**NAME:** Tejal Desai  
**NJIT UCID:** td344  
**Email Address:** td344@njit.edu  
**Date:** 08/10/2024  
**Professor:** Yasser Abduallah  
**CS 634 Data Mining**  
 **Midterm Project Report**

**Implementation and Code Usage: Apriori Algorithm in Retail Data Mining**

**Abstract**

In this project, I investigate the Apriori Algorithm, a fundamental data mining technique for discovering associations within retail transactions. By implementing the algorithm and applying essential data mining principles, I evaluate its effectiveness in uncovering frequent itemsets and producing actionable association rules. This report details the design and development of a customized tool aimed at extracting valuable insights from retail transaction data.

**Introduction**

Data mining involves revealing hidden patterns and relationships within large datasets. In the retail sector, understanding these associations can inform business strategies. The Apriori Algorithm is a well-established technique for mining frequent itemsets and generating association rules. This project investigates its application in retail, with a focus on the Nike dataset. We apply essential data mining principles, including support, confidence, and the generation of frequent itemsets.

**Core Concepts and Principles:**

* **Frequent Itemset Discovery:** The Apriori Algorithm identifies groups of items that frequently appear together in transactions.
* **Support and Confidence:**
  + Support measures how often an item or itemset occurs within the dataset.
  + Confidence reflects the likelihood of items being purchased together.
* **Association Rules:** Strong rules are generated from frequent itemsets, indicating the probability of co-purchases.

**Project Workflow**

**Data Loading and Preprocessing:**

**The transaction data is imported from a CSV file and undergoes cleaning and preprocessing to ensure that items are unique and properly sorted.**

**Setting Minimum Support and Confidence:**

**User-defined thresholds for minimum support and confidence are established to filter out less significant patterns.**

**Generating Frequent Itemsets:**

**The Apriori Algorithm processes item combinations, incrementally increasing the size of the itemsets (K = 1, 2, 3, etc.), and filters out frequent itemsets based on the defined support threshold.**

**Generating Association Rules:**

**Association rules are constructed from the frequent itemsets and assessed using confidence metrics.**

**Implementation**

This project utilizes Python's **mlxtend** library to implement both the Apriori and FP-Growth algorithms, along with a brute-force approach for comparative analysis. The dataset consists of transactions related to Nike products, featuring items such as "Running Shoes" and "Sweatshirts."

* **Brute Force Implementation:** This method calculates all possible combinations of items, retaining those that meet the specified minimum support threshold.
* **Apriori Algorithm:** This algorithm efficiently generates frequent itemsets by minimizing the number of combinations to be evaluated.
* **FP-Growth Algorithm:** Another effective technique for frequent itemset mining, employed here for comparison purposes.

**Results and Evaluation**

The project assesses the performance of each algorithm by comparing their execution times and outcomes:

* **Brute Force:** Provides accurate results but tends to be slow when dealing with large datasets.
* **Apriori:** Offers a good balance of efficiency, speed, and accuracy.
* **FP-Growth:** Comparable to Apriori in functionality but demonstrates faster execution times for larger datasets.

| **Algorithm** | **Time (seconds)** | **Frequent Itemsets** | **Association Rules** |
| --- | --- | --- | --- |
| Brute Force | 0.75 | 12 | 6 |
| Apriori | 0.30 | 12 | 6 |
| FP-Growth | 0.25 | 12 | 6 |

**Conclusion**

This project showcases the practical use of the Apriori Algorithm in retail data mining. By uncovering strong associations within Nike transactions, it highlights the significance of frequent itemset mining for informed retail decision-making. The findings indicate that both the Apriori and FP-Growth algorithms are more efficient than the brute force method while still delivering accurate results.

**Screenshots**

Screenshots of the code and outputs have been included to demonstrate functionality. The source code, datasets, and visualizations are attached to this report for reference.

Figure=amazon\_items,bestbuy\_items,kmart\_items

A computer screen with text

Description automatically generated

Below are the screenshots of the code from the python file:

A computer screen with text

Description automatically generated

The list of transactions for three stores—Amazon,Bestbuy and kmart—is generated by the script, with randomly selected items for each store. 20 transactions are created per store, and each transaction contains 2 to 6 items. The transactions are then saved into a CSV file named store\_transactions.csv. The code effectively produces a DataFrame with transactions and saves the outcome in the CSV.

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

This code generates a series of predetermined transactions for three different stores: Best buy ,Amazon, and kmart. Each store has its own specific inventory of items, and the transactions are stored in separate CSV files based on the store and database number. For example, the transactions for amazon are stored as transactions\_amazon.csv, and so forth.

The main steps involved in this process are:

- Defining the list of items for each store.

- Predefining sets of transactions for each store.

- Creating a function to store each store's transactions in a CSV file.

- Storing the transactions for Best buy ,Amazon, and kmart.

Creating the dataitems and creating 20 transactions.

A screen shot of a computer program

Description automatically generated

The performance of three algorithms—Brute Force, Apriori, and FP-Growth—is compared using transaction datasets from Best buy ,Amazon, and kmart. Each algorithm is assessed for frequent itemset mining and association rule generation based on user-defined minimum support and confidence thresholds.

The key steps are as follows:

- Using predefined transaction data from Best buy ,Amazon, and kmart.

- Taking minimum support and confidence levels from the user to determine frequent itemsets and association rules.

- Implementing the Brute Force Algorithm, which checks all combinations of items to find frequent itemsets.

- Applying the Apriori & FP-Growth algorithms on the transaction data for standard frequent itemset mining.

- Printing the results for each algorithm, including frequent itemsets, association rules, and execution times, for comparison.

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated

A black screen with many small colored lines

Description automatically generated with medium confidence

Kmart

A black rectangular object with white lines

Description automatically generated

A computer screen shot of a program

Description automatically generated

Best buy  
A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Other

**GitHub Repository:**

The complete code and dataset are available on my GitHub repository:

https://github.com/Tejal05131998/Data-mining-\_midterm

Colab link: https://colab.research.google.com/drive/1h7bAd3jsyVH0ZXZjOrrMR6T85FCW\_NOf#scrollTo=ziIzyV819HHd