MIDTERM PROJECT

-Anish Panicker (ap2938)

Abstract:

In this project, I explore the use of three different algorithms to find frequent itemsets and generate association rules for retail transaction data: the Brute Force method, the Apriori algorithm, and FP-Growth algorithm. These algorithms help uncover patterns in customer purchasing behavior by identifying which items are often bought together. I created and analyzed five customized transactional datasets for different retail stores. Additionally, I compare the execution times of each algorithm to determine which is the most effective for analyzing large datasets.

Introduction:

In this project, I implemented three algorithms to find frequent itemsets and generate association rules from retail transaction data: the Brute Force method, the Apriori algorithm, and the FP-Growth algorithm. These algorithms help identify patterns in customer purchases, showing which items are often bought together.

I created five datasets, each containing 25 transactions with items typically found in stores like Amazon, BestBuy, Walmart, Target, and Kmart. The datasets represent common transactions with a variety of products. These datasets are used to test and compare the three algorithms.

The Brute Force method checks all possible combinations of items to find frequent itemsets. Although simple, it becomes inefficient as the number of items increases because it checks every possible combination.

The Apriori algorithm improves on this by using a more efficient approach. It eliminates itemsets that don't meet the minimum support early, which reduces the number of combinations that need to be checked. Apriori works by first finding frequent single items, then extending to larger itemsets.

The FP-Growth algorithm is faster than both Brute Force and Apriori. It uses a structure called an FP-tree to store the data in a compact form. This allows it to find frequent itemsets without generating all possible combinations, making it very efficient for large datasets.

For the implementation, I used Python libraries like 'pandas' for loading and processing the datasets, 'itertools' for generating item combinations, and 'mlxtend' for Apriori and FP-Growth.

In this project, the user can specify the minimum support and confidence levels to analyze the results. These parameters control how frequent an itemset must be to be considered and the strength of the association between items. By adjusting these, we can explore how the algorithms perform under different conditions.

Dataset Creation & Preprocessing Dataset:

I created five datasets, each representing transactions from Amazon, Kmart, BestBuy, Target, and Walmart. Each dataset contains 10 unique items and 25 transactions using these items. The transactions were built using example data provided, and I expanded on it to ensure a realistic set of transactions for each store. These datasets were designed to simulate typical purchases and help analyze patterns using the three algorithms in this project.

The data preprocessing begins by loading the dataset from a CSV file into a pandas DataFrame. Each transaction, initially a string of items, is split into a list of individual items. Next, all unique items across the transactions are extracted and sorted. The transactions are then transformed into a Boolean DataFrame, where each row represents a transaction, and each column represents an item, with True or False indicating the presence of an item. This structure prepares the data for applying association rule mining algorithms like Apriori, FP-Growth, and Brute Force.

```
dataset_choice = int(input("Enter the dataset number you want to load (1-5): "))

if dataset_choice in datasets:
    dataset_path = datasets[dataset_choice]
    df = pd.read_csv(dataset_path)
    print(f"\nLoaded dataset from: {dataset_path}")
    print(df.head())

    transactions = df['Items'].apply(lambda x: x.split(', ')).tolist()
    all_items = sorted(set(item for transaction in transactions for item in transaction))

    print("Transactions:", transactions)
    print("Number of Transactions:", len(transactions))
    print("All Unique Items:", all_items)
    print("Number of Unique Items:", len(all_items))
else:
    print("Invalid choice. Please select a number between 1 and 5.")

min_support = float(input("Enter the minimum support value: "))
min_confidence = float(input("Enter the minimum confidence value: "))

transaction_df = pd.DataFrame([{item: (item in transaction) for item in all_items} for transaction in transactio
```

Fig1:Data Preprocessing

```
Add code cell
⌘/Ctrl+M B
Datasets:
nt/amazon_items - Sheet1.csv
2: /content/bestbuy_items - Sheet1.csv
3: /content/kmart_items - Sheet1 (1).csv
4: /content/target_items - Sheet1 (1).csv
5: /content/walmart_items - Sheet1.csv
Enter the dataset number you want to load (1-5): 1

Loaded dataset from: /content/amazon_items - Sheet1.csv
   TransactionID                                  Items
0              1  Kindle, Cable, Stand, Smartwatch, Headset
1              2       Fire, Kindle, Echo, Charger, Speaker
2              3    Smartwatch, Stand, Cable, Charger, Fire
3              4      Echo, Kindle, Fire, Smartwatch, Cable
4              5       Laptop, Charger, Smartwatch, Headset
Transactions: [['Kindle', 'Cable', 'Stand', 'Smartwatch', 'Headset'], ['Fire', 'Kindle', 'Echo', 'Charger', 'Speaker'], ['Smartwatch', 'Stand', 'C
Number of Transactions: 25
All Unique Items: ['Cable', 'Charger', 'Echo', 'Fire', 'Headset', 'Kindle', 'Laptop', 'Smartwatch', 'Speaker', 'Stand']
Number of Unique Items: 10
Enter the minimum support value: 0.2
Enter the minimum confidence value: 0.5
```

Fig2: Inputs given by user , Support: 0.2 , Confidence: 0.5

Unique Items in Amazon Dataset:

- Cable
- Charger
- Echo
- Fire
- Headset
- Kindle
- Laptop
- Smartwatch
- Speaker
- Stand

TransactionID	Items
1	Kindle, Cable, Stand, Smartwatch, Headset
2	Fire, Kindle, Echo, Charger, Speaker
3	Smartwatch, Stand, Cable, Charger, Fire
4	Echo, Kindle, Fire, Smartwatch, Cable
5	Laptop, Charger, Smartwatch, Headset
6	Kindle, Cable, Speaker, Echo, Smartwatch
7	Stand, Fire, Speaker, Charger
8	Kindle, Headset, Cable, Echo
9	Laptop, Speaker, Echo, Fire, Smartwatch
10	Kindle, Charger, Headset, Smartwatch, Cable
11	Fire, Smartwatch, Speaker, Stand
12	Kindle, Headset, Stand, Cable
13	Smartwatch, Echo, Fire, Cable
14	Kindle, Charger, Echo, Smartwatch
15	Speaker, Laptop, Cable, Fire
16	Echo, Kindle, Headset, Charger
17	Smartwatch, Stand, Speaker, Echo
18	Fire, Cable, Charger, Smartwatch
19	Laptop, Kindle, Headset, Cable
20	Echo, Speaker, Stand, Smartwatch
21	Cable, Charger, Fire, Smartwatch
22	Headset, Kindle, Laptop, Speaker
23	Fire, Echo, Charger, Cable
24	Kindle, Echo, Smartwatch, Speaker
25	Stand, Charger, Smartwatch, Fire

Unique Items in Bestbuy Dataset:

- Antivirus
- Camera
- Case
- Desktop
- Flash
- Hard Drive
- Laptop
- Office
- Printer
- Speakers

TransactionID	Items
1	Desktop, Printer, Flash, Office, Speakers, Antivirus
2	Laptop, Flash, Office, Case, Antivirus
3	Laptop, Printer, Flash, Office, Antivirus, Case, Harddrive
4	Laptop, Printer, Flash, Antivirus, Harddrive, Case
5	Laptop, Flash, Case, Antivirus
6	Laptop, Printer, Flash, Office
7	Desktop, Printer, Flash, Office
8	Laptop, Harddrive, Antivirus
9	Desktop, Printer, Flash, Office, Case, Antivirus, Speakers, Harddrive
10	Camera, Laptop, Desktop, Printer, Flash, Office, Case, Antivirus, Harddrive, Speakers
11	Laptop, Desktop, Case, Harddrive, Speakers, Antivirus
12	Camera, Laptop, Case, Harddrive, Antivirus, Speakers
13	Camera, Speakers
14	Camera, Desktop, Printer, Flash, Office
15	Printer, Flash, Office, Antivirus, Case, Speakers, Harddrive
16	Camera, Flash, Office, Antivirus, Case, Harddrive, Speakers
17	Camera, Laptop, Case
18	Camera, Case, Speakers
19	Camera, Laptop, Printer, Flash, Office, Speakers, Case, Antivirus
20	Camera, Laptop, Speakers, Antivirus, Case
21	Desktop, Flash, Harddrive, Antivirus
22	Laptop, Speakers, Office, Printer
23	Case, Harddrive, Camera, Laptop
24	Printer, Antivirus, Desktop, Speakers
25	Flash, Case, Laptop, Camera

Unique Items in K-mart Dataset:

- Babywear
- Backpack
- Clothes
- Cookware
- Furniture
- Garden
- Kitchenware
- Shoes
- Bedding
- Toys

TransactionID	Items
1	Clothes, Shoes, Backpack, Kitchenware, Furniture
2	Toys, Shoes, Babywear, Kitchenware, Bedding
3	Furniture, Shoes, Clothes, Cookware, Garden
4	Toys, Furniture, Backpack, Shoes, Bedding
5	Kitchenware, Clothes, Shoes, Babywear
6	Shoes, Furniture, Clothes, Backpack, Cookware
7	Toys, Kitchenware, Clothes, Furniture
8	Garden, Furniture, Shoes, Cookware
9	Clothes, Toys, Shoes, Backpack, Bedding
10	Furniture, Garden, Clothes, Kitchenware, Shoes
11	Toys, Babywear, Shoes, Clothes
12	Clothes, Kitchenware, Furniture, Garden
13	Shoes, Toys, Backpack, Bedding
14	Kitchenware, Clothes, Furniture, Cookware, Bedding
15	Toys, Shoes, Clothes, Furniture
16	Garden, Babywear, Cookware, Clothes
17	Shoes, Furniture, Toys, Kitchenware
18	Clothes, Bedding, Garden, Cookware
19	Toys, Shoes, Clothes, Kitchenware
20	Furniture, Backpack, Garden, Clothes
21	Shoes, Kitchenware, Clothes, Garden
22	Furniture, Toys, Clothes, Bedding
23	Cookware, Babywear, Shoes, Furniture
24	Garden, Clothes, Furniture, Shoes
25	Babywear, Shoes, Furniture, Bedding

Unique Items in Walmart Dataset:

- Clothes
- Electronics
- Furniture
- Groceries
- Office
- Shoes
- Sporting
- Toys

TransactionID	Items
1	Electronics, Furniture, Groceries, Sporting, Office
2	Clothes, Groceries, Shoes, Sporting, Toys
3	Office, Electronics, Sporting, Furniture
4	Groceries, Electronics, Furniture, Shoes, Office
5	Sporting, Toys, Groceries, Furniture
6	Electronics, Groceries, Shoes, Sporting
7	Furniture, Office, Clothes, Shoes
8	Electronics, Sporting, Groceries, Office
9	Groceries, Sporting, Clothes, Toys
10	Electronics, Sporting, Office, Shoes
11	Clothes, Groceries, Furniture, Sporting
12	Office, Electronics, Furniture, Shoes
13	Toys, Sporting, Groceries, Electronics
14	Groceries, Clothes, Sporting, Toys
15	Shoes, Furniture, Office, Sporting
16	Groceries, Electronics, Clothes, Shoes
17	Toys, Electronics, Furniture, Sporting
18	Clothes, Groceries, Shoes, Toys
19	Office, Furniture, Electronics, Shoes
20	Sporting, Electronics, Clothes, Groceries
21	Shoes, Sporting, Electronics, Office
22	Furniture, Clothes, Toys, Groceries
23	Electronics, Sporting, Office, Shoes
24	Clothes, Groceries, Furniture, Toys
25	Office, Shoes, Electronics, Sporting

Unique Items in Target Dataset:

- Bedding
- Clothes
- Furniture
- Groceries
- Shampoo
- Shoes
- Toys
- Utensils

TransactionID	Items
1	Groceries, Shoes, Clothes, Utensils, Furniture
2	Toys, Bedding, shampoo, Furniture, Groceries
3	Clothes, Groceries, Shoes, Utensils
4	Toys, shampoo, Shoes, Furniture, Clothes
5	Groceries, Bedding, Utensils, Shoes
6	Furniture, Groceries, shampoo, Clothes
7	Toys, Shoes, Bedding, Furniture
8	shampoo, Groceries, Shoes, Clothes
9	Toys, Groceries, Shoes, Clothes
10	Bedding, Utensils, Furniture, Groceries
11	shampoo, Clothes, Shoes, Bedding
12	Groceries, Furniture, Shoes, Utensils
13	Toys, shampoo, Groceries, Clothes
14	Bedding, Utensils, Clothes, Shoes
15	Groceries, Shoes, shampoo, Clothes
16	Toys, Furniture, Bedding, Groceries
17	Utensils, Clothes, Furniture, Shoes
18	Groceries, shampoo, Furniture, Shoes
19	Clothes, Shoes, Groceries, shampoo
20	Bedding, Groceries, Utensils, Shoes
21	Toys, shampoo, Clothes, Furniture
22	Shoes, Groceries, Utensils, shampoo
23	Furniture, Clothes, Shoes, Groceries
24	Bedding, Furniture, Clothes, Utensils
25	Toys, Shoes, Groceries, Furniture

Steps To Run The Code :

Install Python and Required Packages:

- Ensure that Python is installed on your system.
- Open your terminal or command prompt and install the required packages by running the following command: pip install pandas mlxtend
- Upload the 5 datasets into the same local folder where the .ipynb or .py file is present
- Open the terminal or command prompt and navigate to the directory containing Python file and datasets.
- After doing this run this command python Anish_Panicker_midtermproj
- After running the program, it will show a list of datasets (Amazon, BestBuy, Kmart, Target, Walmart). Enter the corresponding number (1-5) for the dataset and press Enter.
- Input the support and confidence values, and the program will proceed to analyze the selected dataset.
- The program will first show the frequent itemsets generated using the Brute Force method, followed by Apriori and FP-Growth algorithms.
- It will also display the association rules for each algorithm based on the support and confidence thresholds entered earlier
- At the end of the process, the program will compare the execution times of the three algorithms (Brute Force, Apriori, and FP-Growth) and show which one was the fastest.
- It will also indicate which algorithm generated the most frequent itemsets.

Brute Force Algorithm:

Brute Force method checks all possible combinations of items to find frequent itemsets. Although simple, it becomes inefficient as the number of items increases because it checks every possible combination.
The function generate_frequent_itemsets uses the brute force method to find frequent itemsets. It starts by generating candidate itemsets of size k (beginning with 1-itemsets), and for each candidate, it calculates its support . If the support is greater than or equal to the user-specified min_support, the itemset is considered frequent and added to the list. The function continues increasing the size of the itemsets until no more frequent itemsets are found, then returns all frequent itemsets.

```python
def generate_candidates(itemset, length):
    """Generate all combinations of itemsets of a given length."""
    return list(itertools.combinations(itemset, length))

def calculate_support(itemset, transactions):
    """Calculate support of an itemset."""
    count = 0
    for transaction in transactions:
        if all(item in transaction for item in itemset):
            count += 1
    return count / len(transactions)
```

Fig 3: Functions to Generate Candidates and Calculate Support

I. Frequent Itemsets:

The total number of frequent itemsets generated at min_support = 0.2 are 31 for Amazon Dataset

Number of 1 Itemsets : 10 , Number of 2 Itemsets : 18 ,Number of 3 Itemsets : 3

```
Frequent Itemsets:
Itemset: ('Cable',), Support: 0.52
Itemset: ('Charger',), Support: 0.44
Itemset: ('Echo',), Support: 0.48
Itemset: ('Fire',), Support: 0.48
Itemset: ('Headset',), Support: 0.32
Itemset: ('Kindle',), Support: 0.48
Itemset: ('Laptop',), Support: 0.2
Itemset: ('Smartwatch',), Support: 0.64
Itemset: ('Speaker',), Support: 0.4
Itemset: ('Stand',), Support: 0.32
Itemset: ('Cable', 'Charger'), Support: 0.2
Itemset: ('Cable', 'Echo'), Support: 0.2
Itemset: ('Cable', 'Fire'), Support: 0.28
Itemset: ('Cable', 'Headset'), Support: 0.2
Itemset: ('Cable', 'Kindle'), Support: 0.28
Itemset: ('Cable', 'Smartwatch'), Support: 0.32
Itemset: ('Charger', 'Fire'), Support: 0.28
Itemset: ('Charger', 'Smartwatch'), Support: 0.28
Itemset: ('Echo', 'Fire'), Support: 0.2
Itemset: ('Echo', 'Kindle'), Support: 0.28
Itemset: ('Echo', 'Smartwatch'), Support: 0.32
Itemset: ('Echo', 'Speaker'), Support: 0.24
Itemset: ('Fire', 'Smartwatch'), Support: 0.32
Itemset: ('Fire', 'Speaker'), Support: 0.2
Itemset: ('Headset', 'Kindle'), Support: 0.28
Itemset: ('Kindle', 'Smartwatch'), Support: 0.24
Itemset: ('Smartwatch', 'Speaker'), Support: 0.24
Itemset: ('Smartwatch', 'Stand'), Support: 0.24
Itemset: ('Cable', 'Fire', 'Smartwatch'), Support: 0.2
Itemset: ('Cable', 'Headset', 'Kindle'), Support: 0.2
Itemset: ('Echo', 'Smartwatch', 'Speaker'), Support: 0.2
```

Fig 4: Frequent Itemsets using Brute Force

```python
def generate_frequent_itemsets(transactions, all_items, min_support):

    frequent_itemsets = []
    k = 1
    while True:
        candidate_itemsets = generate_candidates(all_items, k)
        current_frequent_itemsets = []


        for itemset in candidate_itemsets:
            support = calculate_support(itemset, transactions)
            if support >= min_support:
                current_frequent_itemsets.append((itemset, support))

        if not current_frequent_itemsets:
            break


        frequent_itemsets.extend(current_frequent_itemsets)
        k += 1

    return frequent_itemsets


start_time = time.time()
frequent_itemsets = generate_frequent_itemsets(transactions, all_items, min_support)
end_time = time.time()
bf_time = end_time - start_time

print("\nFrequent Itemsets:")
for itemset, support in frequent_itemsets:
    print(f"Itemset: {itemset}, Support: {support}")
```

Fig 5: Frequent Itemsets Bruteforce code

II. Association Rules:

This function `generate_association_rules` generates association rules from the frequent itemsets. For each itemset with at least two items, it identifies all possible pairs and calculates the confidence for each rule. If the confidence of a rule meets or exceeds the user-specified `min_confidence`, the rule is added to the list. Function returns all the rules that meet the confidence threshold.**There are 37 association rules in total**

```
Association Rules (Brute Force):
Rule: ('Cable',) -> ('Fire',), Support: 0.28, Confidence: 0.5384615384615385
Rule: ('Fire',) -> ('Cable',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Headset',) -> ('Cable',), Support: 0.2, Confidence: 0.625
Rule: ('Cable',) -> ('Kindle',), Support: 0.28, Confidence: 0.5384615384615385
Rule: ('Kindle',) -> ('Cable',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6153846153846154
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.32, Confidence: 0.5
Rule: ('Charger',) -> ('Fire',), Support: 0.28, Confidence: 0.6363636363636365
Rule: ('Fire',) -> ('Charger',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.28, Confidence: 0.6363636363636365
Rule: ('Echo',) -> ('Kindle',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Kindle',) -> ('Echo',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6666666666666667
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.32, Confidence: 0.5
Rule: ('Echo',) -> ('Speaker',), Support: 0.24, Confidence: 0.5
Rule: ('Speaker',) -> ('Echo',), Support: 0.24, Confidence: 0.6
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6666666666666667
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.32, Confidence: 0.5
Rule: ('Speaker',) -> ('Fire',), Support: 0.2, Confidence: 0.5
Rule: ('Headset',) -> ('Kindle',), Support: 0.28, Confidence: 0.8750000000000001
Rule: ('Kindle',) -> ('Headset',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.5
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.6
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.75
Rule: ('Cable', 'Fire') -> ('Smartwatch',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Cable', 'Smartwatch') -> ('Fire',), Support: 0.2, Confidence: 0.625
Rule: ('Fire', 'Smartwatch') -> ('Cable',), Support: 0.2, Confidence: 0.625
Rule: ('Headset',) -> ('Cable', 'Kindle'), Support: 0.2, Confidence: 0.625
Rule: ('Cable', 'Headset') -> ('Kindle',), Support: 0.2, Confidence: 1.0
Rule: ('Cable', 'Kindle') -> ('Headset',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Headset', 'Kindle') -> ('Cable',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Speaker',) -> ('Echo', 'Smartwatch'), Support: 0.2, Confidence: 0.5
Rule: ('Echo', 'Smartwatch') -> ('Speaker',), Support: 0.2, Confidence: 0.625
Rule: ('Echo', 'Speaker') -> ('Smartwatch',), Support: 0.2, Confidence: 0.8333333333333334
Rule: ('Smartwatch', 'Speaker') -> ('Echo',), Support: 0.2, Confidence: 0.8333333333333334
```

Fig 6: Association rule using Bruteforce min confidence : 0.5

```python
def generate_association_rules(frequent_itemsets, min_confidence):
    rules = []
    for itemset, support in frequent_itemsets:
        if len(itemset) < 2:
            continue
        for i in range(1, len(itemset)):
            antecedents = list(itertools.combinations(itemset, i))
            for antecedent in antecedents:
                consequent = tuple(item for item in itemset if item not in antecedent)
                antecedent_support = calculate_support(antecedent, transactions)
                if antecedent_support > 0:
                    confidence = support / antecedent_support
                    if confidence >= min_confidence:
                        rules.append((antecedent, consequent, support, confidence))
    return rules

association_rules_bf = generate_association_rules(frequent_itemsets, min_confidence)
print("\nAssociation Rules (Brute Force):")
for antecedent, consequent, support, confidence in association_rules_bf:
    print(f"Rule: {antecedent} -> {consequent}, Support: {support}, Confidence: {confidence}")
```

Fig 7: Association rules Bruteforce code

Apriori Algorithm:

The Apriori algorithm improves on this by using a more efficient approach. It eliminates itemsets that don't meet the minimum support early, which reduces the number of combinations that need to be checked. Apriori works by first finding frequent single items, then extending to larger itemsets.

In this section, I used the `apriori` function from the `mlxtend` library to generate frequent itemsets based on the user-defined minimum support. The time taken to execute the Apriori algorithm is measured and stored in `apriori_time`. Once the frequent itemsets are generated, the `association_rules` function is used to derive association rules based on the specified minimum confidence. The frequent itemsets and their corresponding association rules are printed, along with the support and confidence values for each rule. The execution time for the Apriori algorithm is an important aspect of the analysis, allowing us to compare its efficiency with other algorithms.

```python
start_time = time.time()
frequent_itemsets_apriori = apriori(transaction_df, min_support=min_support, use_colnames=True)
end_time = time.time()
apriori_time = end_time - start_time

association_rules_apriori = association_rules(frequent_itemsets_apriori, metric="confidence", min_threshold=min_confidence)

print("\nFrequent Itemsets (Apriori):")
print(frequent_itemsets_apriori)
print("\nAssociation Rules (Apriori):")
for _, row in association_rules_apriori.iterrows():
    print(f"Rule: {tuple(row['antecedents'])} -> {tuple(row['consequents'])}, Support: {row['support']:.4f}, Confidence: {row['confidence']
```

Fig 8: Code for Apriori Algorithm

I. Frequent Itemsets:

The total number of frequent itemsets generated at min_support = 0.2 are 31 for Amazon Dataset

Number of 1 Itemsets : 10 , Number of 2 Itemsets : 18 ,Number of 3 Itemsets : 3

Frequent Itemsets (Apriori):

	support	itemsets
0	0.52	(Cable)
1	0.44	(Charger)
2	0.48	(Echo)
3	0.48	(Fire)
4	0.32	(Headset)
5	0.48	(Kindle)
6	0.20	(Laptop)
7	0.64	(Smartwatch)
8	0.40	(Speaker)
9	0.32	(Stand)
10	0.20	(Charger, Cable)
11	0.20	(Echo, Cable)
12	0.28	(Cable, Fire)
13	0.20	(Headset, Cable)
14	0.28	(Kindle, Cable)
15	0.32	(Smartwatch, Cable)
16	0.28	(Charger, Fire)
17	0.28	(Charger, Smartwatch)
18	0.20	(Echo, Fire)
19	0.28	(Kindle, Echo)
20	0.32	(Smartwatch, Echo)
21	0.24	(Speaker, Echo)
22	0.32	(Smartwatch, Fire)
23	0.20	(Speaker, Fire)
24	0.28	(Kindle, Headset)
25	0.24	(Smartwatch, Kindle)
26	0.24	(Speaker, Smartwatch)
27	0.24	(Smartwatch, Stand)
28	0.20	(Smartwatch, Cable, Fire)
29	0.20	(Kindle, Headset, Cable)
30	0.20	(Speaker, Smartwatch, Echo)

Fig9: Frequent Itemsets generated by Apriori Algorithm Support: 0.2

II. Association Rules:

There are 37 association rules in total in Amazon dataset when confidence is 0.5

```
Association Rules (Apriori):
Rule: ('Cable',) -> ('Fire',), Support: 0.2800, Confidence: 0.5385
Rule: ('Fire',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Headset',) -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Kindle',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Cable',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5385
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.3200, Confidence: 0.5000
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6154
Rule: ('Charger',) -> ('Fire',), Support: 0.2800, Confidence: 0.6364
Rule: ('Fire',) -> ('Charger',), Support: 0.2800, Confidence: 0.5833
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.2800, Confidence: 0.6364
Rule: ('Kindle',) -> ('Echo',), Support: 0.2800, Confidence: 0.5833
Rule: ('Echo',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5833
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.3200, Confidence: 0.5000
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Speaker',) -> ('Echo',), Support: 0.2400, Confidence: 0.6000
Rule: ('Echo',) -> ('Speaker',), Support: 0.2400, Confidence: 0.5000
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.3200, Confidence: 0.5000
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Speaker',) -> ('Fire',), Support: 0.2000, Confidence: 0.5000
Rule: ('Kindle',) -> ('Headset',), Support: 0.2800, Confidence: 0.5833
Rule: ('Headset',) -> ('Kindle',), Support: 0.2800, Confidence: 0.8750
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.5000
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.6000
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.7500
Rule: ('Smartwatch', 'Cable') -> ('Fire',), Support: 0.2000, Confidence: 0.6250
Rule: ('Smartwatch', 'Fire') -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable', 'Fire') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.7143
Rule: ('Kindle', 'Headset') -> ('Cable',), Support: 0.2000, Confidence: 0.7143
Rule: ('Kindle', 'Cable') -> ('Headset',), Support: 0.2000, Confidence: 0.7143
Rule: ('Headset', 'Cable') -> ('Kindle',), Support: 0.2000, Confidence: 1.0000
Rule: ('Headset',) -> ('Kindle', 'Cable'), Support: 0.2000, Confidence: 0.6250
Rule: ('Speaker', 'Smartwatch') -> ('Echo',), Support: 0.2000, Confidence: 0.8333
Rule: ('Speaker', 'Echo') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.8333
Rule: ('Smartwatch', 'Echo') -> ('Speaker',), Support: 0.2000, Confidence: 0.6250
Rule: ('Speaker',) -> ('Smartwatch', 'Echo'), Support: 0.2000, Confidence: 0.5000
```

Fig10 Association Rules generated by Apriori Confidence: 0.5

FP-Tree Algorithm:

The FP-Growth algorithm is faster than both Brute Force and Apriori. It uses a structure called an FP-tree to store the data in a compact form. This allows it to find frequent itemsets without generating all possible combinations, making it very efficient for large datasets.
I applied the FP-Growth algorithm using the FP-growth function from the ‘mlxtend’ library. The algorithm identifies frequent itemsets based on the user-specified minimum support. The time taken to execute FP-Growth is measured and stored in fpgrowth_time, which allows for comparison with other algorithms. After generating the frequent itemsets, the association_rules function is used to compute association rules based on the given minimum confidence. The frequent itemsets and their corresponding association rules, along with their support and confidence values, are printed. The execution time provides insight into the efficiency of the FP-Growth algorithm.

```python
start_time = time.time()
frequent_itemsets_fpgrowth = fpgrowth(transaction_df, min_support=min_support, use_colnames=True)
end_time = time.time()
fpgrowth_time = end_time - start_time

association_rules_fpgrowth = association_rules(frequent_itemsets_fpgrowth, metric="confidence", min_threshold=min_confidence)

print("\nFrequent Itemsets (FP-Growth):")
print(frequent_itemsets_fpgrowth)
print("\nAssociation Rules (FP-Growth):")
for _, row in association_rules_fpgrowth.iterrows():
    print(f"Rule: {tuple(row['antecedents'])} -> {tuple(row['consequents'])}, Support: {row['support']:.4f}, Confidence: {row['confidence
```

Fig 11: Code for FP-Tree Algorithm

I. Frequent Itemsets:

The total number of frequent itemsets generated at min_support = 0.2 are 31 for Amazon Dataset

Number of 1 Itemsets : 10 , Number of 2 Itemsets : 18 ,Number of 3 Itemsets : 3

```
Frequent Itemsets (FP-Growth):
    support                    itemsets
0      0.64                 (Smartwatch)
1      0.52                       (Cable)
2      0.48                      (Kindle)
3      0.32                       (Stand)
4      0.32                     (Headset)
5      0.48                        (Fire)
6      0.48                        (Echo)
7      0.44                     (Charger)
8      0.40                     (Speaker)
9      0.20                      (Laptop)
10     0.32           (Smartwatch, Cable)
11     0.28               (Kindle, Cable)
12     0.24           (Smartwatch, Kindle)
13     0.24           (Smartwatch, Stand)
14     0.28             (Kindle, Headset)
15     0.20              (Headset, Cable)
16     0.20      (Kindle, Headset, Cable)
17     0.32            (Smartwatch, Fire)
18     0.28                (Cable, Fire)
19     0.20     (Smartwatch, Cable, Fire)
20     0.28                (Kindle, Echo)
21     0.20                 (Echo, Fire)
22     0.32           (Smartwatch, Echo)
23     0.20                (Echo, Cable)
24     0.28               (Charger, Fire)
25     0.28         (Charger, Smartwatch)
26     0.20              (Charger, Cable)
27     0.24               (Speaker, Echo)
28     0.20               (Speaker, Fire)
29     0.24         (Speaker, Smartwatch)
30     0.20   (Speaker, Smartwatch, Echo)
```

Fig12: Frequent Itemsets generated by Apriori Algorithm Support: 0.2

II. Association Rules:

There are 37 association rules in total in Amazon dataset when confidence is 0.5

```
Association Rules (FP-Growth):
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.3200, Confidence: 0.5000
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6154
Rule: ('Kindle',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Cable',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5385
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.500
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.7500
Rule: ('Kindle',) -> ('Headset',), Support: 0.2800, Confidence: 0.5833
Rule: ('Headset',) -> ('Kindle',), Support: 0.2800, Confidence: 0.8750
Rule: ('Headset',) -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Kindle', 'Headset') -> ('Cable',), Support: 0.2000, Confidence:
Rule: ('Kindle', 'Cable') -> ('Headset',), Support: 0.2000, Confidence:
Rule: ('Headset', 'Cable') -> ('Kindle',), Support: 0.2000, Confidence:
Rule: ('Headset',) -> ('Kindle', 'Cable'), Support: 0.2000, Confidence:
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.3200, Confidence: 0.5000
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Cable',) -> ('Fire',), Support: 0.2800, Confidence: 0.5385
Rule: ('Fire',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Smartwatch', 'Cable') -> ('Fire',), Support: 0.2000, Confidence:
Rule: ('Smartwatch', 'Fire') -> ('Cable',), Support: 0.2000, Confidence:
Rule: ('Cable', 'Fire') -> ('Smartwatch',), Support: 0.2000, Confidence:
Rule: ('Kindle',) -> ('Echo',), Support: 0.2800, Confidence: 0.5833
Rule: ('Echo',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5833
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.3200, Confidence: 0.5000
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Charger',) -> ('Fire',), Support: 0.2800, Confidence: 0.6364
Rule: ('Fire',) -> ('Charger',), Support: 0.2800, Confidence: 0.5833
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.2800, Confidence: 0.63
Rule: ('Speaker',) -> ('Echo',), Support: 0.2400, Confidence: 0.6000
Rule: ('Echo',) -> ('Speaker',), Support: 0.2400, Confidence: 0.5000
Rule: ('Speaker',) -> ('Fire',), Support: 0.2000, Confidence: 0.5000
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.60
Rule: ('Speaker', 'Smartwatch') -> ('Echo',), Support: 0.2000, Confidenc
Rule: ('Speaker', 'Echo') -> ('Smartwatch',), Support: 0.2000, Confidenc
Rule: ('Smartwatch', 'Echo') -> ('Speaker',), Support: 0.2000, Confidenc
Rule: ('Speaker',) -> ('Smartwatch', 'Echo'), Support: 0.2000, Confidenc
```

Fig13 Association Rules generated by FP-Tree Confidence: 0.5

Comparative Study:

The code compares the execution times of the three algorithms:Brute Force, Apriori, and FP-Growth by printing the time taken for each. The min function is used to identify the fastest algorithm based on the recorded times. The result shows which algorithm was the quickest to run, helping in evaluating the efficiency of each method.

```
print("\nPerformance Comparison:")
print(f"Brute Force Time: {bf_time:.4f} seconds")
print(f"Apriori Time: {apriori_time:.4f} seconds")
print(f"FP-Growth Time: {fpgrowth_time:.4f} seconds")


fastest_algorithm = min([('Brute Force', bf_time), ('Apriori', apriori_time), ('FP-Growth', fpgrowth_time)], key=lambda x
print(f"The fastest algorithm is {fastest_algorithm[0]} with a time of {fastest_algorithm[1]:.4f} seconds.")
```

Fig14: Code to calculate the time taken by each algorithm

We see that the fastest algorithm is FP-Tree algorithm.

```
Brute Force Time: 0.0082 seconds
Apriori Time: 0.0133 seconds
FP-Growth Time: 0.0063 seconds
The fastest algorithm is FP-Growth with a time of 0.0063 seconds.
```

Fig15: Time taken by each algorithm and the fastest algorithm

Executing the project on command prompt:

Step1: User input for dataset, minimum support , confidence

```
PS C:\Users\archi\Downloads\Anish_Panicker_midtermproj\Anish_Panicker_midtermproj> python .\anish_panicker_midtermproj.py
Available Datasets:
1: amazon_items.csv
2: bestbuy_items.csv
3: kmart_items.csv
4: target_items.csv
5: walmart_items.csv
Enter the dataset number you want to load (1-5): 1

Loaded dataset from: amazon_items.csv
   TransactionID                                Items
0              1  Kindle, Cable, Stand, Smartwatch, Headset
1              2       Fire, Kindle, Echo, Charger, Speaker
2              3    Smartwatch, Stand, Cable, Charger, Fire
3              4      Echo, Kindle, Fire, Smartwatch, Cable
4              5       Laptop, Charger, Smartwatch, Headset
Transactions: [['Kindle', 'Cable', 'Stand', 'Smartwatch', 'Headset'], ['Fire', 'Kindle', 'Echo', 'Charger', 'Speaker'], ['Smartwatch'
, 'Stand', 'Cable', 'Charger', 'Fire'], ['Echo', 'Kindle', 'Fire', 'Smartwatch', 'Cable'], ['Laptop', 'Charger', 'Smartwatch', 'Heads
et'], ['Kindle', 'Cable', 'Speaker', 'Echo', 'Smartwatch'], ['Stand', 'Fire', 'Speaker', 'Charger'], ['Kindle', 'Headset', 'Cable', '
Echo'], ['Laptop', 'Speaker', 'Echo', 'Fire', 'Smartwatch'], ['Kindle', 'Charger', 'Headset', 'Smartwatch', 'Cable'], ['Fire', 'Smart
watch', 'Speaker', 'Stand'], ['Kindle', 'Headset', 'Stand', 'Cable'], ['Smartwatch', 'Echo', 'Fire', 'Cable'], ['Kindle', 'Charger',
'Echo', 'Smartwatch'], ['Speaker', 'Laptop', 'Cable', 'Fire'], ['Echo', 'Kindle', 'Headset', 'Charger'], ['Smartwatch', 'Stand', 'Spe
aker', 'Echo'], ['Fire', 'Cable', 'Charger', 'Smartwatch'], ['Laptop', 'Kindle', 'Headset', 'Cable'], ['Echo', 'Speaker', 'Stand', 'S
martwatch'], ['Cable', 'Charger', 'Fire', 'Smartwatch'], ['Headset', 'Kindle', 'Laptop', 'Speaker'], ['Fire', 'Echo', 'Charger', 'Cab
le'], ['Kindle', 'Echo', 'Smartwatch', 'Speaker'], ['Stand', 'Charger', 'Smartwatch', 'Fire']]
Number of Transactions: 25
All Unique Items: ['Cable', 'Charger', 'Echo', 'Fire', 'Headset', 'Kindle', 'Laptop', 'Smartwatch', 'Speaker', 'Stand']
Number of Unique Items: 10
Enter the minimum support value: 0.2
Enter the minimum confidence value: 0.5
```

Step2: Frequent itemsets generated by Brute Force

```
Frequent Itemsets:
Itemset: ('Cable',), Support: 0.52
Itemset: ('Charger',), Support: 0.44
Itemset: ('Echo',), Support: 0.48
Itemset: ('Fire',), Support: 0.48
Itemset: ('Headset',), Support: 0.32
Itemset: ('Kindle',), Support: 0.48
Itemset: ('Laptop',), Support: 0.2
Itemset: ('Smartwatch',), Support: 0.64
Itemset: ('Speaker',), Support: 0.4
Itemset: ('Stand',), Support: 0.32
Itemset: ('Cable', 'Charger'), Support: 0.2
Itemset: ('Cable', 'Echo'), Support: 0.2
Itemset: ('Cable', 'Fire'), Support: 0.28
Itemset: ('Cable', 'Headset'), Support: 0.2
Itemset: ('Cable', 'Kindle'), Support: 0.28
Itemset: ('Cable', 'Smartwatch'), Support: 0.32
Itemset: ('Charger', 'Fire'), Support: 0.28
Itemset: ('Charger', 'Smartwatch'), Support: 0.28
Itemset: ('Echo', 'Fire'), Support: 0.2
Itemset: ('Echo', 'Kindle'), Support: 0.28
Itemset: ('Echo', 'Smartwatch'), Support: 0.32
Itemset: ('Echo', 'Speaker'), Support: 0.24
Itemset: ('Fire', 'Smartwatch'), Support: 0.32
Itemset: ('Fire', 'Speaker'), Support: 0.2
Itemset: ('Headset', 'Kindle'), Support: 0.28
```

Step3: Association rules for brute force:

```
Association Rules (Brute Force):
Rule: ('Cable',) -> ('Fire',), Support: 0.28, Confidence: 0.5384615384615385
Rule: ('Fire',) -> ('Cable',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Headset',) -> ('Cable',), Support: 0.2, Confidence: 0.625
Rule: ('Cable',) -> ('Kindle',), Support: 0.28, Confidence: 0.5384615384615385
Rule: ('Kindle',) -> ('Cable',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6153846153846154
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.32, Confidence: 0.5
Rule: ('Charger',) -> ('Fire',), Support: 0.28, Confidence: 0.6363636363636365
Rule: ('Fire',) -> ('Charger',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.28, Confidence: 0.6363636363636365
Rule: ('Echo',) -> ('Kindle',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Kindle',) -> ('Echo',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6666666666666667
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.32, Confidence: 0.5
Rule: ('Echo',) -> ('Speaker',), Support: 0.24, Confidence: 0.5
Rule: ('Speaker',) -> ('Echo',), Support: 0.24, Confidence: 0.6
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.32, Confidence: 0.6666666666666667
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.32, Confidence: 0.5
Rule: ('Speaker',) -> ('Fire',), Support: 0.2, Confidence: 0.5
Rule: ('Headset',) -> ('Kindle',), Support: 0.28, Confidence: 0.8750000000000001
Rule: ('Kindle',) -> ('Headset',), Support: 0.28, Confidence: 0.5833333333333334
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.5
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.6
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.24, Confidence: 0.75
Rule: ('Cable', 'Fire') -> ('Smartwatch',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Cable', 'Smartwatch') -> ('Fire',), Support: 0.2, Confidence: 0.625
Rule: ('Fire', 'Smartwatch') -> ('Cable',), Support: 0.2, Confidence: 0.625
Rule: ('Headset',) -> ('Cable', 'Kindle'), Support: 0.2, Confidence: 0.625
Rule: ('Cable', 'Headset') -> ('Kindle',), Support: 0.2, Confidence: 1.0
Rule: ('Cable', 'Kindle') -> ('Headset',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Headset', 'Kindle') -> ('Cable',), Support: 0.2, Confidence: 0.7142857142857143
Rule: ('Speaker',) -> ('Echo', 'Smartwatch'), Support: 0.2, Confidence: 0.5
Rule: ('Echo', 'Smartwatch') -> ('Speaker',), Support: 0.2, Confidence: 0.625
Rule: ('Echo', 'Speaker') -> ('Smartwatch',), Support: 0.2, Confidence: 0.8333333333333334
```

Step4: Frequent itemsets generated by Apriori

```
Frequent Itemsets (Apriori):
    support                    itemsets
0      0.52                     (Cable)
1      0.44                    (Charger)
2      0.48                       (Echo)
3      0.48                       (Fire)
4      0.32                    (Headset)
5      0.48                     (Kindle)
6      0.20                     (Laptop)
7      0.64                  (Smartwatch)
8      0.40                     (Speaker)
9      0.32                       (Stand)
10     0.20             (Cable, Charger)
11     0.20                (Echo, Cable)
12     0.28                (Fire, Cable)
13     0.20             (Cable, Headset)
14     0.28              (Cable, Kindle)
15     0.32          (Cable, Smartwatch)
16     0.28              (Fire, Charger)
17     0.28        (Smartwatch, Charger)
18     0.20                 (Fire, Echo)
19     0.28               (Echo, Kindle)
20     0.32           (Echo, Smartwatch)
21     0.24              (Echo, Speaker)
22     0.32           (Fire, Smartwatch)
23     0.20              (Fire, Speaker)
24     0.28            (Headset, Kindle)
25     0.24         (Kindle, Smartwatch)
26     0.24        (Speaker, Smartwatch)
27     0.24          (Smartwatch, Stand)
28     0.20    (Fire, Cable, Smartwatch)
29     0.20     (Cable, Headset, Kindle)
30     0.20  (Echo, Speaker, Smartwatch)
```

Step5: Association rules for Apriori:

```
Association Rules (Apriori):
Rule: ('Fire',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Cable',) -> ('Fire',), Support: 0.2800, Confidence: 0.5385
Rule: ('Headset',) -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5385
Rule: ('Kindle',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6154
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.3200, Confidence: 0.5000
Rule: ('Fire',) -> ('Charger',), Support: 0.2800, Confidence: 0.5833
Rule: ('Charger',) -> ('Fire',), Support: 0.2800, Confidence: 0.6364
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.2800, Confidence: 0.6364
Rule: ('Echo',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5833
Rule: ('Kindle',) -> ('Echo',), Support: 0.2800, Confidence: 0.5833
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.3200, Confidence: 0.5000
Rule: ('Echo',) -> ('Speaker',), Support: 0.2400, Confidence: 0.5000
Rule: ('Speaker',) -> ('Echo',), Support: 0.2400, Confidence: 0.6000
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.3200, Confidence: 0.5000
Rule: ('Speaker',) -> ('Fire',), Support: 0.2000, Confidence: 0.5000
Rule: ('Headset',) -> ('Kindle',), Support: 0.2800, Confidence: 0.8750
Rule: ('Kindle',) -> ('Headset',), Support: 0.2800, Confidence: 0.5833
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.5000
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.6000
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.7500
Rule: ('Fire', 'Cable') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.7143
Rule: ('Fire', 'Smartwatch') -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable', 'Smartwatch') -> ('Fire',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable', 'Headset') -> ('Kindle',), Support: 0.2000, Confidence: 1.0000
Rule: ('Cable', 'Kindle') -> ('Headset',), Support: 0.2000, Confidence: 0.7143
Rule: ('Headset', 'Kindle') -> ('Cable',), Support: 0.2000, Confidence: 0.7143
Rule: ('Headset',) -> ('Cable', 'Kindle'), Support: 0.2000, Confidence: 0.6250
Rule: ('Echo', 'Speaker') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.8333
Rule: ('Echo', 'Smartwatch') -> ('Speaker',), Support: 0.2000, Confidence: 0.6250
Rule: ('Speaker', 'Smartwatch') -> ('Echo',), Support: 0.2000, Confidence: 0.8333
Rule: ('Speaker',) -> ('Echo', 'Smartwatch'), Support: 0.2000, Confidence: 0.5000
```

Step6: Frequent itemsets generated by FP-Tree

```
Frequent Itemsets (FP-Growth):
    support                    itemsets
0      0.64                 (Smartwatch)
1      0.52                       (Cable)
2      0.48                      (Kindle)
3      0.32                       (Stand)
4      0.32                     (Headset)
5      0.48                        (Fire)
6      0.48                        (Echo)
7      0.44                     (Charger)
8      0.40                     (Speaker)
9      0.20                      (Laptop)
10     0.32          (Cable, Smartwatch)
11     0.28              (Cable, Kindle)
12     0.24         (Kindle, Smartwatch)
13     0.24          (Smartwatch, Stand)
14     0.28            (Headset, Kindle)
15     0.20             (Cable, Headset)
16     0.20     (Cable, Headset, Kindle)
17     0.32           (Fire, Smartwatch)
18     0.28                (Fire, Cable)
19     0.20    (Fire, Cable, Smartwatch)
20     0.28               (Echo, Kindle)
21     0.20                (Fire, Echo)
22     0.32           (Echo, Smartwatch)
23     0.20                (Echo, Cable)
24     0.28              (Fire, Charger)
25     0.28        (Smartwatch, Charger)
26     0.20             (Cable, Charger)
27     0.24              (Echo, Speaker)
28     0.20              (Fire, Speaker)
29     0.24        (Speaker, Smartwatch)
30     0.20  (Echo, Speaker, Smartwatch)
```

Step7: Association rules for FP-Tree:

```
Association Rules (FP-Growth):
Rule: ('Cable',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6154
Rule: ('Smartwatch',) -> ('Cable',), Support: 0.3200, Confidence: 0.5000
Rule: ('Cable',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5385
Rule: ('Kindle',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Kindle',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.5000
Rule: ('Stand',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.7500
Rule: ('Headset',) -> ('Kindle',), Support: 0.2800, Confidence: 0.8750
Rule: ('Kindle',) -> ('Headset',), Support: 0.2800, Confidence: 0.5833
Rule: ('Headset',) -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable', 'Headset') -> ('Kindle',), Support: 0.2000, Confidence: 1.0000
Rule: ('Cable', 'Kindle') -> ('Headset',), Support: 0.2000, Confidence: 0.7143
Rule: ('Headset', 'Kindle') -> ('Cable',), Support: 0.2000, Confidence: 0.7143
Rule: ('Headset',) -> ('Cable', 'Kindle'), Support: 0.2000, Confidence: 0.6250
Rule: ('Fire',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Smartwatch',) -> ('Fire',), Support: 0.3200, Confidence: 0.5000
Rule: ('Fire',) -> ('Cable',), Support: 0.2800, Confidence: 0.5833
Rule: ('Cable',) -> ('Fire',), Support: 0.2800, Confidence: 0.5385
Rule: ('Fire', 'Cable') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.7143
Rule: ('Fire', 'Smartwatch') -> ('Cable',), Support: 0.2000, Confidence: 0.6250
Rule: ('Cable', 'Smartwatch') -> ('Fire',), Support: 0.2000, Confidence: 0.6250
Rule: ('Echo',) -> ('Kindle',), Support: 0.2800, Confidence: 0.5833
Rule: ('Kindle',) -> ('Echo',), Support: 0.2800, Confidence: 0.5833
Rule: ('Echo',) -> ('Smartwatch',), Support: 0.3200, Confidence: 0.6667
Rule: ('Smartwatch',) -> ('Echo',), Support: 0.3200, Confidence: 0.5000
Rule: ('Fire',) -> ('Charger',), Support: 0.2800, Confidence: 0.5833
Rule: ('Charger',) -> ('Fire',), Support: 0.2800, Confidence: 0.6364
Rule: ('Charger',) -> ('Smartwatch',), Support: 0.2800, Confidence: 0.6364
Rule: ('Echo',) -> ('Speaker',), Support: 0.2400, Confidence: 0.5000
Rule: ('Speaker',) -> ('Echo',), Support: 0.2400, Confidence: 0.6000
Rule: ('Speaker',) -> ('Fire',), Support: 0.2000, Confidence: 0.5000
Rule: ('Speaker',) -> ('Smartwatch',), Support: 0.2400, Confidence: 0.6000
Rule: ('Echo', 'Speaker') -> ('Smartwatch',), Support: 0.2000, Confidence: 0.8333
Rule: ('Echo', 'Smartwatch') -> ('Speaker',), Support: 0.2000, Confidence: 0.6250
```

Step 8: Comparing the number of itemsets created by each algorithm

```
Number of Frequent Itemsets Generated:
Brute Force: 31
Apriori: 31
FP-Growth: 31
The algorithm that generated the most frequent itemsets is Brute Force with 31 itemsets.
```

Conclusion:

In conclusion, this project highlights the use of three key data mining algorithms Brute Force, Apriori, and FP-Growth to identify frequent itemsets and generate association rules from retail transaction data. By applying these algorithms to custom datasets and analyzing their performance, we were able to extract meaningful patterns in customer purchases. We also find that FP-Tree is most suitable for large datasets and takes the least amount of time.