CS634 – Data Mining Midterm Project Report

Student Name: Taymar Walters
Email: tw237@njit.edu
Course: CS634 - Data Mining
Instructor: Dr. Yasser Abduallah

1. Introduction

The purpose of this project is to explore how frequent itemsets and association rules can be discovered from transactional data using three methods:

1. A Brute Force algorithm built entirely from scratch in Python,
2. The Apriori algorithm implemented via the mlxtend library, and
3. The FP-Growth algorithm, also from mlxtend.

The project compares each approach in terms of correctness, execution time, and scalability. Each method was applied to five different transactional datasets, each representing a real-world scenario such as retail shopping or product associations. The report follows a tutorial style, providing step-by-step explanations so that readers can easily reproduce the results.

2. How to run the code

Tools Used:

- Python: 3.11
- Environment: VS Code / PowerShell
- Libraries: pandas, mlxtend, and tabulate

Open this file in Visual Studio Code and run the code through the powershell terminal

3. Project Layout (Actual Setup)

walters_taymar_midtermproj/
 │
 ├── .git/
 ├── associationruleminer.py
 ├── associationruleminer_notebook.ipynb
 ├── generic_items.csv
 ├── generic_transactions. csv
 ├── nike_products. csv
 ├── nike_product_transactions. csv
 ├── bestbuy_products. csv
 ├── bestbuy_transactions. csv
 ├── coffee_items. csv
 ├── coffee_transactions. csv
 ├── k-mart_items. csv
 ├── k-mart_transactions.csv

Each dataset pair represents product listings (*_items.csv) and transactions (*_transactions.csv). The main script runs all algorithms with user-specified parameters and displays them on the console.

4. Dataset Creation

Each transaction dataset contains exactly 20 transactions that are deterministic which means no random generation.

Examples include:
 - Generic: Letters A-F
 - Nike Products: athletic wear and accessories
 - BestBuy: consumer electronics
 - Coffee Items: café menu products
 - K-Mart: mixed retail inventory

*The Coffee Items dataset was founded on Kaggle and every other dataset was provided through the file: Midterm_Project_Items_Datasets_Examples.pdf

5. Algorithm Explanations

Brute Force: Enumerates all combinations and counts occurrences to determine support. It's slow but guarantees correctness.
Apriori: Improves efficiency by pruning infrequent itemsets. Uses a bottom-up search to discover frequent sets.
FP-Growth: Avoids candidate generation by compressing data into an FP-tree and directly mining frequent patterns.

7. Results

Example outputs from generic_transactions.csv with min_support = 0.3 and min_confidence = 0.6:

```
================================================
🔹 FREQUENT ITEMS FOUND BY BRUTE FORCE:
================================================
        ('A',) | support: 1.00
        ('B',) | support: 0.40
        ('C',) | support: 0.60
        ('D',) | support: 0.45
        ('E',) | support: 0.70
        ('A', 'B') | support: 0.40
        ('A', 'C') | support: 0.60
        ('A', 'D') | support: 0.45
        ('A', 'E') | support: 0.70
        ('C', 'D') | support: 0.30
        ('C', 'E') | support: 0.35
        ('A', 'C', 'D') | support: 0.30
        ('A', 'C', 'E') | support: 0.35

================================================
🔸 ASSOCIATION RULES — BRUTE FORCE
================================================

        ('B',) → ('A',) (support: 0.40, confidence: 1.00)
        ('A',) → ('C',) (support: 0.60, confidence: 0.60)
        ('C',) → ('A',) (support: 0.60, confidence: 1.00)
        ('D',) → ('A',) (support: 0.45, confidence: 1.00)
        ('A',) → ('E',) (support: 0.70, confidence: 0.70)
        ('E',) → ('A',) (support: 0.70, confidence: 1.00)
        ('D',) → ('C',) (support: 0.30, confidence: 0.67)
        ('D',) → ('C', 'A') (support: 0.30, confidence: 0.67)
        ('A', 'D') → ('C',) (support: 0.30, confidence: 0.67)
        ('C', 'D') → ('A',) (support: 0.30, confidence: 1.00)
        ('C', 'E') → ('A',) (support: 0.35, confidence: 1.00)
```

8. Reproducibility

All datasets are deterministic and identical values yield identical results across all algorithms.

10. Key Takeaways

- Building Brute Force helps understand the fundamentals.
- Apriori introduces pruning efficiency.
- FP-Growth is the most efficient for larger datasets.

11. Links

Github Repository: https://github.com/tw237njit/walters_taymar_midtermproj.git