NAME: Hem Ramesh Thumar

NJIT UCID: ht296

Email Address: ht296@njit.edu

13th October 2024

Professor: Yasser Abduallah

CS 634 - Data Mining

Midterm Project Report

Implementation and Code Usage

Apriori Algorithm Implementation in Retail Data Mining

Abstract

This project explores the Apriori Algorithm, a fundamental data mining technique, to uncover associations within retail transactions. The algorithm is implemented, and various data mining concepts, principles, and methods are employed to assess its effectiveness and efficiency. Custom data mining tools are developed to create a model for extracting valuable insights from transaction data.

Introduction

Data mining is a powerful approach for uncovering hidden patterns and associations within large datasets. This project focuses on the Apriori Algorithm, a classic method for association rule mining, and its application in a retail context. The core data mining concepts and principles applied in the work are outlined.

The main idea behind the Apriori Algorithm is to create associations. To do this, the algorithm first identifies the most frequent items in the list of transactions. Then, based on a user-defined support parameter, the support value for each item is calculated. Items that do not meet the user-defined support parameter are eliminated, and the remaining items are used to create associations. The Apriori algorithm is a well-known data mining technique that uses a brute-force approach to find frequently occurring sets of items and generate association rules. It works by repeatedly increasing the size of the item sets and removing those that don't meet a minimum support threshold.

In this particular implementation, the Apriori algorithm was applied to a custom dataset related to a retail store. This process involved several key steps:

1. Creating dictionaries to store candidate and frequent item sets.
2. Loading the dataset and item sets from CSV files.
3. Preprocessing the dataset to ensure proper item order and uniqueness.
4. Collecting user input for the minimum support and confidence thresholds.
5. Iteratively generating candidate item sets and updating the frequent item sets using the Apriori algorithm's brute-force approach, which considers all possible combinations of items.

This implementation allows us to identify the frequent item sets and association rules within the retail store's data, providing valuable insights for business decisions and strategies.

Core concepts and Principles

Frequent Itemset Discovery

The Apriori Algorithm is a fundamental method used in data mining that focuses on discovering frequent itemsets. These itemsets are essentially groups of items that frequently appear together in transactions. Understanding these itemsets is crucial as they provide valuable insights into customer purchasing behavior and preferences, helping businesses tailor their strategies to meet consumer needs more effectively.

Support and Confidence

In the realm of data mining, two essential metrics play a pivotal role: support and confidence. Support is a measure that indicates how often a particular item or itemset appears in the dataset. It helps us understand the popularity of items among customers. On the other hand, confidence assesses the likelihood that items will be purchased together. This metric is vital for determining the strength of the relationship between items. Together, these metrics guide our analysis and help us make informed decisions based on the data.

Association Rules

By identifying strong association rules, I can determine which items are commonly purchased together. These rules are not just theoretical; they are instrumental in optimizing sales strategies. For instance, they can be used to create effective product recommendations, enhancing the shopping experience for customers and potentially increasing sales for retailers. Understanding these associations allows businesses to strategically position products and create promotions that resonate with their target audience.

Project Workflow

Our project follows a structured workflow that involves several stages, all centered around the application of the Apriori Algorithm:

Data Loading and Preprocessing

The first step in our project is to load transaction data from a retail store dataset. Each transaction consists of a list of items that a customer has purchased. To ensure the accuracy and reliability of our analysis, we preprocess the dataset. This preprocessing step includes filtering out unique items and sorting them based on a predefined order. By doing this, we prepare the data for further analysis, ensuring that it is clean and organized.

Determination of Minimum Support and Confidence

User input is crucial in the data mining process. We actively collect the user's preferences regarding the minimum support and confidence levels they wish to set. This input is essential as it helps filter out less significant patterns that may not provide valuable insights. By establishing these thresholds, we can focus our analysis on the most relevant and impactful itemsets.

Iteration Through Candidate Itemsets

The iterative application of the Apriori Algorithm involves generating candidate itemsets of increasing sizes. We start with single items, referred to as itemset size K = 1, and then progress to K = 2, K = 3, and so forth. This iterative process employs a "brute force" method, where we generate all possible combinations of itemsets. This thorough approach ensures that we do not overlook any potential associations.

Support Count Calculation

For each candidate itemset generated, we calculate its support by counting how many transactions contain that specific itemset. Itemsets that meet or exceed the minimum support threshold are retained for further analysis, while those that do not meet this criterion are discarded. This step is crucial for narrowing down our focus to the most relevant itemsets.

Confidence Calculation

Next, we evaluate the confidence of the association rules. This step indicates the strength of the associations between items. It requires careful comparison of support values for individual items and itemsets. By analyzing these values, we can determine which associations are strong and worth considering in our recommendations.

Association Rule Generation

Finally, we extract association rules that satisfy both the minimum support and minimum confidence requirements. These rules reveal valuable insights into which items are often purchased together. By leveraging these insights, businesses can make data-driven decisions that enhance their marketing strategies and improve customer satisfaction.

Result and Evaluation

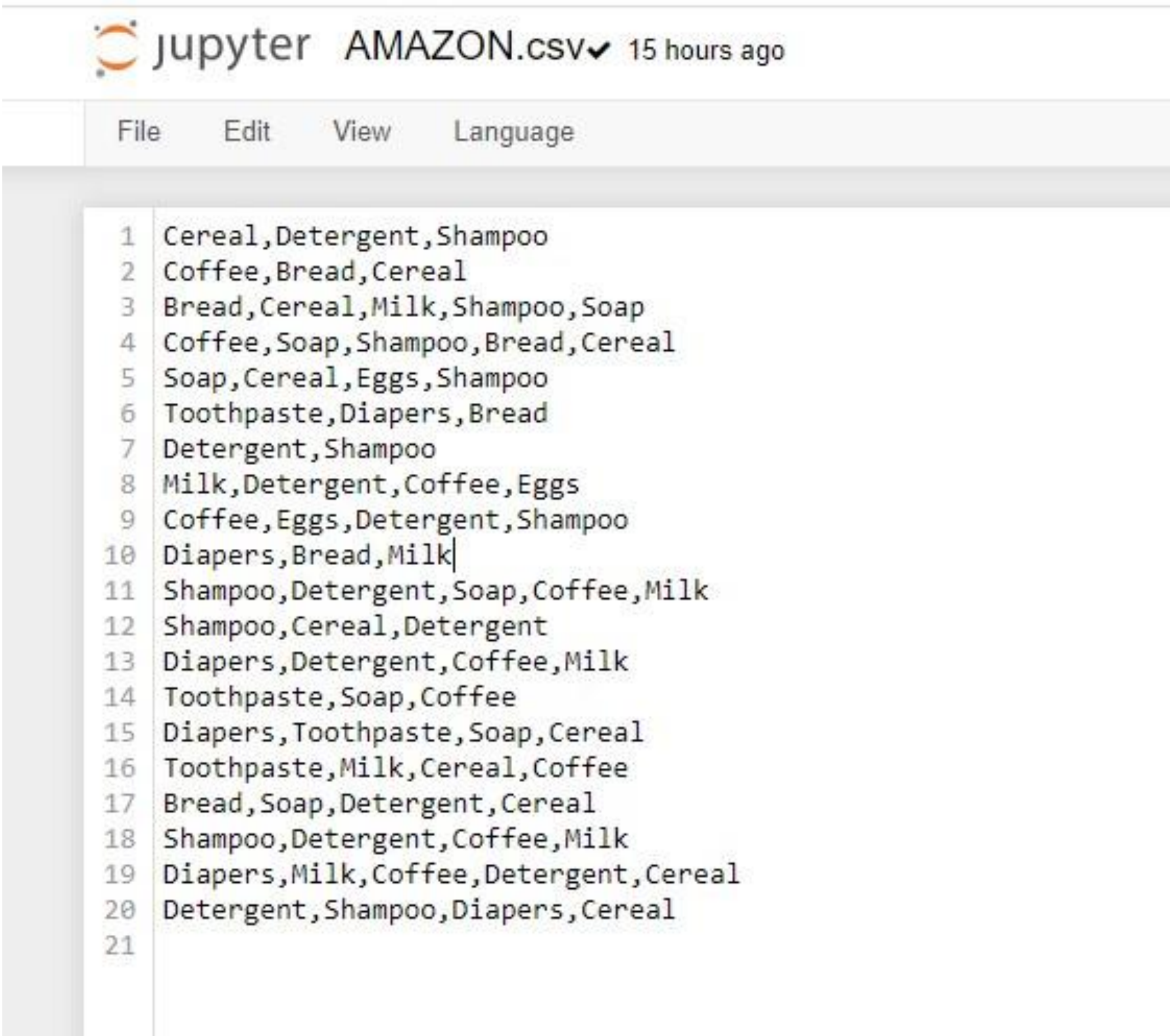
The project's effectiveness and efficiency are thoroughly evaluated using specific performance measures, which include support, confidence, and the resulting association rules derived from the data. These metrics help us understand how well the algorithm performs in identifying relationships within the data. Additionally, we conduct a comparison between our custom implementation of the Apriori Algorithm and the established Apriori library. This comparison is crucial as it allows us to assess the reliability and accuracy of our approach, ensuring that our results are both valid and trustworthy.

Conclusion

In conclusion, our project effectively demonstrates the practical application of data mining concepts, principles, and methods in a real-world context. We successfully implemented the Apriori Algorithm to extract meaningful association rules from retail transaction data, showcasing its utility. The iterative, "brute force" approach, along with our custom algorithm design and strict adherence to user-defined parameters, exemplifies the power of data mining. This process reveals valuable patterns that can significantly enhance decision-making in the retail industry, ultimately leading to improved business strategies and outcomes.

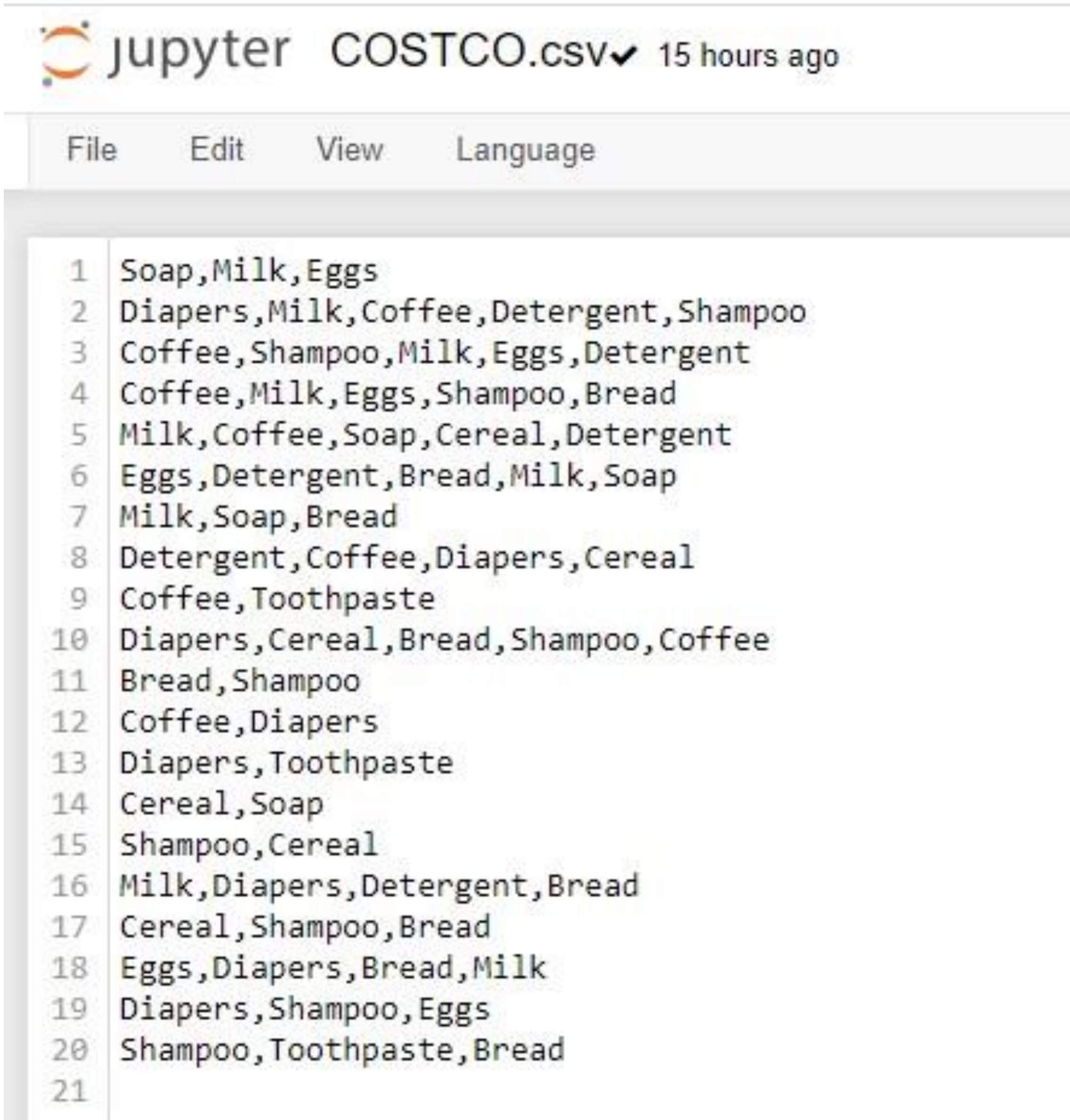
Screenshots

Here are the csv files for the project.




```
Cereal,Detergent,Shampoo
Coffee,Bread,Cereal
Bread,Cereal,Milk,Shampoo,Soap
Coffee,Soap,Shampoo,Bread,Cereal
Soap,Cereal,Eggs,Shampoo
Toothpaste,Diapers,Bread
Detergent,Shampoo
Milk,Detergent,Coffee,Eggs
Coffee,Eggs,Detergent,Shampoo
Diapers,Bread,Milk
Shampoo,Detergent,Soap,Coffee,Milk
Shampoo,Cereal,Detergent
Diapers,Detergent,Coffee,Milk
Toothpaste,Soap,Coffee
Diapers,Toothpaste,Soap,Cereal
Toothpaste,Milk,Cereal,Coffee
Bread,Soap,Detergent,Cereal
Shampoo,Detergent,Coffee,Milk
Diapers,Milk,Coffee,Detergent,Cereal
Detergent,Shampoo,Diapers,Cereal
```

Fig1: Amazon_CSV

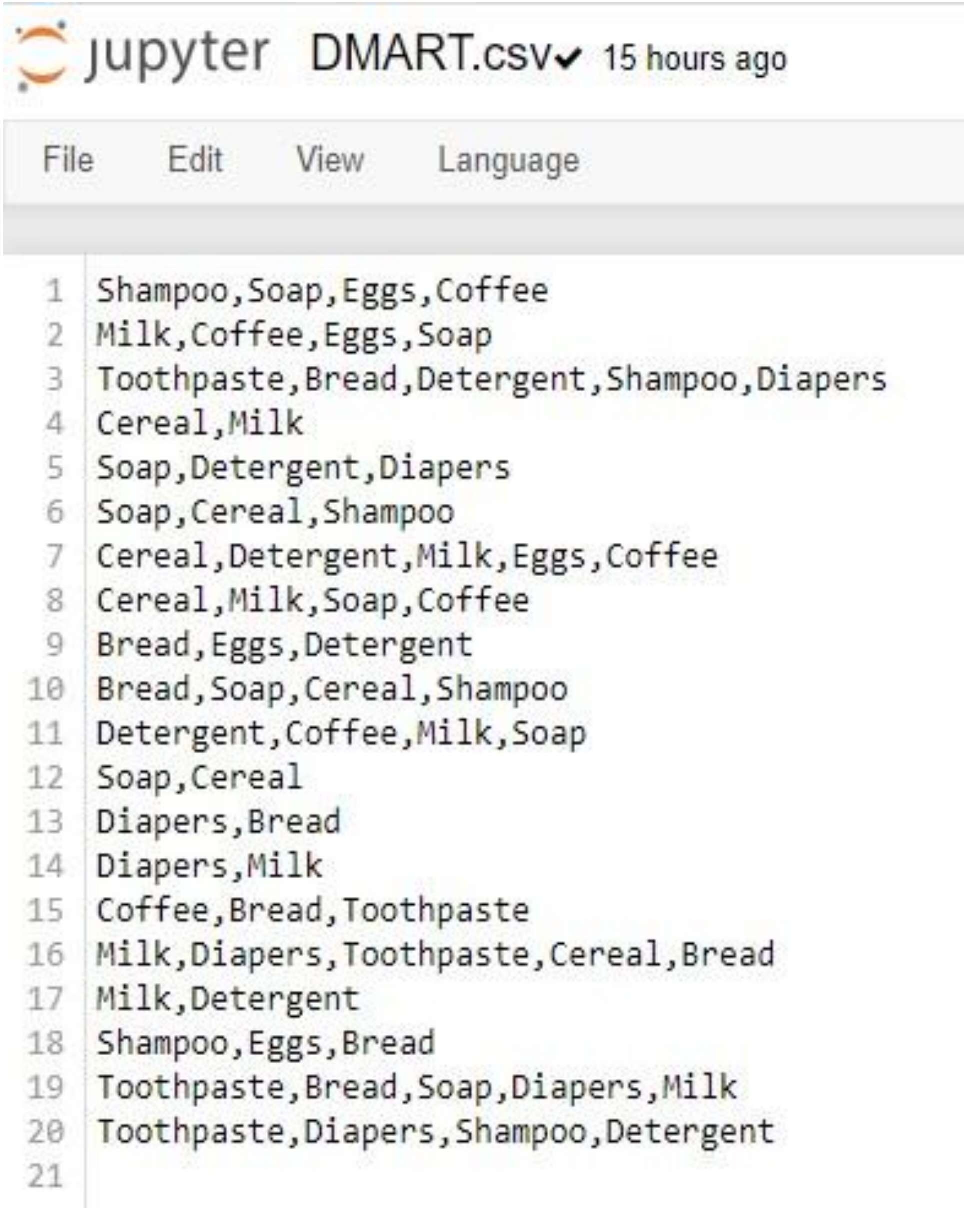


jupyter COSTCO.csv 15 hours ago

File Edit View Language

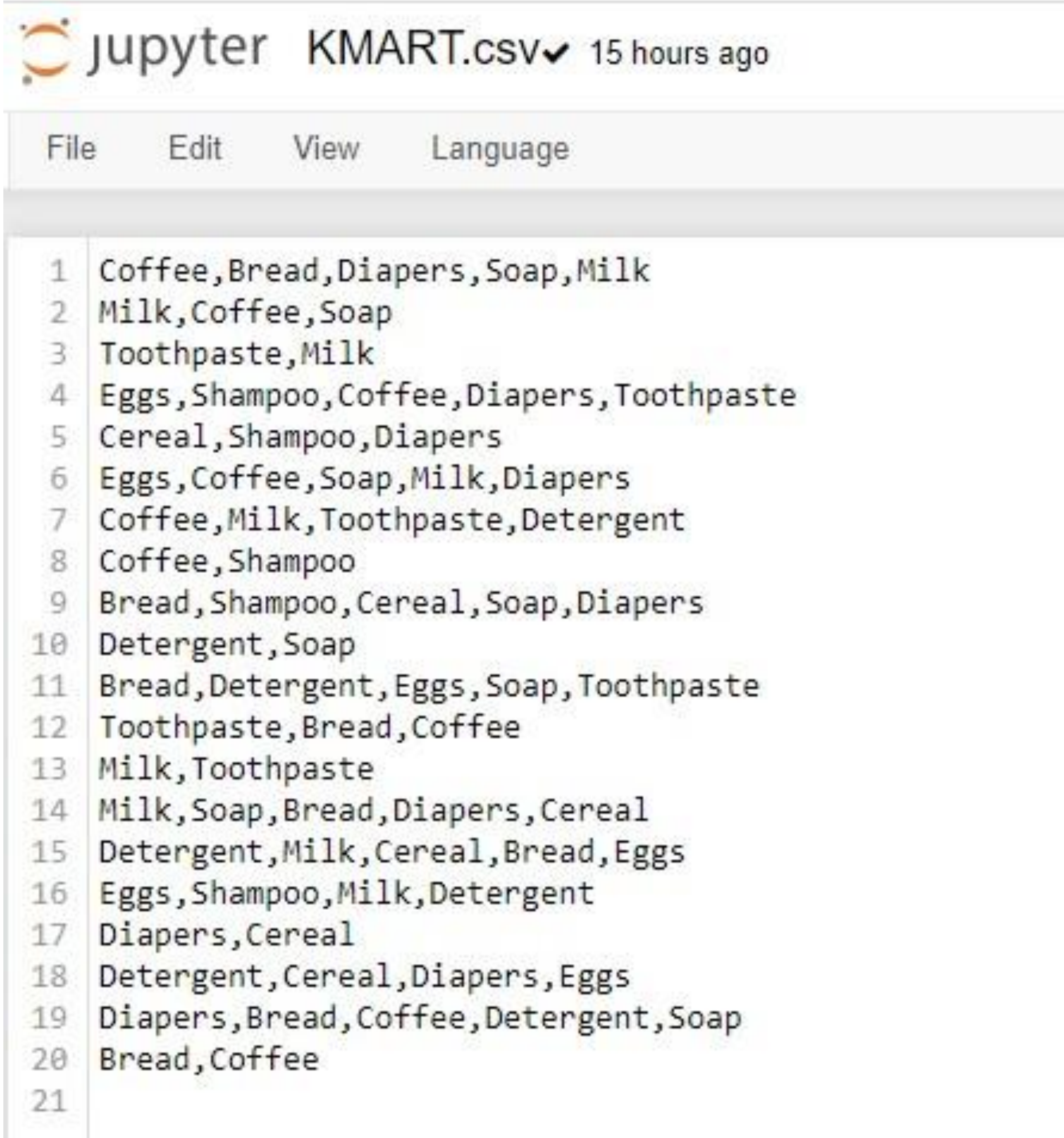
```
Soap,Milk,Eggs
Diapers,Milk,Coffee,Detergent,Shampoo
Coffee,Shampoo,Milk,Eggs,Detergent
Coffee,Milk,Eggs,Shampoo,Bread
Milk,Coffee,Soap,Cereal,Detergent
Eggs,Detergent,Bread,Milk,Soap
Milk,Soap,Bread
Detergent,Coffee,Diapers,Cereal
Coffee,Toothpaste
Diapers,Cereal,Bread,Shampoo,Coffee
Bread,Shampoo
Coffee,Diapers
Diapers,Toothpaste
Cereal,Soap
Shampoo,Cereal
Milk,Diapers,Detergent,Bread
Cereal,Shampoo,Bread
Eggs,Diapers,Bread,Milk
Diapers,Shampoo,Eggs
Shampoo,Toothpaste,Bread
```

Fig 2: Costco_CSV



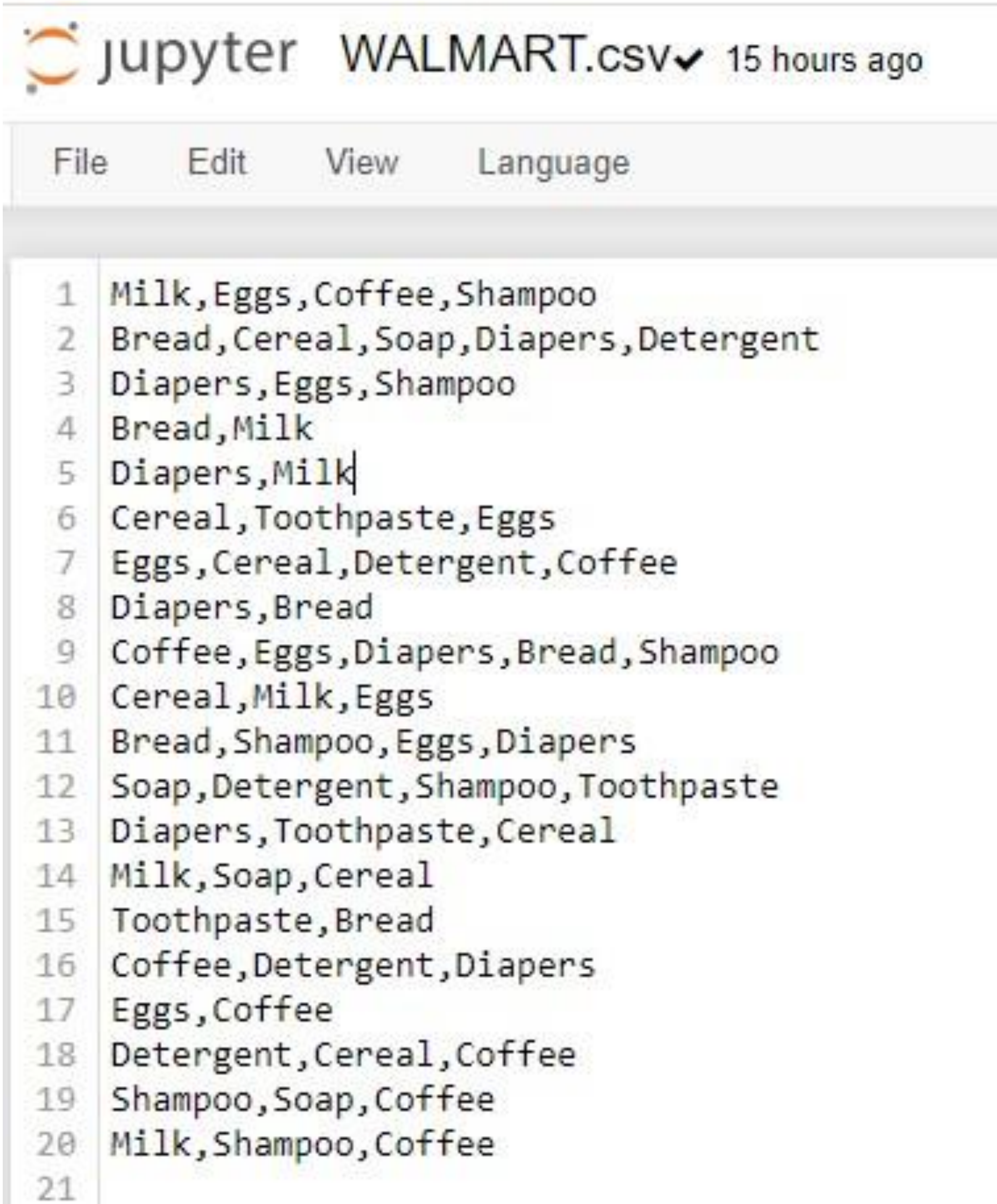

```
Shampoo,Soap,Eggs,Coffee
Milk,Coffee,Eggs,Soap
Toothpaste,Bread,Detergent,Shampoo,Diapers
Cereal,Milk
Soap,Detergent,Diapers
Soap,Cereal,Shampoo
Cereal,Detergent,Milk,Eggs,Coffee
Cereal,Milk,Soap,Coffee
Bread,Eggs,Detergent
Bread,Soap,Cereal,Shampoo
Detergent,Coffee,Milk,Soap
Soap,Cereal
Diapers,Bread
Diapers,Milk
Coffee,Bread,Toothpaste
Milk,Diapers,Toothpaste,Cereal,Bread
Milk,Detergent
Shampoo,Eggs,Bread
Toothpaste,Bread,Soap,Diapers,Milk
Toothpaste,Diapers,Shampoo,Detergent
```

Fig 3: DMART_CSV




```
Coffee,Bread,Diapers,Soap,Milk
Milk,Coffee,Soap
Toothpaste,Milk
Eggs,Shampoo,Coffee,Diapers,Toothpaste
Cereal,Shampoo,Diapers
Eggs,Coffee,Soap,Milk,Diapers
Coffee,Milk,Toothpaste,Detergent
Coffee,Shampoo
Bread,Shampoo,Cereal,Soap,Diapers
Detergent,Soap
Bread,Detergent,Eggs,Soap,Toothpaste
Toothpaste,Bread,Coffee
Milk,Toothpaste
Milk,Soap,Bread,Diapers,Cereal
Detergent,Milk,Cereal,Bread,Eggs
Eggs,Shampoo,Milk,Detergent
Diapers,Cereal
Detergent,Cereal,Diapers,Eggs
Diapers,Bread,Coffee,Detergent,Soap
Bread,Coffee
```

Fig 4: KMART_CSV




```
Milk,Eggs,Coffee,Shampoo
Bread,Cereal,Soap,Diapers,Detergent
Diapers,Eggs,Shampoo
Bread,Milk
Diapers,Milk
Cereal,Toothpaste,Eggs
Eggs,Cereal,Detergent,Coffee
Diapers,Bread
Coffee,Eggs,Diapers,Bread,Shampoo
Cereal,Milk,Eggs
Bread,Shampoo,Eggs,Diapers
Soap,Detergent,Shampoo,Toothpaste
Diapers,Toothpaste,Cereal
Milk,Soap,Cereal
Toothpaste,Bread
Coffee,Detergent,Diapers
Eggs,Coffee
Detergent,Cereal,Coffee
Shampoo,Soap,Coffee
Milk,Shampoo,Coffee
```

Fig 5: Walmart_csv

Below are the screenshots of the python code file:

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

Fig 1: Libraries used

```
# Dataset Paths
dataset_files = {
    "AMAZON": r"AMAZON.csv",
    "COSTCO": r"COSTCO.csv",
    "DMART": r"DMART.csv",
    "WALMART": r"WALMART.csv",
    "KMART": r"KMART.csv"
}

class TransactionAnalyzer:
    def __init__(self, filepath):
        self.filepath = filepath
        self.transactions = self.extract_transactions()

    def extract_transactions(self):
        """Load transaction data from CSV."""
        with open(self.filepath, newline='') as file:
            reader = csv.reader(file)
            return [list(filter(None, row)) for row in reader]

    def compute_frequent_itemsets(self, support_min):
        """Brute force method for generating frequent itemsets."""
        item_frequency = {}
        for transaction in self.transactions:
            for item in transaction:
                item_frequency[item] = item_frequency.get(item, 0) + 1

        frequent_sets = {1: {item: count for item, count in item_frequency.items() if count / len(self.transactions) >= support_m

        k = 2
        while True:
            prev_itemset = list(frequent_sets[k - 1].keys())
            new_item_combinations = list(combinations(prev_itemset, k))
            current_count = {}

            for transaction in self.transactions:
                transaction_set = set(transaction)
                for combination in new_item_combinations:
                    if set(combination).issubset(transaction_set):
                        current_count[combination] = current_count.get(combination, 0) + 1

            frequent_sets[k] = {combo: count for combo, count in current_count.items() if count / len(self.transactions) >= suppo
            if not frequent_sets[k]:
                del frequent_sets[k]
                break
            k += 1

        return frequent_sets
```

Fig 2: Reading csv files and generating frequent items

```python
    def run_apriori(self, support_min, confidence_min):
        """Run the Apriori algorithm."""
        encoder = TransactionEncoder()
        transformed_data = encoder.fit(self.transactions).transform(self.transactions)
        df_transactions = pd.DataFrame(transformed_data, columns=encoder.columns_)

        frequent_itemsets = apriori(df_transactions, min_support=support_min, use_colnames=True)
        rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=confidence_min)

        return frequent_itemsets, rules

    def run_fpgrowth(self, support_min, confidence_min):
        """Run the FP-Growth algorithm."""
        encoder = TransactionEncoder()
        transformed_data = encoder.fit(self.transactions).transform(self.transactions)
        df_transactions = pd.DataFrame(transformed_data, columns=encoder.columns_)

        frequent_itemsets = fpgrowth(df_transactions, min_support=support_min, use_colnames=True)
        rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=confidence_min)

        return frequent_itemsets, rules

def time_algorithm_execution(algorithm_func, *params):
    """Measure execution time of an algorithm."""
    start_time = time.time()
    output = algorithm_func(*params)
    elapsed_time = time.time() - start_time
    return output, elapsed_time
```

Fig 3: Applying apriori and fp growth algorithm

```
def main_menu():
    """User interaction interface."""
    while True:
        print("\nWelcome to the Data Transaction Analysis Tool!")
        print("Please select a dataset to analyze or exit the program:")

        # Listing of available datasets
        for index, name in enumerate(dataset_files.keys(), start=1):
            print(f"{index}. {name}")
        print("0. Exit")

        try:
            user_input = int(input("Enter the dataset number for analyzing (0 to exit): "))
            if user_input < 0 or user_input > len(dataset_files):
                raise ValueError("Invalid choice. Please select a valid option.")

            if user_input == 0:
                print("Thank you for using the program! Goodbye!")
                break

            selected_dataset = list(dataset_files.keys())[user_input - 1]
            analyzer = TransactionAnalyzer(dataset_files[selected_dataset])
            print(f"Successfully loaded {len(analyzer.transactions)} transactions from {selected_dataset}.")

            # Get user-defined thresholds for both operations
            support_input = float(input("Please enter the minimum support threshold (Example: 10 for 10%): ")) / 100
            confidence_input = float(input("Please enter the minimum confidence threshold (Example: 10 for 10%): ")) / 100

            print(f"\nAnalyzing dataset: {selected_dataset} with support threshold: {support_input * 100:.2f}% and confidence thr

            # Brute Force Frequent Itemsets
            brute_force_result, bf_execution_time = time_algorithm_execution(analyzer.compute_frequent_itemsets, support_input)
            print(f"\nBrute Force Frequent Itemsets:\n{brute_force_result}")
            print(f"Execution Time (Brute Force): {bf_execution_time:.4f} seconds")

            # Apriori Algorithm
            apriori_result, apriori_time = time_algorithm_execution(analyzer.run_apriori, support_input, confidence_input)
            print(f"\nFrequent Itemsets (Apriori):\n{apriori_result}")
            print(f"Execution Time (Apriori): {apriori_time:.4f} seconds")

            # FP-Growth Algorithm
            fpgrowth_result, fp_execution_time = time_algorithm_execution(analyzer.run_fpgrowth, support_input, confidence_input)
            print(f"\nFrequent Itemsets (FP-Growth):\n{fpgrowth_result}")
            print(f"Execution Time (FP-Growth): {fp_execution_time:.4f} seconds")

            # Ask user if they want to analyze another dataset
            retry = input("\nWould you like to analyze a different dataset? (yes/no): ").strip().lower()
            if retry != 'yes':
                print("Thank you for using the program! Goodbye!")
                break
```

```
        except ValueError as e:
            print(f"Error: {e}. Please try again.")

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

Fig 5: Main source code

Output:

```
Welcome to the Data Transaction Analysis Tool!
Please select a dataset to analyze or exit the program:
1. AMAZON
2. COSTCO
3. DMART
4. WALMART
5. KMART
0. Exit
Enter the dataset number for analyzing (0 to exit): 1
Successfully loaded 20 transactions from AMAZON.
Please enter the minimum support threshold (Example: 10 for 10%): 20
Please enter the minimum confidence threshold (Example: 10 for 10%): 20
```

Fig 1: User defined datasets

```
Brute Force Frequent Itemsets:
{1: {'Cereal': 11, 'Detergent': 11, 'Shampoo': 10, 'Coffee': 10, 'Bread': 6, 'Milk': 8, 'Soap': 7, 'Toothpaste': 4, 'Diapers':
6}, 2: {('Cereal', 'Detergent'): 5, ('Cereal', 'Shampoo'): 6, ('Detergent', 'Shampoo'): 7, ('Cereal', 'Coffee'): 4, ('Cereal',
'Bread'): 4, ('Cereal', 'Soap'): 5, ('Shampoo', 'Soap'): 4, ('Shampoo', 'Coffee'): 4, ('Detergent', 'Coffee'): 6, ('Detergent',
'Milk'): 5, ('Coffee', 'Milk'): 6}}
Execution Time (Brute Force): 0.0050 seconds
```

Fig 2: Brute Force time

```
Frequent Itemsets (Apriori):
(    support                   itemsets
0      0.30                    (Bread)
1      0.55                   (Cereal)
2      0.50                   (Coffee)
3      0.55                (Detergent)
4      0.30                  (Diapers)
5      0.40                     (Milk)
6      0.50                  (Shampoo)
7      0.35                     (Soap)
8      0.20               (Toothpaste)
9      0.20            (Cereal, Bread)
10     0.20           (Coffee, Cereal)
11     0.25        (Detergent, Cereal)
12     0.30          (Shampoo, Cereal)
13     0.25             (Cereal, Soap)
14     0.30        (Coffee, Detergent)
15     0.30             (Coffee, Milk)
16     0.20          (Coffee, Shampoo)
17     0.25          (Detergent, Milk)
18     0.35       (Shampoo, Detergent)
19     0.20            (Shampoo, Soap)
20     0.25  (Coffee, Detergent, Milk),           antecedents           consequents  antecedent support  \
0            (Cereal)               (Bread)                0.55
1             (Bread)              (Cereal)                0.30
2            (Coffee)              (Cereal)                0.50
3            (Cereal)              (Coffee)                0.55
4         (Detergent)              (Cereal)                0.55
5            (Cereal)           (Detergent)                0.55
6           (Shampoo)              (Cereal)                0.50
7            (Cereal)             (Shampoo)                0.55
8            (Cereal)                (Soap)                0.55
9              (Soap)              (Cereal)                0.35
10           (Coffee)           (Detergent)                0.50
11        (Detergent)              (Coffee)                0.55
12           (Coffee)                (Milk)                0.50
13             (Milk)              (Coffee)                0.40
14           (Coffee)             (Shampoo)                0.50
15          (Shampoo)              (Coffee)                0.50
16        (Detergent)                (Milk)                0.55
17             (Milk)           (Detergent)                0.40
18          (Shampoo)           (Detergent)                0.50
19        (Detergent)             (Shampoo)                0.55
20          (Shampoo)                (Soap)                0.50
21             (Soap)             (Shampoo)                0.35
22  (Coffee, Detergent)                (Milk)                0.30
23       (Coffee, Milk)           (Detergent)                0.30
24    (Detergent, Milk)              (Coffee)                0.25
25           (Coffee)     (Detergent, Milk)                0.50
26        (Detergent)        (Coffee, Milk)                0.55
27             (Milk)   (Coffee, Detergent)                0.40
```

53f614e

Fig 3: Apriori algorithm and rules

	consequent support	support	confidence	lift	leverage	conviction \
0	0.30	0.20	0.363636	1.212121	0.0350	1.100000
1	0.55	0.20	0.666667	1.212121	0.0350	1.350000
2	0.55	0.20	0.400000	0.727273	-0.0750	0.750000
3	0.50	0.20	0.363636	0.727273	-0.0750	0.785714
4	0.55	0.25	0.454545	0.826446	-0.0525	0.825000
5	0.55	0.25	0.454545	0.826446	-0.0525	0.825000
6	0.55	0.30	0.600000	1.090909	0.0250	1.125000
7	0.50	0.30	0.545455	1.090909	0.0250	1.100000
8	0.35	0.25	0.454545	1.298701	0.0575	1.191667
9	0.55	0.25	0.714286	1.298701	0.0575	1.575000
10	0.55	0.30	0.600000	1.090909	0.0250	1.125000
11	0.50	0.30	0.545455	1.090909	0.0250	1.100000
12	0.40	0.30	0.600000	1.500000	0.1000	1.500000
13	0.50	0.30	0.750000	1.500000	0.1000	2.000000
14	0.50	0.20	0.400000	0.800000	-0.0500	0.833333
15	0.50	0.20	0.400000	0.800000	-0.0500	0.833333
16	0.40	0.25	0.454545	1.136364	0.0300	1.100000
17	0.55	0.25	0.625000	1.136364	0.0300	1.200000
18	0.55	0.35	0.700000	1.272727	0.0750	1.500000
19	0.50	0.35	0.636364	1.272727	0.0750	1.375000
20	0.35	0.20	0.400000	1.142857	0.0250	1.083333
21	0.50	0.20	0.571429	1.142857	0.0250	1.166667
22	0.40	0.25	0.833333	2.083333	0.1300	3.600000
23	0.55	0.25	0.833333	1.515152	0.0850	2.700000
24	0.50	0.25	1.000000	2.000000	0.1250	inf
25	0.25	0.25	0.500000	2.000000	0.1250	1.500000
26	0.30	0.25	0.454545	1.515152	0.0850	1.283333
27	0.30	0.25	0.625000	2.083333	0.1300	1.866667

	zhangs_metric
0	0.388889
1	0.250000
2	-0.428571
3	-0.454545
4	-0.318182
5	-0.318182
6	0.166667
7	0.185185
8	0.511111
9	0.353846
10	0.166667
11	0.185185
12	0.666667
13	0.555556
14	-0.333333
15	-0.333333
16	0.266667
17	0.200000
18	0.428571
19	0.476190

Fig 4: Calculations and metrics

```
Frequent Itemsets (FP-Growth):
(     support                  itemsets
0     0.55              (Detergent)
1     0.55                 (Cereal)
2     0.50                (Shampoo)
3     0.50                 (Coffee)
4     0.30                  (Bread)
5     0.40                   (Milk)
6     0.35                   (Soap)
7     0.30                (Diapers)
8     0.20             (Toothpaste)
9     0.25      (Detergent, Cereal)
10    0.35     (Shampoo, Detergent)
11    0.30        (Shampoo, Cereal)
12    0.20         (Coffee, Cereal)
13    0.20        (Coffee, Shampoo)
14    0.30      (Coffee, Detergent)
15    0.20          (Cereal, Bread)
16    0.30           (Coffee, Milk)
17    0.25        (Detergent, Milk)
18    0.25  (Coffee, Detergent, Milk)
19    0.25           (Cereal, Soap)
20    0.20          (Shampoo, Soap),            antecedents          consequents  antecedent support  \
0            (Detergent)             (Cereal)                0.55
1               (Cereal)          (Detergent)                0.55
2              (Shampoo)          (Detergent)                0.50
3            (Detergent)            (Shampoo)                0.55
4              (Shampoo)             (Cereal)                0.50
5               (Cereal)            (Shampoo)                0.55
6               (Coffee)             (Cereal)                0.50
7               (Cereal)             (Coffee)                0.55
8               (Coffee)            (Shampoo)                0.50
9              (Shampoo)             (Coffee)                0.50
10              (Coffee)          (Detergent)                0.50
11           (Detergent)             (Coffee)                0.55
12              (Cereal)              (Bread)                0.55
13               (Bread)             (Cereal)                0.30
14              (Coffee)               (Milk)                0.50
15                (Milk)             (Coffee)                0.40
16           (Detergent)               (Milk)                0.55
17                (Milk)          (Detergent)                0.40
18   (Coffee, Detergent)               (Milk)                0.30
19        (Coffee, Milk)          (Detergent)                0.30
20     (Detergent, Milk)             (Coffee)                0.25
21              (Coffee)    (Detergent, Milk)                0.50
22           (Detergent)       (Coffee, Milk)                0.55
23                (Milk)  (Coffee, Detergent)                0.40
24              (Cereal)               (Soap)                0.55
25                (Soap)             (Cereal)                0.35
26             (Shampoo)               (Soap)                0.50
27                (Soap)            (Shampoo)                0.35
```

Fig 5: FP-Growth algorithm and rules

	consequent support	support	confidence	lift	leverage	conviction \
0	0.55	0.25	0.454545	0.826446	-0.0525	0.825000
1	0.55	0.25	0.454545	0.826446	-0.0525	0.825000
2	0.55	0.35	0.700000	1.272727	0.0750	1.500000
3	0.50	0.35	0.636364	1.272727	0.0750	1.375000
4	0.55	0.30	0.600000	1.090909	0.0250	1.125000
5	0.50	0.30	0.545455	1.090909	0.0250	1.100000
6	0.55	0.20	0.400000	0.727273	-0.0750	0.750000
7	0.50	0.20	0.363636	0.727273	-0.0750	0.785714
8	0.50	0.20	0.400000	0.800000	-0.0500	0.833333
9	0.50	0.20	0.400000	0.800000	-0.0500	0.833333
10	0.55	0.30	0.600000	1.090909	0.0250	1.125000
11	0.50	0.30	0.545455	1.090909	0.0250	1.100000
12	0.30	0.20	0.363636	1.212121	0.0350	1.100000
13	0.55	0.20	0.666667	1.212121	0.0350	1.350000
14	0.40	0.30	0.600000	1.500000	0.1000	1.500000
15	0.50	0.30	0.750000	1.500000	0.1000	2.000000
16	0.40	0.25	0.454545	1.136364	0.0300	1.100000
17	0.55	0.25	0.625000	1.136364	0.0300	1.200000
18	0.40	0.25	0.833333	2.083333	0.1300	3.600000
19	0.55	0.25	0.833333	1.515152	0.0850	2.700000
20	0.50	0.25	1.000000	2.000000	0.1250	inf
21	0.25	0.25	0.500000	2.000000	0.1250	1.500000
22	0.30	0.25	0.454545	1.515152	0.0850	1.283333
23	0.30	0.25	0.625000	2.083333	0.1300	1.866667
24	0.35	0.25	0.454545	1.298701	0.0575	1.191667
25	0.55	0.25	0.714286	1.298701	0.0575	1.575000
26	0.35	0.20	0.400000	1.142857	0.0250	1.083333
27	0.50	0.20	0.571429	1.142857	0.0250	1.166667

	zhangs_metric
0	-0.318182
1	-0.318182
2	0.428571
3	0.476190
4	0.166667
5	0.185185
6	-0.428571
7	-0.454545
8	-0.333333
9	-0.333333
10	0.166667
11	0.185185
12	0.388889
13	0.250000
14	0.666667
15	0.555556
16	0.266667
17	0.200000
18	0.742857
19	0.485714
20	0.666667

Fig 6: calculations and metrics for fp-growth

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/HemThumar/DM_midtermproj