DMBI Experiment 9 – Association Rule Mining using Python

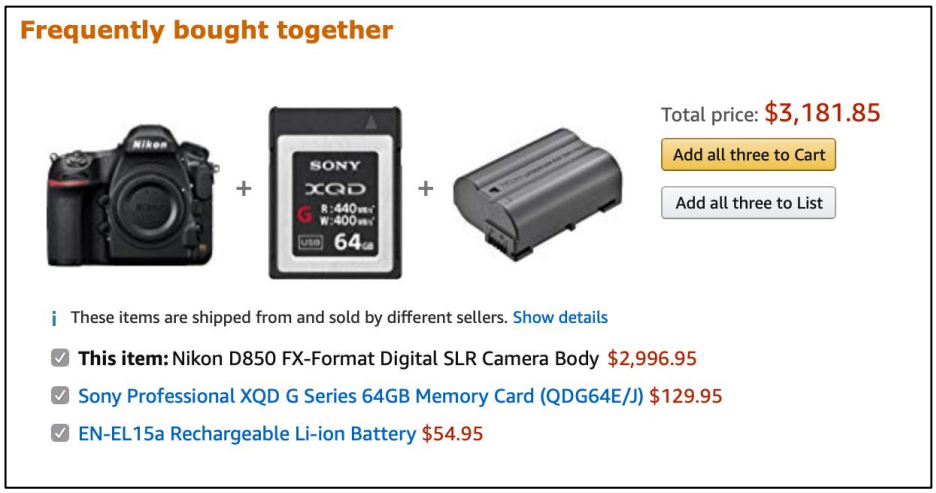
Name: Atharva Lotankar
Class: D15C; **Rnum:** 27
Batch B

Aim: To implement Association Rule Mining (Apriori Algorithm) using Python.

Theory:

The Apriori algorithm is a key tool in market basket analysis, used to identify frequent itemsets and generate association rules that reveal products often purchased together. This insight enables retailers to optimize product placement, promotions, and cross-selling strategies by understanding customer buying patterns effectively.

Python offers robust support for implementing Apriori through libraries such as mlxtend. It streamlines the process of data preprocessing, frequent itemset generation, and rule extraction, allowing for efficient analysis of large transaction datasets. Python's flexibility and rich ecosystem make it ideal for customizing and automating market basket analysis workflows.



A real-world case study is Amazon's "Frequently Bought Together" feature, which leverages association rules derived from Apriori to recommend complementary products. This application significantly boosts customer experience and sales by suggesting items that previous customers commonly bought together, showcasing the practical impact of Apriori-driven market basket analysis in e-commerce.

Algorithm / Code Snippet:

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

# Load dataset
df = pd.read_csv('cosmetics_fraud_2025.csv')

# Combine Product_SKU and Product_Category for each item in each transaction
df['Item'] = df['Product_SKU'] + ' | ' + df['Product_Category']

# Select relevant columns for market basket: TransactionID and combined Item
basket_data = df[['Transaction_ID', 'Item']]

# Create list of transactions - group combined items by TransactionID
transactions = basket_data.groupby('Transaction_ID')['Item'].apply(list).tolist()

# Encode transactions to one-hot format
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)

# Generate frequent itemsets with minimum support of 0.01
frequent_itemsets = apriori(df_encoded, min_support=0.01, use_colnames=True)

# Filter for single itemsets (length 1)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
single_itemsets = frequent_itemsets[frequent_itemsets['length'] == 1]

# Display frequent single itemsets with support
print(single_itemsets[['itemsets', 'support']])
```

Code Implementation:

- **Import libraries**: pandas for data handling, TransactionEncoder for one-hot encoding transactions, and apriori from mlxtend for association mining.
- **Load dataset**: read the CSV file containing transactional data with product SKU and category.
- **Combine product attributes**: create a combined "Item" column by joining Product_SKU and Product_Category to analyze them together.
- **Group transactions**: aggregate items bought together in each transaction into a list, forming a list of transactions.
- **One-hot encode**: transform transactions into a boolean matrix indicating whether each item is present or not, required by the Apriori algorithm.
- **Apply Apriori algorithm**: extract frequent itemsets with a minimum support threshold (e.g., 0.01), meaning itemsets that appear in at least 1% of transactions.
- **Filter itemsets**: select only single itemsets to analyze individual products' frequency in transactions.
- **Print results**: display the frequent single itemsets and their support values for insight into popular items and categories.

This workflow mimics market basket analysis to identify commonly purchased items and associations.

Inference:

- The most frequent items include specific product SKUs combined with their categories, showing which individual products and product types dominate sales.
- These frequent single itemsets indicate customer preferences and can guide stock management, marketing focus, and targeted promotions.
- The results provide foundational insight into product popularity, which can be further expanded into more complex association rules to identify product bundles or cross-sell opportunities.

Result:

	itemsets	support
0	(AURORA-LIP-01 | Blush)	0.012189
1	(AURORA-LIP-01 | Eyeshadow Palette)	0.010783
2	(AURORA-LIP-01 | Highlighter)	0.013127
3	(AURORA-LIP-01 | Lipstick)	0.010314
4	(AURORA-LIP-01 | Mascara)	0.011252
5	(AURORA-LIP-01 | Serum)	0.015471
6	(AURORA-LIP-01 | Setting Spray)	0.010314
7	(CELESTE-EYE-05 | Lipstick)	0.010314
8	(CELESTE-EYE-05 | Mascara)	0.011252
9	(CELESTE-EYE-05 | Serum)	0.010314
10	(COSMIC-HIGHLIGHT-06 | Mascara)	0.012189

Conclusion:

The experiment successfully demonstrated the application of the Apriori algorithm for market basket analysis using Python. By preprocessing transactional data to combine product SKU and category, frequent itemsets representing popular products were identified. This approach provides valuable insights into customer purchasing behavior, enabling targeted marketing and inventory optimization. The experiment validates Apriori's effectiveness in uncovering actionable product associations from large datasets.