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 different methods:

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

The python script being demonstrated runs all algorithms with user-specified parameters and displays them on the console. This project compares each of the three approaches that are 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 or higher
Environment: VS Code / PowerShell
Libraries: pandas, mlxtend, and time

IMPORTANT:
Ensure that the python file is saved in the same folder as the dataset files (.csv) or it won't work.

Run in VS Code
Open the python file in Visual Studio Code and run the code through the powershell terminal.

Run through Command Prompt
1) Open Command Prompt (press Win + R, type cmd, hit Enter), then type: *python --version*
2) If you see something like *Python 3.11* or higher, you're good. Otherwise, install Python from https://www.python.org/downloads/ and during installation, check the box that says:
Add Python to PATH
3) Use the cd command to go to the folder where you saved the file.
4) Once you're inside the correct folder, just run: *python lastname_firstname_midterm.py*

3. Project Layout (Actual Setup)

walters_taymar_midtermproj/
 │
 ├── .git/
 ├── walters_taymar_midterm.py
 ├── walters_taymar_midterm_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
 ├── walters_taymar_midterm_report.pdf
 ├── k-mart_items. csv
 ├── k-mart_transactions.csv

4. Dataset Creation

There are five pairs of datasets. Each dataset pair represents product listings (*_*items.csv* or *_*products.csv*) and transactions (*_*transactions*.csv).Each product listing has at least 5 unique items while each transaction dataset contains exactly 20 transactions that are deterministic which means no random generation.

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

Note: The Coffee Items dataset was founded on Kaggle.com 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.

6. Results

Example outputs from ***generic_transactions.csv*** with *minimum_support* = 0.3 and *minimum_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)
```

7. Reproducibility

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

9. Key Takeaways

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

10. Links

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

➔ **Coffee Dataset from Kaggle:**
https://www.kaggle.com/datasets/ayeshasiddiqa123/coffee-dataset