NAME: <student name>
NJIT UCID: <ucid>
Email Address: <email>
<date>
Professor: Yasser Abduallah
CS 634 <section> Data Mining

Midterm Project Report

Implementation and Code Usage

Apriori Algorithm Implementation in Retail Data Mining

Abstract:

In this project, I explore the Apriori Algorithm, a fundamental technique in data mining, to uncover associations within retail transactions. By implementing the algorithm and employing various data mining concepts, principles, and methods, I assess its effectiveness and efficiency. Through the design and development of custom data mining tools, I create a custom model for mining valuable insights from transaction data.

Introduction:

Data mining is a powerful approach for uncovering hidden patterns and associations within large datasets. Our project focuses on the Apriori Algorithm, a classic method for association rule mining, and its application in a retail context. We'll outline the core data mining concepts and principles applied in our work.

Writing and displaying association rules.The main concept of Apriori Algorithm is to create associations. In order to create associations, I had to figure out what items were most frequent when given the list of transactions. Once the items that are most frequent were found, depending on the user's support parameter, the support would have to be calculated for each item. After calculating the support value for each item, we can eliminate the items that do not meet the user-defined support parameter. The Apriori algorithm is a classic data mining algorithm that utilizes a brute force approach to find frequent itemsets and generate association rules. It works by iteratively increasing the size of itemsets and filtering out those that do not meet a minimum support threshold.

In this implementation, I applied the Apriori algorithm to a custom dataset associated with a retail store, allowing us to find frequent itemsets and association rules. Key steps in this process included:

- Initializing dictionaries for candidate and frequent itemsets.
- Loading the dataset and itemsets from CSV files.
- Preprocessing the dataset to ensure item order and uniqueness.
- Collecting user input for minimum support and confidence thresholds.
- Iteratively generating candidate itemsets and updating frequent itemsets using the Apriori algorithm, which employs a brute force approach by considering all possible combinations of items.

Core Concepts and Principles:

Frequent Itemset Discovery:
The Apriori Algorithm revolves around discovering frequent itemsets, i.e., sets of items that frequently co-occur in transactions. These itemsets provide insights into customer purchase behavior and preferences.

Support and Confidence:
Two key metrics in data mining are support and confidence. Support measures how frequently an item or itemset occurs, while confidence assesses the likelihood of items being purchased together. These metrics guide our analysis.

Association Rules:
By determining strong association rules, I identify which items are commonly purchased together. These rules are instrumental for optimizing sales strategies, such as recommendations.

Project Workflow:

Our project follows a structured workflow involving various stages and the application of the Apriori Algorithm:

Data Loading and Preprocessing:
We begin by loading transaction data from a retail store dataset. Each transaction consists of a list of items bought by a customer. To ensure data accuracy, we preprocess the dataset, filtering unique items and sorting them based on a predefined order.

Determination of Minimum Support and Confidence:
User input is crucial in data mining. We collect the user's preferences for minimum support and confidence levels to filter out less significant patterns.

Iteration Through Candidate Itemsets:
The iterative application of the Apriori Algorithm involves generating candidate itemsets of increasing sizes. We start with single items (itemset size K = 1) and proceed to K = 2, K = 3, and so on. This iterative process involves a "brute force" method of generating all possible itemset combinations.

Support Count Calculation:
For each candidate itemset, we calculate its support by counting how many transactions contain the itemset. Itemsets that meet the minimum support threshold are retained, while others are discarded.

Confidence Calculation:
We evaluate the confidence of association rules, indicating the strength of associations between items. This step requires careful comparison of support values for individual items and itemsets.

Association Rule Generation:
Association rules that satisfy both the minimum support and minimum confidence requirements are extracted. These rules reveal valuable insights into which items are often purchased together.

Results and Evaluation:

The project's effectiveness and efficiency are evaluated based on performance measures such as support, confidence, and the resulting association rules. We also compare our custom Apriori Algorithm implementation with the Apriori library to assess its reliability.

Conclusion:

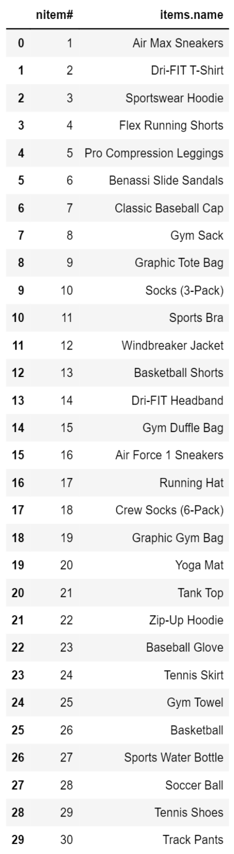
In conclusion, our project demonstrates the application of data mining concepts, principles, and methods. We successfully implemented the Apriori Algorithm to extract meaningful association rules from retail transaction data. The iterative, "brute force" approach, custom algorithm design, and adherence to user-defined parameters exemplify the power of data mining in revealing valuable patterns for decision-making in the retail industry.

Screenshots

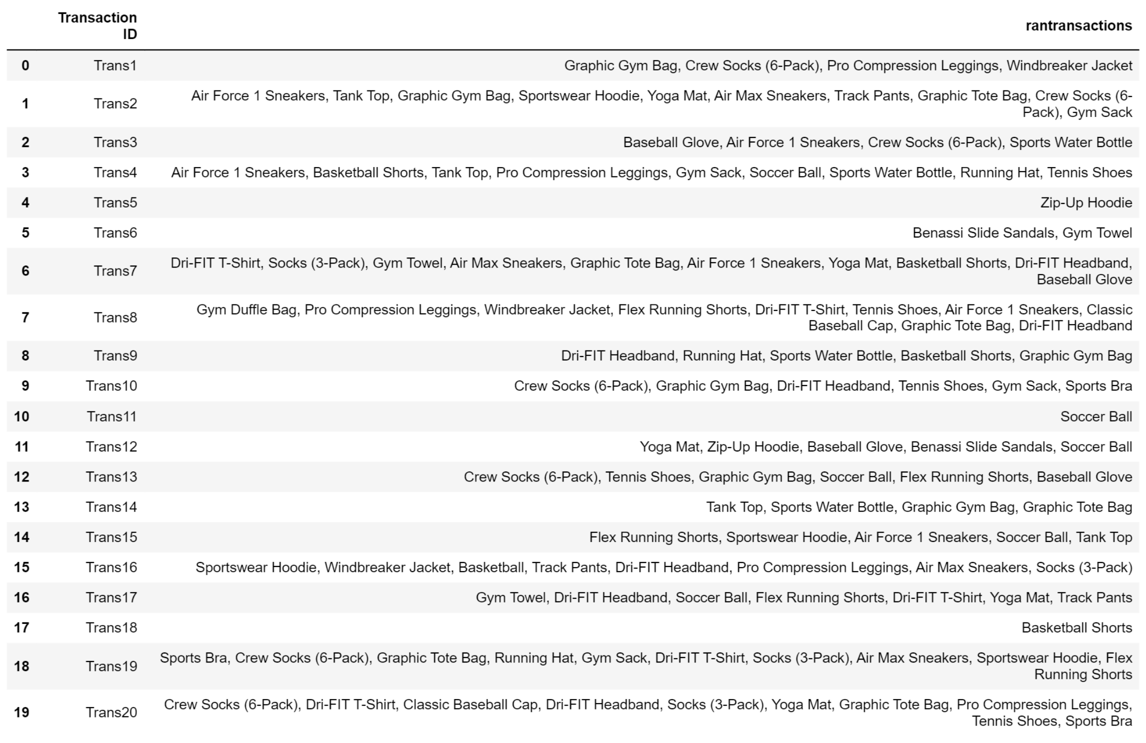
Here are what the csv files (This program takes in two separate csv files: Item Names & Transactions).

Figure 1 : Nike Item Names CSV file

Figure 2 : Nike Transactions CSV file



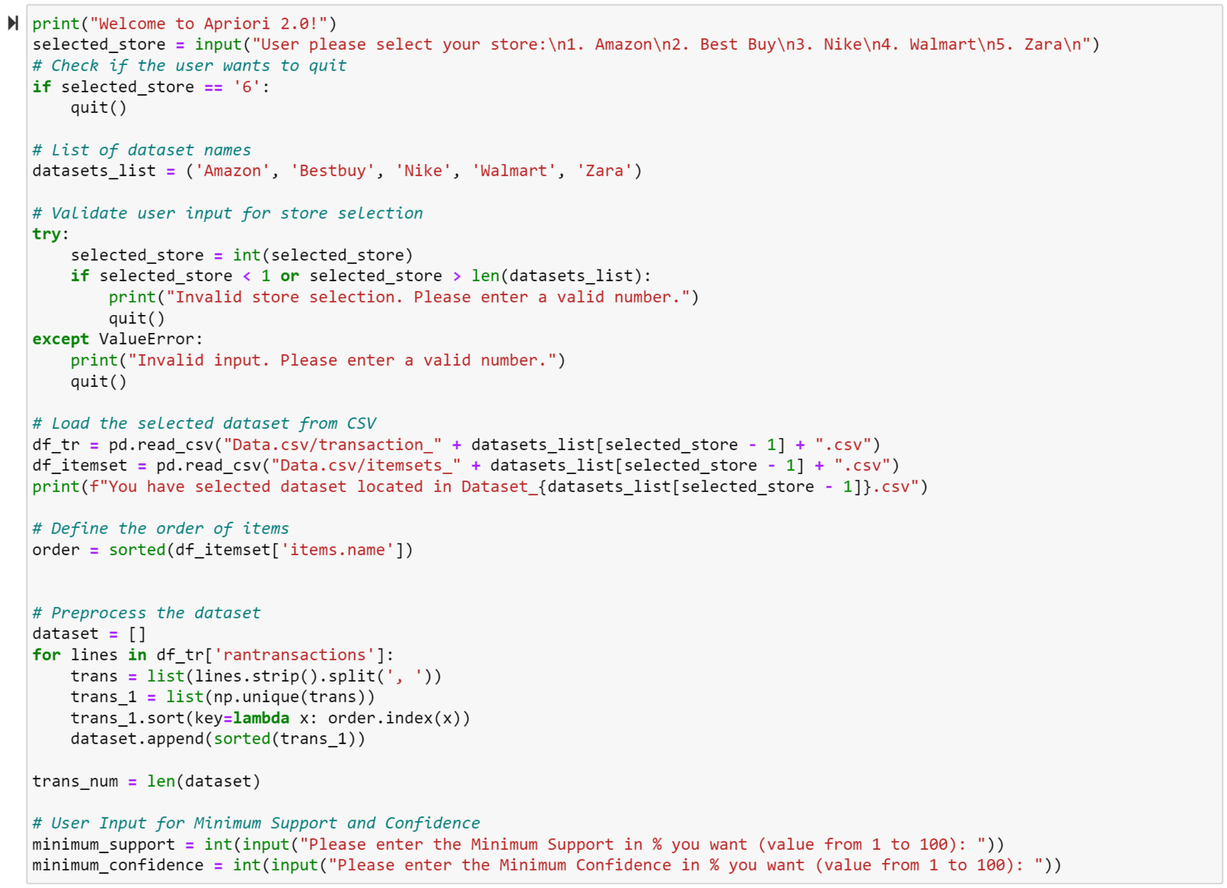
	nitem#	items.name
0	1	Air Max Sneakers
1	2	Dri-FIT T-Shirt
2	3	Sportswear Hoodie
3	4	Flex Running Shorts
4	5	Pro Compression Leggings
5	6	Benassi Slide Sandals
6	7	Classic Baseball Cap
7	8	Gym Sack
8	9	Graphic Tote Bag
9	10	Socks (3-Pack)
10	11	Sports Bra
11	12	Windbreaker Jacket
12	13	Basketball Shorts
13	14	Dri-FIT Headband
14	15	Gym Duffle Bag
15	16	Air Force 1 Sneakers
16	17	Running Hat
17	18	Crew Socks (6-Pack)
18	19	Graphic Gym Bag
19	20	Yoga Mat
20	21	Tank Top
21	22	Zip-Up Hoodie
22	23	Baseball Glove
23	24	Tennis Skirt
24	25	Gym Towel
25	26	Basketball
26	27	Sports Water Bottle
27	28	Soccer Ball
28	29	Tennis Shoes
29	30	Track Pants



	Transaction ID	rantransactions
0	Trans1	Graphic Gym Bag, Crew Socks (6-Pack), Pro Compression Leggings, Windbreaker Jacket
1	Trans2	Air Force 1 Sneakers, Tank Top, Graphic Gym Bag, Sportswear Hoodie, Yoga Mat, Air Max Sneakers, Track Pants, Graphic Tote Bag, Crew Socks (6-Pack), Gym Sack
2	Trans3	Baseball Glove, Air Force 1 Sneakers, Crew Socks (6-Pack), Sports Water Bottle
3	Trans4	Air Force 1 Sneakers, Basketball Shorts, Tank Top, Pro Compression Leggings, Gym Sack, Soccer Ball, Sports Water Bottle, Running Hat, Tennis Shoes
4	Trans5	Zip-Up Hoodie
5	Trans6	Benassi Slide Sandals, Gym Towel
6	Trans7	Dri-FIT T-Shirt, Socks (3-Pack), Gym Towel, Air Max Sneakers, Graphic Tote Bag, Air Force 1 Sneakers, Yoga Mat, Basketball Shorts, Dri-FIT Headband, Baseball Glove
7	Trans8	Gym Duffle Bag, Pro Compression Leggings, Windbreaker Jacket, Flex Running Shorts, Dri-FIT T-Shirt, Tennis Shoes, Air Force 1 Sneakers, Classic Baseball Cap, Graphic Tote Bag, Dri-FIT Headband
8	Trans9	Dri-FIT Headband, Running Hat, Sports Water Bottle, Basketball Shorts, Graphic Gym Bag
9	Trans10	Crew Socks (6-Pack), Graphic Gym Bag, Dri-FIT Headband, Tennis Shoes, Gym Sack, Sports Bra
10	Trans11	Soccer Ball
11	Trans12	Yoga Mat, Zip-Up Hoodie, Baseball Glove, Benassi Slide Sandals, Soccer Ball
12	Trans13	Crew Socks (6-Pack), Tennis Shoes, Graphic Gym Bag, Soccer Ball, Flex Running Shorts, Baseball Glove
13	Trans14	Tank Top, Sports Water Bottle, Graphic Gym Bag, Graphic Tote Bag
14	Trans15	Flex Running Shorts, Sportswear Hoodie, Air Force 1 Sneakers, Soccer Ball, Tank Top
15	Trans16	Sportswear Hoodie, Windbreaker Jacket, Basketball, Track Pants, Dri-FIT Headband, Pro Compression Leggings, Air Max Sneakers, Socks (3-Pack)
16	Trans17	Gym Towel, Dri-FIT Headband, Soccer Ball, Flex Running Shorts, Dri-FIT T-Shirt, Yoga Mat, Track Pants
17	Trans18	Basketball Shorts
18	Trans19	Sports Bra, Crew Socks (6-Pack), Graphic Tote Bag, Running Hat, Gym Sack, Dri-FIT T-Shirt, Socks (3-Pack), Air Max Sneakers, Sportswear Hoodie, Flex Running Shorts
19	Trans20	Crew Socks (6-Pack), Dri-FIT T-Shirt, Classic Baseball Cap, Dri-FIT Headband, Socks (3-Pack), Yoga Mat, Graphic Tote Bag, Pro Compression Leggings, Tennis Shoes, Sports Bra

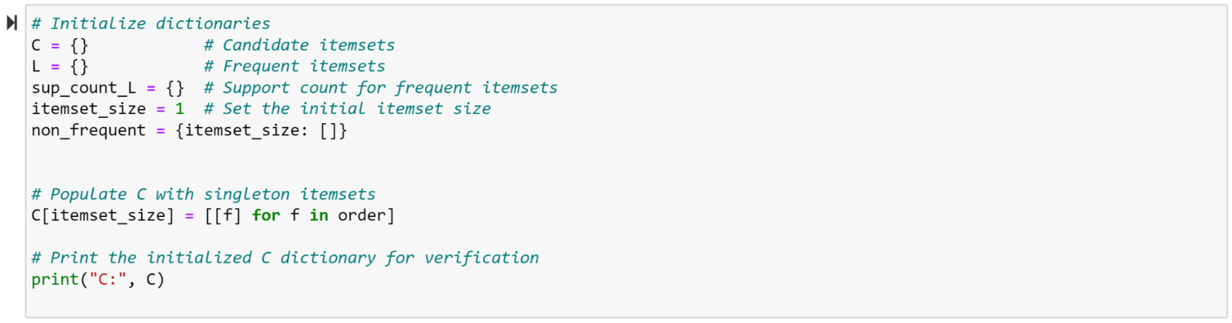
Below are screenshots of the code from python file:

Prompts to choose which store you want. We first have to read in the csv files and make sure that the inputs received from the user are valid



```python
print("Welcome to Apriori 2.0!")
selected_store = input("User please select your store:\n1. Amazon\n2. Best Buy\n3. Nike\n4. Walmart\n5. Zara\n")
# Check if the user wants to quit
if selected_store == '6':
    quit()

# List of dataset names
datasets_list = ('Amazon', 'Bestbuy', 'Nike', 'Walmart', 'Zara')

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

# Load the selected dataset from CSV
df_tr = pd.read_csv("Data.csv/transaction_" + datasets_list[selected_store - 1] + ".csv")
df_itemset = pd.read_csv("Data.csv/itemsets_" + datasets_list[selected_store - 1] + ".csv")
print(f"You have selected dataset located in Dataset_{datasets_list[selected_store - 1]}.csv")

# Define the order of items
order = sorted(df_itemset['items.name'])


# Preprocess the dataset
dataset = []
for lines in df_tr['rantransactions']:
    trans = list(lines.strip().split(', '))
    trans_1 = list(np.unique(trans))
    trans_1.sort(key=lambda x: order.index(x))
    dataset.append(sorted(trans_1))

trans_num = len(dataset)

# User Input for Minimum Support and Confidence
minimum_support = int(input("Please enter the Minimum Support in % you want (value from 1 to 100): "))
minimum_confidence = int(input("Please enter the Minimum Confidence in % you want (value from 1 to 100): "))
```

Initializing Candidate and Frequent Itemset Dictionaries



```python
# Initialize dictionaries
C = {}            # Candidate itemsets
L = {}            # Frequent itemsets
sup_count_L = {}  # Support count for frequent itemsets
itemset_size = 1  # Set the initial itemset size
non_frequent = {itemset_size: []}


# Populate C with singleton itemsets
C[itemset_size] = [[f] for f in order]

# Print the initialized C dictionary for verification
print("C:", C)
```

Counting Items and Finding Frequent Itemsets

```python
#`count_items`It iterates through the dataset and checks if the itemset is a subset of each transaction.
#The count of matching transactions is returned as the support count.

def count_items(itemset, dataset):
    count=0
    for i in range (0, len(dataset)):
        if set(itemset).issubset(set(dataset[i])):
            count += 1
    return count
```

```python
#`frequent_itemsets` function aims to find frequent itemsets within a given list of itemsets.
#It utilizes the Apriori algorithm brute force to filter and identify itemsets that meet the minimum support threshold.

def frequent_itemsets(itemsets, dataset, minimum_support, non_frequent):
    L = []
    sup_count = []
    new_non_frequent = []
    trans_num = len(dataset)
    K = len(non_frequent.keys())
    for i in range(0, len(itemsets)):
        candidate_set = 0
        if K>0:
            for j in non_frequent[K]:
                if set(j).issubset(set(itemsets[i])):
                    candidate_set = 1
                    break
        if candidate_set == 0:
            freq_count = count_items(itemsets[i], dataset)
            if freq_count >= (minimum_support/100)*trans_num:
                L.append(itemsets[i])
                sup_count.append(freq_count)
            else:
                new_non_frequent.append(itemsets[i])
    return L, sup_count, new_non_frequent
```

```python
def print_table (table, sup_count):
        print("Itemset | Count")
        for i in range (0, len(table)):
            print("{} : {}".format(table[i], sup_count[i]))
        print("\n\n")
```

The get_candidate_set function is responsible for generating candidate itemsets from a list of items.

```python
def get_candidate_set(items, order):
    candidate_set = []
    for i in range(len(items)):
        for j in range(i + 1, len(items)):
            joined_itemset = join_itemsets(items[i], items[j], order)
            if len(joined_itemset) > 0:
                candidate_set.append(joined_itemset)
    return candidate_set

def join_itemsets(item_1, item_2, order):
    item_1.sort(key=lambda x: order.index(x))
    item_2.sort(key=lambda x: order.index(x))

    for i in range(len(item_1) - 1):
        if item_1[i] != item_2[i]:
            return []
    if order.index(item_1[-1]) < order.index(item_2[-1]):
        return item_1 + [item_2[-1]]
    return []
```

The all_subsets function generates all possible subsets of a given set

```
def all_subsets(x):
    s = list(x)
    subsets = list(chain.from_iterable(combinations(s, r) for r in range(1, len(s) + 1)))
    return subsets

def write_assoc_rules(item2, item1, conf, support, trans_num, rulenumber):
    association_rules = ""
    association_rules += f"Rule {rulenumber}: {list(item1)} -> {list(item2)}\n"
    association_rules += f"Confidence: {conf * 100:.2f}%\n"
    association_rules += f"Support: {(support / trans_num) * 100:.2f}%\n\n"
    return association_rules
```

```
# Calculate frequent itemsets and their support
freq_set, support, new_non_frequent = frequent_itemsets(C[itemset_size], dataset, minimum_support, non_frequent)

# Update frequent itemsets, non-frequent itemsets, and support counts
L.update({itemset_size: freq_set})
non_frequent.update({itemset_size: new_non_frequent})
sup_count_L.update({itemset_size: support})
```

```
def print_itemset_table(itemset_name, itemsets, sup_counts):
    print(f"\nTable {itemset_name}:\n")
    for i, itemset in enumerate(itemsets):
        itemset_str = ', '.join(itemset)
        count = sup_counts[i]
        print(f"{itemset_str} : {count}")

print_itemset_table("C1", C[1], [count_items(item, dataset) for item in C[1]])
print_itemset_table("L1", L[1], sup_count_L[1])
```

Iterative Generation of Candidate Itemsets

```
# Uutlined the iterative process for generating candidate itemsets and updating frequent itemsets using the Apriori algorithm
#The process involves incrementing the itemset size `K` until no more frequent itemsets can be found.

K = itemset_size + 1
candidate_set = 0

while candidate_set == 0:
    candidate_itemsets = get_candidate_set(L[K - 1], order)
    C.update({K: candidate_itemsets})

    print("Table C{}: \n".format(K))
    print_table(C[K], [count_items(item, dataset) for item in C[K]])

    freq_set, support, new_non_frequent = frequent_itemsets(C[K], dataset, minimum_support, non_frequent)

    L.update({K: freq_set})
    non_frequent.update({K: new_non_frequent})
    sup_count_L.update({K: support})

    if len(L[K]) == 0:
        candidate_set = 1
    else:
        print("Table L{}: \n".format(K))
        print_table(L[K], sup_count_L[K])

    K += 1
```

...

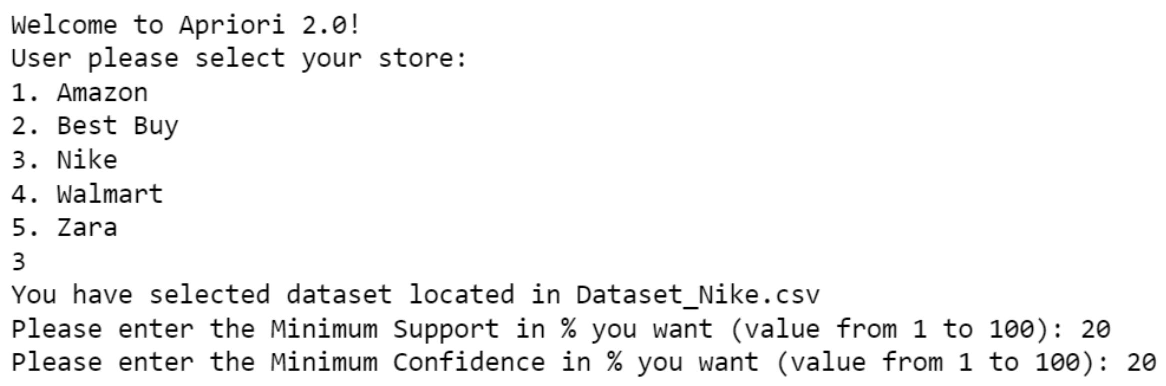
Generated Final Association Rules

```python
# Generate association rules
association_rules = ""
rulenumber = 1

for itemset_size in range(1, len(L)):
    for itemset in L[itemset_size]:
        subsets = list(all_subsets(set(itemset)))
        subsets.pop()

        for subset in subsets:
            item1 = set(subset)
            freq_set1 = set(itemset)
            item2 = set(freq_set1 - item1)
            support_freq_set1 = count_items(freq_set1, dataset)
            support_item1 = count_items(item1, dataset)
            support_item2 = count_items(item2, dataset)
            confidence = support_freq_set1 / support_item1

            if confidence >= (minimum_confidence / 100) and support_freq_set1 >= (minimum_support / 100) * trans_num:
                association_rules += write_assoc_rules(item2, item1, confidence, support_freq_set1, trans_num, rulenumber)
                rulenumber += 1

print("Final Association Rules: \n")
print(association_rules)
```

Verified Results with the built-in python package.

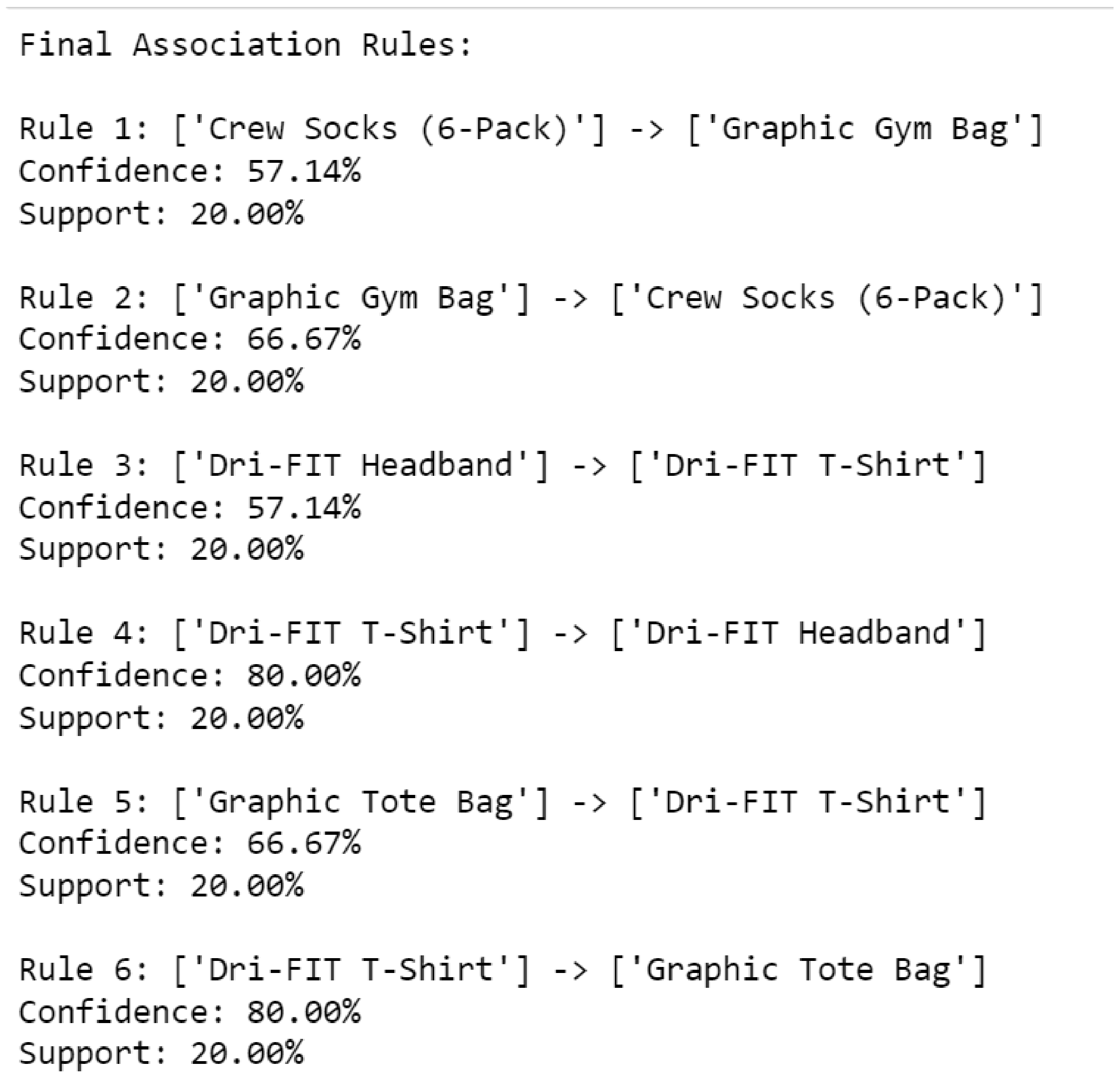
```python
from apriori_python.apriori import apriori

# Your code for loading and preprocessing the dataset

# Specify minSup and minConf values
minSup = 0.2
minConf = 0.19

# Run Apriori algorithm
freqItemSet, rules = apriori(dataset, minSup=minSup, minConf=minConf)

# Print the generated rules
for i, rule in enumerate(rules):
    print(f"Rule {i + 1}: {rule}\n")
```

Below are screenshots to show that the program runs in the Terminal.



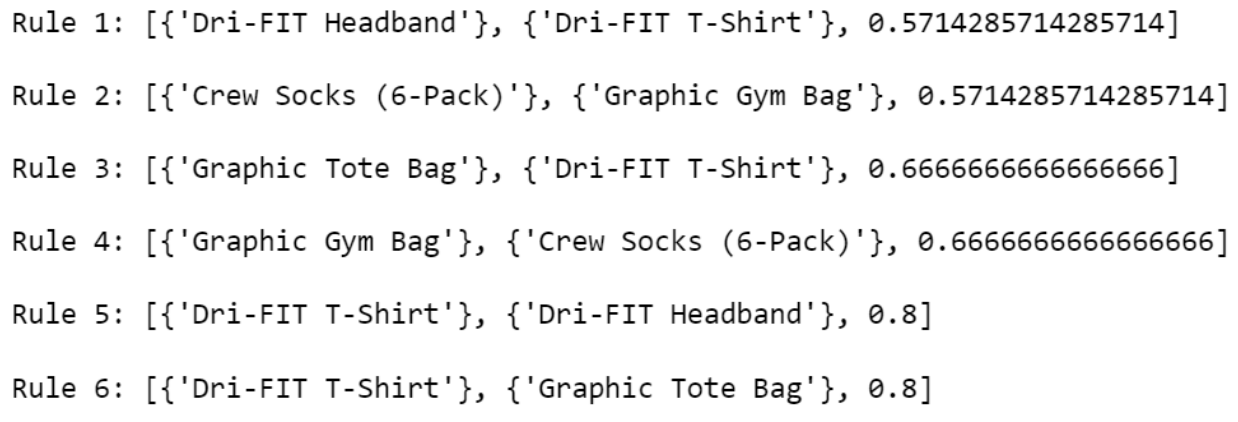
```
Welcome to Apriori 2.0!
User please select your store:
1. Amazon
2. Best Buy
3. Nike
4. Walmart
5. Zara
3
You have selected dataset located in Dataset_Nike.csv
Please enter the Minimum Support in % you want (value from 1 to 100): 20
Please enter the Minimum Confidence in % you want (value from 1 to 100): 20
```

The final output should be the following:



```
Final Association Rules:

Rule 1: ['Crew Socks (6-Pack)'] -> ['Graphic Gym Bag']
Confidence: 57.14%
Support: 20.00%

Rule 2: ['Graphic Gym Bag'] -> ['Crew Socks (6-Pack)']
Confidence: 66.67%
Support: 20.00%

Rule 3: ['Dri-FIT Headband'] -> ['Dri-FIT T-Shirt']
Confidence: 57.14%
Support: 20.00%

Rule 4: ['Dri-FIT T-Shirt'] -> ['Dri-FIT Headband']
Confidence: 80.00%
Support: 20.00%

Rule 5: ['Graphic Tote Bag'] -> ['Dri-FIT T-Shirt']
Confidence: 66.67%
Support: 20.00%

Rule 6: ['Dri-FIT T-Shirt'] -> ['Graphic Tote Bag']
Confidence: 80.00%
Support: 20.00%
```

Verified With Built in Package:



```
Rule 1: [{'Dri-FIT Headband'}, {'Dri-FIT T-Shirt'}, 0.5714285714285714]

Rule 2: [{'Crew Socks (6-Pack)'}, {'Graphic Gym Bag'}, 0.5714285714285714]

Rule 3: [{'Graphic Tote Bag'}, {'Dri-FIT T-Shirt'}, 0.6666666666666666]

Rule 4: [{'Graphic Gym Bag'}, {'Crew Socks (6-Pack)'}, 0.6666666666666666]

Rule 5: [{'Dri-FIT T-Shirt'}, {'Dri-FIT Headband'}, 0.8]

Rule 6: [{'Dri-FIT T-Shirt'}, {'Graphic Tote Bag'}, 0.8]
```

Other

The source code (.py file) and data sets (.csv files) will be attached to the zip file.

Link to Git Repository

https://github.com/mahumabid/Apriori_Algorithm