Name – Mahesh Chandra Chowdary Bodepudi

NJIT UCID – mb2327

Email Address – [mb2327@njit.edu](mailto:mb2327@njit.edu)

Professor – Yasser Abduallah

Course – CS634 Data Mining

**CS 634 – Data Mining Project Report**

**Abstract**

Finding patterns and connections in big datasets can be achieved through data mining. It aids in identifying products that are frequently bought together and in understanding consumer purchasing behavior in retail.  
Three algorithms—Brute Force, Apriori, and FP-Growth—are used in this project to generate association rules and mine common itemsets. Comparing these algorithms' effectiveness in identifying frequently occurring itemsets and producing association rules is the primary objective.  
Transactional datasets from several retail establishments, including Amazon, Walmart, Target, BestBuy, and CVS, were used for the analysis. Support and confidence values that were specified by the user were used to test each method. The comparison is based on how many itemsets each algorithm identified and how long it took.

**Introduction**

Large data sets can have valuable information extracted from them with the use of data mining. It is often used in retail to identify things that customers regularly buy together and to research their purchasing trends.  
In order to identify frequently occurring itemsets and produce association rules from transaction records, this research employs three data mining algorithms: Brute Force, Apriori, and FP-Growth. The speed and scalability of these algorithms are compared to see which one works best.  
Transaction information from various retail establishments is included in the databases used. The Jupyter Notebook is used to conduct the analysis in Python. The user provides the minimum support and confidence levels, and the same values are used for comparison across all three algorithms.

**Core Concepts and Principles**

In order to identify products that frequently appear together in transaction data and to characterize the relationships between them in a way that is helpful for retail decisions, frequent itemset mining and association rule generation are employed. This project compares how three algorithms, with varying speeds and scalability, find common itemsets and generate rules using the same dataset and user-selected thresholds.   
  
**Brute Force:** This approach methodically compares the dataset with every conceivable item combination. Although it is easy to comprehend and ensures comprehensive coverage, its runtime increases rapidly with the number of items, making it only useful for small datasets or as a baseline for verification.

**Apriori:** By eliminating infrequent items and itemsets beforehand and then constructing larger itemsets from smaller, more frequent ones, this algorithm outperforms brute force. It eliminates pointless checks by gradually shrinking the search space, and on moderately sized data, it usually performs quicker than brute force.   
  
**FP-Growth:** By using this method, candidate itemsets are not generated. The dataset is compressed into an FP-tree, and patterns are extracted straight from that structure. For large or dense datasets, it is typically the most effective and scalable of the three since it minimizes the number of repeated scans and concentrates on compact tree routes.

**Implementation Overview**

The solution is built up as a straightforward, repeatable pipeline that compares the timing and outputs of the three algorithms after running the same dataset through them all with the same inputs. The code loads the CSV file for a chosen store, turns each transaction into a list of items, requests the user's minimum level of support and confidence, then sequentially executes Brute Force, Apriori, and FP-Growth, creates association rules using the itemsets it finds, and logs the execution time and the number of frequently found itemsets for comparison.

**Script Used for creation of Datasets:**

A total of 5 datasets are created for the project and the datasets are made deterministic and not random as per the instructions. This is the working script used for the creation of the datasets.

To avoid randomness and create consistent, reproducible transactions, I followed a structured pattern based on transaction number. Each dataset contains **30 transactions**, where specific item groupings repeat in a logical sequence.  
For example:

* Transactions **1, 6, 11, 16, 21, 26** → Items 1–5
* Transactions **2, 7, 12, 17, 22, 27** → Items 2–6
* Transactions **3, 8, 13, 18, 23, 28** → Items 3–7
* Transactions **4, 9, 14, 19, 24, 29** → Items 4–8
* Transactions **5, 10, 15, 20, 25, 30**→ Items 6–10

This deterministic structure ensures that each dataset always produces the same set of transactions and captures realistic co-purchase behavior.

A screenshot of a computer program

AI-generated content may be incorrect.

**Selection and loading of datasets:** The application starts by allowing the user to select a retail dataset (e.g., home improvement, electronics, groceries, clothes, or books). After that, it reads the relevant CSV file and converts the transactions into mining-ready string lists. The report procedure, in which the user chooses a store, loads data, and then moves on to analysis, is mirrored here.

A screen shot of a computer

AI-generated content may be incorrect.  
  
**inputs from users:** Numerical inputs are used for minimum support and minimum confidence. To provide for a fair comparison, these thresholds, which determine which itemsets are retained and which rules are generated, are maintained constant between algorithms.

A screenshot of a computer program

AI-generated content may be incorrect.

**Algorithm Execution:** The code executes the three methods in the same session following loading and preprocessing. Brute Force lists potential itemsets and sorts them according to support; While FP-Growth creates an FP-tree and mines patterns without candidate generation, Apriori generates larger itemsets from frequent smaller ones with pruning. The comparison is simple when they are run consecutively on the same data.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Creation of association rules:** The end output contains both the frequent itemsets and interpretable rules for retail co-purchases since the code generates association rules that pass the user's confidence threshold after frequent itemsets are received.

**Performance Measurement:** The software counts the number of frequent itemsets that fulfill the thresholds and records the execution time for each algorithm. Efficiency and scalability are later compared using these metrics.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Reproducibility and the environment:** The Jupyter Notebook is used to run the Python-written implementation. It uses standard libraries (such pandas and, if relevant, mlxtend for Apriori/FP-Growth baselines) and has a predictable workflow: open notebook, install dependencies, choose dataset, establish thresholds, execute algorithms, and examine results.

**Results and Evaluation**

All three algorithms were able to find significant frequent itemsets and association rules when tested on the identical retail transaction datasets using standard support and confidence standards. Because it checks every conceivable combination, the Brute Force approach took the longest and delivered accurate results, making it less useful for huge datasets. The Apriori approach slowed slower as the number of items increased and still required several scans of the dataset, but it performed better by removing infrequent itemsets early and minimizing needless checks. By employing a compact tree structure to mine patterns without creating candidate sets, the FP-Growth method demonstrated the highest overall performance, finishing the operation more quickly and managing larger datasets more effectively. FP-Growth was the most successful of the three in terms of execution time and scalability.

**Conclusion**

Using transactional retail datasets, this study examined the Brute Force, Apriori, and FP-Growth algorithms for frequent itemset mining and association rule building. Though their effectiveness and scalability differed, all three approaches were able to find significant item connections. Although Brute Force produced comprehensive findings, its lengthy processing time made it unsuitable for bigger datasets. By using trimming techniques, Apriori enhanced efficiency, but it still needed to repeatedly scan the data. By employing a tree-based methodology, FP-Growth produced the greatest results and was able to handle larger datasets more quickly. For real-world data mining applications, FP-Growth was found to be the most effective and feasible method overall, particularly when handling substantial amounts of transaction data.

**Steps to Run the project**

Using Jupyter Notebook, the implementation was done in Python. To execute the project and view the outcomes of all three algorithms, the following procedures were taken:

To set up the workspace, the Jupyter Notebook was installed and opened. To handle datasets and support the algorithm logic, the necessary libraries were loaded, including pandas and itertools. Other libraries, such as mlxtend, may occasionally be installed in order to verify the results of FP-Growth and Apriori. The user was asked to choose a dataset that represented a retail store once the setting was prepared. Every dataset was transformed into a list of transactions for processing after being loaded from a CSV file.

The user was prompted to provide the minimal support and confidence values after the dataset had been loaded. To guarantee a fair comparison, these thresholds were used consistently across all algorithms. The software then generated frequent itemsets and association rules by progressively executing the Brute Force, Apriori, and FP-Growth procedures. To assess performance, the number of itemsets and execution time for each algorithm were noted. In order to compare the three methods in a single run, the output finally showed the patterns that were found together with their association rules.

**GitHub Repository**

The source code and datasets used in the project is available in this GitHub repository.

<https://github.com/maheshb2327/Bodepudi_Mahesh_MidtermProject/tree/main/Bodepudi_Mahesh_MidtermProject>