Midterm Project Report

NAME: Charan Reddy Katta
NJIT UCID: ck366
Email Address: ck366@njit.edu
03-10-2024
Professor: Yasser Abduallah
CS 634 (105) Data Mining

Implementation and Code Usage

Title: Comparative Analysis of Data Mining Algorithms

Abstract:

This project undertakes the implementation and comparative assessment of three distinct algorithms for frequent itemset mining and association rule generation: brute force, Apriori, and FP-Growth. Additionally, the project involves the creation of transactional databases and the derivation of association rules based on user-defined parameters

Introduction:

The Comparative Analysis of Data Mining Algorithms project aims to explore and evaluate the effectiveness of three distinct algorithms for frequent itemset mining and association rule generation. Data mining plays a crucial role in uncovering valuable patterns and insights from large datasets, making it a cornerstone in various domains including retail, finance, and healthcare. In this project, we focus on three key algorithms: brute force, Apriori, and FP-Growth, each offering unique approaches to extracting frequent itemset and deriving association rules. Through this comparative analysis, we seek to understand the strengths and weaknesses of each algorithm in terms of efficiency and accuracy, providing insights into their applicability in real-world scenarios.

Core Concepts and Principles:

Frequent Itemset Discovery: The Apriori Algorithm is all about finding frequent item sets – groups of items that frequently show up together in transactions. These item sets give us a peek into what customers tend to buy together, helping us understand their shopping habits and preferences.

Support and Confidence:

Support and confidence are crucial metrics in data mining. Support tells us how often an item or itemset appears in transactions, showing its importance. Confidence measures the likelihood of one item being bought when another is already purchased, indicating how strong the connection between items is. These metrics are our compass in analyzing data and making sense of it.

Association Rules:

Strong association rules help us identify which items are commonly purchased together. These rules are good for refining sales strategies, like making recommendations to customers based on their purchase history.

Project Workflow:

<u>Our project follows a structured path, guided by the Apriori Algorithm:</u>

Data Loading and Preprocessing:

We kick off by loading transaction data from a retail dataset. Each transaction lists the items bought by a customer. To make sure our data is clean and accurate, we tidy it up by removing duplicate items and sorting them out in a specific order.

Determining Minimum Support and Confidence:

User input is key in data mining. We ask the user for their preferred minimum support and confidence levels to filter out less important patterns from our analysis.

Iterating Through Candidate Item sets:

We apply the Apriori Algorithm by generating candidate item sets of increasing sizes. Starting with single items, we move on to pairs, triplets, and so forth. This iterative process is a bit like brute force – we are trying out all combinations of items.

Calculating Support Counts:

For each candidate itemset, we calculate its support by tallying up how many transactions contained in that itemset. We keep the item sets that meet our minimum support threshold and discard the rest.

Calculating Confidence:

Next, we assess the confidence of association rules to gauge the strength of connections between items. This involves comparing the support values of individual items and item sets.

Generating Association Rules:

We extract association rules that meet both the minimum support and confidence requirements. These rules offer valuable insights into which items tend to go hand in hand during purchases.

Results and Evaluation:

We evaluate the project's success based on metrics like support, confidence, and the association rules we've generated. We also compare our custom Apriori Algorithm implementation with existing libraries to ensure its reliability and accuracy.

Conclusion:

The project successfully implemented and compared three different algorithms for frequent itemset mining and association rule generation: brute force, Apriori, and FP-Growth.The results showed that both Apriori and FP-Growth produced the same number of frequent itemsets, while the brute force method did not find any. In terms of performance, FP-Growth was the fastest, followed by Apriori, and then the brute force method. Overall, this project demonstrated the practical application and comparison of different algorithms in data mining.

Screenshots:

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

CHARAN_KATTA_MID_PROJ

Midterm_Project_Items_Datasets_Examples-2		
Midterm_Project_Items_Datasets_Examples-2		
database1.csv'	Amazon	
Item #	Item Name	
1	A Beginner's Guide	
2	Java: The Complete Reference	
3	Java For Dummies	
4	Android Programming: The Big Nerd Ranch	
5	Head First Java 2nd Edition	
6	Beginning Programming with Java	
7	Java 8 Pocket Guide	
8	C++ Programming in Easy Steps	
9	Effective Java (2nd Edition)	
10	HTML and CSS: Design and Build Websites	
Transaction ID	Transaction	
Trans1	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch	
Trans2	A Beginner's Guide, Java: The Complete Reference, Java For Dummies	
Trans3	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition	
Trans4	Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,	
Trans5	Android Programming: The Big Nerd Ranch, Beginning Programming with Java, Java 8 Pocket Guide	
Trans6	A Beginner's Guide, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition	
Trans7	A Beginner's Guide, Head First Java 2nd Edition , Beginning Programming with Java	
Trans8	Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch,	
Trans9	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition , Beginning Programming with Java,	
Trans10	Beginning Programming with Java, Java 8 Pocket Guide, C++ Programming in Easy Steps	
Trans11	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Android Programming: The Big Nerd Ranch	
Trans12	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, HTML and CSS: Design and Build Websites	
Trans13	A Beginner's Guide, Java: The Complete Reference, Java For Dummies, Java 8 Pocket Guide,	HTML
Trans14	Java For Dummies, Android Programming: The Big Nerd Ranch, Head First Java 2nd Edition	

	A	B	C
38	**database2.csv'**	**Best Buy**	
39	**Item #**	Item Name	
40	**1**	Digital Camera	
41	**2**	Lab Top	
42	**3**	Desk Top	
43	**4**	Printer	
44	**5**	Flash Drive	
45	**6**	Microsoft Office	
46	**7**	Speakers	
47	**8**	Lab Top Case	
48	**9**	Anti-Virus	
49	**10**	External Hard-Drive	
50			
51			
52	**Transaction ID**	Transaction	
53			
54	**Trans1**	Desk Top, Printer, Flash Drive, Microsoft Office, Speakers, Anti-Virus	
55	**Trans2**	Lab Top, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus	
56	**Trans3**	Lab Top, Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive	
57	**Trans4**	Lab Top, Printer, Flash Drive, Anti-Virus, External Hard-Drive, Lab Top Case	
58	**Trans5**	Lab Top, Flash Drive, Lab Top Case, Anti-Virus	
59	**Trans6**	Lab Top, Printer, Flash Drive, Microsoft Office	
60	**Trans7**	Desk Top, Printer, Flash Drive, Microsoft Office	
61	**Trans8**	Lab Top, External Hard-Drive, Anti-Virus	
62	**Trans9**	Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, Speakers, External Hard-Drive	
63	**Trans10**	Digital Camera , Lab Top, Desk Top, Printer, Flash Drive, Microsoft Office, Lab Top Case, Anti-Virus, External Hard-Drive, Speakers	
64	**Trans11**	Lab Top, Desk Top, Lab Top Case, External Hard-Drive, Speakers, Anti-Virus	
65	**Trans12**	Digital Camera , Lab Top, Lab Top Case, External Hard-Drive, Anti-Virus, Speakers	
66	**Trans13**	Digital Camera , Speakers	
67	**Trans14**	Digital Camera , Desk Top, Printer, Flash Drive, Microsoft Office	
68	**Trans15**	Printer, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, Speakers, External Hard-Drive	
69	**Trans16**	Digital Camera, Flash Drive, Microsoft Office, Anti-Virus, Lab Top Case, External Hard-Drive, Speakers	
70	**Trans17**	Digital Camera , Lab Top, Lab Top Case	
71	**Trans18**	Digital Camera , Lab Top Case, Speakers	
72	**Trans19**	Digital Camera , Lab Top, Printer, Flash Drive, Microsoft Office, Speakers, Lab Top Case, Anti-Virus	
73	**Trans20**	Digital Camera , Lab Top, Speakers, Anti-Virus, Lab Top Case	
74			

Text Best Buy

Sheet 1

	A	B	C
74			
75	**database3.csv'**	K-mart	
76	**Item #**	Item Name	
77	**1**	Quilts	
78	**2**	Bedspreads	
79	**3**	Decorative Pillows	
80	**4**	Bed Skirts	
81	**5**	Sheets	
82	**6**	Shams	
83	**7**	Bedding Collections	
84	**8**	Kids Bedding	
85	**9**	Embroidered	
86	**10**	Bedspread	
87	**11**	Towels	
88			
89	**Transaction ID**	Transaction	
90	**Trans1**	Decorative Pillows, Quilts, Embroidered Bedspread	
91	**Trans2**	Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections, Bed Skirts, Bedspreads, Sheets	
92	**Trans3**	Decorative Pillows, Quilts, Embroidered Bedspread, Shams, Kids Bedding, Bedding Collections	
93	**Trans4**	Kids Bedding, Bedding Collections, Sheets, Bedspreads, Bed Skirts	
94	**Trans5**	Decorative Pillows, Kids Bedding, Bedding Collections, Sheets, Bed Skirts, Bedspreads	
95	**Trans6**	Bedding Collections, Bedspreads, Bed Skirts, Sheets, Shams, Kids Bedding	
96	**Trans7**	Decorative Pillows, Quilts	
97	**Trans8**	Decorative Pillows, Quilts, Embroidered Bedspread	
98	**Trans9**	Bedspreads, Bed Skirts, Shams, Kids Bedding, Sheets	
99	**Trans10**	Quilts, Embroidered Bedspread, Bedding Collections	
100	**Trans11**	Bedding Collections, Bedspreads, Bed Skirts, Kids Bedding, Shams, Sheets	
101	**Trans12**	Decorative Pillows, Quilts	
102	**Trans13**	Embroidered Bedspread, Shams	
103	**Trans14**	Sheets, Shams, Bed Skirts, Kids Bedding	
104	**Trans15**	Decorative Pillows, Quilts	
105	**Trans16**	Decorative Pillows, Kids Bedding, Bed Skirts, Shams	
106	**Trans17**	Decorative Pillows, Shams, Bed Skirts	
107	**Trans18**	Quilts, Sheets, Kids Bedding	
108	**Trans19**	Shams, Bed Skirts, Kids Bedding, Sheets	
109	**Trans20**	Decorative Pillows, Bedspreads, Shams, Sheets, Bed Skirts, Kids Bedding	
110			
111	**database4.csv'**	Nike	

Text Best Buy

Sheet 1

	A	B	C
111	**database4.csv'**	Nike	
112	**Item #**	Item Name	
113	**1**	Running Shoe	
114	**2**	Soccer Shoe	
115	**3**	Socks	
116	**4**	Swimming Shirt	
117	**5**	Dry Fit V-Nick	
118	**6**	Rash Guard	
119	**7**	Sweatshirts	
120	**8**	Hoodies	
121	**9**	Tech Pants	
122	**10**	Modern Pants	
123			
124	**Transaction ID**	Transaction	
125	**Trans1**	Running Shoe, Socks, Sweatshirts, Modern Pants	
126	**Trans2**	Running Shoe, Socks, Sweatshirts	
127	**Trans3**	Running Shoe, Socks, Sweatshirts, Modern Pants	
128	**Trans4**	Running Shoe, Sweatshirts, Modern Pants	
129	**Trans5**	Running Shoe, Socks, Sweatshirts, Modern Pants, Soccer Shoe	
130	**Trans6**	Running Shoe, Socks, Sweatshirts	
131	**Trans7**	Running Shoe, Socks, Sweatshirts, Modern Pants, Tech Pants, Rash Guard, Hoodies	
132	**Trans8**	Swimming Shirt, Socks, Sweatshirts	
133	**Trans9**	Swimming Shirt, Rash Guard, Dry Fit V-Nick, Hoodies, Tech Pants	
134	**Trans10**	Swimming Shirt, Rash Guard, Dry	
135	**Trans11**	Swimming Shirt, Rash Guard, Dry Fit V-Nick	
136	**Trans12**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Hoodies, Tech Pants, Dry Fit V-Nick	
137	**Trans13**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Tech Pants, Dry Fit V-Nick, Hoodies	
138	**Trans14**	Running Shoe, Swimming Shirt, Rash Guard, Tech Pants, Hoodies, Dry Fit V-Nick	
139	**Trans15**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Dry Fit V-Nick, Rash Guard, Tech Pants	
140	**Trans16**	Swimming Shirt, Soccer Shoe, Hoodies, Dry Fit V-Nick, Tech Pants, Rash Guard	
141	**Trans17**	Running Shoe, Socks	
142	**Trans18**	Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Rash Guard, Tech Pants, Dry Fit V-Nick	
143	**Trans19**	Running Shoe, Swimming Shirt, Rash Guard	
144	**Trans20**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Tech Pants, Rash Guard, Dry Fit V-Nick	
145			
146	**database5.csv'**	Generic	
147	**Item #**	Item Name	
148	**1**	A	

Text | Best Buy

Sheet 1

	A	B	C
	Trans6	Running Shoe, Socks, Sweatshirts	
131	**Trans7**	Running Shoe, Socks, Sweatshirts, Modern Pants, Tech Pants, Rash Guard, Hoodies	
132	**Trans8**	Swimming Shirt, Socks, Sweatshirts	
133	**Trans9**	Swimming Shirt, Rash Guard, Dry Fit V-Nick, Hoodies, Tech Pants	
134	**Trans10**	Swimming Shirt, Rash Guard, Dry	
135	**Trans11**	Swimming Shirt, Rash Guard, Dry Fit V-Nick	
136	**Trans12**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Hoodies, Tech Pants, Dry Fit V-Nick	
137	**Trans13**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Rash Guard, Tech Pants, Dry Fit V-Nick, Hoodies	
138	**Trans14**	Running Shoe, Swimming Shirt, Rash Guard, Tech Pants, Hoodies, Dry Fit V-Nick	
139	**Trans15**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Dry Fit V-Nick, Rash Guard, Tech Pants	
140	**Trans16**	Swimming Shirt, Soccer Shoe, Hoodies, Dry Fit V-Nick, Tech Pants, Rash Guard	
141	**Trans17**	Running Shoe, Socks	
142	**Trans18**	Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Rash Guard, Tech Pants, Dry Fit V-Nick	
143	**Trans19**	Running Shoe, Swimming Shirt, Rash Guard	
144	**Trans20**	Running Shoe, Swimming Shirt, Socks, Sweatshirts, Modern Pants, Soccer Shoe, Hoodies, Tech Pants, Rash Guard, Dry Fit V-Nick	
145			
146	**database5.csv'**	Generic	
147	**Item #**	Item Name	
148	**1**	A	

	A	B	C
149	**2**	B	
150	**3**	C	
151	**4**	D	
152	**5**	E	
153	**6**	F	
154			
155			
156	**Transaction ID**	Transaction	
157	**Trans1**	A, B, C	
158	**Trans2**	A, B, C	
159	**Trans3**	A, B, C, D	
160	**Trans4**	A, B, C, D, E	
161	**Trans5**	A, B, D, E	
162	**Trans6**	A, D, E	
163	**Trans7**	A, E	
164	**Trans8**	A, E	
165	**Trans9**	A, C, E	
166	**Trans10**	A, C, E	
167	**Trans11**	A, C, E	

Text | Best Buy

Below are screenshots of the code from python file:

It prompts users to select their desired store. Initially, it reads CSV files and validates user inputs. It begins by initializing dictionaries for Candidate and Frequent Item sets.

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [27]:
```python
# import the necessary modules
import random
import csv
import os
import itertools
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, fpgrowth
import time
```

In [15]:
```
## Part 1: Data Preparation
```

In [16]:
```python
# List of items seen in supermarkets
items = ['Milk', 'Cheese', 'Yogurt', 'Chicken', 'Beef', 'Bread', 'Chips', 'Cookies', 'Soda', 'Juice']

# Create a transaction
def transaction(items):
    return random.sample(items, random.randint(1, len(items)))

# Create a database with the transactions
def database(items, transactions, filename):
    with open(filename, 'w', newline='') as file:
        writer = csv.writer(file)
        for _ in range(transactions):
            writer.writerow(transaction(items))
```

In [17]:
```python
# For the initial database
database(items, 20, 'database1.csv')

# Creating 4 additional, different databases
for i in range(2, 6):
    database(items, 20, f'database{i}.csv')

# Test creation of database
print("Databases have been created successfully.")
```

```
Databases have been created successfully.
```

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [18]:
```
## Part 2: Algorithm Implementation and Comparison
```

In [19]:
```
### Brute Force Algorithm
```

In [20]:
```python
# Read transactions from a CSV file
def read_transactions(filename):
    with open(filename, 'r') as file:
        reader = csv.reader(file)
        return list(reader)

# Count the frequency of an itemset in the transactions
def count_frequency(itemset, transactions):
    return sum(1 for transaction in transactions if set(itemset).issubset(transaction))

# Generate all possible itemsets of a certain size
def generate_itemsets(items, size):
    return list(itertools.combinations(items, size))

# Identify frequent itemsets using the brute force method
def find_frequent_itemsets(items, transactions, min_frequency):
    size = 1
    while True:
        itemsets = generate_itemsets(items, size)
        frequent_itemsets = [itemset for itemset in itemsets if count_frequency(itemset, transactions) >= min_freque
        if not frequent_itemsets:
            break
        size += 1
    return frequent_itemsets

# Generate association rules from the frequent itemsets
def generate_association_rules(frequent_itemsets, min_confidence):
    rules = []
    for itemset in frequent_itemsets:
        for i in range(1, len(itemset)):
            for antecedent in itertools.combinations(itemset, i):
                consequent = tuple(item for item in itemset if item not in antecedent)
                confidence = count_frequency(itemset, transactions) / count_frequency(antecedent, transactions)
                if confidence >= min_confidence:
                    rules.append((antecedent, consequent))
    return rules
```

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
    return rules

# Read the transactions from the CSV file
transactions = read_transactions('database1.csv')

# Find the frequent itemsets
frequent_itemsets = find_frequent_itemsets(items, transactions, min_frequency=10)

# Generate the association rules
rules = generate_association_rules(frequent_itemsets, min_confidence=0.5)

print("Association rules generated successfully.")
```

```
Association rules generated successfully.
```

In [21]:

```
### Brute Force Algorithm
```

In [22]:

```
# Read transactions from a CSV file
def read_transactions(filename):
    with open(filename, 'r') as file:
        reader = csv.reader(file)
        return list(reader)

# Count the frequency of an itemset in the transactions
def count_frequency(itemset, transactions):
    return sum(1 for transaction in transactions if set(itemset).issubset(transaction))

# Generate all possible itemsets of a certain size
def generate_itemsets(items, size):
    return list(itertools.combinations(items, size))

# Identify frequent itemsets using the brute force method
def find_frequent_itemsets(items, transactions, min_frequency):
    size = 1
    while True:
        itemsets = generate_itemsets(items, size)
        frequent_itemsets = [itemset for itemset in itemsets if count_frequency(itemset, transactions) >= min_freque
        if not frequent_itemsets:
            break
        size += 1
```

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [23]:

```
### Apriori and FP-Growth
```

In [24]:

```
# Read the databases
def read_all_databases(database_filenames):
    all_transactions = []
    for filename in database_filenames:
        transactions = read_transactions(filename)
        all_transactions.extend(transactions)
    return all_transactions

# List of database filenames
database_filenames = ['database1.csv', 'database2.csv', 'database3.csv', 'database4.csv', 'database5.csv']

# Read all the transactions from all databases
all_transactions = read_all_databases(database_filenames)

# Convert the transactions into a DataFrame
te = TransactionEncoder()
te_ary = te.fit(all_transactions).transform(all_transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Minimum support
min_support = 0.1

# Minimum confidence
min_confidence = 0.5

# Run the brute force algorithm and measure the time
start = time.time()
frequent_itemsets_brute_force = find_frequent_itemsets(items, all_transactions, min_support)
rules_brute_force = generate_association_rules(frequent_itemsets_brute_force, min_confidence)
end = time.time()
print(f"Brute force method took {end - start} seconds.")

# Run the Apriori algorithm and measure the time
start = time.time()
frequent_itemsets_apriori = apriori(df, min_support=min_support, use_colnames=True)
end = time.time()
print(f"Apriori algorithm took {end - start} seconds.")

# Run the FP-Growth algorithm and measure the time
```

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [23]:
```
### Apriori and FP-Growth
```

In [24]:
```
# Read the databases
def read_all_databases(database_filenames):
    all_transactions = []
    for filename in database_filenames:
        transactions = read_transactions(filename)
        all_transactions.extend(transactions)
    return all_transactions

# List of database filenames
database_filenames = ['database1.csv', 'database2.csv', 'database3.csv', 'database4.csv', 'database5.csv']

# Read all the transactions from all databases
all_transactions = read_all_databases(database_filenames)

# Convert the transactions into a DataFrame
te = TransactionEncoder()
te_ary = te.fit(all_transactions).transform(all_transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Minimum support
min_support = 0.1

# Minimum confidence
min_confidence = 0.5

# Run the brute force algorithm and measure the time
start = time.time()
frequent_itemsets_brute_force = find_frequent_itemsets(items, all_transactions, min_support)
rules_brute_force = generate_association_rules(frequent_itemsets_brute_force, min_confidence)
end = time.time()
print(f"Brute force method took {end - start} seconds.")

# Run the Apriori algorithm and measure the time
start = time.time()
frequent_itemsets_apriori = apriori(df, min_support=min_support, use_colnames=True)
end = time.time()
print(f"Apriori algorithm took {end - start} seconds.")

# Run the FP-Growth algorithm and measure the time
```

jupyter CHARAN_KATTA_MID_PROJ Last Checkpoint: Last Monday at 9:51 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
rules_brute_force = generate_association_rules(frequent_itemsets_brute_force, min_confidence)
end = time.time()
print(f"Brute force method took {end - start} seconds.")

# Run the Apriori algorithm and measure the time
start = time.time()
frequent_itemsets_apriori = apriori(df, min_support=min_support, use_colnames=True)
end = time.time()
print(f"Apriori algorithm took {end - start} seconds.")

# Run the FP-Growth algorithm and measure the time
start = time.time()
frequent_itemsets_fpgrowth = fpgrowth(df, min_support=min_support, use_colnames=True)
end = time.time()
print(f"FP-Growth algorithm took {end - start} seconds.")

# Compare the results
print("Brute force method found {} frequent itemsets.".format(len(frequent_itemsets_brute_force)))
print("Apriori algorithm found {} frequent itemsets.".format(len(frequent_itemsets_apriori)))
print("FP-Growth algorithm found {} frequent itemsets.".format(len(frequent_itemsets_fpgrowth)))
```

```
Brute force method took 0.024338960647583008 seconds.
Apriori algorithm took 0.007256984710693359 seconds.
FP-Growth algorithm took 0.0064220428466796875 seconds.
Brute force method found 0 frequent itemsets.
Apriori algorithm found 1023 frequent itemsets.
FP-Growth algorithm found 1023 frequent itemsets.
```

In [25]:
```
## Performance Analysis & Conclusion
```

In [26]:
```
##In conclusion,
#The project successfully implemented and compared three different algorithms for frequent itemset mining and associ
#brute force, Apriori, and FP-Growth.
#The results showed that both Apriori and FP-Growth produced the same number of frequent itemsets, while the brute f
```

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

```
Please select one out of 6 databases
Enter 1 for Amazon
Enter 2 for BestBuy
Enter 3 for kmart
Enter 4 for Nike
Enter 5 for Generic
Enter 6 for Custom.
 3
Count of each items :

                 Element  Count
0                  Shams     11
1             Bed Skirts     11
2             Bedspreads      7
3  Embroidered Bedspread      6
4                 Sheets     10
5    Bedding Collections      7
6           Kids Bedding     12
7     Decorative Pillows     10
8                 Quilts      8
This is a database of 20 transactions.Enter any value of support and confidence between 10 to 100%
Enter the support in percent:  70
Enter the confidence in percent:  70
Minimum support in quantity is  14
```

The Final output should be the following Verified With Built in Package:

Please select one out of 6 databases:

Enter 1 for Amazon

Enter 2 for BestBuy

Enter 3 for kmart

Enter 4 for Nike

Enter 5 for Generic

Enter 6 for Custom

3

Enter the support in percent: 20

Enter the confidence in percent: 50

Frequent items are as below: [['Graphic Gym Bag', 'Crew Socks (6-Pack)'], [('Crew Socks (6-Pack)',), ('Graphic Gym Bag',)]]

Final Association Rules: Rule 1: Graphic Gym Bag -> Crew Socks (6-Pack) Confidence: 57.14%

Support: 20% Rule 2: Crew Socks (6-Pack) -> Graphic Gym Bag Confidence: 66.67% Support: 20%

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/charanreddy9866/Charan_Katta_midproj/tree/main