NAME: Emily Pham
NJIT UCID: etp6
Email Address: etp6@njit.edu
March 2nd, 2025
Professor: Yasser Abduallah
CS 634-854 Data Mining

CS634 Midterm Project Report

Data Mining for Frequent Itemsets & Association Rules: Brute-Force vs. Apriori

Implementation and Code Usage

ABSTRACT

This project explores the application of data mining techniques for discovering frequent itemsets and association rules in the transactional datasets of the retail trade industry. Two different algorithms–**Brute-Force** and **Apriori**–are implemented and compared based on performance and accuracy. The dataset consists of lists of items and transactional records from five different retail stores. By implementing each algorithm, the goal is to identify frequently purchased itemsets and generate association rules based on user-defined minimum support and minimum confidence values. The results of this project will be useful for retail businesses looking to optimize their inventory and sales strategies through data-driven insights.

INTRODUCTION

Data mining is the process of deriving patterns, correlations, and useful information from large datasets. It plays a crucial role in optimizing operations and decision-making across various industries, particularly in retail with techniques such as market basket analysis. Frequent itemset mining is a key aspect of data mining that involves identifying sets of items that commonly appear together in transactional databases. This project implements two different approaches for frequent itemset discovery and association rule generation: the Brute-Force method and the Apriori algorithm.

In this project, item and transaction datasets were designed from five different retail stores–Amazon, BestBuy, Nike, Kmart, and Walmart–and formatted into CSV files for analysis. The main concept of both these algorithms is to create associations, which was the ultimate goal of the project. There were several key steps shared by both algorithms:

1. **Dataset Selection**: Transaction data from five different stores is loaded for analysis.
2. **Preprocessing**: The datasets are cleaned and formatted for analysis by extracting item names and transaction histories.
3. **Frequent Itemset Discovery**: The algorithms iterate over transactions to determine frequent *k*-itemsets, where *k* represents the number of items in the set. This ensures that only the most relevant item combinations are considered.
4. **Association Rule Mining**: Association rules are derived based on user-specified confidence thresholds, allowing businesses to understand the relationships between items.
5. **Performance Evaluation**: The final execution time of each algorithm is recorded and printed for comparison.

The Brute-Force algorithm follows a more straightforward approach by generating all possible *k*-itemsets and evaluating each against the user-specified support threshold. Conversely, the Apriori algorithm applies an iterative approach, utilizing a process known as "pruning" to eliminate candidate itemsets that do not meet the minimum support threshold early on in the process to further enhance efficiency. It first identifies frequent 1-itemsets before iteratively generating additional (*k*+1)-itemsets only from those that clear the threshold.

The key difference between the two approaches lies in efficiency. While Brute-Force is exhaustive and guarantees completeness, it suffers from exponential growth in processing time as the dataset scales upwards. On the other hand, Apriori's pruning techniques greatly optimize the process by eliminating unnecessary computations, allowing for a faster and more scalable solution. The project works to demonstrate how different methods can be used to extract useful insights from transaction data, supporting data-driven decision-making in not just retail, but in other industries across the board.

CORE CONCEPTS AND PRINCIPLES

Frequent Itemset Discovery

Frequent itemset discovery is the process of identifying sets of items that appear together in transactions with a frequency above a specified threshold. This provides insight into customer purchase behavior and preferences, thereby proving essential for applications such as customer recommendation systems and inventory management.

Support and Confidence

Data mining revolves around two key metrics: support and confidence. Support measures the frequency at which an itemset appears in a dataset, while confidence measures the likelihood that a single item in a frequent itemset appears in a transaction when given the presence of another item (i.e. the probability that two items will be purchased jointly).

Association Rules

Association rules define relationships between items in transactions. By identifying strong association rules, the program is able to determine which items are frequently purchased together.

RESULTS AND EVALUATIONS

The performance difference between both algorithms was analyzed in terms of execution time and number of generated association rules. The Brute-Force algorithm was observed to produce accurate results, but with a significantly higher computational time as the numbers of items/transactions to be analyzed in the dataset increased. The Apriori algorithm was determined to be more efficient in comparison due to its pruning step, however it was noted to be unable to generate as many rules as the Brute-Force method for datasets that were smaller in size.

CONCLUSION

In conclusion, this project demonstrates the application of two prominent data mining techniques, the Brute-Force method and the Apriori algorithm, for determining frequent itemsets and generating strong association rules. Though the Brute-Force method is more thorough in its analysis, it becomes computationally bottlenecked for larger datasets. The Apriori algorithm provides a more scalable solution by reducing the number of candidate itemsets, thereby significantly reducing computational overhead and increasing efficiency. But despite the differences, both approaches play a crucial role in data mining to reveal valuable patterns and correlations for decision-making in the retail industry.

1) CSV FILES

This is an example of the CSV files used in this program. There are two separate CSV files for each store, detailing its item names (*Screenshot 1.1*) and transactions (*Screenshot 1.2*).

amazon_items

Item ID	Item Name
1	A Beginner's Guide
2	Java: The Complete Reference
3	Java For Dummies
4	Android Programming: The Big Nerd Ranch
5	Head First Java 2nd Edition
6	Beginning Programming with Java
7	Java 8 Pocket Guide
8	C++ Programming in Easy Steps
9	Effective Java (2nd Edition)
10	HTML and CSS: Design and Build Websites

Screenshot 1.1 Amazon Item Names CSV File

amazon_transactions

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	Head First Java 2nd Edition,Java For Dummies,C++ Programming in Easy Steps,A Beginner's Guide
Trans12	Android Programming: The Big Nerd Ranch,Java 8 Pocket Guide,Java For Dummies,Beginning Programming with Java,C++ Programming in Easy Steps
Trans13	Java For Dummies,HTML and CSS: Design and Build Websites,Head First Java 2nd Edition,Android Programming: The Big Nerd Ranch,A Beginner's Guide
Trans14	Beginning Programming with Java,Java 8 Pocket Guide,Effective Java (2nd Edition),HTML and CSS: Design and Build Websites,Android Programming: The Big Nerd Ranch,Head First Java 2nd Edition
Trans15	Java: The Complete Reference,HTML and CSS: Design and Build Websites,Java 8 Pocket Guide,Android Programming: The Big Nerd Ranch,C++ Programming in Easy Steps
Trans16	Java: The Complete Reference,Effective Java (2nd Edition),C++ Programming in Easy Steps,Java For Dummies,Android Programming: The Big Nerd Ranch
Trans17	C++ Programming in Easy Steps,Effective Java (2nd Edition),Java For Dummies
Trans18	Head First Java 2nd Edition,Java: The Complete Reference,Java 8 Pocket Guide,HTML and CSS: Design and Build Websites,Android Programming: The Big Nerd Ranch,Beginning Programming with Java
Trans19	Head First Java 2nd Edition,C++ Programming in Easy Steps,Java: The Complete Reference,A Beginner's Guide
Trans20	Effective Java (2nd Edition),Head First Java 2nd Edition,C++ Programming in Easy Steps,Java For Dummies,Java: The Complete Reference

Screenshot 1.2. Amazon Transactions CSV File

2) CODE

The following screenshots are of the code from the Python file.

Welcome Message and User-Input Functions

The program first welcomes the user, then prompts him/her to select a store from the given list, or to quit the program altogether. After ensuring the validity of the user's input, the program locates the proper CSV files and loads it. To reduce computational overhead, a limit was set for the max number of items per transaction. The user is then asked to input minimum support and minimum confidence values.

```python
import pandas as pd
import time
from itertools import combinations
from mlxtend.frequent_patterns import apriori, association_rules

# Welcome message
print("Welcome to the Frequent Itemset Mining and Association Rule Generation Program!")
selected_store = input("Please select your store:\n1. Amazon\n2. BestBuy\n3. Nike\n4. Kmart\n5. Walmart\n6. Quit program\n")
if selected_store == '6':
    quit()

stores = ['Amazon', 'BestBuy', 'Nike', 'Kmart', 'Walmart']

# Verify input
try:
    selected_store = int(selected_store)
    if selected_store < 1 or selected_store > len(stores):
        print("Invalid store selection. Please enter a valid number.")
        quit()
except ValueError:
    print("Invalid input. Please enter a valid number.")
    quit()

store_name = stores[selected_store - 1]

# Load datasets based on user-selected store
df_tr = pd.read_csv(f"csv files/{store_name}_transactions.csv")
df_items = pd.read_csv(f"csv files/{store_name}_items.csv")

print(f"\nYou have selected {store_name}!\n")
print("\nLoaded Items:")
print(df_items, "\n")

# Set a limit for max items per transaction
MAX_ITEMS_PER_TRANSACTION = 5

# Prepare transactions with item limitation
transactions = df_tr['Transaction'].dropna().apply(lambda x: sorted(set(x.split(',')))[:MAX_ITEMS_PER_TRANSACTION]).tolist()

for i, transaction in enumerate(transactions, 1):
    print(f"Transaction {i}: {transaction}")

# User input for support and confidence thresholds
minimum_support = int(input("\nPlease enter a Minimum Support value (1 to 100): "))
minimum_confidence = int(input("Please enter a Minimum Confidence value (1 to 100): "))
```

Screenshot 2.1. Welcome Message and User-Input Functions

minSupCount & minConfidence

The user-specified minimum support and minimum confidence values are divided by 100 for conversion into absolute counts. For minimum support, the resulting value is then multiplied by the total number of transactions in the dataset; this is to determine how many times an itemset must appear in a dataset to be considered frequent.

```python
minSupCount = (minimum_support / 100) * len(transactions)
minConfidence = minimum_confidence / 100
```

Screenshot 2.2. minSupCount & minConfidence

Brute-Force Algorithm

```python
# Brute-Force Algorithm
def run_brute_force():
    print("\nInitiating Brute-Force Algorithm...\n")
    start_time = time.time()

    def count_occurrences(itemset, transactions):
        return sum(1 for transaction in transactions if set(itemset).issubset(transaction))

    all_items = sorted(set(item for transaction in transactions for item in transaction))
    frequent_itemsets = {}

    for item in all_items:
        support = count_occurrences([item], transactions)
        if support >= minSupCount:
            frequent_itemsets[(item,)] = support / len(transactions)

    k = 2
    while True:
        candidate_itemsets = list(combinations(frequent_itemsets.keys(), k))
        candidate_itemsets = [tuple(sorted(set().union(*itemset))) for itemset in candidate_itemsets]

        new_frequent_itemsets = {}
        for itemset in candidate_itemsets:
            support = count_occurrences(itemset, transactions)
            if support >= minSupCount:
                new_frequent_itemsets[itemset] = support / len(transactions)
        if not new_frequent_itemsets:
            break

        frequent_itemsets.update(new_frequent_itemsets)
        k += 1

    print("\nFrequent Itemsets:")
    if not frequent_itemsets:
        print("No frequent itemsets found with the given support threshold.")
    else:
        for idx, (itemset, support) in enumerate(frequent_itemsets.items(), 1):
            print(f"Itemset {idx}: {list(itemset)}, Support: {support:.2f}")
```

Screenshot 2.3. Brute-Force Algorithm–Initialization & Frequent Itemset Discovery

```python
association_rules_list = []
for itemset, support in frequent_itemsets.items():
    for i in range(1, len(itemset)):
        for antecedent in combinations(itemset, i):
            consequent = tuple(set(itemset) - set(antecedent))
            antecedent_support = count_occurrences(antecedent, transactions) / len(transactions)
            confidence = support / antecedent_support if antecedent_support > 0 else 0
            if confidence >= minConfidence:
                association_rules_list.append((antecedent, consequent, confidence, support))

print("\nGenerated Association Rules (Brute-Force):")
num_rules = len(association_rules_list)

if not association_rules_list:
    print("No association rules found with the given confidence threshold.")
else:
    for i, (antecedent, consequent, confidence, support) in enumerate(association_rules_list, 1):
        print(f"Rule {i}: {list(antecedent)} -> {list(consequent)}")
        print(f"Confidence: {confidence * 100:.2f}%")
        print(f"Support: {support * 100:.2f}%\n")

execution_time = round(time.time() - start_time, 4)
return num_rules, execution_time
```

Screenshot 2.4. Brute-Force Algorithm–Association Rules & Execution Time

Apriori Algorithm

```
def apriori_algorithm():
    print("\nInitiating Apriori Algorithm...")
    start_time = time.time()
    all_items = sorted(set(item for transaction in transactions for item in transaction))
    encoded_data = pd.DataFrame([{item: (item in transaction) for item in all_items} for transaction in transactions])

    frequent_itemsets = apriori(encoded_data, min_support=minimum_support / 100, use_colnames=True)

    if frequent_itemsets.empty:
        print("\nNo frequent itemsets found with the given support threshold.")
    else:
        print("\nFrequent Itemsets (Apriori):")
        for i, row in frequent_itemsets.iterrows():
            print(f"Itemset {i + 1}: {list(row['itemsets'])}, Support: {row['support']:.2f}")

    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=minConfidence)
    num_rules = len(rules)

    print("\nGenerated Association Rules (Apriori):")
    if rules.empty:
        print("No association rules found with the given confidence threshold.")
    else:
        for i, row in rules.iterrows():
            print(f"Rule {i + 1}: {list(row['antecedents'])} -> {list(row['consequents'])}")
            print(f"Confidence: {row['confidence'] * 100:.2f}%")
            print(f"Support: {row['support'] * 100:.2f}%\n")

    execution_time = round(time.time() - start_time, 4)
    return num_rules, execution_time
```

Screenshot 2.5. Apriori Algorithm

Wrapper Function

Executes both algorithms with all user-specified inputs/parameters when called, and prints a complete summary of both algorithms at completion.

```
def run_algorithms():
    brute_force_rules, brute_force_time = brute_force_algorithm()
    apriori_rules, apriori_time = apriori_algorithm()

    print("\n" + "="*40)
    print("FINAL SUMMARY\n")
    print(f"Brute-Force Algorithm:")
    print(f"  - Total Rules Generated: {brute_force_rules}")
    print(f"  - Execution Time: {brute_force_time} seconds\n")

    print(f"Apriori Algorithm:")
    print(f"  - Total Rules Generated: {apriori_rules}")
    print(f"  - Execution Time: {apriori_time} seconds")
    print("="*40)

run_algorithms()
```

Screenshot 2.6. Wrapper Function

3. OUTPUT

The following screenshots are to demonstrate performance of the program in the Terminal.

Welcome Message & Store Selection

```
Welcome to the Frequent Itemset Mining and Association Rule Generation Program!
Please select your store:
1. Amazon
2. BestBuy
3. Nike
4. Kmart
5. Walmart
6. Quit program
```

Screenshot 3.1. Welcome Message & Store Selection

Example #1 – Store = Amazon; MinSupp/MinConf = 20/80

```
You have selected Amazon!


Loaded Items:
   Item ID                            Item Name
0        1                 A Beginner's Guide
1        2          Java: The Complete Reference
2        3                     Java For Dummies
3        4  Android Programming: The Big Nerd Ranch
4        5            Head First Java 2nd Edition
5        6         Beginning Programming with Java
6        7                    Java 8 Pocket Guide
7        8           C++ Programming in Easy Steps
8        9            Effective Java (2nd Edition)
9       10  HTML and CSS: Design and Build Websites

Transaction 1: ['A Beginner's Guide', 'Android Programming: The Big Nerd Ranch', 'Java For Dummies', 'Java: The Complete Reference']
Transaction 2: ['A Beginner's Guide', 'Java For Dummies', 'Java: The Complete Reference']
Transaction 3: ['A Beginner's Guide', 'Android Programming: The Big Nerd Ranch', 'Head First Java 2nd Edition', 'Java For Dummies', 'Java: The Complete Reference']
Transaction 4: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'Head First Java 2nd Edition']
Transaction 5: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'Java 8 Pocket Guide']
Transaction 6: ['A Beginner's Guide', 'Android Programming: The Big Nerd Ranch', 'Head First Java 2nd Edition']
Transaction 7: ['A Beginner's Guide', 'Beginning Programming with Java', 'Head First Java 2nd Edition']
Transaction 8: ['Android Programming: The Big Nerd Ranch', 'Java For Dummies', 'Java: The Complete Reference']
Transaction 9: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'Head First Java 2nd Edition', 'Java For Dummies']
Transaction 10: ['Beginning Programming with Java', 'C++ Programming in Easy Steps', 'Java 8 Pocket Guide']
Transaction 11: ['A Beginner's Guide', 'C++ Programming in Easy Steps', 'Head First Java 2nd Edition', 'Java For Dummies']
Transaction 12: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'C++ Programming in Easy Steps', 'Java 8 Pocket Guide', 'Java For Dummies']
Transaction 13: ['A Beginner's Guide', 'Android Programming: The Big Nerd Ranch', 'HTML and CSS: Design and Build Websites', 'Head First Java 2nd Edition', 'Java For Dummies']
Transaction 14: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'Effective Java (2nd Edition)', 'HTML and CSS: Design and Build Websites', 'Head First Java 2nd Edition']
Transaction 15: ['Android Programming: The Big Nerd Ranch', 'C++ Programming in Easy Steps', 'HTML and CSS: Design and Build Websites', 'Java 8 Pocket Guide', 'Java: The Complete Reference']
Transaction 16: ['Android Programming: The Big Nerd Ranch', 'C++ Programming in Easy Steps', 'Effective Java (2nd Edition)', 'Java For Dummies', 'Java: The Complete Reference']
Transaction 17: ['C++ Programming in Easy Steps', 'Effective Java (2nd Edition)', 'Java For Dummies']
Transaction 18: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'HTML and CSS: Design and Build Websites', 'Head First Java 2nd Edition', 'Java 8 Pocket Guide']
Transaction 19: ['A Beginner's Guide', 'C++ Programming in Easy Steps', 'Head First Java 2nd Edition', 'Java: The Complete Reference']
Transaction 20: ['C++ Programming in Easy Steps', 'Effective Java (2nd Edition)', 'Head First Java 2nd Edition', 'Java For Dummies', 'Java: The Complete Reference']

Please enter a Minimum Support value (1 to 100): 20
Please enter a Minimum Confidence value (1 to 100): 80
```

Screenshot 3.2.1–Loaded Dataset & MinSup/MinConf Input

```
Initiating Brute-Force Algorithm...


Frequent Itemsets (Brute-Force):
Itemset 1: ['A Beginner's Guide'], Support: 0.40
Itemset 2: ['Android Programming: The Big Nerd Ranch'], Support: 0.65
Itemset 3: ['Beginning Programming with Java'], Support: 0.40
Itemset 4: ['C++ Programming in Easy Steps'], Support: 0.40
Itemset 5: ['Effective Java (2nd Edition)'], Support: 0.20
Itemset 6: ['HTML and CSS: Design and Build Websites'], Support: 0.20
Itemset 7: ['Head First Java 2nd Edition'], Support: 0.55
Itemset 8: ['Java 8 Pocket Guide'], Support: 0.25
Itemset 9: ['Java For Dummies'], Support: 0.55
Itemset 10: ['Java: The Complete Reference'], Support: 0.40
Itemset 11: ['A Beginner's Guide', 'Android Programming: The Big Nerd Ranch'], Support: 0.20
Itemset 12: ['A Beginner's Guide', 'Head First Java 2nd Edition'], Support: 0.30
Itemset 13: ['A Beginner's Guide', 'Java For Dummies'], Support: 0.25
Itemset 14: ['A Beginner's Guide', 'Java: The Complete Reference'], Support: 0.20
Itemset 15: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java'], Support: 0.30
Itemset 16: ['Android Programming: The Big Nerd Ranch', 'HTML and CSS: Design and Build Websites'], Support: 0.20
Itemset 17: ['Android Programming: The Big Nerd Ranch', 'Head First Java 2nd Edition'], Support: 0.35
Itemset 18: ['Android Programming: The Big Nerd Ranch', 'Java 8 Pocket Guide'], Support: 0.20
Itemset 19: ['Android Programming: The Big Nerd Ranch', 'Java For Dummies'], Support: 0.35
Itemset 20: ['Android Programming: The Big Nerd Ranch', 'Java: The Complete Reference'], Support: 0.25
Itemset 21: ['Beginning Programming with Java', 'Head First Java 2nd Edition'], Support: 0.25
Itemset 22: ['Beginning Programming with Java', 'Java 8 Pocket Guide'], Support: 0.20
Itemset 23: ['C++ Programming in Easy Steps', 'Java For Dummies'], Support: 0.25
Itemset 24: ['C++ Programming in Easy Steps', 'Java: The Complete Reference'], Support: 0.20
Itemset 25: ['Head First Java 2nd Edition', 'Java For Dummies'], Support: 0.25
Itemset 26: ['Java For Dummies', 'Java: The Complete Reference'], Support: 0.30
Itemset 27: ['Android Programming: The Big Nerd Ranch', 'Beginning Programming with Java', 'Head First Java 2nd Edition'], Support: 0.20
Itemset 28: ['Android Programming: The Big Nerd Ranch', 'Java For Dummies', 'Java: The Complete Reference'], Support: 0.20

Generated Association Rules (Brute-Force):
Rule 1: ['HTML and CSS: Design and Build Websites'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 100.00%
Support: 20.00%

Rule 2: ['Java 8 Pocket Guide'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 80.00%
Support: 20.00%

Rule 3: ['Java 8 Pocket Guide'] -> ['Beginning Programming with Java']
Confidence: 80.00%
Support: 20.00%

Rule 4: ['Beginning Programming with Java', 'Head First Java 2nd Edition'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 80.00%
Support: 20.00%

Rule 5: ['Android Programming: The Big Nerd Ranch', 'Java: The Complete Reference'] -> ['Java For Dummies']
Confidence: 80.00%
Support: 20.00%
```

Screenshot 3.2.2–Brute Force Algorithm

```
Initiating Apriori Algorithm...

Frequent Itemsets (Apriori):
Itemset 1: ['A Beginner's Guide'], Support: 0.40
Itemset 2: ['Android Programming: The Big Nerd Ranch'], Support: 0.65
Itemset 3: ['Beginning Programming with Java'], Support: 0.40
Itemset 4: ['C++ Programming in Easy Steps'], Support: 0.40
Itemset 5: ['Effective Java (2nd Edition)'], Support: 0.20
Itemset 6: ['HTML and CSS: Design and Build Websites'], Support: 0.20
Itemset 7: ['Head First Java 2nd Edition'], Support: 0.55
Itemset 8: ['Java 8 Pocket Guide'], Support: 0.25
Itemset 9: ['Java For Dummies'], Support: 0.55
Itemset 10: ['Java: The Complete Reference'], Support: 0.40
Itemset 11: ['Android Programming: The Big Nerd Ranch', 'A Beginner's Guide'], Support: 0.20
Itemset 12: ['Head First Java 2nd Edition', 'A Beginner's Guide'], Support: 0.30
Itemset 13: ['A Beginner's Guide', 'Java For Dummies'], Support: 0.25
Itemset 14: ['A Beginner's Guide', 'Java: The Complete Reference'], Support: 0.20
Itemset 15: ['Beginning Programming with Java', 'Android Programming: The Big Nerd Ranch'], Support: 0.30
Itemset 16: ['HTML and CSS: Design and Build Websites', 'Android Programming: The Big Nerd Ranch'], Support: 0.20
Itemset 17: ['Head First Java 2nd Edition', 'Android Programming: The Big Nerd Ranch'], Support: 0.35
Itemset 18: ['Java 8 Pocket Guide', 'Android Programming: The Big Nerd Ranch'], Support: 0.20
Itemset 19: ['Android Programming: The Big Nerd Ranch', 'Java For Dummies'], Support: 0.35
Itemset 20: ['Android Programming: The Big Nerd Ranch', 'Java: The Complete Reference'], Support: 0.25
Itemset 21: ['Beginning Programming with Java', 'Head First Java 2nd Edition'], Support: 0.25
Itemset 22: ['Beginning Programming with Java', 'Java 8 Pocket Guide'], Support: 0.20
Itemset 23: ['Java For Dummies', 'C++ Programming in Easy Steps'], Support: 0.25
Itemset 24: ['C++ Programming in Easy Steps', 'Java: The Complete Reference'], Support: 0.20
Itemset 25: ['Head First Java 2nd Edition', 'Java For Dummies'], Support: 0.25
Itemset 26: ['Java For Dummies', 'Java: The Complete Reference'], Support: 0.30
Itemset 27: ['Beginning Programming with Java', 'Head First Java 2nd Edition', 'Android Programming: The Big Nerd Ranch'], Support: 0.20
Itemset 28: ['Android Programming: The Big Nerd Ranch', 'Java For Dummies', 'Java: The Complete Reference'], Support: 0.20

Generated Association Rules (Apriori):
Rule 1: ['HTML and CSS: Design and Build Websites'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 100.00%
Support: 20.00%

Rule 2: ['Java 8 Pocket Guide'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 80.00%
Support: 20.00%

Rule 3: ['Java 8 Pocket Guide'] -> ['Beginning Programming with Java']
Confidence: 80.00%
Support: 20.00%

Rule 4: ['Beginning Programming with Java', 'Head First Java 2nd Edition'] -> ['Android Programming: The Big Nerd Ranch']
Confidence: 80.00%
Support: 20.00%

Rule 5: ['Android Programming: The Big Nerd Ranch', 'Java: The Complete Reference'] -> ['Java For Dummies']
Confidence: 80.00%
Support: 20.00%
```

Screenshot 3.2.3–Apriori Algorithm

```
==========================================
FINAL SUMMARY

Brute-Force Algorithm:
  - Total Rules Generated: 5
  - Execution Time: 28.1868 seconds

Apriori Algorithm:
  - Total Rules Generated: 5
  - Execution Time: 0.0031 seconds
==========================================
```

Screenshot 3.2.4–Final Summary

Example #2 – Store = Best Buy; MinSupp/MinConf = 40/75

```
You have selected BestBuy!


Loaded Items:
   Item #           Item Name
0       1       Digital Camera
1       2             Lab Top
2       3            Desk Top
3       4             Printer
4       5         Flash Drive
5       6     Microsoft Office
6       7            Speakers
7       8        Lab Top Case
8       9          Anti-Virus
9      10  External Hard-Drive

Transaction 1: ['Anti-Virus', 'Desk Top', 'Flash Drive', 'Microsoft Office', 'Printer']
Transaction 2: ['Anti-Virus', 'Flash Drive', 'Lab Top', 'Lab Top Case', 'Microsoft Office']
Transaction 3: ['Anti-Virus', 'External Hard-Drive', 'Flash Drive', 'Lab Top', 'Lab Top Case']
Transaction 4: ['Anti-Virus', 'External Hard-Drive', 'Flash Drive', 'Lab Top', 'Lab Top Case']
Transaction 5: ['Anti-Virus', 'Flash Drive', 'Lab Top', 'Lab Top Case']
Transaction 6: ['Flash Drive', 'Lab Top', 'Microsoft Office', 'Printer']
Transaction 7: ['Desk Top', 'Flash Drive', 'Microsoft Office', 'Printer']
Transaction 8: ['Anti-Virus', 'External Hard-Drive', 'Lab Top']
Transaction 9: ['Anti-Virus', 'Desk Top', 'External Hard-Drive', 'Flash Drive', 'Lab Top Case']
Transaction 10: ['Anti-Virus', 'Desk Top', 'Digital Camera', 'External Hard-Drive', 'Flash Drive']
Transaction 11: ['Flash Drive', 'Lab Top Case', 'Microsoft Office', 'Printer']
Transaction 12: ['Desk Top', 'Digital Camera', 'Speakers']
Transaction 13: ['Lab Top Case', 'Microsoft Office', 'Printer', 'Speakers']
Transaction 14: ['Anti-Virus', 'Flash Drive', 'Lab Top Case', 'Microsoft Office', 'Printer']
Transaction 15: ['Anti-Virus', 'External Hard-Drive', 'Flash Drive', 'Printer']
Transaction 16: ['Anti-Virus', 'Digital Camera', 'Flash Drive', 'Lab Top']
Transaction 17: ['Anti-Virus', 'Desk Top', 'Digital Camera', 'External Hard-Drive', 'Lab Top']
Transaction 18: ['Anti-Virus', 'Desk Top', 'Lab Top', 'Printer']
Transaction 19: ['Desk Top', 'Flash Drive', 'Lab Top', 'Lab Top Case', 'Microsoft Office']
Transaction 20: ['Anti-Virus', 'External Hard-Drive', 'Flash Drive', 'Lab Top', 'Printer']

Please enter a Minimum Support value (1 to 100): 40
Please enter a Minimum Confidence value (1 to 100): 75
```

Screenshot 3.3.1–Loaded Dataset & MinSup/MinConf Input

```
Initiating Brute-Force Algorithm...


Frequent Itemsets (Brute-Force):
Itemset 1: ['Anti-Virus'], Support: 0.70
Itemset 2: ['Desk Top'], Support: 0.40
Itemset 3: ['External Hard-Drive'], Support: 0.40
Itemset 4: ['Flash Drive'], Support: 0.75
Itemset 5: ['Lab Top'], Support: 0.55
Itemset 6: ['Lab Top Case'], Support: 0.45
Itemset 7: ['Microsoft Office'], Support: 0.40
Itemset 8: ['Printer'], Support: 0.45
Itemset 9: ['Anti-Virus', 'External Hard-Drive'], Support: 0.40
Itemset 10: ['Anti-Virus', 'Flash Drive'], Support: 0.55
Itemset 11: ['Anti-Virus', 'Lab Top'], Support: 0.45
Itemset 12: ['Flash Drive', 'Lab Top'], Support: 0.40
Itemset 13: ['Flash Drive', 'Lab Top Case'], Support: 0.40

Generated Association Rules (Brute-Force):
Rule 1: ['External Hard-Drive'] -> ['Anti-Virus']
Confidence: 100.00%
Support: 40.00%

Rule 2: ['Anti-Virus'] -> ['Flash Drive']
Confidence: 78.57%
Support: 55.00%

Rule 3: ['Lab Top'] -> ['Anti-Virus']
Confidence: 81.82%
Support: 45.00%

Rule 4: ['Lab Top Case'] -> ['Flash Drive']
Confidence: 88.89%
Support: 40.00%
```

Screenshot 3.3.2–Brute Force Algorithm

```
Initiating Apriori Algorithm...

Frequent Itemsets (Apriori):
Itemset 1: ['Anti-Virus'], Support: 0.70
Itemset 2: ['Desk Top'], Support: 0.40
Itemset 3: ['External Hard-Drive'], Support: 0.40
Itemset 4: ['Flash Drive'], Support: 0.75
Itemset 5: ['Lab Top'], Support: 0.55
Itemset 6: ['Lab Top Case'], Support: 0.45
Itemset 7: ['Microsoft Office'], Support: 0.40
Itemset 8: ['Printer'], Support: 0.45
Itemset 9: ['External Hard-Drive', 'Anti-Virus'], Support: 0.40
Itemset 10: ['Flash Drive', 'Anti-Virus'], Support: 0.55
Itemset 11: ['Lab Top', 'Anti-Virus'], Support: 0.45
Itemset 12: ['Lab Top', 'Flash Drive'], Support: 0.40
Itemset 13: ['Flash Drive', 'Lab Top Case'], Support: 0.40

Generated Association Rules (Apriori):
Rule 1: ['External Hard-Drive'] -> ['Anti-Virus']
Confidence: 100.00%
Support: 40.00%

Rule 2: ['Anti-Virus'] -> ['Flash Drive']
Confidence: 78.57%
Support: 55.00%

Rule 3: ['Lab Top'] -> ['Anti-Virus']
Confidence: 81.82%
Support: 45.00%

Rule 4: ['Lab Top Case'] -> ['Flash Drive']
Confidence: 88.89%
Support: 40.00%
```

Screenshot 3.3.3–Apriori Algorithm

```
========================================
FINAL SUMMARY

Brute-Force Algorithm:
  - Total Rules Generated: 4
  - Execution Time: 0.019 seconds

Apriori Algorithm:
  - Total Rules Generated: 4
  - Execution Time: 0.0065 seconds
========================================
```

Screenshot 3.2.4–Final Summary

Example #3 – Store = Walmart; MinSupp/MinConf = 20/20

```
You have selected Walmart!


Loaded Items:
   Item #          Item Name
0       1       Toilet Paper
1       2  Laundry Detergent
2       3          Dish Soap
3       4            Shampoo
4       5         Toothpaste
5       6             Cereal
6       7      Bottled Water
7       8              Bread
8       9               Eggs
9      10               Milk

Transaction 1: ['Shampoo', 'Toilet Paper', 'Toothpaste']
Transaction 2: ['Cereal', 'Dish Soap', 'Milk', 'Shampoo']
Transaction 3: ['Dish Soap', 'Laundry Detergent', 'Toilet Paper']
Transaction 4: ['Bottled Water', 'Bread', 'Eggs']
Transaction 5: ['Dish Soap', 'Laundry Detergent', 'Shampoo']
Transaction 6: ['Cereal', 'Shampoo', 'Toothpaste']
Transaction 7: ['Cereal', 'Dish Soap', 'Laundry Detergent', 'Shampoo', 'Toilet Paper']
Transaction 8: ['Bottled Water', 'Bread', 'Eggs']
Transaction 9: ['Dish Soap', 'Laundry Detergent', 'Toilet Paper']
Transaction 10: ['Bottled Water', 'Cereal', 'Dish Soap', 'Laundry Detergent', 'Shampoo']
Transaction 11: ['Laundry Detergent', 'Milk', 'Toilet Paper']
Transaction 12: ['Cereal', 'Dish Soap', 'Eggs', 'Milk', 'Toilet Paper']
Transaction 13: ['Bottled Water', 'Dish Soap', 'Eggs', 'Laundry Detergent', 'Milk']
Transaction 14: ['Bottled Water', 'Bread', 'Eggs']
Transaction 15: ['Bottled Water', 'Dish Soap', 'Laundry Detergent', 'Milk']
Transaction 16: ['Bread', 'Laundry Detergent', 'Toothpaste']
Transaction 17: ['Bottled Water', 'Cereal', 'Laundry Detergent']
Transaction 18: ['Bottled Water', 'Bread', 'Eggs', 'Milk', 'Shampoo']
Transaction 19: ['Cereal', 'Laundry Detergent', 'Toilet Paper']
Transaction 20: ['Cereal', 'Eggs', 'Toilet Paper', 'Toothpaste']

Please enter a Minimum Support value (1 to 100): 20
Please enter a Minimum Confidence value (1 to 100): 20
```

Screenshot 3.4.1–Loaded Dataset & MinSup/MinConf Input

```
Initiating Brute-Force Algorithm...


Frequent Itemsets (Brute-Force):
Itemset 1: ['Bottled Water'], Support: 0.40
Itemset 2: ['Bread'], Support: 0.25
Itemset 3: ['Cereal'], Support: 0.40
Itemset 4: ['Dish Soap'], Support: 0.45
Itemset 5: ['Eggs'], Support: 0.35
Itemset 6: ['Laundry Detergent'], Support: 0.55
Itemset 7: ['Milk'], Support: 0.30
Itemset 8: ['Shampoo'], Support: 0.35
Itemset 9: ['Toilet Paper'], Support: 0.40
Itemset 10: ['Toothpaste'], Support: 0.20
Itemset 11: ['Bottled Water', 'Bread'], Support: 0.20
Itemset 12: ['Bottled Water', 'Eggs'], Support: 0.25
Itemset 13: ['Bottled Water', 'Laundry Detergent'], Support: 0.20
Itemset 14: ['Bread', 'Eggs'], Support: 0.20
Itemset 15: ['Cereal', 'Dish Soap'], Support: 0.20
Itemset 16: ['Cereal', 'Laundry Detergent'], Support: 0.20
Itemset 17: ['Cereal', 'Shampoo'], Support: 0.20
Itemset 18: ['Cereal', 'Toilet Paper'], Support: 0.20
Itemset 19: ['Dish Soap', 'Laundry Detergent'], Support: 0.35
Itemset 20: ['Dish Soap', 'Milk'], Support: 0.20
Itemset 21: ['Dish Soap', 'Shampoo'], Support: 0.20
Itemset 22: ['Dish Soap', 'Toilet Paper'], Support: 0.20
Itemset 23: ['Laundry Detergent', 'Toilet Paper'], Support: 0.25
Itemset 24: ['Bottled Water', 'Bread', 'Eggs'], Support: 0.20

Generated Association Rules (Brute-Force):
Rule 1: ['Bottled Water'] -> ['Bread']
Confidence: 50.00%
Support: 20.00%

Rule 2: ['Bread'] -> ['Bottled Water']
Confidence: 80.00%
Support: 20.00%

Rule 3: ['Bottled Water'] -> ['Eggs']
Confidence: 62.50%
Support: 25.00%

Rule 4: ['Eggs'] -> ['Bottled Water']
Confidence: 71.43%
Support: 25.00%

Rule 5: ['Bottled Water'] -> ['Laundry Detergent']
Confidence: 50.00%
Support: 20.00%

Rule 6: ['Laundry Detergent'] -> ['Bottled Water']
Confidence: 36.36%
Support: 20.00%

Rule 7: ['Bread'] -> ['Eggs']
Confidence: 80.00%
Support: 20.00%

Rule 8: ['Eggs'] -> ['Bread']
Confidence: 57.14%
Support: 20.00%
```

Screenshot 3.2.2–Brute Force Algorithm

```
Initiating Apriori Algorithm...

Frequent Itemsets (Apriori):
Itemset 1: ['Bottled Water'], Support: 0.40
Itemset 2: ['Bread'], Support: 0.25
Itemset 3: ['Cereal'], Support: 0.40
Itemset 4: ['Dish Soap'], Support: 0.45
Itemset 5: ['Eggs'], Support: 0.35
Itemset 6: ['Laundry Detergent'], Support: 0.55
Itemset 7: ['Milk'], Support: 0.30
Itemset 8: ['Shampoo'], Support: 0.35
Itemset 9: ['Toilet Paper'], Support: 0.40
Itemset 10: ['Toothpaste'], Support: 0.20
Itemset 11: ['Bottled Water', 'Bread'], Support: 0.20
Itemset 12: ['Bottled Water', 'Eggs'], Support: 0.25
Itemset 13: ['Bottled Water', 'Laundry Detergent'], Support: 0.20
Itemset 14: ['Eggs', 'Bread'], Support: 0.20
Itemset 15: ['Dish Soap', 'Cereal'], Support: 0.20
Itemset 16: ['Laundry Detergent', 'Cereal'], Support: 0.20
Itemset 17: ['Shampoo', 'Cereal'], Support: 0.20
Itemset 18: ['Toilet Paper', 'Cereal'], Support: 0.20
Itemset 19: ['Dish Soap', 'Laundry Detergent'], Support: 0.35
Itemset 20: ['Dish Soap', 'Milk'], Support: 0.20
Itemset 21: ['Dish Soap', 'Shampoo'], Support: 0.20
Itemset 22: ['Dish Soap', 'Toilet Paper'], Support: 0.20
Itemset 23: ['Laundry Detergent', 'Toilet Paper'], Support: 0.25
Itemset 24: ['Bottled Water', 'Eggs', 'Bread'], Support: 0.20

Generated Association Rules (Apriori):
Rule 1: ['Bottled Water'] -> ['Bread']
Confidence: 50.00%
Support: 20.00%

Rule 2: ['Bread'] -> ['Bottled Water']
Confidence: 80.00%
Support: 20.00%

Rule 3: ['Bottled Water'] -> ['Eggs']
Confidence: 62.50%
Support: 25.00%

Rule 4: ['Eggs'] -> ['Bottled Water']
Confidence: 71.43%
Support: 25.00%

Rule 5: ['Bottled Water'] -> ['Laundry Detergent']
Confidence: 50.00%
Support: 20.00%

Rule 6: ['Laundry Detergent'] -> ['Bottled Water']
Confidence: 36.36%
Support: 20.00%

Rule 7: ['Eggs'] -> ['Bread']
Confidence: 57.14%
Support: 20.00%

Rule 8: ['Bread'] -> ['Eggs']
Confidence: 80.00%
Support: 20.00%
```

Screenshot 3.3.3–Apriori Algorithm

```
==========================================
FINAL SUMMARY

Brute-Force Algorithm:
  - Total Rules Generated: 32
  - Execution Time: 7.4248 seconds

Apriori Algorithm:
  - Total Rules Generated: 32
  - Execution Time: 0.0037 seconds
==========================================
```

Screenshot 3.3.4–Final Summary

The source code (py.) and the necessary datasets (.csv files) will be contained within the .zip file.

Link to the GitHub repository:
https://github.com/etp6/DataMining-MTP