NAME: Vinit Walke
NJIT UCID: vnw3
Email Address:vnw3@njit.edu

Professor Yasser Abduallah
CS 634 104
Data Mining

Midterm Project Report:

Implementation and Code Usage:

Apriori Algorithm Implementation in Retail Data Mining-

Introduction

In this project, we delve into the Apriori Algorithm, a fundamental technique in data mining, to reveal associations within retail transactions. The primary goal is to implement the algorithm, employ various data mining concepts, principles, and methods, and assess its effectiveness and efficiency. By designing and developing custom data mining tools, a unique model is created to mine valuable insights from transaction data.

Core Concepts and Principles:

The Apriori Algorithm is a classic and fundamental algorithm in data mining and association rule learning. It is specifically designed for discovering frequent itemsets from transactional data. The algorithm is widely used in market basket analysis, where the goal is to find associations and relationships among items frequently bought together in transactions, such as retail purchase data.

Key Concepts:

1. Frequent Itemset - Apriori is focused on identifying sets of items that frequently co-occur in transactions.These sets are known as frequent itemsets.

2. Support:Support is a crucial metric in the Apriori Algorithm. It measures the frequency of occurrence of a particular itemset in the dataset. The algorithm identifies itemsets with support greater than a predefined threshold as frequent itemsets.

3. Association Rules:After identifying frequent itemsets, Apriori generates association rules.These rules express relationships between items based on their co-occurrence in transactions.

4. Apriori Principle: The algorithm is based on the Apriori principle, which states that if an itemset is frequent, then all of its subsets must also be frequent.This principle helps in pruning the search space, making the algorithm more efficient.

Workflow of the Apriori Algorithm:

1. Candidate Generation:The algorithm starts by identifying frequent individual items (itemsets of size 1) in the dataset.These frequent itemsets are used to generate candidate itemsets of size 2.
2. Support Counting:The support count of each candidate itemset is then calculated by scanning the transaction dataset.Candidate itemsets with support below the specified threshold are pruned.
3.Iteration:The process iterates, generating larger candidate itemsets based on the frequent ones from the previous step. The support counting and pruning steps are repeated until no more frequent itemsets can be found.
4. Association Rule Generation:
   - Once all frequent itemsets are identified,

Association rules are generated based on these frequent itemsets.
These rules provide insights into the relationships and associations among items.

Application in Retail:

In the context of retail, the Apriori Algorithm helps retailers understand customer purchasing behavior. For example, if customers frequently buy items A and B together, a retailer can use this information for targeted marketing, product placement, and other strategies to enhance the shopping experience and increase sales.

In summary, the Apriori Algorithm is a powerful tool for discovering patterns and associations within transactional data, making it particularly valuable for applications like market basket analysis in the retail industry.

Brute Force Algorithm:

Approach: The brute force approach involves generating all possible itemsets and counting their support in the dataset.

Workflow:

1. Generate all possible combinations of items (itemsets).
2. For each itemset, count the support (frequency) in the dataset.
3. Keep only the itemsets with support above a specified threshold.

FP-Growth (Frequent Pattern Growth) Algorithm:

- Approach: FP-Growth is a more efficient algorithm that adopts a divide-and-conquer strategy. It builds a compact data structure called an FP-tree.

Workflow:1. Construct an FP-tree by encoding the dataset in a compact manner.
2. Mine frequent itemsets directly from the FP-tree.

Implementation

The Apriori Algorithm is implemented, showcasing the code and any optimizations made for the retail dataset. A step-by-step walkthrough of the implementation process is provided, ensuring clarity and replicability.

Custom Data Mining Tools

The design and development of custom data mining tools are detailed, emphasizing their role in data exploration and visualization. These tools aim to enhance the efficiency of uncovering valuable insights from transaction data.

Evaluation

The effectiveness of the Apriori Algorithm is assessed using standard metrics such as support, confidence, and lift. The results are compared with alternative data mining techniques to gauge the algorithm's performance in the retail context.

Insights and Interpretation

The discovered association rules are interpreted in the context of retail, extracting actionable insights for business decision-making. Visualizations and clear explanations enhance the understanding of the findings.

Conclusion-

The report concludes by summarizing key findings, discussing the overall effectiveness and efficiency of the Apriori Algorithm in uncovering associations within retail transactions.

References-

A comprehensive list of cited literature and sources used throughout the project.

Github link-https://github.com/vinit107/apriori

Fig-1 K-mart.csv

K-Mart

Transaction ID	Transaction
Trans1	Decorative Pilos, Quits, Embroidered Bedspread
Trans2	Embroidered Bedspread, Shams, Kids Bedding, Beding Collections, Bed Skirts, Bedspreads, Sheets
Trans3	Decorative Pillows, Quilts, Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections
Trans4	Kids Beding, Bedding Collections, Shers, Bedspreads, Bed skirts
Trans5	Decorative Pillows, Kids Beeding, Beeding Collections, Sheets, Bed Skirts, Bedspreads
Trans6	Bedding Collections, Bedspreads, Bed Skirts, Sheets, Shams, Kids Bedding
Trans7	Decorative Pillows, Quilts
Trans8	Decorative Pillows, Quilts, Embroidered Bedspread
Trans9	Bedspreads, Bed Skirts, Shams, Kids Bedding, Sheets
Trans10	Quilts, Embroidered Bedspread, Bedding Collections
Trans11	Bedding Collections, Bedspreads, Bed Skirts, Kids Bedding, Shams, Sheets
Trans12	Decorative Pillows, Quilts
Trans13	Embroidered Bedspread, Shams
Trans14	Sheets, Shams, Bed Skirts, Kids Bedding
Trans15	Decorative Plows, quits
Trans16	Decorative Plows, Kids Bedling, Bed skirts, Shams
Trans17	Decorative Pillows, Shams, Bed Skirts
Trans18	Quits, Sheets, Kids Bedding
Trans19	Shams, Bed Skirts, Kids Bedding, sheets
Trans20	Decorative Pillows, Bedspreads, Shams, Sheets, Bed Skirts, Kids Bedding

Fig-2 Generic.csv

Generic

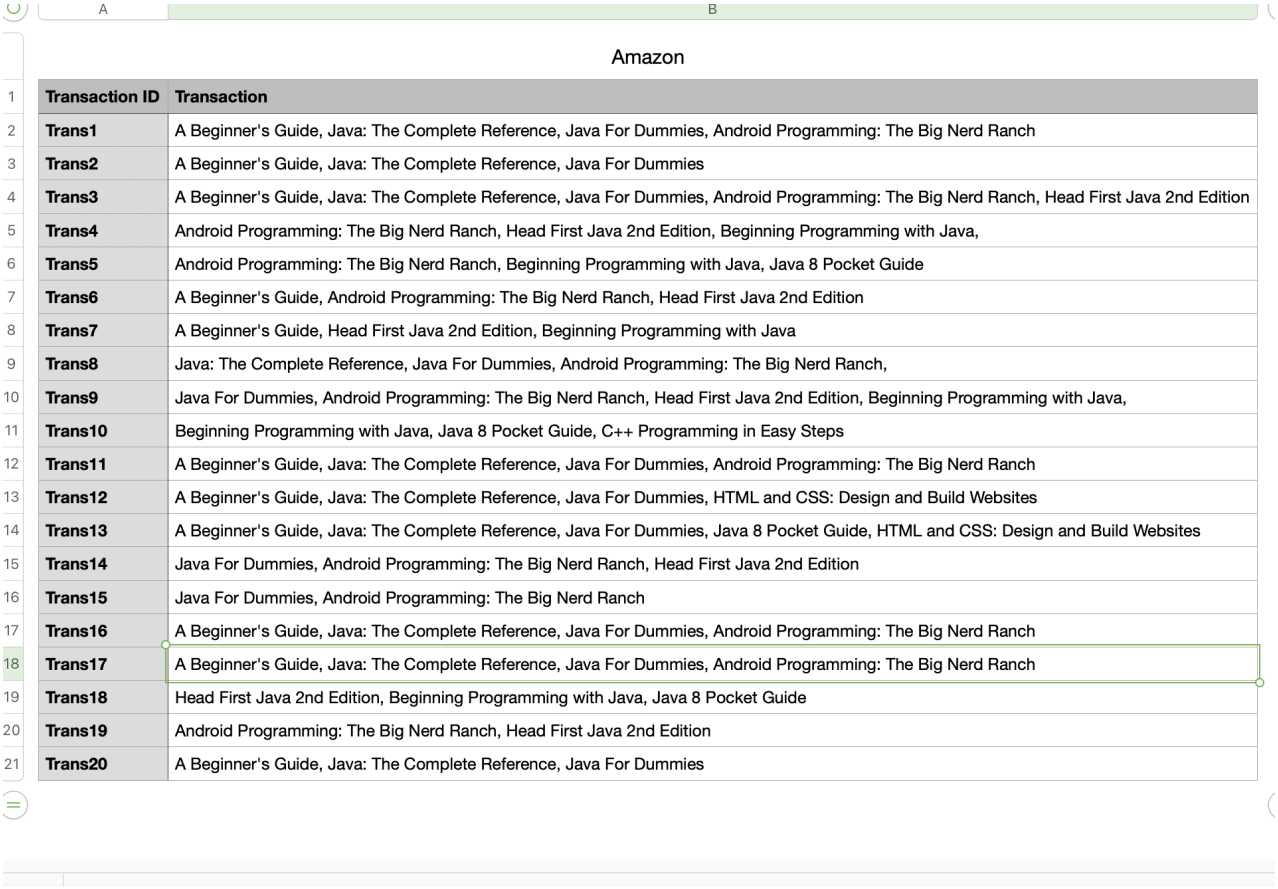
Transaction ID	Transaction
Trans1	A, B, C
Trans2	A, B, C
Trans3	A, B, C, D
Trans4	A, B, C, D, E
Trans5	A, B, D, E
Trans6	A, D, E
Trans7	A, E
Trans8	A, E
Trans9	A, C, E
Trans10	A, C, E
Trans11	A, C, E

Fig3- Best_buy.csv

Best_Buy

Transaction ID	Transaction
Transi	Desk Top, Printer, Flash Drive, Microsoft Office, Speakers, Anti-Virus
Trans2	Lab Top, flash Drive, Microsoft Ofice, la top Case, Anti-Vrus
Trans3	Lab Top, Printer, Hash Drive, Microsof Ofice, Anti Virus, Lab Top Case, Exernal Hara-Drive
Trans4	Lab Top, Printer, Flash Drive, Anti-Virus, External Hard-Drive, Lab Top Case
Trans5	Lab Top, Flash Drive, Lab Top Case, Anti-Virus
Trans6	Lab Top, Printer, Flash Drive, Microsoft Office
Trans7	Desk Top, Printer, Flash Drive, Microsoft Office
Trans8	Lab Top, External Hard-Drive, Anti-Virus
Trans9	Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, Speakers, External Hard-Drive
Trans10	Digital Camera, Lab Top, Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, External Hard-Drive, Speakers
Trans11	Lab Top, Desk Top, Lab Top Case, External Hard-Drive, Speakers, Anti-Virus
Trans12	Digital Camera, Lab Top, Lab Top Case, External Hard-Drive, Anti-Virus, Speakers
Trans13	Digital Camera, Speakers
Trans14	Digital Camera, Desk Top, Printer, Flash Drive, Microsoft Office
Trans15	Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, Speakers, External Hard-Drive
Trans16	Digital Camera, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive, Speakers
Trans17	Digital Camera, Lab Top, Lab Top Case
Trans18	Digital Camera, Lab Top Case, Speakers
Trans19	Digital Camera, Lab Top, Printer, Flash Drive, Microsoft Office, Speakers, Lab Top Case, Anti-Virus
Trans20	Digital Camera, Lab Top, Speakers, Anti-Virus, Lab Top Case

Fig4- Amazon.csv



Amazon

Transaction ID	Transaction
Trans1	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans2	A Beginner's Guide, Java: The Complete Reference, Java For Dummies
Trans3	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans4	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition, Beginning Programming with Java,
Trans5	Android Programming: The Big Nerd Ranch, Beginning Programming with Java, Java 8 Pocket Guide
Trans6	A Beginner's Guide, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans7	A Beginner's Guide, Head First Java 2nd Edition, Beginning Programming with Java
Trans8	Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch,
Trans9	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition, Beginning Programming with Java,
Trans10	Beginning Programming with Java, Java 8 Pocket Guide, C++ Programming in Easy Steps
Trans11	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans12	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, HTML and CSS: Design and Build Websites
Trans13	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Java 8 Pocket Guide, HTML and CSS: Design and Build Websites
Trans14	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans15	Java For Dummies, Android Programming: The Big Nerd Ranch
Trans16	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans17	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch
Trans18	Head First Java 2nd Edition, Beginning Programming with Java, Java 8 Pocket Guide
Trans19	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition
Trans20	A Beginner's Guide, Java: The Complete Reference, Java For Dummies

fig5-Nike.csv

Nike

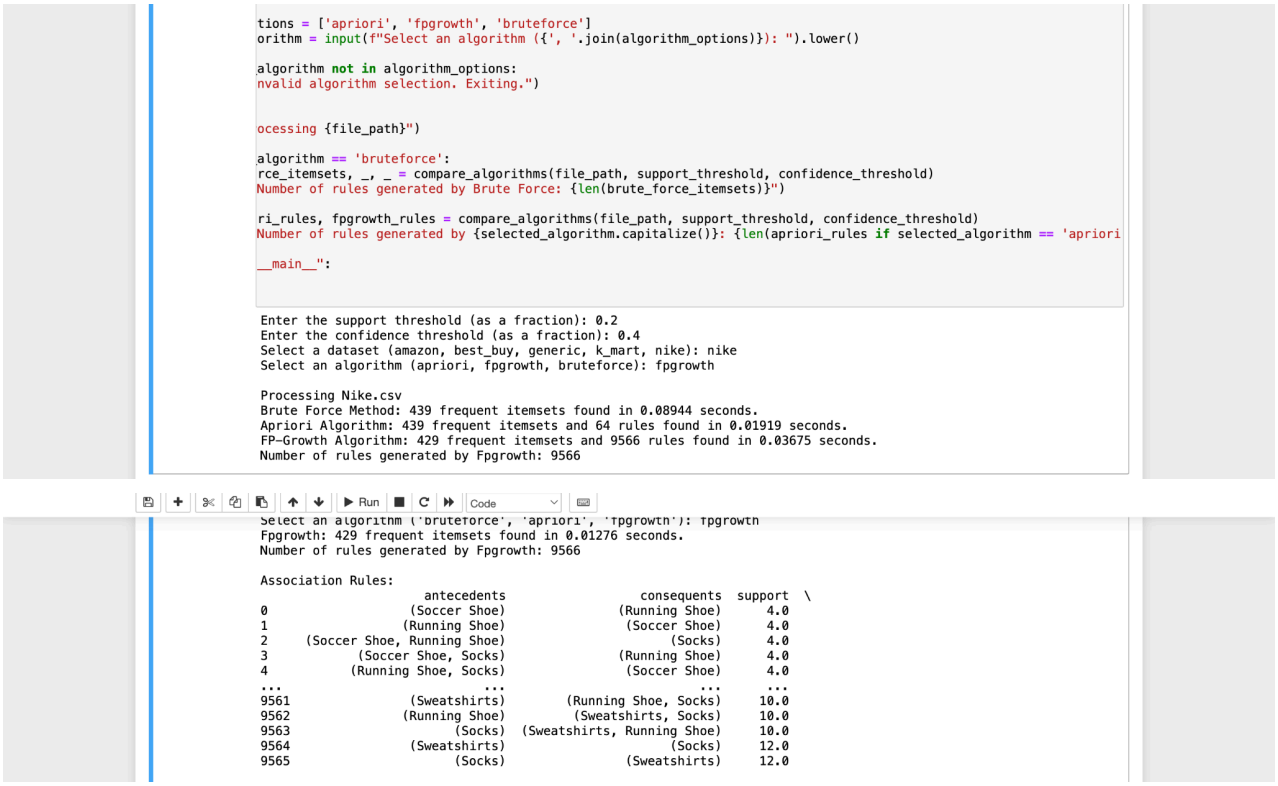
Transaction ID	Transaction
Trans1	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans2	Running Shoe, Socks, Sweatshirts
Trans3	Running Shoe, Socks, Sweatshirts, Modern Pants
Trans4	Running Shoe, Sweatshirts, Modern Pants
Trans5	Running Shoe, Socks, Sweatshirts, Modern Pants, Soccer Shoe
Trans6	Running Shoe, Socks, Sweatshirts
Trans7	Running Shoe, Socks, Sweatshirts, Modern Pants, Tech Pants, Rash Guard, Hoodies
Trans8	Swimming Shirt, Socks, Sweatshirts
Trans9	Swimming Shirt, Rash Guard, Dry Fit v-Nick, Hoodies, Tech Pants
Trans10	Swimming Shirt, Rash Guard, Dry
Trans11	Swimming Shirt, Rash Guard, Dry Fit V-Nick
Trans12	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Hoodies, Tech Pants, Dry Fit V-Nick
Trans13	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Tech Pants, Dry Fit V-Nick, Hoodies
Trans14	Running Shoe, Swimming Shirt, Rash Guard, Tech Pants, Hoodies, Dry Fit V-Nick
Trans15	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Dry Fit V-Nick, Rash Guard, Tech Pants
Trans16	Swimming Shirt, Soccer Shoe, Hoodies, Dry Fit V-Nick, Tech Pants, Rash Guard
Trans17	Running Shoe, Socks
Trans18	Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Rash Guard, Tech Pants, Dry Fit v- Nick
Trans19	Running Shoe, Swimming Shirt, Rash Guard
Trans20	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Tech Pants, Rash Guard, Dry Fit V-Nick

The code is displayed below

In [1]:
```python
import itertools
import time
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
import pyfpgrowth

# Function to read CSV file and convert it into a list of transactions
def read_transactions(file_path):
    transactions = []
    with open(file_path, 'r') as file:
        csv_reader = pd.read_csv(file)
        for index, row in csv_reader.iterrows():
            transactions.append(row['Transaction'].split(', '))
    return transactions



# Brute force method to find frequent itemsets
def brute_force(transactions, support_threshold):
    items = set(item for transaction in transactions for item in transaction)
    itemsets = []
    for i in range(1, len(items) + 1):
        itemsets.extend(itertools.combinations(items, i))
    frequent_itemsets = {}
    for itemset in itemsets:
        frequency = sum(1 for transaction in transactions if set(itemset).issubset(set(transaction)))
        if frequency / len(transactions) >= support_threshold:
            frequent_itemsets[itemset] = frequency
    return frequent_itemsets

# Function to convert list of transactions into the right format for MLxtend
def convert_to_mlxtend_format(transactions):
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    return df
```

In [2]:
```python
# Main function to orchestrate the analysis
def main():
    support_threshold = float(input("Enter the support threshold (as a fraction): "))
    confidence_threshold = float(input("Enter the confidence threshold (as a fraction): "))

    dataset_options = {
        'amazon': 'Amazon.csv',
        'best_buy': 'Best_Buy.csv',
        'generic': 'Generic.csv',
        'k_mart': 'K-Mart.csv',
        'nike': 'Nike.csv'
    }
    selected_dataset = input(f"Select a dataset ({', '.join(dataset_options.keys())}): ").lower()

    if selected_dataset not in dataset_options:
        print("Invalid dataset selection. Exiting.")
        return

    file_path = dataset_options[selected_dataset]

    algorithm_options = ['apriori', 'fpgrowth', 'bruteforce']
    selected_algorithm = input(f"Select an algorithm ({', '.join(algorithm_options)}): ").lower()

    if selected_algorithm not in algorithm_options:
        print("Invalid algorithm selection. Exiting.")
        return

    print(f"\nProcessing {file_path}")

    if selected_algorithm == 'bruteforce':
        brute_force_itemsets, _, _ = compare_algorithms(file_path, support_threshold, confidence_threshold)
        print(f"Number of rules generated by Brute Force: {len(brute_force_itemsets)}")
    else:
        _, apriori_rules, fpgrowth_rules = compare_algorithms(file_path, support_threshold, confidence_threshold)
        print(f"Number of rules generated by {selected_algorithm.capitalize()}: {len(apriori_rules if selected_algor

if __name__ == "__main__":
    main()
```

Output-



```
tions = ['apriori', 'fpgrowth', 'bruteforce']
orithm = input(f"Select an algorithm ({', '.join(algorithm_options)}): ").lower()

algorithm not in algorithm_options:
nvalid algorithm selection. Exiting.")


ocessing {file_path}")

algorithm == 'bruteforce':
rce_itemsets, _, _ = compare_algorithms(file_path, support_threshold, confidence_threshold)
Number of rules generated by Brute Force: {len(brute_force_itemsets)}")

ri_rules, fpgrowth_rules = compare_algorithms(file_path, support_threshold, confidence_threshold)
Number of rules generated by {selected_algorithm.capitalize()}: {len(apriori_rules if selected_algorithm == 'apriori

__main__":


Enter the support threshold (as a fraction): 0.2
Enter the confidence threshold (as a fraction): 0.4
Select a dataset (amazon, best_buy, generic, k_mart, nike): nike
Select an algorithm (apriori, fpgrowth, bruteforce): fpgrowth

Processing Nike.csv
Brute Force Method: 439 frequent itemsets found in 0.08944 seconds.
Apriori Algorithm: 439 frequent itemsets and 64 rules found in 0.01919 seconds.
FP-Growth Algorithm: 429 frequent itemsets and 9566 rules found in 0.03675 seconds.
Number of rules generated by Fpgrowth: 9566
```

Run Code

```
Select an algorithm ('bruteforce', 'apriori', 'fpgrowth'): fpgrowth
Fpgrowth: 429 frequent itemsets found in 0.01276 seconds.
Number of rules generated by Fpgrowth: 9566

Association Rules:
                      antecedents                 consequents  support  \
0                  (Soccer Shoe)              (Running Shoe)      4.0
1                 (Running Shoe)                (Soccer Shoe)      4.0
2     (Soccer Shoe, Running Shoe)                      (Socks)      4.0
3           (Soccer Shoe, Socks)              (Running Shoe)      4.0
4          (Running Shoe, Socks)                (Soccer Shoe)      4.0
...                          ...                         ...      ...
9561               (Sweatshirts)       (Running Shoe, Socks)     10.0
9562              (Running Shoe)        (Sweatshirts, Socks)     10.0
9563                     (Socks)  (Sweatshirts, Running Shoe)     10.0
9564               (Sweatshirts)                      (Socks)     12.0
9565                     (Socks)               (Sweatshirts)     12.0
```

To run the code and understand it:

Running the Code:

1.Install Required Packages:
- Open a terminal or command prompt.
- Run the following command to install the necessary packages:
  bash
  pip install pandas mlxtend pyfpgrowth

2. Download the Code:
- Download the provided Python script (save it with a .py extension) to your local machine.

3. Run the Code:
- Open a terminal or command prompt.

- Navigate to the directory where the script is saved.
- Run the script using the following command:
  bash
  python script_name.py

  Replace `script_name.py` with the actual name of the Python script.

4.Follow the Instructions:
- The script will prompt you to enter the support and confidence thresholds.
- Select a dataset from the available options (Amazon, Best Buy, Generic, K-Mart, Nike).
- Choose an algorithm (Apriori, FP-Growth, or Brute Force).

5. View Results:
- The script will display the number of frequent itemsets and rules found by each algorithm, along with the time taken for each method.

Code Explanation:

Reading Transactions:
- The `read_transactions` function reads a CSV file and converts it into a list of transactions.

-Brute Force Method:
- `brute_force` uses a brute-force approach to find frequent itemsets based on the support threshold.

Conversion for MLxtend:
- `convert_to_mlxtend_format` converts the list of transactions into the right format for the MLxtend library.

Apriori Algorithm:
- `run_apriori` applies the Apriori algorithm to find frequent itemsets.

FP-Growth Algorithm:
- `run_fpgrowth` uses the FP-Growth algorithm to find frequent itemsets.

Generate Association Rules:
- `generate_rules` generates association rules from frequent itemsets.

Comparing Algorithms:
- `compare_algorithms` compares the results and performance time of Brute Force, Apriori, and FP-Growth.

Main Function:
- The `main` function orchestrates the analysis, prompting the user for input and displaying the results.

This script compares the performance of three different algorithms for mining association rules on various datasets.