CS634 – Data Mining Midterm Project Report

Student Name: Shivaji Burle

Email: sb2857@njit.edu

Course: CS634 – Data Mining

Instrucator: Dr. Yasser Abduallah

1.Introduction

This report documents the process of implementing frequent itemset mining and association rule learning using three methods:

Brute Force Method: This method checks all possible item combinations one by one. It is simple but takes a lot of time when there are many items.

Apriori Algorithm: Apriori finds patterns by removing item combinations that do not appear often. It is faster than brute force.

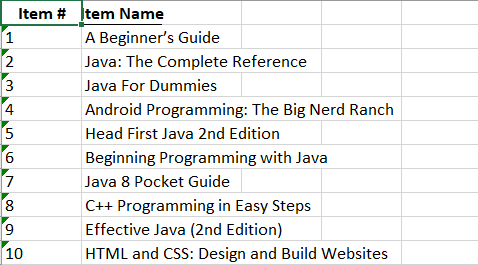
FP-Growth Algorithm: FP-Growth builds a special tree to find frequent patterns quickly. It works faster on large datasets.

The report explains how to prepare data, find itemsets, and create rules clearly.

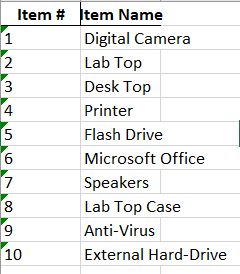
2.Dataset Creation

2.1. Data Items:

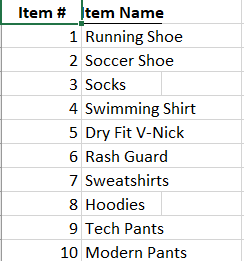
Amazon Data:



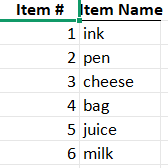
Best Buy Data:



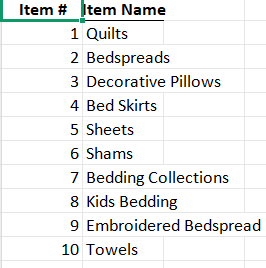
Nike Data:



Custom Data:

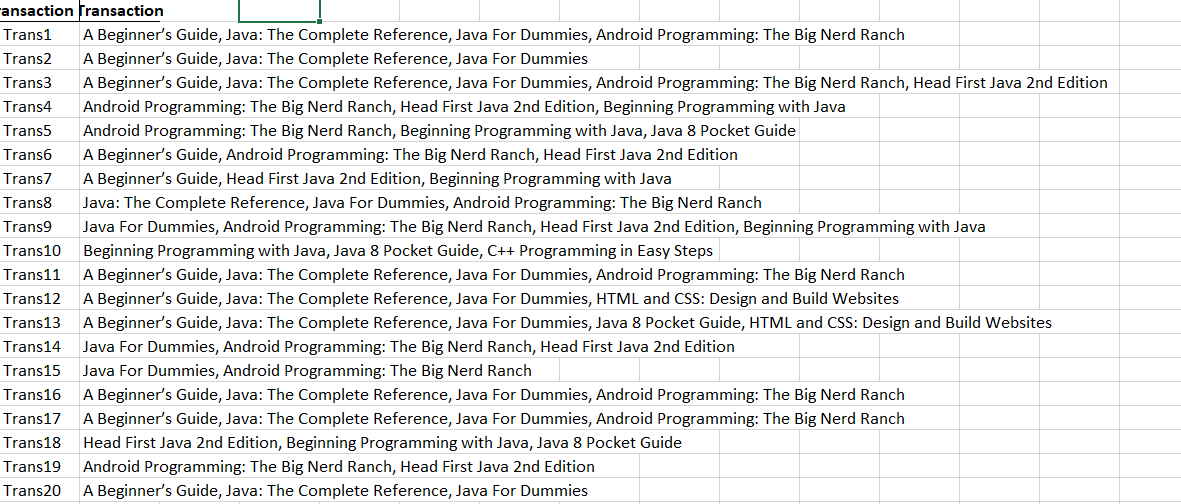


K-mart Data:



2.2 Transactions

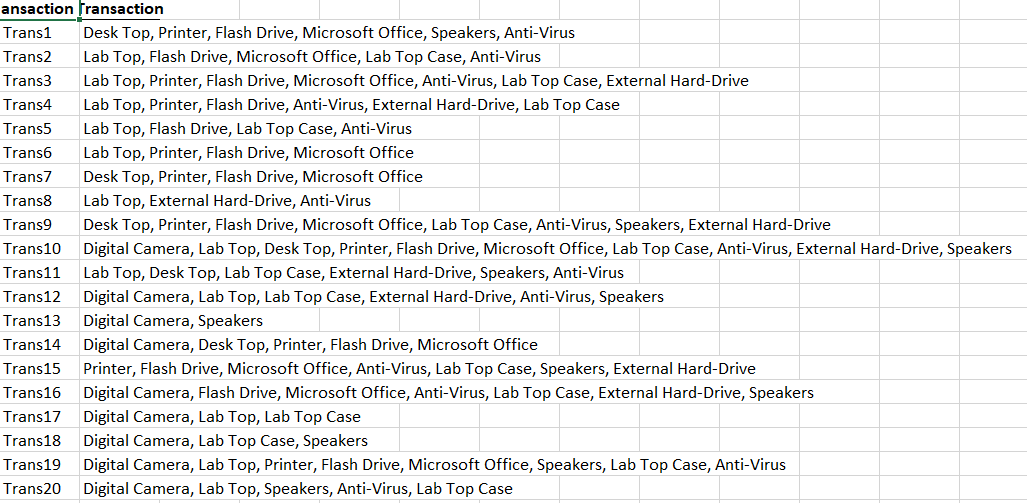
Amazon:



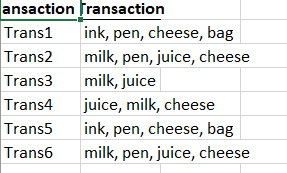
K-Mart:



BestBuy:



Custom:



Nike:



2.3 Datsets Note:

• The datasets were provided by the faculty for this project.

• I have converted and saved them in CSV file format.

• No random or artificial data was used in this analysis.

3. Brute Force Algorithm

3.1 Method

1. The code first converts all transactions into sets.
2. It counts how often each single item appears.
3. It then generates larger itemsets (pairs, triplets) and checks their support one by one.
4. If the itemset meets the minimum support, it’s kept as frequent.
5. Finally, it forms rules (A → B) and prints their confidence and support.

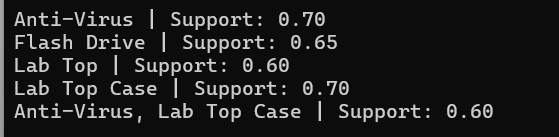
3.2 Example run

Dataset: Bestbuy.csv

Parameters: Support = 0.6, Confidence = 0.6

Output:

Frequent Itemsets:



Association rules:



4. Apriori and FP-Growth

4.1 Apriori :

1. The code encodes all transactions using **TransactionEncoder** to create a one-hot DataFrame.
2. It runs the **apriori()** function from mlxtend to find frequent itemsets.
3. It then applies **association\_rules()** to get rules based on confidence.
4. It displays itemsets with their support and rules with confidence values.
5. Execution time is recorded and printed for comparison.

Results same as Brute Force.

3. FP-Growth

1. Similar to Apriori, it encodes transactions using TransactionEncoder.
2. It runs the fpgrowth() function from mlxtend instead of apriori.
3. FP-Growth builds an FP-Tree internally to find frequent patterns efficiently.
4. The association\_rules() function is used again to form rules.
5. It displays results and runtime alongside the other methods for comparison.

Results same as Brute Force