Association Rule Mining Project Report

**Name:** Sushil Saindane  
**NJIT UCID:** sbs8  
**Email Address:** sbs8@njit.edu  
**Date:** 10/10/2024  
**Professor:** Yasser Abduallah  
**Course:** CS 634101 Data Mining

# Abstract

* This project implements and compares three association rule mining algorithms: Brute Force, Apriori, and FP-Growth. By applying these algorithms to retail transaction data from five different stores (Amazon, Best Buy, Nike, Walmart, and Target), we explore their effectiveness in uncovering purchase patterns and generating association rules. The project demonstrates the practical application of data mining techniques in a retail context, providing insights into algorithm performance, customer behavior, and the trade-offs between different approaches to association rule mining.

# Introduction

* Association rule mining is a fundamental technique in data analytics, particularly valuable in retail and e-commerce sectors. This project focuses on implementing and comparing three key algorithms:

1. **Brute Force**: A custom implementation that exhaustively checks all possible itemsets.
2. **Apriori**: An efficient algorithm for frequent itemset generation, implemented using the mlxtend library.
3. **FP-Growth**: A tree-based approach for mining frequent patterns, also implemented using mlxtend.

* These algorithms are applied to transaction data from five major retailers: Amazon, Best Buy, Nike, Walmart, and Target. By comparing their performance and results, we aim to gain insights into the strengths and weaknesses of each approach in the context of retail data analysis.

# Core Concepts and Principles

**• Frequent Itemset Discovery:**

* The core of association rule mining is identifying sets of items that frequently appear together in transactions. This project implements and compares different approaches to this task, from the exhaustive brute force method to more optimized algorithms like Apriori and FP-Growth.

**• Support and Confidence Metrics:**

Two key metrics guide our analysis:

Support: Measures how frequently an itemset appears in the dataset.

Confidence: Assesses the likelihood of an item being purchased given the purchase of another item.

These metrics are crucial for filtering and ranking association rules.

**• Association Rules:**

The ultimate goal of the project is to generate meaningful association rules that reveal patterns in customer purchasing behavior. These rules can be used for various business applications, such as product recommendations or store layout optimization.

**• Algorithm Efficiency:**

A key aspect of this project is comparing the computational efficiency of different algorithms. We measure and compare execution times to understand the trade-offs between exhaustive searches and more optimized approaches.

**• Data Preprocessing:**

The project includes steps for loading and preprocessing transaction data, demonstrating the importance of data preparation in data mining tasks.

# Project Workflow

1. **Data Preparation and Loading:** 
   * CSV File Creation: Transaction data provided by the professor was expanded and organized into Excel sheets for each store (Amazon, Best Buy, Nike, Walmart, and Target).
   * CSV Loading: A custom function (load\_transactions\_from\_csv) was implemented to read these CSV files and convert them into a format suitable for analysis.
2. **User Interaction:** 
   * The program implements a user-friendly interface allowing selection of the store to analyze and input of support and confidence thresholds.
   * Input validation functions ensure that user inputs are within acceptable ranges.
3. **Algorithm Implementation:**  
   **a. Brute Force Algorithm:**
   * Custom implementation using Python's itertools for combination generation.
   * Exhaustively checks all possible itemsets against the minimum support threshold.
   * Generates association rules based on frequent itemsets and minimum confidence.

**b. Apriori Algorithm:**

* + Utilizes the mlxtend library's implementation of Apriori.
  + Efficiently generates frequent itemsets using the apriori principle.

**c. FP-Growth Algorithm:**

* + Also uses mlxtend library's implementation.
  + Employs a tree-based approach for mining frequent patterns.

1. **Performance Measurement:** 
   * Execution time is measured for each algorithm using Python's time module.
   * Allows for direct comparison of algorithmic efficiency.
2. **Result Comparison and Analysis:** 
   * Compares the number of frequent itemsets and rules generated by each algorithm.
   * Checks for consistency in results across all three methods.
   * Identifies the fastest algorithm for each analysis.
3. **Rule Display and Interpretation:** 
   * Implements functions to display association rules in a readable format.
   * Provides insights into the strength of associations through support and confidence metrics.
4. **Item Analysis:** 
   * Calculates and displays item counts and support values.
   * Indicates whether individual items meet the specified support threshold.

# Implementation Details

**Data Handling**

* Utilized pandas for data manipulation
* Implemented custom functions for CSV file operations

**Brute Force Method**

* Iteratively generated candidate itemsets
* Calculated support and confidence for each potential rule

**Library Methods**

* Used mlxtend's implementations of Apriori and FP-Growth
* Leveraged pandas DataFrames for efficient data processing

**Performance Measurement**

* Used Python's time module to measure execution time for each algorithm

# Results and Analysis

When analyzing the Nike store transactions with a minimum support of 55% and minimum confidence of 70%, we obtained the following results:

1. **Brute Force Method:** 
   * Generated 6 association rules
   * Execution time: 0.0010 seconds
2. **Apriori Algorithm:** 
   * Generated 6 association rules
   * Execution time: 0.0080 seconds
3. **FP-Growth Algorithm:** 
   * Generated 6 association rules
   * Execution time: 0.0020 seconds

**Sample rules generated:**

**Rule 1**: {'Socks'} -> {'Sweatshirts'} (Confidence: 0.92, Support: 0.60)

**Rule 2**: {'Sweatshirts'} -> {'Socks'} (Confidence: 0.92, Support: 0.60)

**Rule 3**: {'Sweatshirts'} -> {'Running Shoe'} (Confidence: 0.85, Support: 0.55)

**Rule 4**: {'Running Shoe'} -> {'Sweatshirts'} (Confidence: 0.79, Support: 0.55)

# Comparison of Algorithms

* Our analysis showed that while all three methods produced the same number of rules for the given parameters, their execution times varied:

1. Brute Force Method: Was the fastest, provides a baseline for comparison
2. Apriori Algorithm: Was the slowest
3. FP-Growth: Was faster than Apriori but slower than Brute Force.

# Conclusion

* This project successfully implemented and compared three different approaches to association rule mining. We found that while the custom brute force method provides a good understanding of the underlying process. The association rules generated from our retail transaction data provide valuable insights into customer purchasing behavior, which can be used for product placement, marketing strategies, and inventory management.

# Screenshots

My CSV files look like this:

A screenshot of a computer

Description automatically generated

Figure 1: Best Buy Transactions

Following screenshots are snippets from my code:

A screen shot of a computer program

Description automatically generated

Figure 2: Loading transactions from CSV & function validation

A screen shot of a computer program

Description automatically generated

Figure 3: Brute Force & Association Rule generation

A screen shot of a computer program

Description automatically generated

Figure 4: Apriori, FP-Growth Algorithm & Comparison of all Algorithms

A screen shot of a computer program

Description automatically generated

Figure 5: Main function for User Interaction

Below screenshots show the output and user interaction:

A screenshot of a computer

Description automatically generated

Figure 6: User selects a store and enters minimum support and confidence for that store.

After entering minimum support and confidence values, we get the following:

A screenshot of a computer program

Description automatically generated

Figure 7: We get each algorithm time, fastest algorithm, association rules & item count

**GitHub Repository**

<https://github.com/sushilsaindane/saindane_sushil_midtermproj>

**References**

1. [Mlxtend library documentation](http://rasbt.github.io/mlxtend/)
2. [Brute Force](https://stackoverflow.com/questions/25899839/how-to-write-a-brute-force-algorithm)
3. [Association Rule Mining](https://towardsdatascience.com/apriori-association-rule-mining-explanation-and-python-implementation-290b42afdfc6)
4. [Github](https://github.com/Dhruvish97/Brute-Force-Rule-Mining)
5. [Perplexity](https://www.perplexity.ai/)