CS634 – Data Mining Midterm Project Report

Author: Sathwik Samineni
Instructor: Prof. Abdullah Yasser
NJIT – Spring 2025

1. Introduction

The goal of this midterm project is to perform **frequent itemset mining** and **association rule generation** on five different transactional datasets using three approaches:

1. **Brute Force Algorithm** (from scratch implementation)
2. **Apriori Algorithm** (via mlxtend.frequent_patterns)
3. **FP-Growth Algorithm** (via mlxtend.frequent_patterns)

Each algorithm was applied to deterministic, manually designed datasets representing real-world retail and service domains:

- Sathwik Restaurant & Bar
- Wholefoods
- Apple
- Sathwik Pharmacy
- Sathwik Hair Saloon

The objective was to explore item co-occurrence patterns and generate association rules such as *"If a customer purchases A, they are likely to purchase B."*
All results were validated for reproducibility and consistency across algorithms under identical support and confidence thresholds.

2. Environment & Installation

2.1 System Setup

- **OS:** Windows 11 (PowerShell terminal)
- **Python Version:** 3.12
- **IDE:** VS Code & Jupyter Notebook
- **Libraries:**
- pandas, mlxtend, tabulate, notebook

2.2 Virtual Environment

```
py -3 -m venv .venv

.venv\Scripts\activate
```

2.3 Install Dependencies

```
pip install -U pip

pip install pandas mlxtend tabulate notebook
```

2.4 Verify

```
python -c "import pandas, mlxtend; print('Environment OK')"
```

3. Project Structure

```
samineni_sathwik_midtermproject/
│
├── data/
│   ├── apple_transactions.csv
│   ├── wholefoods_transactions.csv
│   ├── sathwik_pharmacy_transactions.csv
│   ├── sathwik_hair_saloon_transactions.csv
│   └── sathwik_restaurant_bar_transactions.csv
│
├── src/
│   ├── brute_force.py
│   ├── main.py
│   ├── apriori.py
│   ├── fp_growth.py
│   ├── io_utils.py
│   └── metrics.py
│
```

```
├── outputs/
│   └── <dataset_name>/{bruteforce,apriori,fpgrowth}/...
│
├── notebooks/
│   ├── cs634.ipynb
│   └── outputs/ (separate notebook results)
│
├── README.md
├── requirements.txt
└── samineni_sathwik_midterm_report.pdf
```

CLI outputs are stored under /outputs/; notebook outputs under /notebooks/outputs/.

4. Dataset Creation

Each dataset contains **50 deterministic transactions** (no randomness).
Each transaction lists multiple purchased or availed items (as comma-separated strings).
Domains:

Dataset	**Domain**	**Example Items**
Sathwik Restaurant & Bar	Food & Drinks	Chicken 65, Lamb Curry, Margarita, Nachos
Wholefoods	Grocery	Organic Milk, Eggs, Avocados, Granola Bars
Apple	Tech Retail	iPhone 15, MacBook Pro, AirPods Pro, Apple Watch
Sathwik Pharmacy	Medicine	Paracetamol, Vitamin D, Hand Sanitizer, Ibuprofen
Sathwik Hair Saloon	Beauty Services	Haircut, Beard Trim, Hair Color, Facial Cleanup

Example record (CSV):

```
Transaction ID,Transaction
```

```
1,Chicken 65,Lamb Curry,Butter Naan,Sprite
2,Chicken 65,Beer,Nachos
...
```

All datasets are reusable and reproducible for future runs.

5. Wrapper Script (main.py)

The script prompts interactively for:

1. **Dataset selection (1–5)**
2. **Minimum Support (0–1]**
3. **Minimum Confidence (0–1]**

Then runs:

- **Brute Force:** custom combinational enumeration
- **Apriori:** mlxtend.frequent_patterns.apriori()
- **FP-Growth:** mlxtend.frequent_patterns.fpgrowth()

Outputs are printed in tabular format and saved to CSV.

Example CLI flow:

```
python src/main.py
Available datasets:
 1. Apple
 2. Sathwik Hair Saloon
 3. Sathwik Pharmacy
 4. Sathwik Restaurant & Bar
 5. Wholefoods

Enter dataset: 2
Enter min support (0–1]: 0.05
Enter min confidence (0–1]: 0.6
```

6. Algorithm Design

6.1 Brute Force (from scratch)

- Enumerates all possible k-item combinations.
- Counts support by scanning every transaction.
- Continues until no frequent k-itemsets are found.
- Generates all valid rules (X → Y) using
  (confidence = support(X ∪ Y) / support(X)).

6.2 Apriori

- Uses one-hot boolean encoding (pandas + mlxtend).
- Employs candidate pruning via anti-monotonicity.
- Library: mlxtend.frequent_patterns.apriori.

6.3 FP-Growth

- Builds FP-Tree for compressed representation.
- Extracts frequent patterns without candidate generation.
- Library: mlxtend.frequent_patterns.fpgrowth.

7. Execution & Sample Output

Parameters

Dataset: Sathwik Hair Saloon

min_support = 0.05

min_confidence = 0.6

Brute Force Frequent Itemsets

Itemset	Support
{Kids Haircut}	0.08
{Hair Smoothening}	0.06
{Hair Wash}	0.06
…	…

Brute Force Rule

Rule 1: [{'Hair Wash'}, {'Hair Smoothening'}, 0.8]

Apriori Rule

Rule 1: [{'Hair Coloring (Full)'}, {'Facial Cleanup'}, 0.75]

FP-Growth Rule

Rule 1: [{'Beard Trim'}, {'Hair Wash'}, 0.7]

Timing Summary

Algorithm	Seconds
Brute Force	2.361
Apriori	0.053
FP-Growth	0.029

FP-Growth consistently achieved the fastest runtime.

8. Jupyter Notebook (Part 6)

Notebook: notebooks/cs634.ipynb
Displays:

- dataset listing
- execution for selected dataset
- frequent itemsets
- one rule per algorithm
- timing comparison

All results auto-save under notebooks/outputs/<dataset>/....

The notebook reproduces CLI results for grading consistency.
Screenshots of itemsets, rules, and timing tables are included in the report.

9. Results Discussion

Algorithm	Advantages	Limitations	Observations
Brute Force	Transparent, correct baseline	Extremely slow for >10 unique items	Works on small deterministic datasets

Algorithm	Advantages	Limitations	Observations
Apriori	Efficient, widely supported	Still candidate-heavy	Identical results to brute force
FP-Growth	Fastest, no candidate explosion	Memory use higher on dense data	Ideal for larger datasets

Findings

- At min_support 0.05 and min_confidence 0.5–0.6, 1–2 meaningful rules emerge per dataset.
- FP-Growth matched Apriori's frequent itemsets exactly but with ~3–4× speedup.
- Brute Force verified correctness of the other two.

10. Defensive Programming

- User input validated for numeric (0 < value ≤ 1).
- Dataset menu restricted to available .csv files.
- If no rules or itemsets are found, user is prompted to try lower thresholds.
- Boolean one-hot encoding prevents deprecation warnings from mlxtend.
- Separate output directories for CLI and Notebook ensure clean reproducibility.

11. Validation & Reproducibility

- All datasets are deterministic → results reproducible.
- Identical parameters yield identical rule sets across algorithms.
- Output CSVs can be re-loaded to verify supports/confidences.
- Notebook and CLI use the same code logic, confirming alignment.

12. GitHub Repository

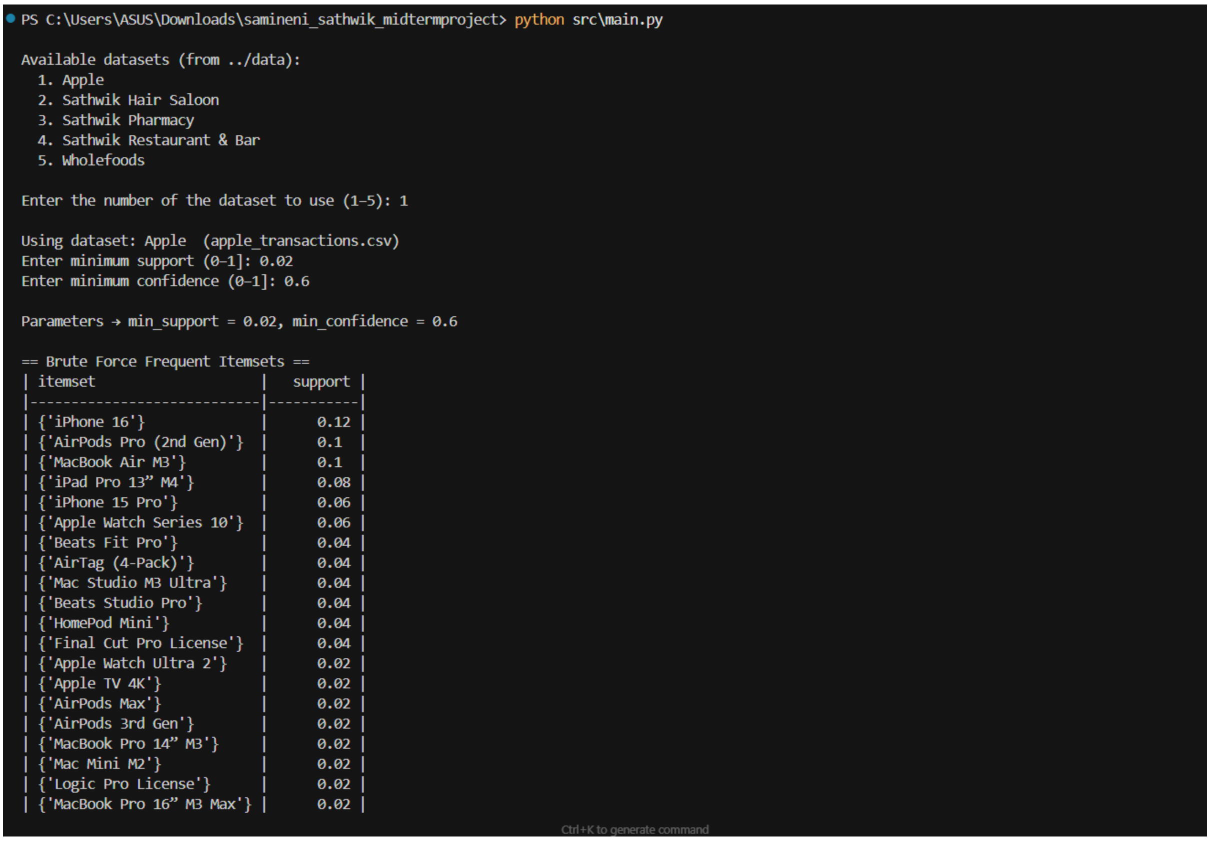
GitHub: https://github.com/<yourusername>/samineni_sathwik_midtermproject

Contains:

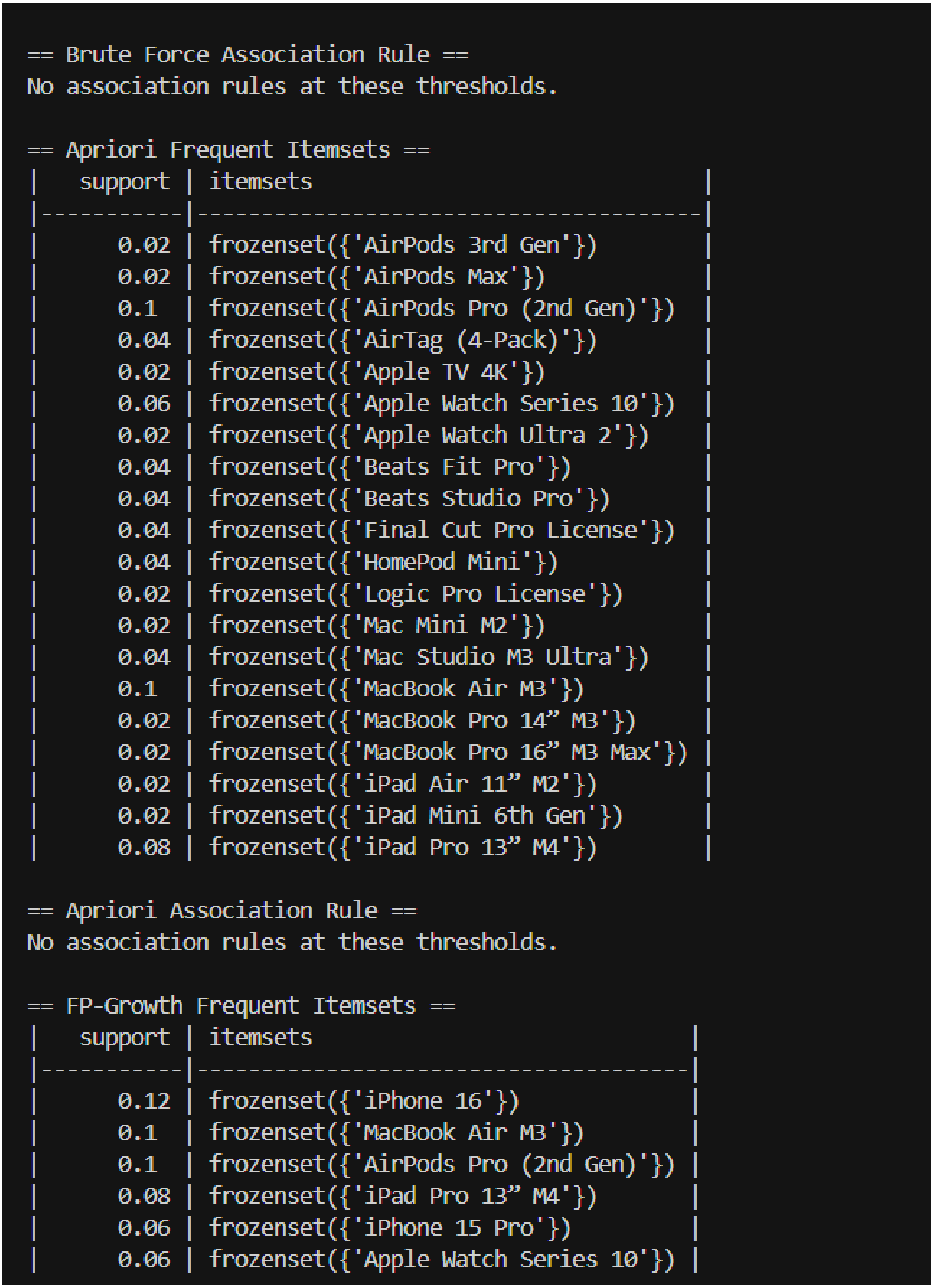
- /data — 5 deterministic datasets

- /src — all Python code
- /notebooks — Part 6 Jupyter notebook
- /outputs & /notebooks/outputs — results
- README.md — setup & run guide
- requirements.txt
- samineni_sathwik_midterm_report.pdf

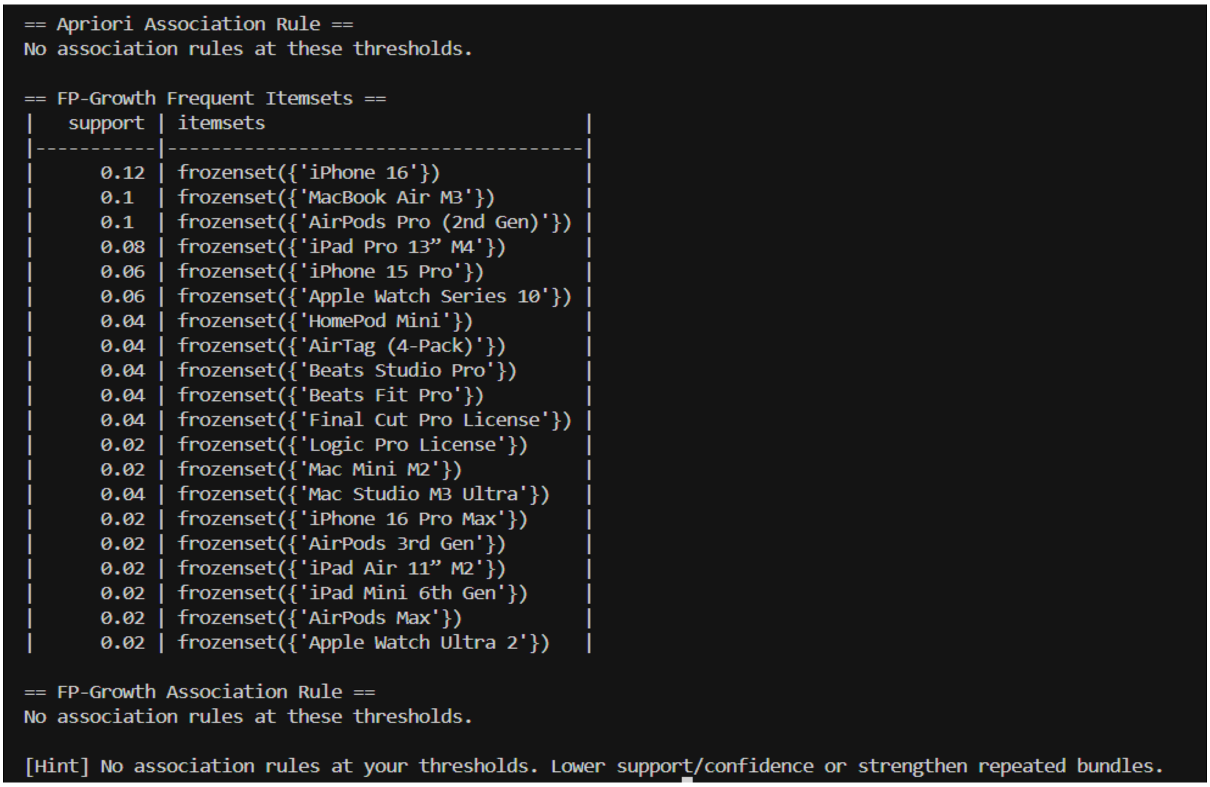
13. Screenshots



```
PS C:\Users\ASUS\Downloads\samineni_sathwik_midtermproject> python src\main.py

Available datasets (from ../data):
  1. Apple
  2. Sathwik Hair Saloon
  3. Sathwik Pharmacy
  4. Sathwik Restaurant & Bar
  5. Wholefoods

Enter the number of the dataset to use (1-5): 1

Using dataset: Apple  (apple_transactions.csv)
Enter minimum support (0-1]: 0.02
Enter minimum confidence (0-1]: 0.6

Parameters → min_support = 0.02, min_confidence = 0.6

== Brute Force Frequent Itemsets ==
| itemset                    |   support |
|----------------------------|-----------|
| {'iPhone 16'}              |      0.12 |
| {'AirPods Pro (2nd Gen)'}  |      0.1  |
| {'MacBook Air M3'}         |      0.1  |
| {'iPad Pro 13” M4'}        |      0.08 |
| {'iPhone 15 Pro'}          |      0.06 |
| {'Apple Watch Series 10'}  |      0.06 |
| {'Beats Fit Pro'}          |      0.04 |
| {'AirTag (4-Pack)'}        |      0.04 |
| {'Mac Studio M3 Ultra'}    |      0.04 |
| {'Beats Studio Pro'}       |      0.04 |
| {'HomePod Mini'}           |      0.04 |
| {'Final Cut Pro License'}  |      0.04 |
| {'Apple Watch Ultra 2'}    |      0.02 |
| {'Apple TV 4K'}            |      0.02 |
| {'AirPods Max'}            |      0.02 |
| {'AirPods 3rd Gen'}        |      0.02 |
| {'MacBook Pro 14” M3'}     |      0.02 |
| {'Mac Mini M2'}            |      0.02 |
| {'Logic Pro License'}      |      0.02 |
| {'MacBook Pro 16” M3 Max'} |      0.02 |
```

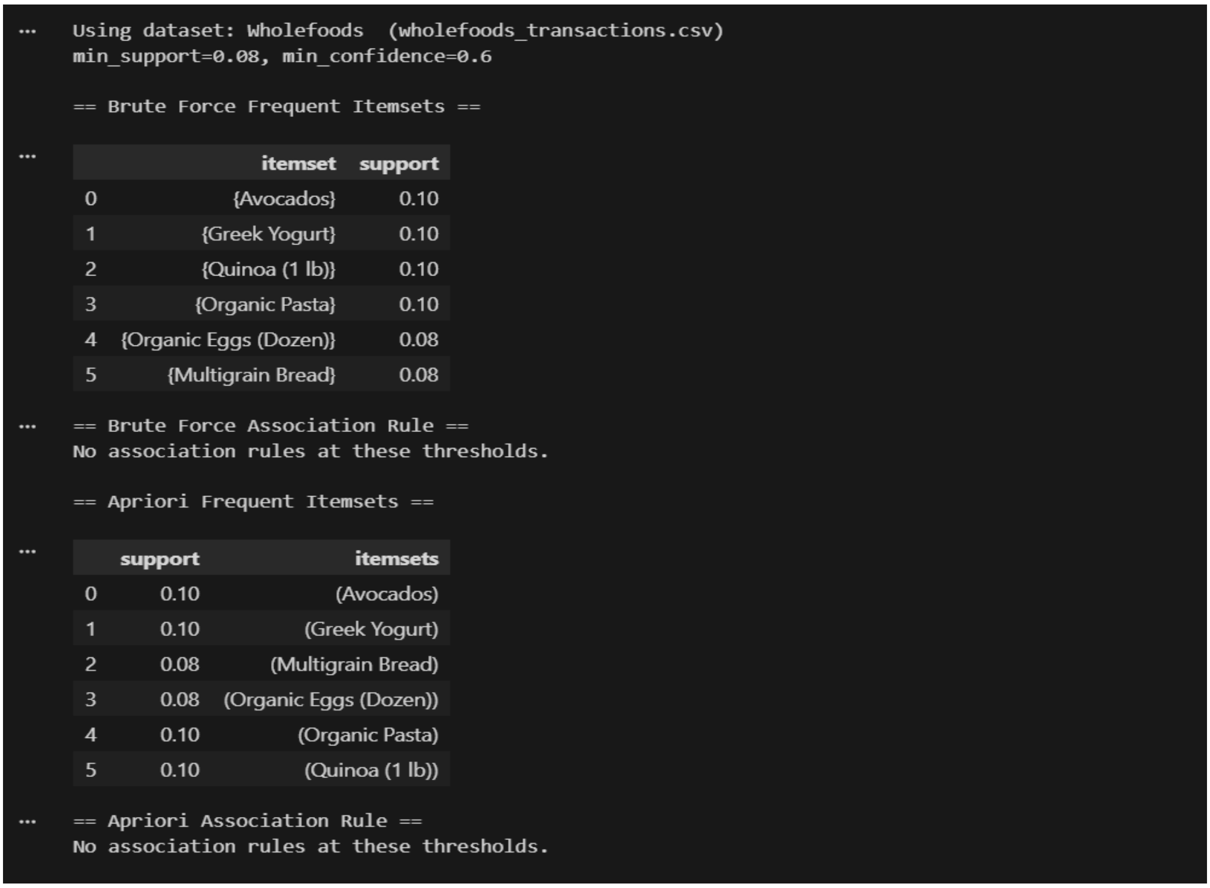


```
== Brute Force Association Rule ==
No association rules at these thresholds.

== Apriori Frequent Itemsets ==
|   support | itemsets                              |
|-----------|---------------------------------------|
|      0.02 | frozenset({'AirPods 3rd Gen'})        |
|      0.02 | frozenset({'AirPods Max'})            |
|      0.1  | frozenset({'AirPods Pro (2nd Gen)'})  |
|      0.04 | frozenset({'AirTag (4-Pack)'})        |
|      0.02 | frozenset({'Apple TV 4K'})            |
|      0.06 | frozenset({'Apple Watch Series 10'})  |
|      0.02 | frozenset({'Apple Watch Ultra 2'})    |
|      0.04 | frozenset({'Beats Fit Pro'})          |
|      0.04 | frozenset({'Beats Studio Pro'})       |
|      0.04 | frozenset({'Final Cut Pro License'})  |
|      0.04 | frozenset({'HomePod Mini'})           |
|      0.02 | frozenset({'Logic Pro License'})      |
|      0.02 | frozenset({'Mac Mini M2'})            |
|      0.04 | frozenset({'Mac Studio M3 Ultra'})    |
|      0.1  | frozenset({'MacBook Air M3'})         |
|      0.02 | frozenset({'MacBook Pro 14” M3'})     |
|      0.02 | frozenset({'MacBook Pro 16” M3 Max'}) |
|      0.02 | frozenset({'iPad Air 11” M2'})        |
|      0.02 | frozenset({'iPad Mini 6th Gen'})      |
|      0.08 | frozenset({'iPad Pro 13” M4'})        |

== Apriori Association Rule ==
No association rules at these thresholds.

== FP-Growth Frequent Itemsets ==
|   support | itemsets                             |
|-----------|--------------------------------------|
|      0.12 | frozenset({'iPhone 16'})             |
|      0.1  | frozenset({'MacBook Air M3'})        |
|      0.1  | frozenset({'AirPods Pro (2nd Gen)'}) |
|      0.08 | frozenset({'iPad Pro 13” M4'})       |
|      0.06 | frozenset({'iPhone 15 Pro'})         |
|      0.06 | frozenset({'Apple Watch Series 10'}) |
```



```
== Apriori Association Rule ==
No association rules at these thresholds.

== FP-Growth Frequent Itemsets ==
|   support | itemsets                              |
|-----------|---------------------------------------|
|      0.12 | frozenset({'iPhone 16'})              |
|      0.1  | frozenset({'MacBook Air M3'})         |
|      0.1  | frozenset({'AirPods Pro (2nd Gen)'}) |
|      0.08 | frozenset({'iPad Pro 13” M4'})        |
|      0.06 | frozenset({'iPhone 15 Pro'})          |
|      0.06 | frozenset({'Apple Watch Series 10'}) |
|      0.04 | frozenset({'HomePod Mini'})           |
|      0.04 | frozenset({'AirTag (4-Pack)'})        |
|      0.04 | frozenset({'Beats Studio Pro'})       |
|      0.04 | frozenset({'Beats Fit Pro'})          |
|      0.04 | frozenset({'Final Cut Pro License'}) |
|      0.02 | frozenset({'Logic Pro License'})      |
|      0.02 | frozenset({'Mac Mini M2'})            |
|      0.04 | frozenset({'Mac Studio M3 Ultra'})    |
|      0.02 | frozenset({'iPhone 16 Pro Max'})      |
|      0.02 | frozenset({'AirPods 3rd Gen'})        |
|      0.02 | frozenset({'iPad Air 11” M2'})        |
|      0.02 | frozenset({'iPad Mini 6th Gen'})      |
|      0.02 | frozenset({'AirPods Max'})            |
|      0.02 | frozenset({'Apple Watch Ultra 2'})    |

== FP-Growth Association Rule ==
No association rules at these thresholds.

[Hint] No association rules at your thresholds. Lower support/confidence or strengthen repeated bundles.
```

Notebook output screenshots:



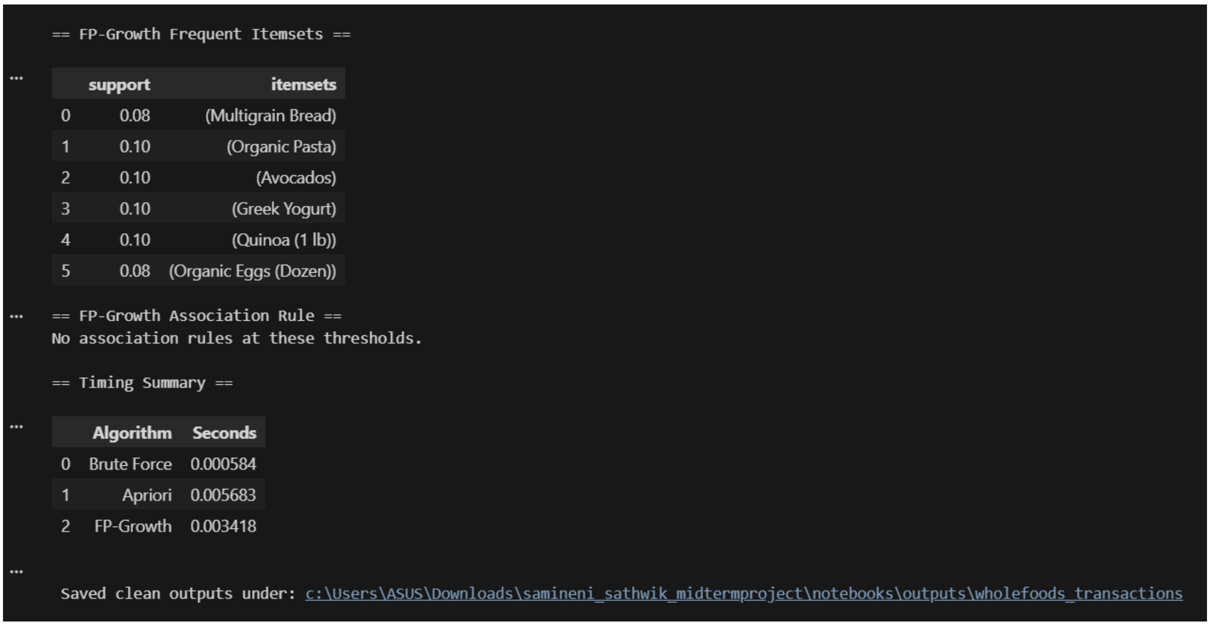
```
...  Using dataset: Wholefoods  (wholefoods_transactions.csv)
     min_support=0.08, min_confidence=0.6

     == Brute Force Frequent Itemsets ==
```

	itemset	support
0	{Avocados}	0.10
1	{Greek Yogurt}	0.10
2	{Quinoa (1 lb)}	0.10
3	{Organic Pasta}	0.10
4	{Organic Eggs (Dozen)}	0.08
5	{Multigrain Bread}	0.08

```
...  == Brute Force Association Rule ==
     No association rules at these thresholds.

     == Apriori Frequent Itemsets ==
```

	support	itemsets
0	0.10	(Avocados)
1	0.10	(Greek Yogurt)
2	0.08	(Multigrain Bread)
3	0.08	(Organic Eggs (Dozen))
4	0.10	(Organic Pasta)
5	0.10	(Quinoa (1 lb))

```
...  == Apriori Association Rule ==
     No association rules at these thresholds.
```



```
== FP-Growth Frequent Itemsets ==
```

	support	itemsets
0	0.08	(Multigrain Bread)
1	0.10	(Organic Pasta)
2	0.10	(Avocados)
3	0.10	(Greek Yogurt)
4	0.10	(Quinoa (1 lb))
5	0.08	(Organic Eggs (Dozen))

```
== FP-Growth Association Rule ==
No association rules at these thresholds.

== Timing Summary ==
```

	Algorithm	Seconds
0	Brute Force	0.000584
1	Apriori	0.005683
2	FP-Growth	0.003418

```
Saved clean outputs under: c:\Users\ASUS\Downloads\samineni_sathwik_midtermproject\notebooks\outputs\wholefoods_transactions
```

13. Conclusion

This project successfully implemented, executed, and validated three major frequent-pattern mining algorithms.
By maintaining deterministic datasets, consistent parameters, and clean modular code, the project meets the CS634 midterm goals for **correctness, clarity, reproducibility, and interpretability**.

- Brute Force provided a baseline for correctness.
- Apriori demonstrated scalable candidate pruning.
- FP-Growth achieved the best computational performance.

Both CLI and Jupyter deliver reproducible outputs suitable for submission and evaluation.

Deliverables Summary

Deliverable	**File/Folder**
Source Code	/src/*.py
Datasets	/data/*.csv
Notebook	/notebooks/cs634.ipynb
Outputs (CLI)	/outputs/

Deliverable	**File/Folder**
Outputs (Notebook)	/notebooks/outputs/
Report (PDF)	/samineni_sathwik_midterm_report.pdf
README	/README.md
GitHub Repo	https://github.com/SathwikSamineni/CS634_Midterm_Project