CS634 – Data Mining
Master Sample Midterm Report
(Python-Only, Single Runner)

Student Name: Amanda

Email: ay365@njit.edi

Course: CS634 – Data Mining

Instructor: Dr. Yasser Abduallah

1. Introduction

This report outlines the implementation process for itemset mining and association rule learning using three different approaches:

- Brute Force (built from scratch)
- Apriori (Python library)
- FP-Growth (Python library)

The report is presented in a tutorial format, enabling other students to follow the steps and reproduce the results.

2. Environment & Installation

2.1 Items

I chose 10 general retail items to represent a variety of products sold across major stores:

Headphones, Laptop, Phone Case, Sneakers, T-shirt, Shampoo, Coffee Maker, Batteries, Backpack, Smartwatch.

2.2 Transactions

I created 5 separate datasets, one for each retailer, with at least 20 transactions each.

Each CSV file represents a unique retailer (Amazon, Walmart, BestBuy, Nike, Costco).

2.3 Dataset Notes

- Each retailer's dataset was created in Excel and exported to CSV.

- Deterministic (no random generation).

3. Brute Force Algorithm

3.1 Method

1. Compute support for all 1-itemsets and keep those meeting the minimum support threshold.
2. Iteratively generate candidate k-itemsets using combinations of all items.
3. For each candidate, calculate its support and keep only frequent itemsets (≥ min_support).
4. Stop when no new frequent itemsets are found.
5. Generate association rules from all frequent itemsets, keeping those with confidence ≥ min_confidence.

3.2 Example Run

```
Parameters: Support = 0.3, Confidence = 0.6
Output:

Frequent Itemsets:
{A Beginner's Guide} (support=0.5)
{Android Programming: The Big Nerd Ranch} (support=0.5)

Association Rules:
Effective Java (2nd Edition) → A Beginner's Guide (confidence=0.6)
A Beginner's Guide → Effective Java (2nd Edition) (confidence=0.6)
```

4. Apriori and FP-Growth

4.1 Apriori

1. Convert transactions into a one-hot encoded DataFrame.
2. Use mlxtend.frequent_patterns.apriori to find frequent itemsets meeting the minimum support threshold.
3. Generate association rules using association_rules() with a minimum confidence threshold.
4. Save both frequent itemsets and rules to CSV files for each retailer.
5. Results: Produced the same frequent itemsets and association rules as the brute force method.

4.2 FP-Growth

1. Use mlxtend.frequent_patterns.fpgrowth on the same one-hot encoded data to identify frequent itemsets.
2. Generate association rules using association_rules() with the same thresholds.
3. Save results (itemsets and rules) for each retailer.
4. Results: Matched Apriori outputs — identical frequent itemsets and rules.

4.3 Timing Comparison

```
=== Average Execution Time (All Retailers) ===
  Algorithm  Execution Time (s)
Brute Force              0.051
```

```
   Apriori              0.638
FP-Growth               0.134
```

5. Multiple Parameters

Tested three settings per dataset:
- Support = 0.3, Confidence = 0.6
- Support = 0.2, Confidence = 0.5
- Support = 0.1, Confidence = 0.4

6. GitHub Repository

Code and datasets uploaded here:
Collaborator access given to ya54@njit.edu.

7. How to Run the Code

7.1 Run Options
Midterm-Part4-Run.py

8. Conclusion

- Brute Force works but is slow.
- Apriori and FP-Growth are faster and give identical results.

9. Appendix

9.1 Dataset Snippet
TID,Items

TransactionID	ItemsPurchased
1	Milk, Eggs, Bread
2	Eggs, Bread, Cheese, Butter
3	Bread, Cheese, Butter, Chicken, Rice
4	Cheese, Butter, Chicken, Rice, Olive Oil, Pasta

9.2 Screenshot Samples

```
=== Available Retailer Datasets ===
1. Amazon.csv
2. Costco.csv
3. Walmart.csv
4. Nike.csv
5. BestBuy.csv

Select a dataset by number: 4
Enter minimum support (0–1): .3
Enter minimum confidence (0–1): .6

=== Execution Summary ===
Selected Dataset : Nike.csv
Minimum Support   : 0.3
Minimum Confidence: 0.6
```

=== Apriori Results Preview ===

Table Apriori:
Itemset | Count
['Cap'] : 10
['Gloves'] : 10
['Hoodie'] : 10
['Running Shoes'] : 10
['Shorts'] : 10
['Soccer Shoes'] : 10
['Socks'] : 10
['Sweatshirt'] : 10
['Track Pants'] : 10
['Training Shirt'] : 10

```
Final Association Rules:
Rule 1: [{'Running Shoes'}], [{'Cap'}], 0.8
Rule 2: [{'Cap'}], [{'Running Shoes'}], 0.8
Rule 3: [{'Cap'}], [{'Shorts'}], 0.6
Rule 4: [{'Shorts'}], [{'Cap'}], 0.6
Rule 5: [{'Cap'}], [{'Soccer Shoes'}], 0.6
Rule 6: [{'Soccer Shoes'}], [{'Cap'}], 0.6
Rule 7: [{'Cap'}], [{'Training Shirt'}], 0.8
Rule 8: [{'Training Shirt'}], [{'Cap'}], 0.8
```

GitHub Repository Structure

cs634_amandayu_midtermproject/

```
│
├── transactional_datasets/          # Main project data and results
│   ├── transactions/
│   │   ├── amazon.csv
│   │   ├── walmart.csv
│   │   ├── bestbuy.csv
│   │   ├── nike.csv
│   │   └── costco.csv
│   │
│   ├── results/                     # Output CSVs
│   │   ├── amazon_frequent_itemsets.csv
│   │   ├── walmart_association_rules.csv
│   │   ├── summary.csv
│   │   └── execution_times.csv
│   │
│   ├── Screenshots/                 # Saved output screenshots for the report
│   │   ├── run1.jpg
│   │   ├── run2.jpg
│   │   └── Timing.jpg
│   │
│   ├── AY-Midterm-Project.ipynb     # Jupyter notebook version of the analysis
│   ├── Midterm-Part4-Run.py         # Python script to run all algorithms
│   └── retailers.csv                # List of retailer dataset names and basic info
│
├── report/                          # Final midterm report
│   └── amandayu_midterm_report.pdf
│
```

└──README.md # Instructions for setup and running the project