Assignment No. 6

Problem Statement: Market Basket Analysis using Apriori Algorithm.

Objective: To perform Market Basket Analysis using the Apriori Algorithm for association rule mining. This assignment focuses on discovering frequent itemsets and deriving association rules from transaction data using the Apriori algorithm.

Prerequisite :

1. Python environment with libraries: pandas, mlxtend, numpy, matplotlib, seaborn.
2. Transaction dataset in list or DataFrame format.
3. Basic understanding of association rule mining and support-confidence-lift metrics.

Theory :

1. Understanding the Dataset

Before applying the Apriori algorithm, it's essential to explore the dataset:

- **Shape of Data: Use `.shape` to get the number of transactions and items.**
- **Data Types: Ensure the dataset contains categorical data suitable for association analysis.**
- **Missing Values: Use `.isnull().sum()` to check for nulls and handle them properly.**

---

2. Data Preprocessing

To prepare the data for the Apriori algorithm:

- **Transaction Format: Convert data into a one-hot encoded matrix using `TransactionEncoder`, where each item in a transaction is marked as 1.**
- **Remove Rare Items: Eliminate items with very low frequency to reduce noise and focus on more relevant itemsets.**

---

3. Applying the Apriori Algorithm

Apriori finds frequent itemsets based on:

- **Support: Frequency of an itemset in the dataset.**
- **Confidence: Probability that item Y is bought when item X is bought.**
- **Lift: Strength of the rule compared to random chance.**

Use the `apriori()` function from `mlxtend` to find frequent itemsets.

---

4. Generating Rules

After identifying frequent itemsets:

- **Use `association_rules()` from `mlxtend` to generate rules.**
- **Analyze rules using support, confidence, and lift.**
- **Sort by lift or confidence to find the strongest patterns.**

---

5. Business Insights

The rules can help:

1. **Identify frequently bought-together products.**
2. **Suggest cross-selling opportunities.**
3. **Improve product placement in stores or on websites.**

4. Code & Output

[19]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv('/Users/pranavashokdivekar/this_mac/Machine Learning/groceries-groceries.csv', on_bad_lines='skip')

# Preview the first few rows
print(df.head())
```

```
   Item(s)             Item 1               Item 2           Item 3  \
0        4       citrus fruit  semi-finished bread        margarine
1        3     tropical fruit               yogurt           coffee
2        1         whole milk                  NaN              NaN
3        4          pip fruit               yogurt     cream cheese
4        4   other vegetables           whole milk  condensed milk

                    Item 4 Item 5 Item 6 Item 7 Item 8 Item 9  ... Item 23  \
0              ready soups    NaN    NaN    NaN    NaN    NaN  ...     NaN
1                      NaN    NaN    NaN    NaN    NaN    NaN  ...     NaN
2                      NaN    NaN    NaN    NaN    NaN    NaN  ...     NaN
3             meat spreads    NaN    NaN    NaN    NaN    NaN  ...     NaN
4  long life bakery product    NaN    NaN    NaN    NaN    NaN  ...     NaN

  Item 24 Item 25 Item 26 Item 27 Item 28 Item 29 Item 30 Item 31 Item 32
0     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
1     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
2     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
3     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN     NaN

[5 rows x 33 columns]
```

[20]:
```python
# Step 2: Convert rows to list of items (ignoring NaN/empty cells)
transactions = df.drop('Item(s)', axis=1).values.tolist()
```

```python
[20]: # Step 2: Convert rows to list of items (ignoring NaN/empty cells)
transactions = df.drop('Item(s)', axis=1).values.tolist()
transactions = [[item for item in transaction if pd.notna(item)] for transaction in transactions]
```

```python
[21]: # Step 3: Encode the transaction data
te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_array, columns=te.columns_)
```

```python
[22]: # Step 4: Apply Apriori to find frequent itemsets
frequent_itemsets = apriori(df_encoded, min_support=0.03, use_colnames=True)
```

```python
[23]: # Step 5: Generate association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
```

```python
[24]: # Step 6: Output top rules
print(rules.sort_values(by="lift", ascending=False).head(10))
```

```
            antecedents          consequents  antecedent support  \
6    (other vegetables)    (root vegetables)            0.193493
7     (root vegetables)   (other vegetables)            0.108998
19            (sausage)          (rolls/buns)            0.093950
18         (rolls/buns)            (sausage)            0.183935
9    (other vegetables)      (tropical fruit)            0.193493
8      (tropical fruit)   (other vegetables)            0.104931
31  (whipped/sour cream)         (whole milk)            0.071683
30         (whole milk)  (whipped/sour cream)            0.255516
26         (whole milk)    (root vegetables)            0.255516
27    (root vegetables)         (whole milk)            0.108998

    consequent support   support  confidence      lift  representativity  \
6             0.108998  0.047382    0.244877  2.246605               1.0
7             0.193493  0.047382    0.434701  2.246605               1.0
19            0.183935  0.030605    0.325758  1.771048               1.0
18            0.093950  0.030605    0.166390  1.771048               1.0
9             0.104931  0.035892    0.185497  1.767790               1.0
8             0.193493  0.035892    0.342054  1.767790               1.0
31            0.255516  0.032232    0.449645  1.759754               1.0
30            0.071683  0.032232    0.126144  1.759754               1.0
26            0.108998  0.048907    0.191405  1.756031               1.0
27            0.255516  0.048907    0.448694  1.756031               1.0

    leverage  conviction  zhangs_metric   jaccard  certainty  kulczynski
6   0.026291    1.179941       0.688008  0.185731   0.152500    0.339789
7   0.026291    1.426693       0.622764  0.185731   0.299078    0.339789
19  0.013324    1.210344       0.480506  0.123766   0.173788    0.246074
18  0.013324    1.086899       0.533490  0.123766   0.079952    0.246074
9   0.015589    1.098913       0.538522  0.136716   0.090010    0.263775
8   0.015589    1.225796       0.485239  0.136716   0.184204    0.263775
31  0.013916    1.352735       0.465077  0.109273   0.260757    0.287895
30  0.013916    1.062323       0.579917  0.109273   0.058667    0.287895
26  0.021056    1.101913       0.578298  0.154961   0.092487    0.320049
27  0.021056    1.350401       0.483202  0.154961   0.259479    0.320049
```

Github :- https://github.com/Pranav-Divekar/Machine-learning-

Conclusion:

This assignment demonstrated the application of the Apriori algorithm to uncover association rules from a transactional dataset. The process involved several key steps, including preprocessing the transaction data, generating frequent itemsets, and extracting meaningful rules based on metrics such as support, confidence, and lift. Through this approach, the Apriori algorithm effectively revealed patterns in customer purchase behavior. These insights can be valuable for making informed decisions in areas like marketing strategies and product placement to enhance customer experience and drive sales.