CS634 – Data Mining

Midterm Project Report

Name: Kenil Pravinbhai Avaiya

NJIT UCID: ka683

Email Address: ka683@njit.edu

Date: 10-16-2025

Professor: Dr. Yasser Abdullah

Course: CS 634 101 Data Mining

# Title

Frequent item set data mining for diverse transactional data sets from different retailers.

# Abstract

The aim of this project is to identify association rules (A -> B, C) and frequent itemset from diverse transaction datasets, using frequent itemset data mining techniques such as Apriori and FP-Growth tree. Additionally, a brute force strategy, which I have implemented from scratch, will be used to perform same function.

The end goal is to select a dataset as well as algorithm chosen by the user. After making the selection, perform data mining on that dataset, the generated association rules will be printed, along with the time taken to generate them.

# Introduction

Nowadays, retailers are increasingly adopting frequent item set data mining as a popular technique. By analyzing transactional datasets, retailers can uncover core relational relationships and co-occurrence patterns among frequently purchased items. This project aims to explore and compare three distinct algorithms for generating association rules from transaction datasets.

1. **Brute Force Algorithm** – A manually implemented method that systematically checks all possible item combinations to find frequent itemset.
2. **Apriori Algorithm** – An algorithm that optimizes the Apriori property to minimize unnecessary computations by pruning infrequent itemset.
3. **FP-Growth Algorithm** – A tree base approach that efficiently discovers frequent patterns without candidate generation.

For implementation, multiple transactional datasets from various retailers, such as Apple and Amazon, were utilized. Each algorithm was evaluated with user-defined thresholds for minimum support and confidence, enabling a comparative analysis of their performance and rule generation efficiency.

# Important Concepts, terminology and Principles

**Support**: It eliminates rare patterns (statistical reliability) based on the frequency of items appearing in the dataset.

**Confidence**: It eliminates unreliable associations (predictive strength) based on the likelihood of an item being purchased given the presence of another item. In other words, it measures how often the rule is likely to be true.

**Frequent** **Itemset**: Itemset that satisfy the minimum support requirement are considered frequent itemset. These are sets of items that appear frequently together in transactions.

**Association** **Rules**: Association rules are like connections between things that happen together a lot. They’re usually written as A->B, which means if A happens, B is probably going to happen too.

# Project Structure and Workflow

1. Preparation of Datasets

This project, I generated five distinct datasets using Kaggle as well as midterm project example pdf, each with around 10 unique items and based on those items, have twenty-five transactions. The datasets represent general purchase records from popular retailers which people willing to buy: Amazon, Apple, Best Buy, Costco, and Nike. While running this project, program ask Users to select any one of these datasets for analysis and specify minimum support and confidence value (thresholds). These parameters help identify association rules during the mining process.

1. Environment and Installing required libraries and modules

To run the project file, we need some python packages:

* Pandas: use to handle dataset, load the data, as well as clean the data and ready for next analysis step.
* Apyori: to implement apriori data mining
* Mlxtend: to implement FP growth technique.

In my files, if you try to run main.ipynb file. I already define the required libraires, so you do not need to install any package just run the cell and you ready to go.

A close-up of a white background

AI-generated content may be incorrect.

**If you are trying to run the main.py file, then followed bellowed the instruction.**

1. **First Install Python if you do not have**

To run python file, we need to install python in our system.

* **Windows:** Download the installer from python.org and add Python to your PATH during installation.
* **Mac:** Python is usually pre-installed; for the latest version, use python.org or brew install python.

## **Set Up a Virtual Environment**

To establish a virtual environment, locate the folder containing all the necessary files. Then, open the terminal within that folder and proceed with the following steps.

* **Windows Command Prompt:**

**python -m venv venv** # create a python environment with name venv

**venv\Scripts\activate**  # activate the python environment

* **Mac / Linux Terminal:**

**python3 -m venv venv** # create a python environment with name venv

**source venv/bin/activate** # activate the python environment

1. **Install Required Python Packages**

For your file, install these libraries:

* Pandas
* apyori
* mlxtend
* tabulate

Run in your command line or terminal (in the activated environment):

**pip install pandas apyori mlxtend openpyxl tabulate**

## **Prepare Your Data Files**

Make sure your dataset CSV files (such as amazon.csv, apple.csv, etc.) are in the same directory as your script or update the script with their full paths.

## **Run the Python Script**

Run your main file in your terminal.

**Windows:**

**python main.py**

**Mac:**

**python3 main.py**

## Code Structure and Project Workflow Overview

This project is organized to allow users to perform association rule mining on transaction datasets from multiple retail stores (Amazon, Apple, Best Buy, Costco, Nike) using different algorithms.

Workflow:

* Store & Dataset Selection:

The program presents a list of stores to the user. The user selects a store, and the corresponding dataset (CSV file) is loaded.

* Preprocessing:

Transactions with empty or null entries are removed from the dataset. Each transaction string is split and converted into a list of items (e.g., ‘Tv, Usb C cable, Ps5’ → [‘Tv, ‘ Usb C cable’, ‘Ps5’]).

* Parameter Input:

The user is prompted to enter minimum support and confidence thresholds (usually expressed as percentages). These values are converted and used as input parameters for the mining algorithms.

A screenshot of a computer program

AI-generated content may be incorrect.

Code responsible for taking input from the user.

* Algorithm Selection & Execution:

Ask the user to choose an algorithm based on their choice.

* Brute Force Algorithm:

A brute force algorithm checks every possible option or combination to find the correct solution. It checks all possible answers one by one, without using any shortcut or smart method. This way, it makes sure that all correct answers are found.

Brute force is easy to understand and use, but it can take a lot of time and computer power when the data set or problem is large.

A screenshot of a computer

AI-generated content may be incorrect.

Code represents implementation of brute force.

* Apriori Algorithm (apyori library):

The Apriori algorithm finds frequent item sets in a unique way by taking combinations that meet the minimum support value condition. It starts with small item sets and gradually builds larger ones using only the frequent items. This helps reduce extra work and makes the process faster.

The algorithm is often used with libraries like apyori to create association rules from transaction data.

A screenshot of a computer code

AI-generated content may be incorrect.

Code represents implementation of Apriori Algorithm.

* FP-Tree Algorithm (mlxtend library):

The FP-Growth algorithm is a fast method that uses a tree structure to find frequent item sets. It stores all transactions in an **FP-tree**, which helps it find common patterns without creating too many possible combinations. By exploring this tree step by step, FP-Growth can quickly find large item sets that meet the minimum support value.

In our project, I used the **mlxtend** python library to find frequent item sets and association rules from the transaction data.

A screenshot of a computer code

AI-generated content may be incorrect.

Code represents implementation of FP-Growth tree Algorithm.

* All Algorithms with Execution Time Comparison:

I also provide an option to select all algorithms simultaneously and execute all three algorithms on the same data. This allows me to compare their execution times and highlight any performance differences.

* Output:

The discovered association rules are displayed in a human-readable format (e.g., **{Mac} → {Kindle}, Confidence: 92.31%**).

When the user selects all algorithms to execute, the execution times and rule outputs are presented. The faster algorithm that takes the minimum time is also displayed, along with the total time taken by the respected algorithm table. Simply compare the times to determine the most efficient algorithm.

My Output when I run the code:

I select the Best Buy data set for running and also select all algorithms for running with the minimum support of 50 and the minimum confidence value of 60.

A black and white image of a person

AI-generated content may be incorrect.

Selecting data set and providing support and confidence value

A screenshot of a computer

AI-generated content may be incorrect.

Selecting algorithm and getting result.

**Fastest Algorithm**: Apriori library with a time of **0.0009 seconds**

* Brute Force: **0.067925 seconds**
* Apriori: **0.0001356 seconds**
* FP Tree: **0.0057804 seconds**

Additionally, I observed that when the dataset contained many transactions, the Apriori library algorithm exhibited the most efficient execution time. Conversely, with a reduced number of transactions (approximately 10) and fewer items, the Brute Force algorithm occasionally demonstrated the best performance.

# Results and Evaluation

# Based on our results, the **Apriori algorithm** worked the fastest, followed by the **Brute Force algorithm**. The **FP-Growth Tree algorithm** was the slowest, however in some cases the rime taken by **Fp-Growth tree algorithm** is smaller than **Brute Force algorithm**, but it still gave correct results. Even though their speeds were different, all three algorithms produced accurate and reliable outcomes in our project.

# Conclusion

This project presents the implementation and comparative analysis of three widely used algorithms for association rule mining: Brute Force, Apriori, and FP-Tree. While the Brute Force algorithm provides simplicity, it suffers from efficiency limitations. In contrast, the Apriori and FP-Tree algorithms exhibit enhanced efficiency and scalability, making them suitable for handling larger datasets.