/dist-packages (from mlxtend) (2.0.2)
Requirement already satisfied: pandas>=0.24.2 in
/usr/local/lib/python3.11/dist-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from mlxtend) (1.6.1)
Requirement already satisfied: joblib>=0.13.2 in
/usr/local/lib/python3.11/dist-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/distpackages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24.2->mlxtend)
(2025.1)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24.2->mlxtend)
(2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

```
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.3.1->mlxtend)
(3.6.0)
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    (Cheese, Milk)
9      0.6  (Cheese, Butter)

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       (Cheese)      (Milk)                0.8                 0.8      0.6
9       (Milk)    (Cheese)                0.8                 0.8      0.6
10      (Cheese)    (Butter)                0.8                 0.8      0.6
        11    (Butter)    (Cheese)                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/distpackages/mlxtend/frequent_patterns/fpcommon.py:
161: DeprecationWarning: DataFrames with non-bool types result in worse
computationalperformance and their support might be discontinued in the
future.Please use a DataFrame with bool type   warnings.warn(
```



Q.1) **What is the Apriori algorithm in Association Rule Mining?**

-> The Apriori algorithm is a fundamental technique in Association Rule Mining used to identify frequent itemsets in large datasets and derive strong association rules. It follows a bottom-up approach, where frequent itemsets are extended one item at a time (the Apriori property), ensuring that all subsets of a frequent itemset are also frequent. The algorithm consists of two main steps: candidate generation (finding potential frequent itemsets) and pruning (removing infrequent ones). It is widely used in market basket analysis, recommendation systems, and data mining applications to uncover relationships between items, such as products frequently purchased together.

Q.2) **What is the significance of support, confidence, and lift in Apriori?**

->In the **Apriori algorithm**, three key metrics evaluate association rules:

1. **Support**: Measures how frequently an itemset appears in the dataset, helping identify popular item combinations.

1. **Confidence**: Indicates the likelihood of item **Y** being purchased when item **X** is bought, assessing rule reliability.

2. **Lift**: Compares confidence to the expected probability of **Y** occurring independently. A **lift > 1** suggests a strong positive correlation, **=1** means independence, and **<1** implies a negative relationship.

These metrics help in filtering meaningful rules in **market basket analysis** and other data mining tasks.