Name:-Deeraj Kumar vecham Subbaraju
UCID:-dl585
SP25-CS634854 Data Mining

MIDTERM PROJECT

Title: Comparative Analysis of Apriori and Brute Force Algorithms with different Datasets

Abstract:

This project focuses on implementing data mining techniques to identify frequent itemsets and association rules using three different approaches: the brute force method and the Apriori algorithm. By applying these methods to multiple datasets, I compare their efficiency and accuracy to determine the most effective algorithm for analyzing retail shop data.

Introduction:

Data mining is the process of analyzing large amounts of data to uncover useful patterns or insights that can help solve problems or improve business decisions.

The brute force method is a straightforward approach that examines all possible solutions to determine the best one. It systematically tests every option, making it through but often slow.

The Apriori algorithm is a machine learning technique used to find frequently occurring itemsets and association rules in databases. It is commonly applied in areas like market basket analysis, disease detection, and customer behavior analysis. This algorithm helps predict the likelihood of different outcomes based on past data.

Implementation and workflow:

1. Take user input for minimum support, minimum confidence, and the dataset. The dataset consists of a mix of clothing and other retail items. Before processing, the data is cleaned by filtering unique items and arranging them in a predefined order to maintain accuracy.

2. Construct dictionaries to store item occurrences and identify frequently appearing items along with their counts.

3. Begin with single-item sets (K=1) and progressively expand to larger itemsets (K=2, K=3, etc.), identifying frequent patterns at each stage. This process continues until no more frequent itemsets

can be found.

4. Apply the brute force approach to generate all possible association rules. Only the rules that meet or exceed the given minimum support and confidence thresholds are selected, revealing important relationships between purchased items.

5. Implement the Apriori algorithm to extract frequent itemsets and association rules, providing alternative methods for pattern discovery.

6. Compare the results of the three approaches based on performance, efficiency, and accuracy to determine the most effective technique for analyzing the dataset.

Core Concepts and Principle:

Association Rules: Association rule learning is a machine learning technique that identifies meaningful relationships between variables within large datasets using a rule-based approach.

Support and Confidence: Support refers to the proportion of transactions that contain a specific itemset, calculated by dividing the number of occurrences by the total transactions. Confidence measures the probability of one itemset appearing when another itemset is present.

Frequent Itemset Discovery: This process involves identifying groups of items that commonly appear together in a dataset, helping to uncover patterns and relationships within the data.

Results:

The Apriori algorithm demonstrated moderate performance, completing the process in approximately 0.04 seconds. While it performed better than the brute force method, it struggled with the overhead of generating candidate itemsets, particularly when dealing with large datasets.

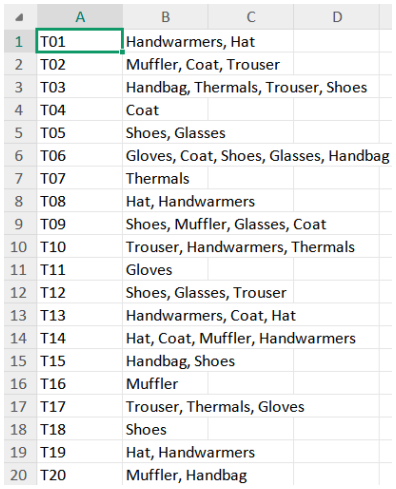
The brute force approach was the slowest, with an execution time of about 0.06 seconds. Its exhaustive search process made it the least efficient, especially for handling large datasets.

Conclusion:

This project successfully applies data mining concepts, principles, and techniques to analyze datasets. A comparative analysis is conducted between the three methods to determine the most effective approach for identifying frequent itemsets and association rules.

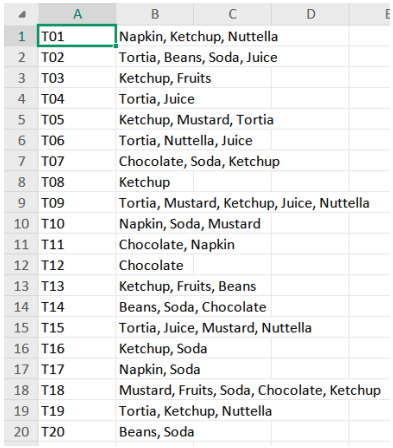
Dataset Screenshot:

1.Burlington dataset:



	A	B	C	D
1	T01	Handwarmers, Hat		
2	T02	Muffler, Coat, Trouser		
3	T03	Handbag, Thermals, Trouser, Shoes		
4	T04	Coat		
5	T05	Shoes, Glasses		
6	T06	Gloves, Coat, Shoes, Glasses, Handbag		
7	T07	Thermals		
8	T08	Hat, Handwarmers		
9	T09	Shoes, Muffler, Glasses, Coat		
10	T10	Trouser, Handwarmers, Thermals		
11	T11	Gloves		
12	T12	Shoes, Glasses, Trouser		
13	T13	Handwarmers, Coat, Hat		
14	T14	Hat, Coat, Muffler, Handwarmers		
15	T15	Handbag, Shoes		
16	T16	Muffler		
17	T17	Trouser, Thermals, Gloves		
18	T18	Shoes		
19	T19	Hat, Handwarmers		
20	T20	Muffler, Handbag		

2.Costco Dataset:



	A	B	C	D
1	T01	Napkin, Ketchup, Nuttella		
2	T02	Tortia, Beans, Soda, Juice		
3	T03	Ketchup, Fruits		
4	T04	Tortia, Juice		
5	T05	Ketchup, Mustard, Tortia		
6	T06	Tortia, Nuttella, Juice		
7	T07	Chocolate, Soda, Ketchup		
8	T08	Ketchup		
9	T09	Tortia, Mustard, Ketchup, Juice, Nuttella		
10	T10	Napkin, Soda, Mustard		
11	T11	Chocolate, Napkin		
12	T12	Chocolate		
13	T13	Ketchup, Fruits, Beans		
14	T14	Beans, Soda, Chocolate		
15	T15	Tortia, Juice, Mustard, Nuttella		
16	T16	Ketchup, Soda		
17	T17	Napkin, Soda		
18	T18	Mustard, Fruits, Soda, Chocolate, Ketchup		
19	T19	Tortia, Ketchup, Nuttella		
20	T20	Beans, Soda		

3.Juice bar Dataset:

	A	B	C	D	E
1	1	Water, Cocoa, Cider			
2	2	Cocoa, Water, Juice			
3	3	Cocktail			
4	4	Tea, Cocktail, Wine, Coffee, Water			
5	5	Cocktail, Juice, ProteinShake, Soda			
6	6	Tea, ProteinShake, Coffee, Soda			
7	7	Water			
8	8	Water, Cocoa, Wine			
9	9	Juice, Cocoa, Water, Coffee, Tea, Cider			
10	10	Cocoa, Tea, Water, Coffee, Wine			
11	11	Soda, Tea, Coffee, ProteinShake, Juice			
12	12	Cocoa, Cocktail, Juice, ProteinShake, Water			
13	13	Coffee, Tea, ProteinShake, Cocktail, Water, Cider			
14	14	Cocktail, Coffee, ProteinShake, Wine, Juice			
15	15	Soda, Coffee, Cider			
16	16	ProteinShake, Water, Coffee			
17	17	ProteinShake, Cocktail, Ciderss			
18	18	Coffee, Cocktail, Cocoa, Wine, ProteinShake, Water			
19	19	Juice, Coffee, Soda			
20	20	Soda			

4.Shoprite Dataset:

	A	B	C	D	E	F
1	1	Crackers, Nuts, Chips Cookies				
2	2	Pretzels, Pudding, MeatSticks, Spreads				
3	3	Pudding, Pretzels, Nuts, Crackers, Dips				
4	4	MeatSticks, Pretzels				
5	5	Chips Cookies, Pudding, Popcorn, Crackers, Dips, Pretzels, Spreads				
6	6	Chips Cookies, Pretzels, Pudding				
7	7	Pudding, Chips Cookies, Nuts, Popcorn, Pretzels, Dips				
8	8	MeatSticks, Pretzels, Dips				
9	9	Popcorn				
10	10	MeatSticks, Pudding, Crackers, Popcorn				
11	11	MeatSticks, Popcorn, Pretzels, Dips, Chips Cookies				
12	12	Chips Cookies, Crackers, Dips, Nuts, Pretzels, Popcorn				
13	13	MeatSticks				
14	14	Pretzels, Pudding, Nuts, Crackers				
15	15	Chips Cookies, Pretzels, Dips, Pudding				
16	16	Chips Cookies, Dips, Nuts, MeatSticks, Pudding, Pretzels				
17	17	Chips Cookies, Dips, Nuts, Crackers				
18	18	Crackers				
19	19	Pretzels, Chips Cookies, Crackers, Pudding, Popcorn, Dips				
20	20	Popcorn, Chips Cookies, Nuts				

5.Walmart:

	A	B	C	D
1	T01	Mustard, Glasses		
2	T02	Trouser, Mustard, Fruits, Shoes		
3	T03	Trouser, Mustard, Fruits, Shoes		
4	T04	Mustard, Gloves, Shoes		
5	T05	Trouser, Mustard, Fruits, Shoes		
6	T06	Trouser, Mustard, Fruits, Shoes		
7	T07	Beans, Mustard, Shoes		
8	T08	Trouser, Mustard, Fruits, Shoes		
9	T09	Trouser, Fruits, Shoes		
10	T10	Gloves, Fruits		
11	T11	Trouser, Mustard, Fruits, Shoes		
12	T12	Trouser, Soda		
13	T13	Beans, Mustard, Glasses, Shoes		
14	T14	Trouser, Mustard, Fruits, Shoes		
15	T15	Trouser, Gloves, Fruits		
16	T16	Beans, Fruits		
17	T17	Trouser, Mustard, Shoes		
18	T18	Trouser, Beans		
19	T19	Trouser, Mustard, Fruits		
20	T20	Trouser, Soda		

Code snippets:

Apriori algorithm:

```
def apriori_from_csv(file_name: str, minimumSup: float, minimumConfi: float):
    global aTime
    start_time=time.time()
    df = pd.read_csv(file_name, header=None)
    transactions = []
    for row in df.itertuples(index=False):
        transactions.append(row[1].split(", "))
    from mlxtend.preprocessing import TransactionEncoder
    encoder = TransactionEncoder()
    encoded_data = encoder.fit(transactions).transform(transactions)
    df = pd.DataFrame(encoded_data, columns=encoder.columns_)
    print(df)

    fi = apriori(df, min_support=minimumSup / 100, use_colnames=True)
    rules = association_rules(fi, metric="confidence", min_threshold=minimumConfi / 100)
    print("Frequent Itemsets:")
    print(fi)
    print("\nAssociation Rules:")
    print(rules)
    end_time= time.time()
    aTime= end_time-start_time
    print(f"Time taken to complete the process using FP-Growth method:{fpTime:.6f}")

apriori_from_csv(file_name, minimumSup, minimumConfi)
```

Brute force algorithm:

```python
def bruteforce(file_name: str, min_supp: int, min_conf: int):
    global bfTime
    global mini_conf
    mini_conf=min_conf

    start_time=time.time()
    with open(file_name, "r") as file_object:
        reader = csv.reader(file_object)
        all_tx = []
        counttot = 0
        support_of_all_item_set = {}
        c1 = {}
        item_set_list = []


        for row1 in reader:
            transaction_id = row1[0]
            items = row1[1].split(", ")
            all_tx.append(transaction_id)
            seen = set()
            for item in items:
                c1[(item,)] = c1.get((item,), 0) + 1
                seen.add(item)
            item_set_list.append(seen)
            counttot += 1

        frequent_set = {}
        rejected_set = []
        print()
        for i in c1:
            if (c1[i] / counttot) * 100 >= min_supp:
                frequent_set[i] = c1[i]
            else:
```

```python
    else:
        rejected_set.append(set(i))
support_of_all_item_set.update(c1)

list1 = [item[0] for item in frequent_set.keys()]
print()
print(itemprint(frequent_set, 1, counttot))

item_set_size = 1
while len(list1) > item_set_size:
    frequent_set1, rejected_set1 = frequentset(
        list1, rejected_set, item_set_list, item_set_size + 1,
        min_supp, counttot, support_of_all_item_set
    )
    if not frequent_set1:
        break
    item_list = [items for item_tuples in frequent_set1.keys() for items in item_tuples]
    list1 = list(set(item_list))
    rejected_set = rejected_set1
    frequent_set = frequent_set1
    item_set_size += 1

association_rules(frequent_set, support_of_all_item_set, min_conf)
end_time= time.time()
bfTime= end_time-start_time
print(f"Time taken to complete the process using brute force method:{bfTime:.6f}")
```

User input snippet:

```python
def get_user_input():
    global choice, minimumSup, minimumConfi, file_name

    print("Please select a dataset:")
    print("1. Juice Bar")
    print("2. Burlington")
    print("3. Costco")
    print("4. Walmart")
    print("5. ShopRite")

    choice = input("Enter the number corresponding to your choice: ")
    minimumSup = int(input("Enter minimum support as %: "))
    minimumConfi = int(input("Enter minimum confidence as %: "))

    file_names = {
        '1': 'juicebar.csv',
        '2': 'burlington.csv',
        '3': 'costco.csv',
        '4': 'walmart.csv',
        '5': 'shoprite.csv'
    }

    if choice in file_names:
        file_name = file_names[choice]
        print(f"You selected: {file_name}")
    else:
        print("Invalid choice. Please try again.")
        get_user_input()
```

Output:

```
PS C:\Users\deera\OneDrive\Desktop\Datamining_project> cd "C:\Users\deera\OneDrive\Desktop\Datamining_project\dl585_dmsmidtermproject"
PS C:\Users\deera\OneDrive\Desktop\Datamining_project\dl585_dmsmidtermproject> python dmsmidtermproject.py
Please select a dataset:
1. Juice Bar
2. Burlington
3. Costco
4. Walmart
5. ShopRite
Enter the number corresponding to your choice: 3
Enter minimum support as %: 30
Enter minimum confidence as %: 40
You selected: costco.csv


Frequent itemsets 1 iteration
('Ketchup',) 50.0
('Tortia',) 35.0
('Soda',) 40.0

None


Association Rule for itemset - ('Ketchup',)

Association Rule for itemset - ('Tortia',)

Association Rule for itemset - ('Soda',)

Time taken to complete the process using brute force method:0.014302
    Beans  Chocolate  Fruits  Juice  Ketchup  Mustard  Napkin  Nuttella   Soda  Tortia
0   False      False   False  False     True    False    True      True  False   False
1    True      False   False   True    False    False   False     False   True    True
2   False      False    True  False     True    False   False     False  False   False
3   False      False   False   True    False    False   False     False  False    True
4   False      False   False  False     True     True   False     False  False    True
5   False      False   False   True    False    False   False      True  False    True
6   False       True   False  False     True    False   False     False   True   False
10  False       True   False  False    False    False    True     False  False   False
11  False       True   False  False    False    False   False     False  False   False
12   True      False    True  False     True    False   False     False  False   False
13   True       True   False  False    False    False   False     False   True   False
14  False      False   False   True    False     True   False      True  False    True
15  False      False   False  False     True    False   False     False   True   False
16  False      False   False  False    False    False    True     False   True   False
17  False       True    True  False     True     True   False     False   True   False
18  False      False   False  False     True    False   False      True  False    True
19   True      False   False  False    False    False   False     False   True   False
Frequent Itemsets:
   support   itemsets
0     0.50  (Ketchup)
1     0.40     (Soda)
2     0.35   (Tortia)

Association Rules:
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, representativity, leverage, conviction, zhangs_metric, jaccard,
 certainty, kulczynski]
Index: []
Brute Force Time: 0.014302 seconds
Apriori Time: 2.486263 seconds
Brute Force is the fastest method.
PS C:\Users\deera\OneDrive\Desktop\Datamining_project\dl585_dmsmidtermproject>
```