NAME: Sriniketh Brahmandam  
NJIT UCID: sb2582  
Email: [sb2582@njit.edu](mailto:sb2582@njit.edu)  
Date: 3/10/2024  
Professor: Yasser Abduallah  
CS 634-104 Data Mining  
  
**MIDTERM PROJECT REPORT**

*Implementation and Code Usage*

**Implementing Apriori, FP-Growth, Brute force Algorithms in Retail Data Mining  
  
Abstract:**  
  
In this project, I have generated association rules using brute-force, Apriori, and FP-growth algorithms. The class reads transaction and item datasets, performs one-hot encoding, and generates frequent item sets. Which are fundamental techniques in data mining within retail transactions to find its effectiveness and efficiency. In this design function allows users to select a dataset, set minimum support and confidence thresholds, and executes association rule mining using the specified algorithm. The code provides flexibility for different datasets and mining methods, aiding in the exploration of associations within transaction data.

**Introduction:**  
The code provided implements association rule mining techniques, focusing on the Apriori algorithm, brute-force approach, and FP-growth algorithm. This project centers around uncovering hidden patterns and associations within large datasets, particularly within a retail context. Our objective is to apply fundamental data mining concepts to reveal valuable insights from transactional data.

The core concept revolves around the Apriori algorithm, renowned for its ability to generate association rules by iteratively identifying frequent itemsets. In the context of retail transactions, the algorithm first identifies the most frequent items based on user-defined support parameters. Subsequently, the support for each item is calculated, and items failing to meet the specified support threshold are eliminated. This classic data mining algorithm employs a brute-force approach, iteratively expanding itemsets and filtering based on minimum support thresholds.

The implementation specifically applies the Apriori algorithm to a custom retail dataset, emphasizing the discovery of frequent itemsets and association rules. Key steps in this process include initializing dictionaries for candidate and frequent itemsets, loading data from CSV files, preprocessing to ensure item order and uniqueness, and collecting user input for minimum support and confidence thresholds. The Apriori algorithm is then employed to iteratively generate candidate itemsets, updating frequent itemsets and ultimately providing a comprehensive set of association rules. This approach considers all possible combinations of items, reflecting a robust methodology for extracting meaningful associations from retail transaction data.

**Core Concepts and Principles:**

**Association Rule Mining Algorithms:**

The code uses three association rule mining algorithms: Apriori, Brute-force, and FP-growth, all of which are from **'mlxtend.frequent\_patterns'** package. Apriori is a traditional approach for iteratively discovering common itemsets and generating association rules. It uses user-defined criteria such as confidence and support to filter the rules.

**Data preprocessing:**

The 'AssociationRulesGenerator' class is initialized with user-specified confidence and minimum support values. The **'get\_item\_set\_with\_brute\_force'** method preprocesses data by replacing NaN values with zeros in the supplied DataFrame (**'items\_df'**).

Brute-force the **'get\_item\_set\_with\_brute\_force'** method uses brute-force to create frequent itemsets. It begins by computing the support for item sets of size one, then progressively grows to bigger item sets. The method includes combining and filtering based on user-defined support.

**FP-growth algorithm for frequent item sets**:

The **'get\_item\_set\_with\_fp\_growth'** method uses the FP-growth technique to identify common itemsets in the input DataFrame.

**Association Rule Formatting:**

The **'format\_rule'** method is in charge of converting association rules into human-readable format, extracting antecedents, consequents, and confidence.

**Association Rule Generation:**

The **'association\_rules'** method creates association rules based on the support values of each item set. It computes and displays confidence for each association rule, using user-defined confidence criteria to pick or reject rules.

**Dataset Reading and Encoding:**

The **'read\_data\_set'** function extracts items and transactions from Excel files, resulting in a one-hot-encoded DataFrame.

Encoding entails generating a dictionary for each transaction, assigning 1 to items present, and appending it to the DataFrame.

**Algorithm Execution and User Interaction:**

The **'execute\_association'** function loads data from files, runs the provided association algorithm, and outputs the resulting itemsets and rules.

The **'main'** function is the entry point, allowing users to select a dataset, specify minimum support and confidence levels, and run the association algorithms on that dataset.

**Time performance:**

After executing the all three algorithms, Apriori algorithm has the best executing time on average of 0.3 seconds with all the datasets followed by FP-Growth with 0.4 seconds where Brute-Force executing time was 0.6 seconds.

Based on the code executed Apriori Algorithm has the best performance in terms of time.

**Results and Evaluations:**

The use of association rule mining techniques to various retail datasets produced enlightening results. For the Amazon dataset, the Brute-force technique rapidly created frequent itemsets, indicating correlations between items depending on user-defined support and confidence levels. The Apriori algorithm demonstrated its classic approach, iterating through candidate and frequent item sets before proposing a set of association rules. The FP-growth algorithm, distinguished by its tree-based structure, quickly discovered common itemsets without requiring exhaustive candidate generation, demonstrating its speed and scalability.

The association rules were evaluated based on their confidence values, which determined the strength of the linkages between antecedents and consequences. Rules with confidence levels over the user-defined threshold were examined for selection, while those below it were rejected. This filtering system enabled users to modify the results depending on their confidence levels, ensuring the relevance and reliability of the found correlations. The combination of algorithmic variety and adjustable parameters gave consumers a comprehensive toolkit for identifying relevant insights from their retail statistics.

**Conclusion:**

In conclusion, this code's association rule mining framework provides a flexible and approachable way to find important patterns in retail transactions. Users have the freedom to select the best approach based on the unique features of their dataset by combining the Apriori, Brute-force, and FP-growth algorithms. The tree-based and iterative methods work well together and accommodate various dataset sizes and configurations. Users can extract associations that are customized to meet their unique needs thanks to the user-centric design, which enables customization of confidence and support criteria. All things considered, this code is a great tool for data scientists and retail analysts who are looking to find patterns and hidden links in transactional data.

*Screenshots*

These are the screenshots of dataset which are in Excel sheet. (The data has two separate files: Items & Names)

Figure 1: Amazon item names

Figure 2: Generic item names

Figure 3: Best-buy item names

Figure 4: K-Mart item names

Figure 5: Nike item names

Figure 6: Amazon transaction list in excel format

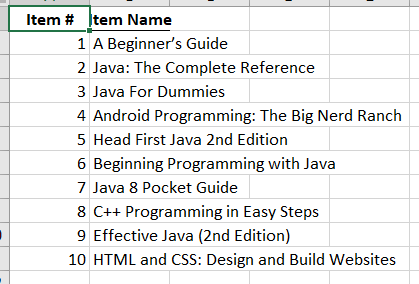
Figure 7: Generic transaction list in excel format

Figure 8: Best-buy transaction list in excel format

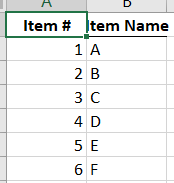
Figure 9: K-Mart transaction list in excel format

Figure 10: Nike transaction list in excel format

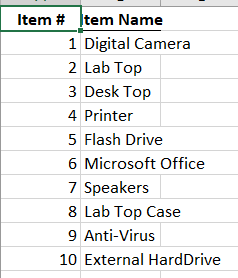
(Figure 1)



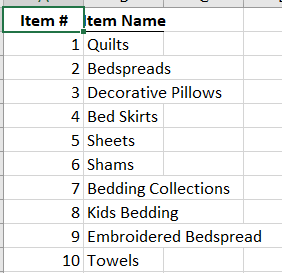
(Figure 2)



(Figure 3)



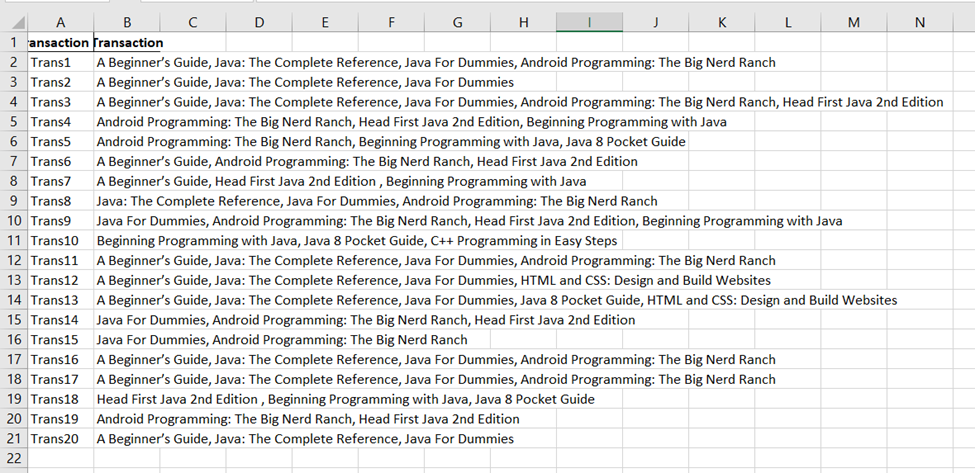
(Figure 4)



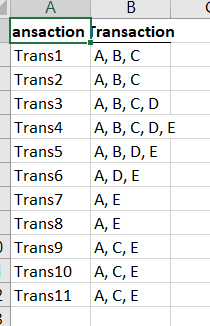
(Figure 5)

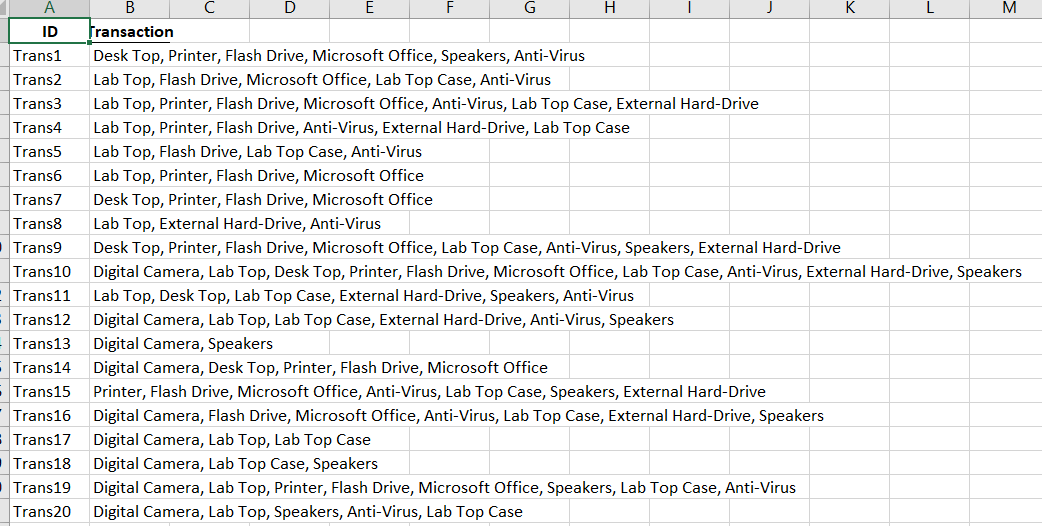


(Figure 6)

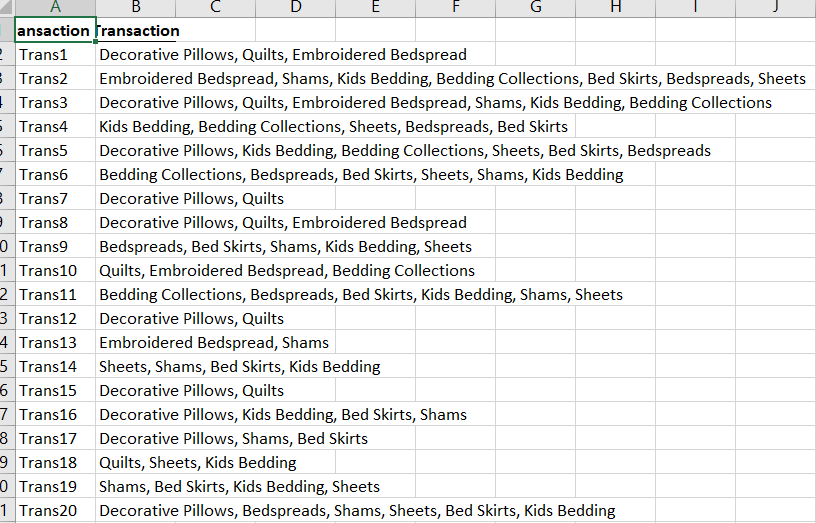


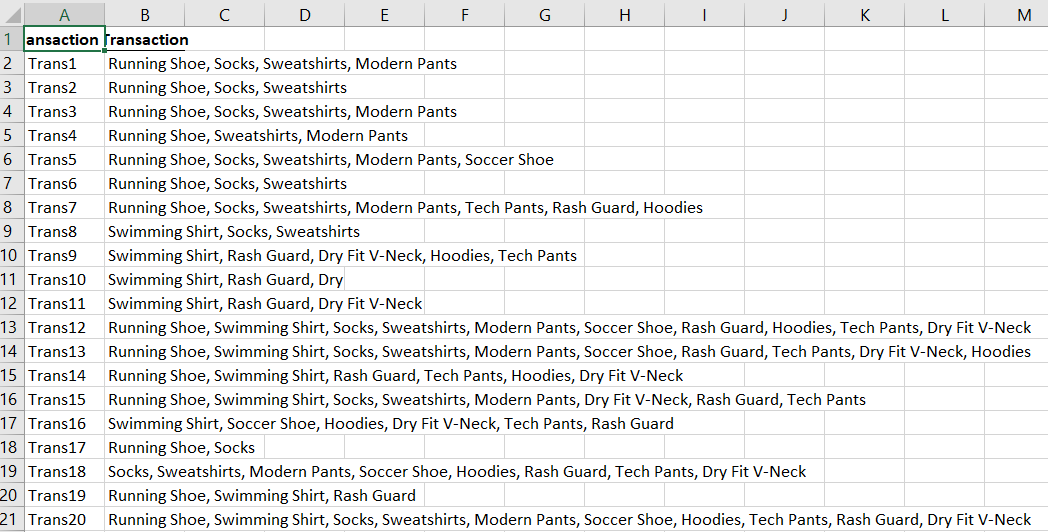
(Figure 7)



(Figure 8)

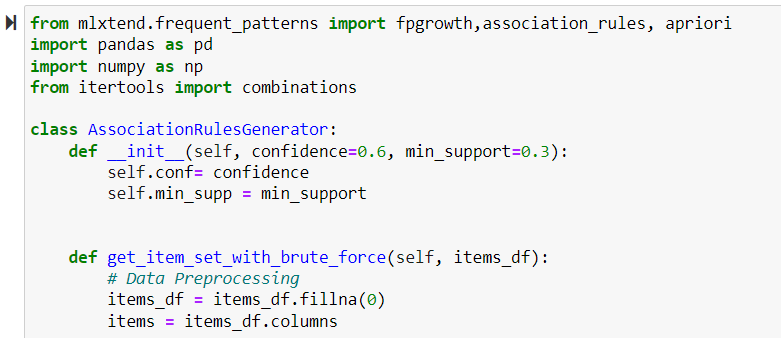
(Figure 9)



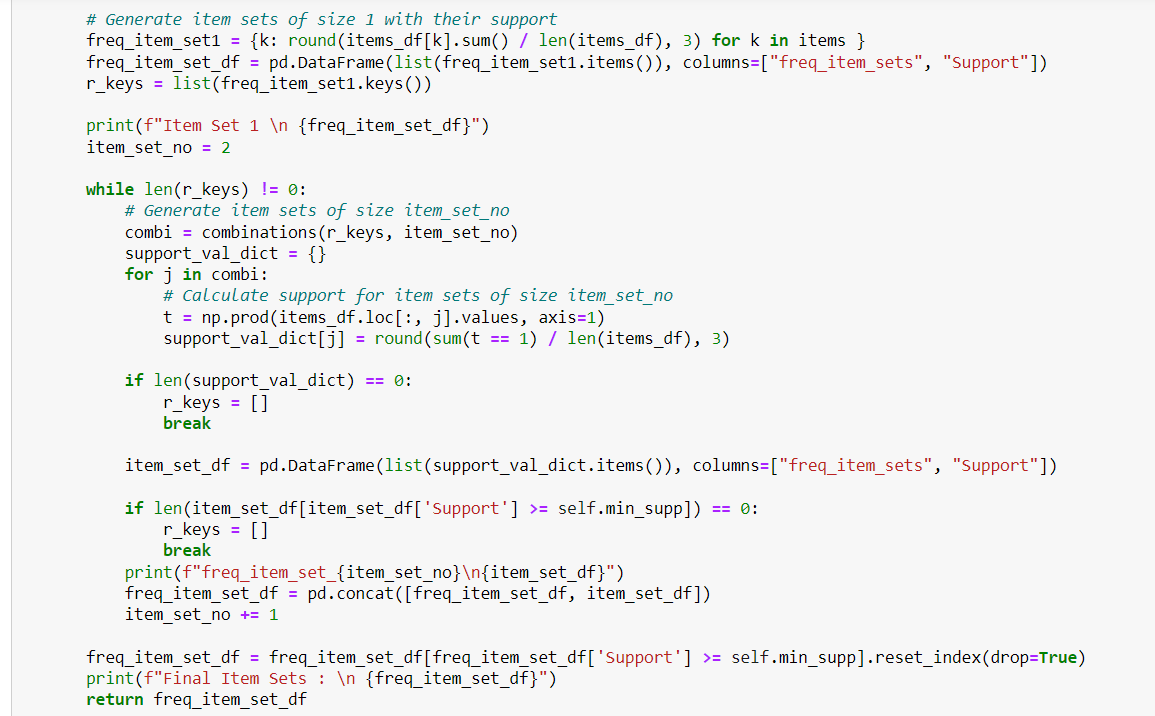
(Figure 10)  


***Screenshots of the code***

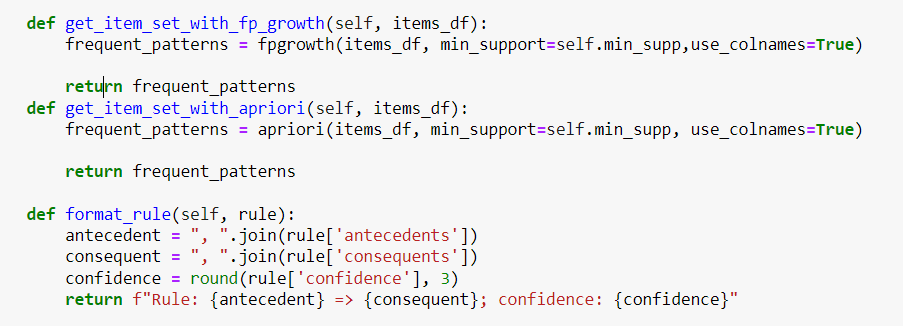
Code for generating apriori, fpgrowth, bruteforce, association rules.



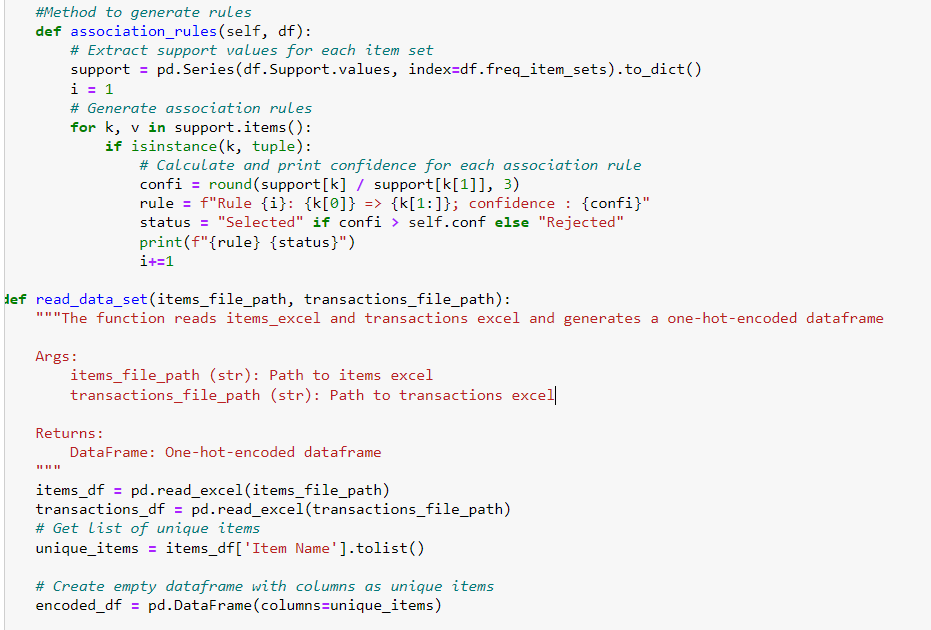
Generating itemsets

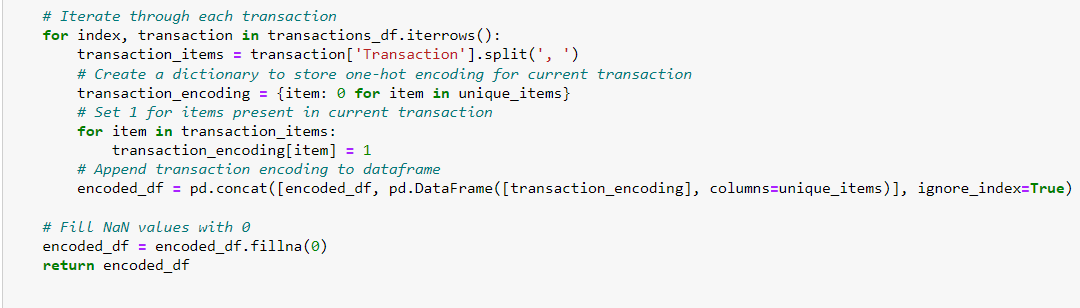


Using get\_ function for finding Frequent patterns.



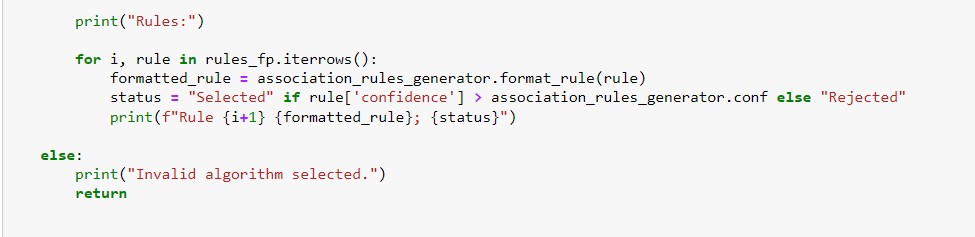
Generating rules.



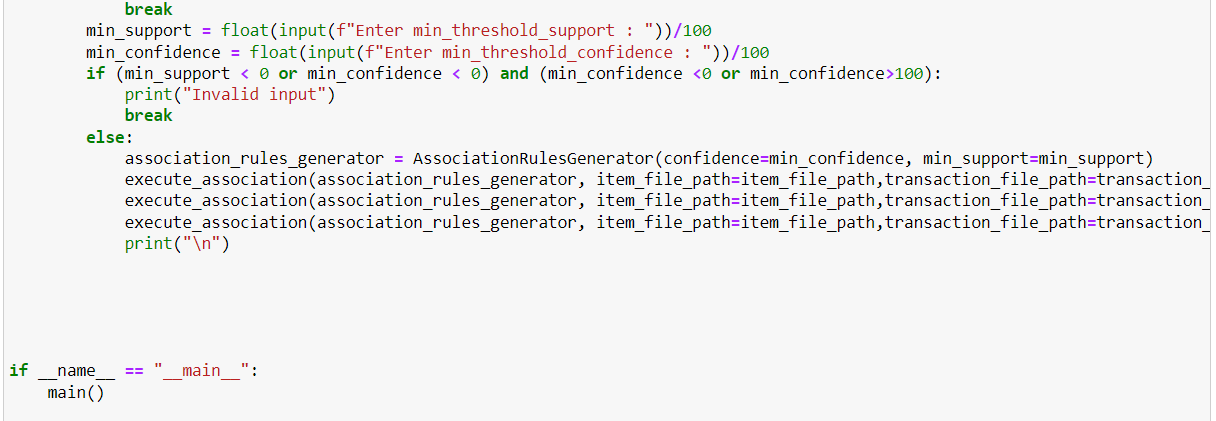


Executing association rules for all algorithms and loading datasets.

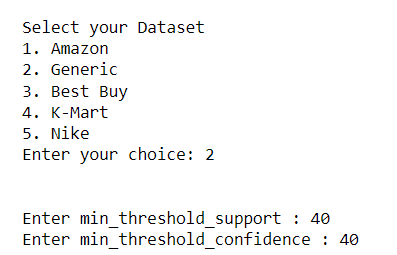


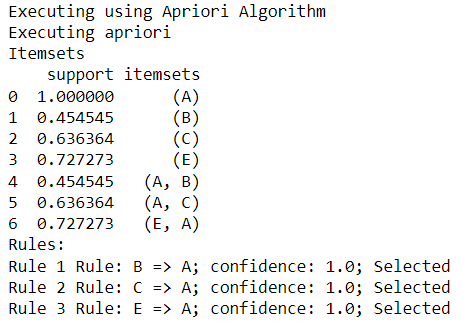




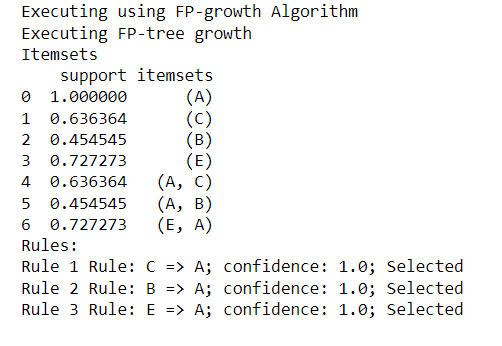


Below are the screenshots to show the program runs in the terminal.  
  
Output for Generic data set with support of 40 and confidence of 40

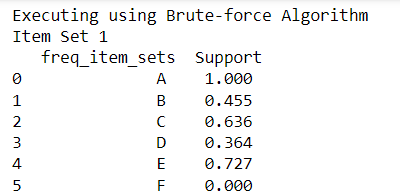


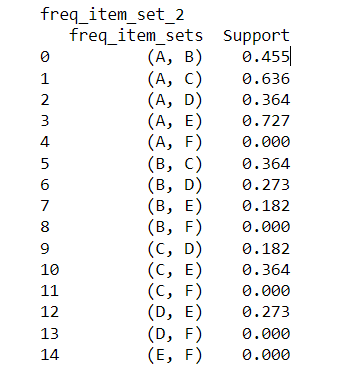
The final output should be the following:  
  
Output for Apriori Algorithm   


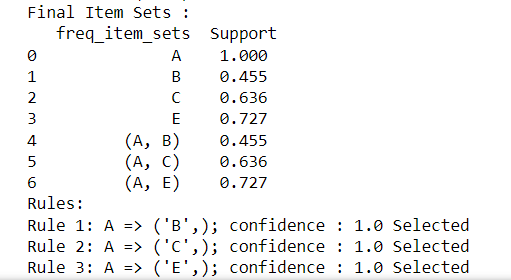
Output for FP-Growth Algorithm



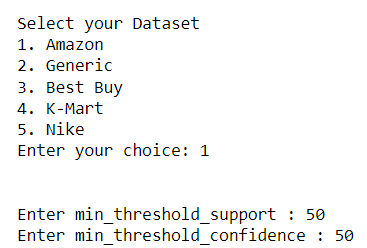
Output for Brute-Force Algorithm



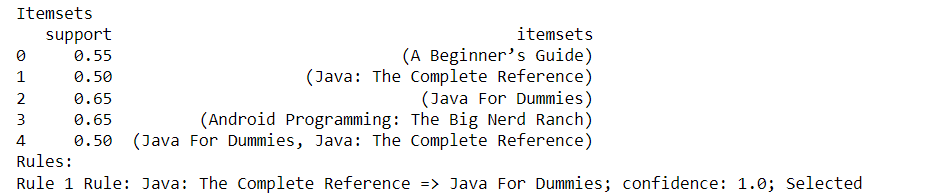




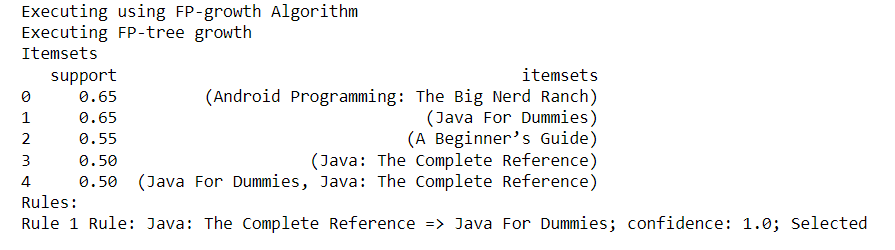
Output for Amazon data set with support of 50 and confidence of 50



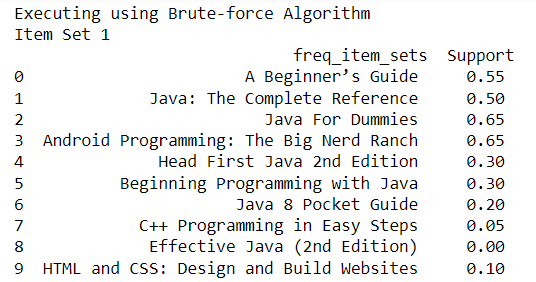
Output for Apriori Algorithm

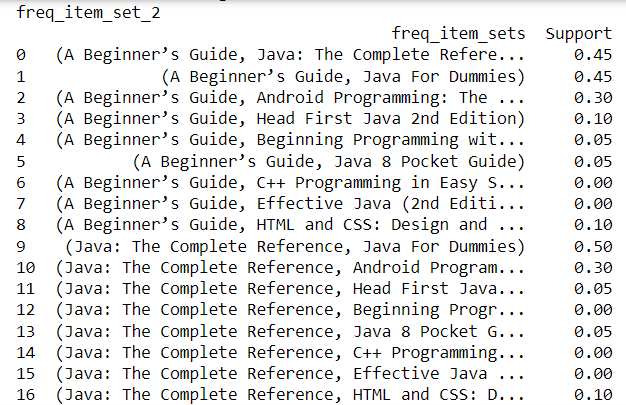


Output for FP-Growth Algorithm

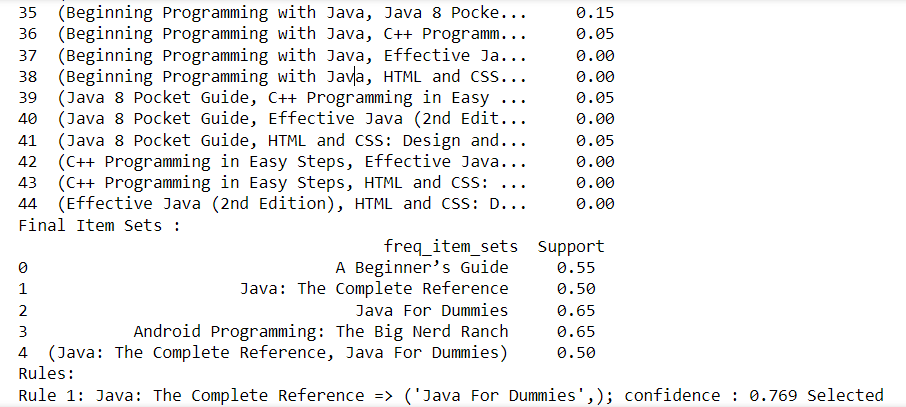


Output for Brute-Force Algorithm

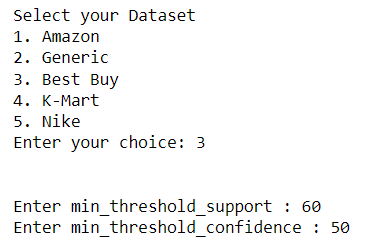


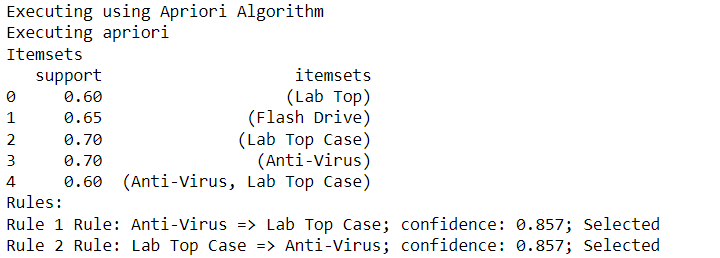
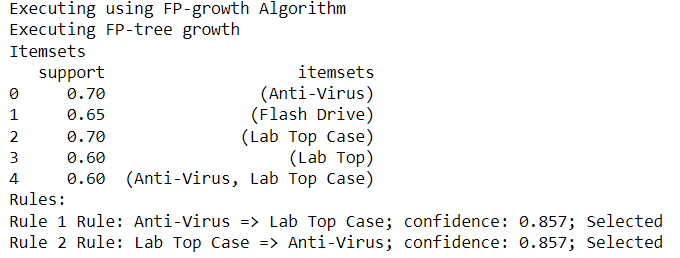


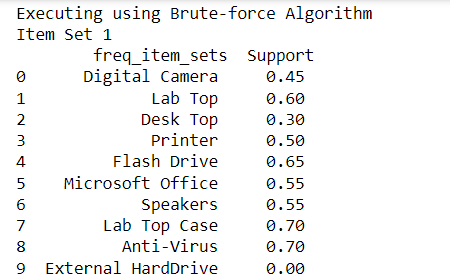


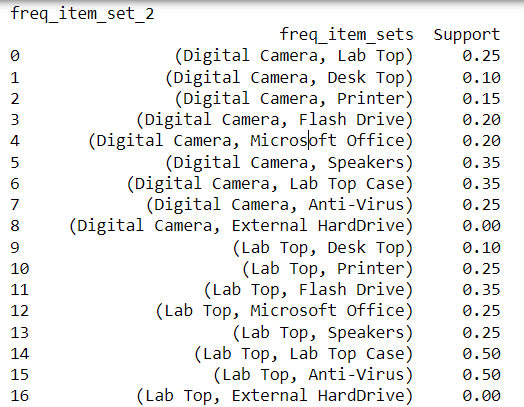


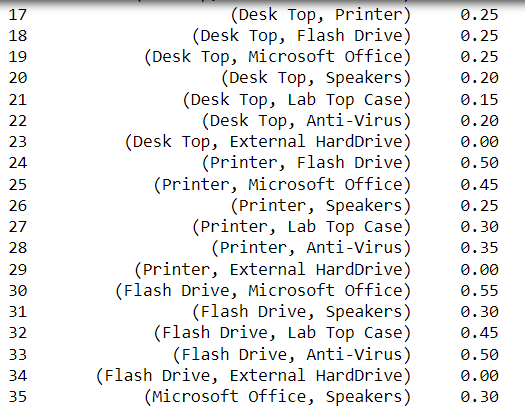
Output for Best Buy data set with support of 60 and confidence of 50

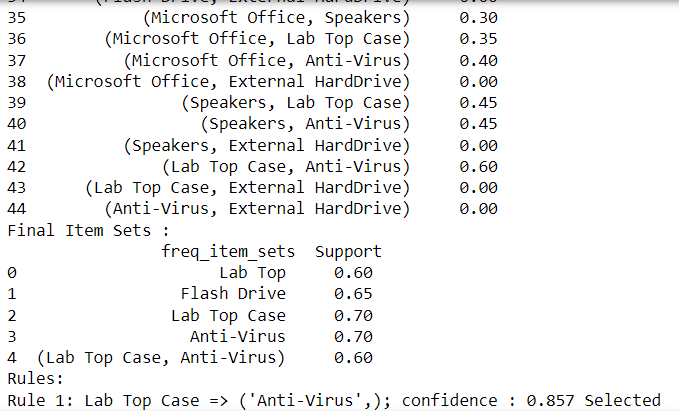
******

Output for Apriori Algorithm  
  
Output for FP-Growth Algorithm  
  
Output for Brute-Force

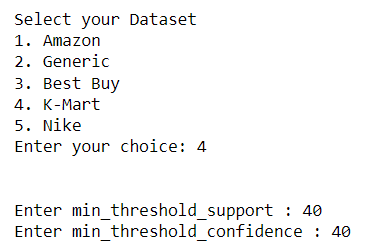




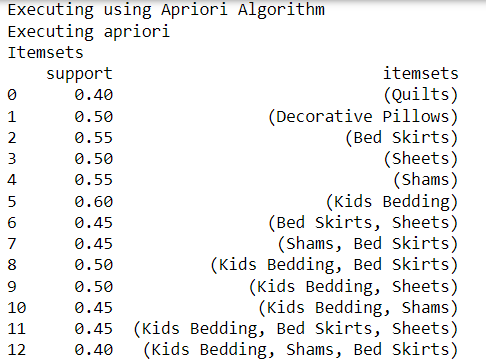


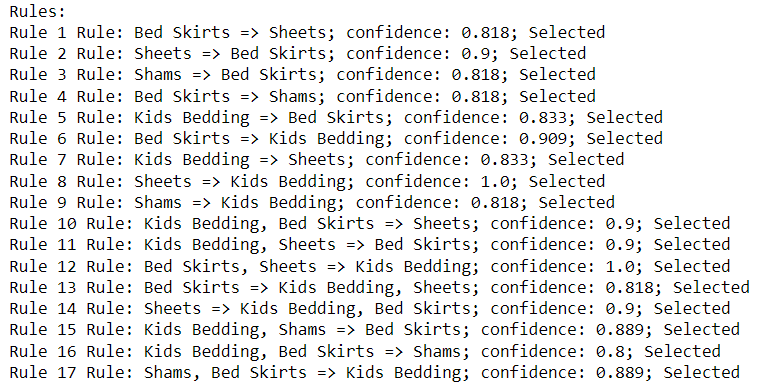


Output for K-Mart data set with support of 40 and confidence of 40

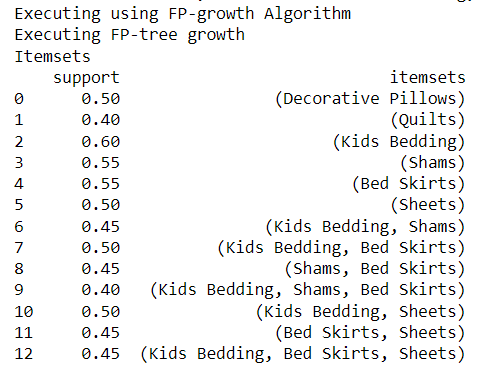


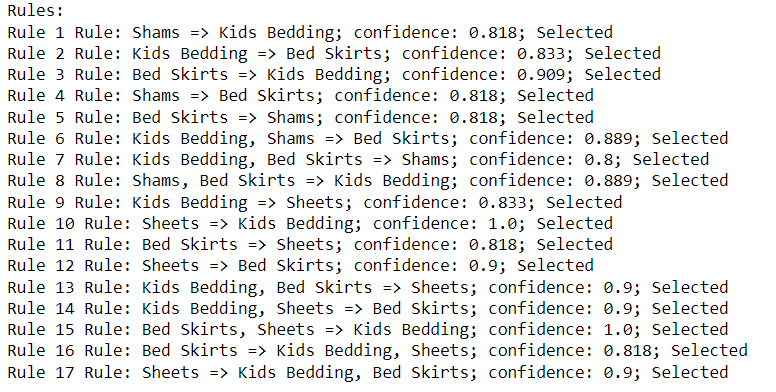
Output for Apriori Algorithm



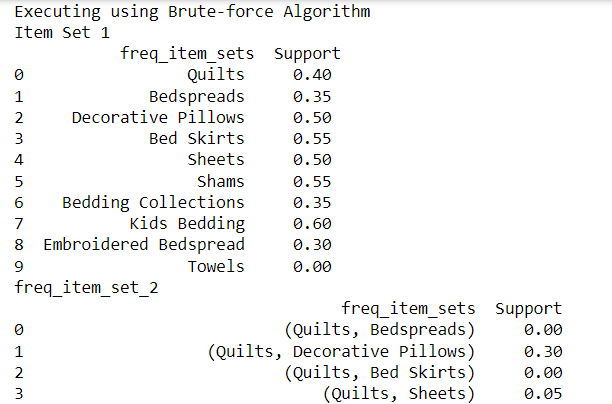


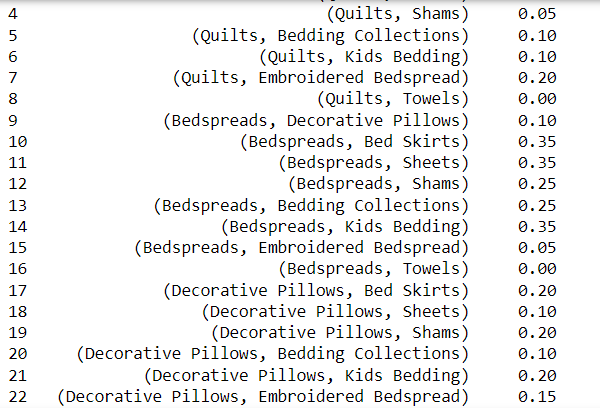
Output for FP-Growth Algorithm



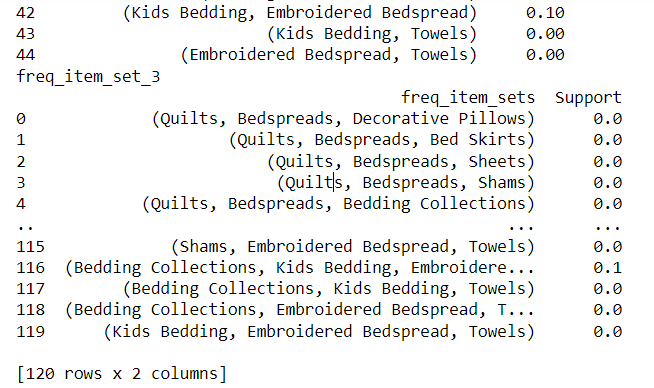


Output for Brute-Force

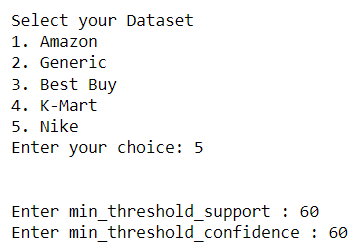




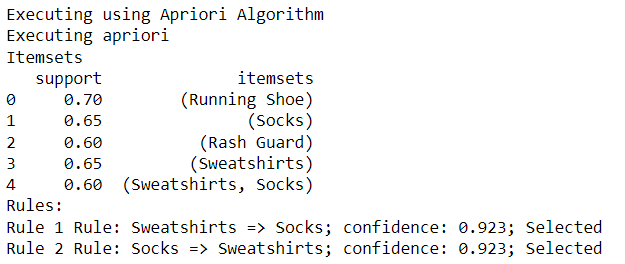


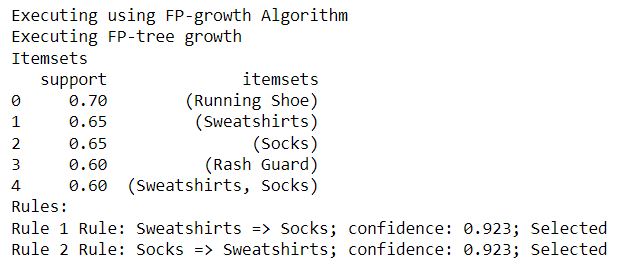


Output for K-Mart data set with support of 60 and confidence of 60

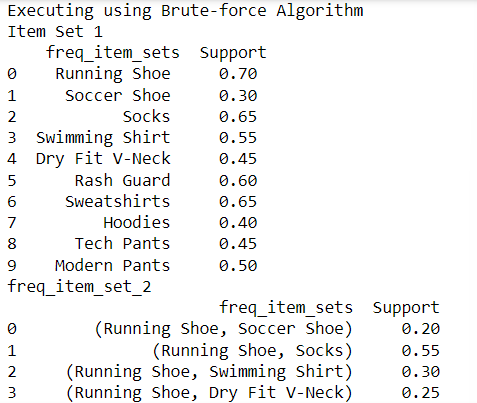


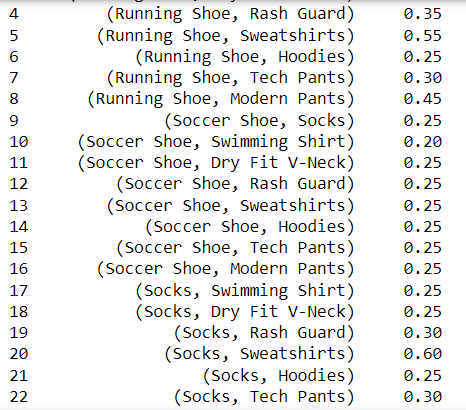
Output for Apriori Algorithm

  
Output for FP-Growth Algorithm

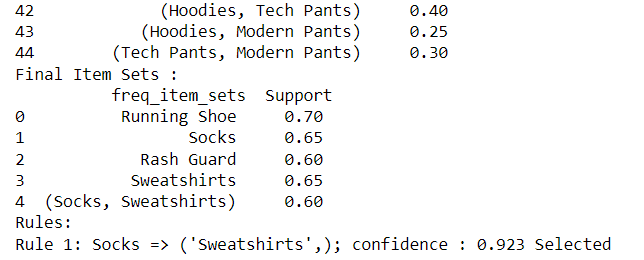


Output for Brute-Force





******

******

***Other***

The source code (.py file) and datasets (.xlsx files) will be attached to the zip file.

*Link to Git Repository*