Data Mining Techniques for Transaction Database Analysis

CS 634 Data Mining

Midterm Term Project

Dr. Yasser Abduallah

Department of Computer Science

New Jersey Institute of Technology

Tarun Deshmukh Tetapally (tt362)

tt362@njit.edu

9th Mar 2024

Index

Github Link:

https://github.com/TarunDeshmukhTetapally/CS634_DataMining_MidTerm

Introduction

This project explores the application of data mining techniques to analyze transaction databases commonly found in retail settings, such as supermarkets. The primary objective is to apply and compare different algorithms, including a brute force method, Apriori, and FP-Growth, for finding frequent itemsets and generating association rules.

Dataset Creation

I have created five datasets with 20 transactions in each namely amazon, kmart, bestbuy, nike and generic. These datasets are created from the data provided. The datasets are saved in csv formats and saved in the folder folder where our code is present. The dataset file names I created are as follows: amazon_transactions.csv, bestbuy_transactions.csv, kmart_transactions.csv, nike_transactions.csv, generic_transactions.csv. The user will be prompted for the dataset to select while running the apriori algorithm.

Project File Structure

The zip file contains the python file with the code implementation, a jupyter notebook where the same code implementation is present with the outputs of the runs and also the datasets used : amazon_transactions.csv, bestbuy_transactions.csv, kmart_transactions.csv, nike_transactions.csv, generic_transactions.csv

Running the Program

This project was developed using Python 3.8, but it should be compatible with Python 3.6 and newer versions.

Required Packages

The program relies on several Python packages for data processing and analysis:

- **mlxtend**: Used for applying the Apriori algorithm for frequent itemset generation and association rule mining.

- **pyfpgrowth**: Implements the FP-Growth algorithm for efficient frequent itemset and association rule generation.
- **pandas**: Provides high-performance, easy-to-use data structures, and data analysis tools.
- **csv**: Facilitates reading from and writing to CSV files, where our transaction data is stored.

To install these packages, open a terminal or command prompt and execute the following command:

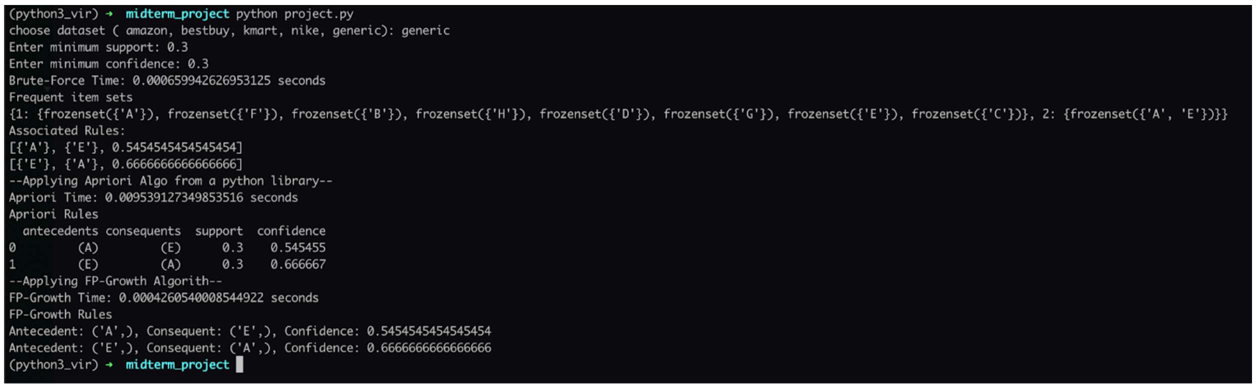
```
pip install mlxtend pyfpgrowth pandas
```

Running the Program using Python file

Open the Terminal and navigate to the directory containing the project files and run the following command. When prompted, provide the dataset, support and confidence as inputs.

```
Python project.py
```

The below screenshot shows running the python file in terminal by selecting generic dataset with support as 0.3 and confidence as 0.3



```
(python3_vir) ➜  midterm_project python project.py
choose dataset ( amazon, bestbuy, kmart, nike, generic): generic
Enter minimum support: 0.3
Enter minimum confidence: 0.3
Brute-Force Time: 0.000659942626953125 seconds
Frequent item sets
{1: {frozenset({'A'}), frozenset({'F'}), frozenset({'B'}), frozenset({'H'}), frozenset({'D'}), frozenset({'G'}), frozenset({'E'}), frozenset({'C'})}, 2: {frozenset({'A', 'E'})}}
Associated Rules:
[{'A'}, {'E'}, 0.5454545454545454]
[{'E'}, {'A'}, 0.6666666666666666]
--Applying Apriori Algo from a python library--
Apriori Time: 0.009539127349853516 seconds
Apriori Rules
  antecedents consequents  support  confidence
0         (A)         (E)      0.3    0.545455
1         (E)         (A)      0.3    0.666667
--Applying FP-Growth Algorith--
FP-Growth Time: 0.0004260540008544922 seconds
FP-Growth Rules
Antecedent: ('A',), Consequent: ('E',), Confidence: 0.5454545454545454
Antecedent: ('E',), Consequent: ('A',), Confidence: 0.6666666666666666
(python3_vir) ➜  midterm_project
```

Running the Program using Jupyter Notebook

Install Jupyter Lab in your machine to open and run a jupyter notebook.

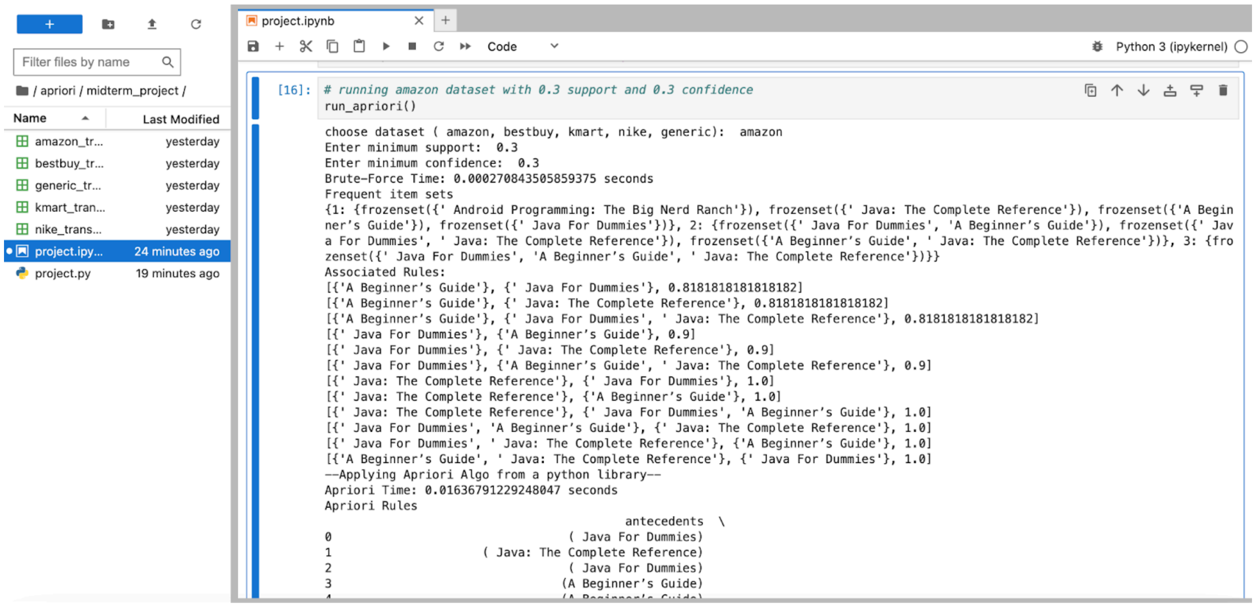
```
Pip install jupyterlab
```

Now, navigate to the directory containing the project files, run the following command to start Jupyter Notebook:

```
jupyter lab
```

Navigate to the project.ipynb file within the Jupyter Notebook interface and open it.

Execute the cells in order by clicking on each and pressing **Shift + Enter**, and providing the appropriate inputs for the prompts to select the desired dataset, support and confidence.



```
[16]: # running amazon dataset with 0.3 support and 0.3 confidence
      run_apriori()

choose dataset ( amazon, bestbuy, kmart, nike, generic):  amazon
Enter minimum support:  0.3
Enter minimum confidence:  0.3
Brute-Force Time: 0.000270843505859375 seconds
Frequent item sets
{1: {frozenset({' Android Programming: The Big Nerd Ranch'}), frozenset({' Java: The Complete Reference'}), frozenset({'A Begin
ner’s Guide'}), frozenset({' Java For Dummies'})}, 2: {frozenset({' Java For Dummies', 'A Beginner’s Guide'}), frozenset({' Jav
a For Dummies', ' Java: The Complete Reference'}), frozenset({'A Beginner’s Guide', ' Java: The Complete Reference'})}, 3: {fro
zenset({' Java For Dummies', 'A Beginner’s Guide', ' Java: The Complete Reference'})}}
Associated Rules:
[{'A Beginner’s Guide'}, {' Java For Dummies'}, 0.8181818181818182]
[{'A Beginner’s Guide'}, {' Java: The Complete Reference'}, 0.8181818181818182]
[{'A Beginner’s Guide'}, {' Java For Dummies', ' Java: The Complete Reference'}, 0.8181818181818182]
[{' Java For Dummies'}, {'A Beginner’s Guide'}, 0.9]
[{' Java For Dummies'}, {' Java: The Complete Reference'}, 0.9]
[{' Java For Dummies'}, {'A Beginner’s Guide', ' Java: The Complete Reference'}, 0.9]
[{' Java: The Complete Reference'}, {' Java For Dummies'}, 1.0]
[{' Java: The Complete Reference'}, {'A Beginner’s Guide'}, 1.0]
[{' Java: The Complete Reference'}, {' Java For Dummies', 'A Beginner’s Guide'}, 1.0]
[{' Java For Dummies', 'A Beginner’s Guide'}, {' Java: The Complete Reference'}, 1.0]
[{' Java For Dummies', ' Java: The Complete Reference'}, {'A Beginner’s Guide'}, 1.0]
[{'A Beginner’s Guide', ' Java: The Complete Reference'}, {' Java For Dummies'}, 1.0]
--Applying Apriori Algo from a python library--
Apriori Time: 0.01636791229248047 seconds
Apriori Rules
                                         antecedents  \
0                                 ( Java For Dummies)
1                    ( Java: The Complete Reference)
2                                 ( Java For Dummies)
3                                (A Beginner’s Guide)
```

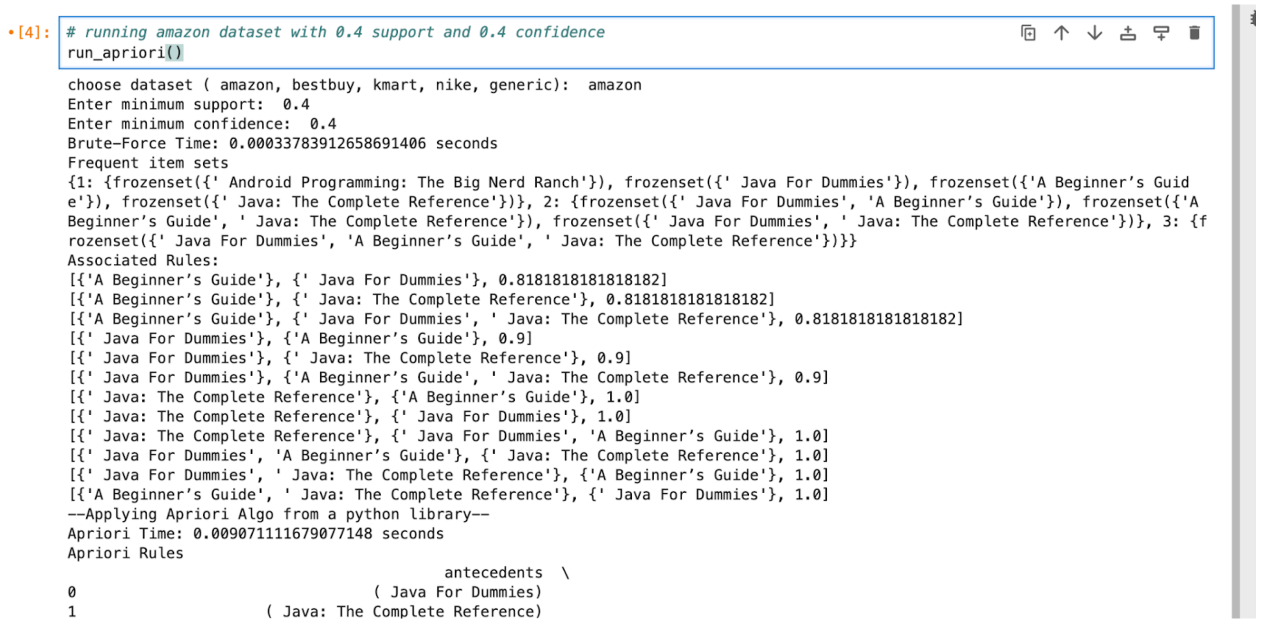
The program will prompt you to select a dataset (e.g., Amazon, BestBuy, Nike, Kmart, Generic) and enter minimum support and confidence levels for the analysis. Follow the prompts to input your choices. The program will then proceed to execute the selected data mining algorithm(s) and display the results, including frequent itemsets and association rules.

Few examples of running multiple different datasets with different support and confidence.

[11]:
```
# running generic dataset with 0.3 support and 0.3 confidence
run_apriori()
```
```
choose dataset ( amazon, bestbuy, kmart, nike, generic):  generic
Enter minimum support:  0.3
Enter minimum confidence:  0.3
Brute-Force Time: 0.00040912628173828125 seconds
Frequent item sets
{1: {frozenset({'A'}), frozenset({'E'}), frozenset({'D'}), frozenset({'H'}), frozenset({'C'}), frozenset({'G'}), frozenset
({'B'}), frozenset({'F'})}, 2: {frozenset({'A', 'E'})}}
Associated Rules:
[{'A'}, {'E'}, 0.5454545454545454]
[{'E'}, {'A'}, 0.6666666666666666]
--Applying Apriori Algo from a python library--
Apriori Time: 0.01174473762512207 seconds
Apriori Rules
  antecedents consequents  support  confidence
0         (A)         (E)      0.3    0.545455
1         (E)         (A)      0.3    0.666667
--Applying FP-Growth Algorith--
FP-Growth Time: 0.0006399154663085938 seconds
FP-Growth Rules
Antecedent: ('A',), Consequent: ('E',), Confidence: 0.5454545454545454
Antecedent: ('E',), Consequent: ('A',), Confidence: 0.6666666666666666
```

[12]:
```
# running nike dataset with 0.4 support and 0.4 confidence
run_apriori()
```
```
choose dataset ( amazon, bestbuy, kmart, nike, generic):  nike
Enter minimum support:  0.4
Enter minimum confidence:  0.4
Brute-Force Time: 0.0006840229034423828 seconds
Frequent item sets
{1: {frozenset({' Modern Pants'}), frozenset({' Sweatshirts'}), frozenset({' Socks'}), frozenset({' Rash Guard'}), frozenset
({'Running Shoe'}), frozenset({' Tech Pants'}), frozenset({' Dry Fit V-Nick'})}, 2: {frozenset({' Sweatshirts', ' Modern Pant
s'}), frozenset({' Socks', 'Running Shoe'}), frozenset({' Modern Pants', 'Running Shoe'}), frozenset({' Tech Pants', ' Rash Gua
rd'}), frozenset({' Sweatshirts', ' Socks'}), frozenset({' Sweatshirts', 'Running Shoe'}), frozenset({' Socks', ' Modern Pant
s'}), frozenset({' Rash Guard', ' Dry Fit V-Nick'})}, 3: {frozenset({' Socks', ' Modern Pants', 'Running Shoe'}), frozenset({'
Socks', 'Running Shoe', ' Sweatshirts'}), frozenset({' Sweatshirts', ' Socks', ' Modern Pants'}), frozenset({' Sweatshirts', '
Modern Pants', 'Running Shoe'})}, 4: {frozenset({' Socks', ' Modern Pants', 'Running Shoe', ' Sweatshirts'})}}
Associated Rules:
[{'Running Shoe'}, {' Socks', ' Modern Pants'}, 0.5714285714285714]
[{'Running Shoe'}, {' Socks', ' Modern Pants', ' Sweatshirts'}, 0.5714285714285714]
[{' Sweatshirts'}, {' Socks', ' Modern Pants'}, 0.6153846153846154]
[{' Sweatshirts'}, {' Socks', ' Modern Pants', 'Running Shoe'}, 0.6153846153846154]
[{'Running Shoe'}, {' Modern Pants'}, 0.6428571428571429]
[{'Running Shoe'}, {' Sweatshirts', ' Modern Pants'}, 0.6428571428571429]
[{' Socks'}, {' Modern Pants'}, 0.6666666666666666]
[{' Rash Guard'}, {' Dry Fit V-Nick'}, 0.6666666666666666]
[{' Socks'}, {' Modern Pants', 'Running Shoe'}, 0.6666666666666666]
[{' Socks'}, {' Sweatshirts', ' Modern Pants'}, 0.6666666666666666]
[{' Socks'}, {' Sweatshirts', ' Modern Pants', 'Running Shoe'}, 0.6666666666666666]
[{' Sweatshirts'}, {' Modern Pants', 'Running Shoe'}, 0.6923076923076923]
[{'Running Shoe'}, {' Socks', ' Sweatshirts'}, 0.7142857142857143]
[{' Socks', 'Running Shoe'}, {' Modern Pants'}, 0.7272727272727273]
```

[8]:
```
# running kmart dataset with 0.4 support and 0.4 confidence
run_apriori()
```
```
choose dataset ( amazon, bestbuy, kmart, nike, generic):  kmart
Enter minimum support:  0.4
Enter minimum confidence:  0.4
Brute-Force Time: 0.0002796649932861328 seconds
Frequent item sets
{1: {frozenset({' Kids Bedding'}), frozenset({'Decorative Pillows'}), frozenset({' Shams'}), frozenset({' Bed Skirts'})}, 2: {f
rozenset({' Kids Bedding', ' Bed Skirts'}), frozenset({' Bed Skirts', ' Shams'})}}
Associated Rules:
[{' Bed Skirts'}, {' Kids Bedding'}, 0.7272727272727273]
[{' Bed Skirts'}, {' Shams'}, 0.7272727272727273]
[{' Kids Bedding'}, {' Bed Skirts'}, 0.8]
[{' Shams'}, {' Bed Skirts'}, 0.8]
--Applying Apriori Algo from a python library--
Apriori Time: 0.008028984069824219 seconds
Apriori Rules
       antecedents      consequents  support  confidence
0   ( Kids Bedding)     ( Bed Skirts)      0.4    0.800000
1     ( Bed Skirts)   ( Kids Bedding)      0.4    0.727273
2     ( Bed Skirts)          ( Shams)      0.4    0.727273
3          ( Shams)     ( Bed Skirts)      0.4    0.800000
--Applying FP-Growth Algorith--
FP-Growth Time: 0.00044798851013183594 seconds
FP-Growth Rules
Antecedent: (' Bed Skirts',), Consequent: (' Shams',), Confidence: 0.7272727272727273
Antecedent: (' Kids Bedding',), Consequent: (' Bed Skirts',), Confidence: 0.8
Antecedent: (' Shams',), Consequent: (' Bed Skirts',), Confidence: 0.8
```



```
•[4]: # running amazon dataset with 0.4 support and 0.4 confidence
      run_apriori()

      choose dataset ( amazon, bestbuy, kmart, nike, generic):  amazon
      Enter minimum support:  0.4
      Enter minimum confidence:  0.4
      Brute-Force Time: 0.00033783912658691406 seconds
      Frequent item sets
      {1: {frozenset({' Android Programming: The Big Nerd Ranch'}), frozenset({' Java For Dummies'}), frozenset({'A Beginner’s Guid
      e'}), frozenset({' Java: The Complete Reference'})}, 2: {frozenset({' Java For Dummies', 'A Beginner’s Guide'}), frozenset({'A
      Beginner’s Guide', ' Java: The Complete Reference'}), frozenset({' Java For Dummies', ' Java: The Complete Reference'})}, 3: {f
      rozenset({' Java For Dummies', 'A Beginner’s Guide', ' Java: The Complete Reference'})}}
      Associated Rules:
      [{'A Beginner’s Guide'}, {' Java For Dummies'}, 0.8181818181818182]
      [{'A Beginner’s Guide'}, {' Java: The Complete Reference'}, 0.8181818181818182]
      [{'A Beginner’s Guide'}, {' Java For Dummies', ' Java: The Complete Reference'}, 0.8181818181818182]
      [{' Java For Dummies'}, {'A Beginner’s Guide'}, 0.9]
      [{' Java For Dummies'}, {' Java: The Complete Reference'}, 0.9]
      [{' Java For Dummies'}, {'A Beginner’s Guide', ' Java: The Complete Reference'}, 0.9]
      [{' Java: The Complete Reference'}, {'A Beginner’s Guide'}, 1.0]
      [{' Java: The Complete Reference'}, {' Java For Dummies'}, 1.0]
      [{' Java: The Complete Reference'}, {' Java For Dummies', 'A Beginner’s Guide'}, 1.0]
      [{' Java For Dummies', 'A Beginner’s Guide'}, {' Java: The Complete Reference'}, 1.0]
      [{' Java For Dummies', ' Java: The Complete Reference'}, {'A Beginner’s Guide'}, 1.0]
      [{'A Beginner’s Guide', ' Java: The Complete Reference'}, {' Java For Dummies'}, 1.0]
      --Applying Apriori Algo from a python library--
      Apriori Time: 0.009071111679077148 seconds
      Apriori Rules
                                          antecedents  \
      0                           ( Java For Dummies)
      1                 ( Java: The Complete Reference)
```

Algorithm Comparison

In our comprehensive analysis comparing the brute force method, Apriori algorithm, and FP-Growth algorithm, all three approaches yielded similar results in terms of identifying frequent itemsets and generating association rules, demonstrating their effectiveness in extracting meaningful patterns from transaction databases. However, a significant distinction emerged when evaluating the computational efficiency of these algorithms. The FP-Growth algorithm, known for its innovative use of the FP-tree data structure to compactly represent the database, significantly outperformed both the brute force method and the Apriori algorithm in terms of time taken to process the datasets.