CS634 – Data Mining

Midterm Project Report

Student Name: Catalin Mocanu

Email: cm788@njit.edu

Course: CS634 – Data Mining

Instructor: Dr. Yasser Abduallah

1. Introduction

The objective of this project is to explore association rule mining techniques for market basket analysis. The goal is to identify frequent itemsets and generate association rules showing relationships between products across five datasets (Amazon, BestBuy, Sephora, Target and IKEA). The project compares three algorithms: a custom Brute Force implementation, Apriori, and FP-Growth, all executed via an interactive Python menu.

2. Dataset Creation

2.1 Items

Each dataset contains 20 transactions representing different product categories, such as cosmetics or household goods.

2.2 Transactions

Below are the transactions from dataset ikea as an example.

Transaction ID	Transaction
Trans1	table,chair,lamp
Trans2	bed frame,pillow,rug
Trans3	desk,lamp,bookshelf
Trans4	lamp,rug,table
Trans5	bed frame,pillow,lamp
Trans6	chair,table,rug
Trans7	desk,bookshelf,chair
Trans8	bed frame,pillow,lamp,rug
Trans9	table,chair,lamp
Trans10	rug,lamp,bookshelf
Trans11	desk,lamp,table
Trans12	bed frame,pillow,rug
Trans13	chair,table,bookshelf
Trans14	desk,bookshelf,lamp
Trans15	rug,lamp,chair
Trans16	bed frame,pillow,lamp,rug
Trans17	chair,table,desk
Trans18	bookshelf,desk,lamp
Trans19	bed frame,pillow,table
Trans20	table,chair,rug,lamp

Fig. 1 – Ikea dataset transactions

2.3 Dataset Notes

Datasets were created manually and stored as CSV files inside a 'data' folder. Each file includes 20 transactions, cleaned and formatted using pandas for analysis.

3. Brute Force Algorithm

3.1 Method

The brute-force algorithm enumerates all item combinations using itertools, counts their occurrences and filters by minimum support. It then generates rules (A->B) if confidence and lift thresholds are satisfied.

3.2 Example for dataset (min support = 0.3 and min conf = 0.5)

ikea_frequent_itemsets

k	itemset	support
1	lamp	0.65
1	rug	0.45
1	table	0.45
1	chair	0.4
1	bed frame	0.3
1	bookshelf	0.3
1	desk	0.3
1	pillow	0.3
2	bed frame,pillow	0.3
2	chair,table	0.3
2	lamp,rug	0.3

Fig. 2 – Brute force frequent itemsets

ikea_bruteforce_rules

antecedent	consequent	support	confidence	lift
bed frame	pillow	0.3	1.0	3.333333
pillow	bed frame	0.3	1.0	3.333333
chair	table	0.3	0.7500000000000000	1.666667
rug	lamp	0.3	0.6666666666666670	1.025641
table	chair	0.3	0.6666666666666670	1.666667

Fig. 3 – Brute force rules

4. Apriori and FP-Growth

4.1 Apriori

The Apriori algorithm was implemented using **mlxtend.frequent_patterns.apriori.** It produced identical results to the brute-force approach.

ikea_apriori_frequent_itemsets

support	itemsets
0.3	frozenset({'bed frame'})
0.3	frozenset({'bookshelf'})
0.4	frozenset({'chair'})
0.3	frozenset({'desk'})
0.65	frozenset({'lamp'})
0.3	frozenset({'pillow'})
0.45	frozenset({'rug'})
0.45	frozenset({'table'})
0.3	frozenset({'pillow', 'bed frame'})
0.3	frozenset({'table', 'chair'})
0.3	frozenset({'lamp', 'rug'})

Fig. 4 – Apriori frequent itemsets

ikea_

antecedents	consequents	antecedent support	consequent support	support	confidence	lift
frozenset({'pillow'})	frozenset({'bed frame'})	0.3	0.3	0.3	1.0	3.3333333333333300
frozenset({'bed frame'})	frozenset({'pillow'})	0.3	0.3	0.3	1.0	3.3333333333333300
frozenset({'table'})	frozenset({'chair'})	0.45	0.4	0.3	0.6666666666666670	1.6666666666666700
frozenset({'chair'})	frozenset({'table'})	0.4	0.45	0.3	0.7500000000000000	1.6666666666666700
frozenset({'rug'})	frozenset({'lamp'})	0.45	0.65	0.3	0.6666666666666670	1.0256410256410300

Fig. 5 – Apriori rules

4.2 FP-Growth

FP-Growth was implemented using **mlxtend.frequent_patterns.fpgrowth**. It achieved the same frequent itemsets and rules as Apriori but with shorter runtime.

ikea_fpgrowth_frequent_itemsets

support	itemsets
0.65	frozenset({'lamp'})
0.45	frozenset({'table'})
0.4	frozenset({'chair'})
0.45	frozenset({'rug'})
0.3	frozenset({'pillow'})
0.3	frozenset({'bed frame'})
0.3	frozenset({'desk'})
0.3	frozenset({'bookshelf'})
0.3	frozenset({'table', 'chair'})
0.3	frozenset({'lamp', 'rug'})
0.3	frozenset({'pillow', 'bed frame'})

Fig. 5 – FP-Growth frequent itemsets

ikea_f

antecedents	consequents	antecedent support	consequent support	support	confidence	lift
frozenset({'table'})	frozenset({'chair'})	0.45	0.4	0.3	0.6666666666666670	1.6666666666666700
frozenset({'chair'})	frozenset({'table'})	0.4	0.45	0.3	0.7500000000000000	1.6666666666666700
frozenset({'rug'})	frozenset({'lamp'})	0.45	0.65	0.3	0.6666666666666670	1.0256410256410300
frozenset({'pillow'})	frozenset({'bed frame'})	0.3	0.3	0.3	1.0	3.3333333333333300
frozenset({'bed frame'})	frozenset({'pillow'})	0.3	0.3	0.3	1.0	3.3333333333333300

Fig. 6 – FP-Growth rules

5. Results across datasets

Summary of all the datasets :

Results were consistent across all methods. Lower support thresholds produced more frequent itemsets and rules, while higher thresholds reduced them.

=== Summary ===

	Dataset	Algorithm	#Transactions	#Frequent Itemsets	#Rules	Time (s)
0	IKEA	Brute Force	20	11	5	0.003198
1	IKEA	Apriori	20	11	5	0.004769
2	IKEA	FP-Growth	20	11	5	0.005679

Fig. 7 – Summary of all 3 algorithms

6. Discussion

The three algorithms produced identical results for the IKEA dataset, each identifying 11 frequent itemsets and 5 association rules. The execution times were also similar, all under 0.006 seconds, due to the small dataset size.

Even though brute force is conceptually the slowest method, it performed efficiently because the dataset only had 20 transactions. As the dataset size grows, its computational cost increases exponentially since it must evaluate every possible combination of items.

Apriori improves upon Brute Force by pruning infrequent itemsets early, reducing the search space, but it still requires multiple passes through the dataset, which can become costly at scale.

FP-Growth is the most scalable method, using a compact FP-tree structure that eliminates the need for candidate generation. This allows FP-Growth to maintain performance advantages for large datasets, making it the preferred choice in real-world applications.

7. Conclusion

This project demonstrated the implementation and comparison of three association rule mining techniques. For small datasets, all methods performed equally well. FP-Growth is the best choice for scalability in larger datasets.

8. Appendix

8.1 Project Structure

```
Mocanu_Catalin_midtermproject/
│
├── Midterm_Project.ipynb
├── main.py
├── midterm_from_notebook.py
├── Mocanu_Catalin_report.pdf
├── data/
│   ├── amazon.csv
│   ├── sephora.csv
│   ├── bestbuy.csv
│   ├── target.csv
│   └── ikea.csv
└── output/
```

8.2 Required packages

- pandas
- mlxtend
- numpy

8.3 How to run the project

GitHub link: https://github.com/catalinm-24/Mocanu_Catalin_midtermproject/tree/main

PS: I added an excel file to the output folder because I couldn't upload it empty to github

Option 1 – Run in Jupyter Notebook

Step 1 – Open the Jupyter Notebook file

Step 2 – Click Run All

Step 3 **-** You will be asked to say which database you want to run, followed by which Algorithm, and then input the minimum support and confidence. The rules and frequent itemsets are generated in excel files in /output and the program gives a quick summary of each algorithm that was run.

Option 2 – Run in terminal

Step 1 – Open a terminal window

Step 2 – Navigate to the project folder:

- .../Midterm_Project_Association_Rules

Step 3 – Create and activate a virtual environment:

- python3 -m venv .venv
- source .venv/bin/activate

Step 4 – Run the project

Step 5 – You will be asked to input which database you want to run, followed by which Algorithm, and then input the minimum support and confidence. The rules and frequent itemsets are generated in excel files in /output and the program gives a quick summary of each algorithm that was run.