NAME: Deepesh Singh Tomar
NJIT UCID: dt394
Email Address: dt394@njit.edu
10-Mar-2024
Professor: Yasser Abduallah
CS 634 104 Data Mining

Midterm Project Report

Implementation and Code Usage

ABSTRACT:-

This paper compares the efficacy and efficiency of three well-known market basket analysis algorithms: FP-Growth, Apriori, and Brute Force. The analysis is based on a thorough Python script. Market basket analysis plays a critical role in identifying patterns of item connection in big transaction datasets, which can lead to valuable insights that improve retail strategies, inventory control, and recommendation systems. The script assesses how well each algorithm performs in recognizing frequently occurring itemsets and producing association rules under predetermined support and confidence criteria by analyzing transaction data from multiple CSV files. The processing time of the algorithms and their capacity to identify significant correlations between objects are the main subjects of the analysis. Based on the results, it can be concluded that although the Brute Force approach offers a comprehensive search capability, the Apriori and FP-Growth algorithms perform faster without sacrificing the depth of analysis. In order to achieve the best possible balance between computing needs and analytical precision, these findings highlight the significance of selecting algorithms based on particular dataset features and intended analytical outputs.

INTRODUCTION:-

This Python script uses three different algorithms (Brute Force, Apriori, and FP-Growth) to perform market basket analysis on transaction data saved in CSV files. Finding common itemsets and producing association rules—two crucial steps in identifying associations between items in huge datasets—are handled differently by each method. The script compares the efficacy and efficiency of different algorithms, paying particular attention to the quantity of frequent itemsets and association rules produced as well as the amount of computing time needed for each technique.

Logic Behind the Code:-

1. **Data Preparation**: The CSV files are used to load transactions, and each transaction shows the combined purchase of all the items. These transactions are read by the `load_data` function, which then uses `encode_txns` to convert them into a one-hot encoded format that is compatible with the `mlxtend` library—a prerequisite for the Apriori algorithm.

2. **Finding Frequent Itemsets**:
   - **Brute Force Search**: In order to determine the frequency of each combination in the dataset, this procedure iterates through all possible item combinations. The itemset is deemed frequent if the frequency is greater than the designated support level. This approach is computationally costly and exhaustive.

```
In [9]: #code implementation for Brute force
        def brute_force_search(t_data, s_thresh):
            all_items = set(item for transaction in t_data for item in transaction)
            max_itemset_size = len(all_items)
            freq_sets = {}
            for l in range(1, max_itemset_size + 1):
                current_lvl_itemsets = list(itertools.combinations(all_items, l))
                current_lvl_freq = 0
                for itemset in current_lvl_itemsets:
                    frequency = sum(1 for transaction in t_data if set(itemset).issubset(set(transaction)))
                    if frequency / len(t_data) >= s_thresh:
                        freq_sets[itemset] = frequency
                        current_lvl_freq += 1
                if current_lvl_freq == 0:
                    break
            return freq_sets
```

   - **Apriori Algorithm**: refines the search space, making it better than brute force. It operates under the tenet that any subset of a frequent itemset needs to likewise be frequent. It starts with smaller itemsets and gradually expands them, discarding those that don't have enough support.

```
In [11]: #code implementation for apriori Algorithm
         def apriori_search(t_data, s_thresh):
             df_encoded = encode_txns(t_data)
             freq_itemsets = apriori(df_encoded, min_support=s_thresh, use_colnames=True)
             return freq_itemsets
```

   - **FP-Growth Algorithm**: eliminates candidate generation and uses a recursive divide-and-conquer strategy to further optimise the search for frequently occurring itemsets. The itemsets in the dataset are compactly represented by an FP-tree (Frequent Pattern Tree), which is then used to extract frequent itemsets straight from the tree.

In [12]:
```
#code implementation for FP-Growth Algorithm
def fpgrowth_search(t_data, s_thresh):
    min_count = int(s_thresh * len(t_data))
    patterns = pyfpgrowth.find_frequent_patterns(t_data, min_count)
    # Constructing the DataFrame with frozenset itemsets and support
    freq_sets = pd.DataFrame([(frozenset(k), v / len(t_data)) for k, v in patterns.items()], columns=['itemsets', 's
    return freq_sets
```

3. **Generating Association Rules**: The script first finds frequently occurring itemsets, then, using a specified confidence level, extracts association rules from these itemsets using the `generate_assoc_rules` function. According to these criteria, there is a certain degree of confidence that the presence of certain items (antecedents) in a transaction indicates the inclusion of other items (consequent).

In [13]:
```
def generate_assoc_rules(freq_sets, t_data, c_thresh):
    # No need for `support_only=True` if the DataFrame is structured correctly.
    if not freq_sets.empty:
        rules = association_rules(freq_sets, metric="confidence", min_threshold=c_thresh, support_only = True)
        return rules[['antecedents', 'consequents', 'support', 'confidence']]
    return pd.DataFrame()
```

4. **Performance Comparison**: Every algorithm is run on the transaction data by the `algo_comparison` function, which logs the number of frequently created itemsets and rules as well as the execution time. The trade-offs between each algorithm's efficiency and computing complexity are shown in this comparison.

In [14]:
```
# Compares the performance and output of Brute Force, Apriori, and FP-Growth algorithms on given transaction data.
def algo_comparison(fp, s_thresh, c_thresh):
    t_data = load_data(fp)

    # Brute Force
    t_start = time.time()
    bf_sets = brute_force_search(t_data, s_thresh)
    t_bf = time.time() - t_start
    print(bf_sets)
    # Apriori
    t_start = time.time()
    ap_sets = apriori_search(t_data, s_thresh)
    t_ap = time.time() - t_start
    ap_rules = generate_assoc_rules(ap_sets, t_data, c_thresh)
    print(ap_sets)
    # FP-Growth
    t_start = time.time()
    fp_sets = fpgrowth_search(t_data, s_thresh)
    t_fp = time.time() - t_start
    fp_rules = generate_assoc_rules(fp_sets, t_data, c_thresh)
    print(fp_sets)
    # Determine the fastest algorithm
    fastest_time = min(t_bf, t_ap, t_fp)
    fastest_algo = 'Brute Force' if fastest_time == t_bf else ('Apriori' if fastest_time == t_ap else 'FP-Growth')

    print(f"Brute Force: {len(bf_sets)} sets in {t_bf:.5f} sec.")
    print(f"Apriori: {len(ap_sets)} sets, {len(ap_rules)} rules in {t_ap:.5f} sec.")
    print(f"FP-Growth: {len(fp_sets)} sets, {len(fp_rules)} rules in {t_fp:.5f} sec.")
    print(f"The fastest algorithm for {fp} is {fastest_algo} with a time of {fastest_time:.5f} sec.\n")

    return bf_sets, ap_rules, fp_rules
```

Output and Analysis

For each dataset (`Amazon.csv`, `Best_Buy.csv`, `Generic.csv`, `K-Mart.csv`, `Nike.csv`), the script outputs the results of the analysis, including:
- The number of frequent itemsets found.

- The execution time for each algorithm.
- The number of association rules generated that meet the confidence threshold.

Example Output Analysis

- **Amazon.csv**: In the shortest amount of time, the Apriori and FP-Growth algorithms found five frequently occurring itemsets and produced two association rules each. Although it took longer, the brute force approach discovered the same number of itemsets. This illustrates how well Apriori and FP-Growth work with datasets that have a well-defined structure of item associations..

- **Best_Buy.csv:** Both optimised algorithms generated 2 rules from 7 itemsets with efficiency, similar to what Amazon does. The fact that the brute force method was much slower highlights the advantages of employing more complex algorithms with bigger datasets.

- **Generic.csv**: Five itemsets were found by all algorithms, and four rules were produced by FP-Growth and Apriori. This dataset demonstrates the algorithms' efficacy in a range of contexts by demonstrating their capacity to handle perfectly distributed data (support of 1.0 for item 'A').

- **K-Mart.csv**: There was just one itemset discovered, and none of the rules met the confidence threshold. This suggests that the threshold settings may be too strict or that there are sparse or weak relationships in the dataset..

- **Nike.csv**: demonstrated the ability of the algorithms to extract complex relationships in datasets with deeper item interactions. Apriori found 12 itemsets and 16 rules, whereas FP-Growth found 11 itemsets and 16 rules. This was the maximum number of rules generated.

```
Support (fraction): 0.5
Confidence (fraction): 0.5
Processing Amazon.csv
Brute Force: 5 sets in 0.00233 sec.
Apriori: 5 sets, 2 rules in 0.00425 sec.
FP-Growth: 5 sets, 2 rules in 0.00039 sec.
The fastest algorithm for Amazon.csv is FP-Growth with a time of 0.00039 sec.

AP: 2 rules, FP: 2 rules

Processing Best_Buy.csv
Brute Force: 7 sets in 0.00413 sec.
Apriori: 7 sets, 2 rules in 0.00111 sec.
FP-Growth: 7 sets, 2 rules in 0.00016 sec.
The fastest algorithm for Best_Buy.csv is FP-Growth with a time of 0.00016 sec.

AP: 2 rules, FP: 2 rules

Processing Generic.csv
Brute Force: 5 sets in 0.00005 sec.
Apriori: 5 sets, 4 rules in 0.00082 sec.
FP-Growth: 7 sets, 4 rules in 0.00011 sec.
The fastest algorithm for Generic.csv is Brute Force with a time of 0.00005 sec.

AP: 4 rules, FP: 4 rules

Processing K-Mart.csv
Brute Force: 1 sets in 0.00090 sec.
Apriori: 1 sets, 0 rules in 0.00066 sec.
FP-Growth: 1 sets, 0 rules in 0.00010 sec.
The fastest algorithm for K-Mart.csv is FP-Growth with a time of 0.00010 sec.

AP: 0 rules, FP: 0 rules

Processing Nike.csv
Brute Force: 12 sets in 0.00434 sec.
Apriori: 12 sets, 16 rules in 0.00115 sec.
FP-Growth: 11 sets, 16 rules in 0.00019 sec.
The fastest algorithm for Nike.csv is FP-Growth with a time of 0.00019 sec.

AP: 16 rules, FP: 16 rules
```

Conclusion

Without compromising the ability to find significant item correlations, the analysis shows how effective the Apriori and FP-Growth algorithms are compared to the brute force method, especially in terms of processing performance. Retail analytics, inventory control, and recommendation systems can all benefit from this information, which help companies better understand consumer behaviour and tailor promotions and product placement. The results highlight how crucial it is to choose the right algorithms depending on the properties of the dataset and the objectives of the analysis, striking a balance between the amount of computing power needed and the depth of understanding needed.

GitHub Link:-

https://github.com/DeepeshSinghTomar/Data_Mining_Mid_term_Project

Screenshots

Transaction ID	Transaction
Trans1	Decorative Pillows, Quilts, Embroidered Bedspread
Trans2	Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections, Bed Skirts, Bedspreads, Sheets
Trans3	Decorative Pillows, Quilts, Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections
Trans4	Kids Bedding, Bedding Collections, Sheets, Bedspreads, Bed Skirts
Trans5	Decorative Pillows, Kids Bedding, Bedding Collections, Sheets, Bed Skirts, Bedspreads
Trans6	Bedding Collections, Bedspreads, Bed Skirts, Sheets, Shams, Kids Bedding
Trans7	Decorative Pillows, Quilts
Trans8	Decorative Pillows, Quilts, Embroidered Bedspread
Trans9	Bedspreads, Bed Skirts, Shams, Kids Bedding, Sheets
Trans10	Quilts, Embroidered Bedspread, Bedding Collections
Trans11	Bedding Collections, Bedspreads, Bed Skirts, Kids Bedding, Shams, Sheets
Trans12	Decorative Pillows, Quilts
Trans13	Embroidered Bedspread, Shams
Trans14	Sheets, Shams, Bed Skirts, Kids Bedding
Trans15	Decorative Pillows, Quilts
Trans16	Decorative Pillows, Kids Bedding, Bed Skirts, Shams
Trans17	Decorative Pillows, Shams, Bed Skirts
Trans18	Quilts, Sheets, Kids Bedding
Trans19	Shams, Bed Skirts, Kids Bedding, Sheets
Trans20	Decorative Pillows, Bedspreads, Shams, Sheets, Bed Skirts, Kids Bedding

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

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

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

---

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