International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 08, Issue – 06, June 2023, PP – 01-13

Comparison of Periodic Pattern Mining Algorithms on Temporal Datasets

[1]Md. Ghouse Mohiuddin, [2]Dr. D. L.Sreenivasa Reddy

[1]*Asst.Prof. (C), Dept. of Computer Science, Palamuru University, Mahabubnagar..*

[2]*Associate Professor, Dept. of Information Technology, CBIT, Hyderabad.*

Abstract: The identification of periodic patterns is of great significance in revealing hidden temporal patterns and regularities across various domains, including finance, healthcare, and social networks. As the availability of large-scale temporal datasets continues to grow, the selection of an appropriate periodic pattern mining algorithm becomes crucial for efficient and accurate analysis. The objective of this research paper is to conduct a comparative evaluation of various periodic pattern mining algorithms applied to temporal datasets. The algorithms under consideration include Apriori-based methods such as Modified-Apriori and LPP-Apriori, as well as Tree-based approaches such as LPP Breadth, and LPP-FP Growth. We have assessed the performance of these algorithms across different datasets, focusing on metrics such as Execution Time, LPP Count, and memory usage.

Keywords: LPP-Apriori, LPP-FP-Growth, Modified-Apriori, Periodic Patterns, Timestamps.

1. Introduction

1.1 Introduction to Periodic Pattern Mining

Periodic pattern mining is a data mining technique that focuses on discovering recurring patterns in temporal datasets. Temporal datasets contain data points associated with timestamps or time intervals, representing events or observations that occur over time. By analyzing these datasets, we can discover hidden regularities and temporal dependencies, providing valuable insights into the underlying dynamics and patterns of various phenomena.

The analysis of temporal data has gained significant importance due to the increasing availability of large-scale datasets in diverse domains such as Retail, finance, healthcare, social networks, transportation, and more. These datasets capture time-varying information, which often exhibits periodic or cyclic behaviour. Examples of periodic patterns include daily stock market fluctuations, Market Basket Analysis, weekly social media trends, seasonal disease outbreaks, and monthly electricity consumption patterns.

The discovery of periodic patterns is crucial for understanding the inherent periodicity in temporal data and extracting meaningful knowledge from it. Periodic pattern mining algorithms are designed to search for recurring patterns that repeat at regular intervals or exhibit cyclical behaviour.

1.2 Motivation for comparing periodic pattern mining algorithms on temporal datasets

Comparing periodic pattern mining algorithms on temporal datasets is essential to identify the best algorithm for extracting recurring patterns from time-dependent data. This comparison is motivated by the need for algorithm selection, performance evaluation, scalability, accuracy, generalization, and advancement of the field. Such comparisons enable researchers and practitioners to make informed decisions, enhance algorithm design, and facilitate effective analysis of temporal data in various domains.

1.3 Research Objectives and Methodology

The research objectives are:

- Compare and evaluate periodic pattern mining algorithms on temporal datasets.
- Assess performance, capabilities, and limitations of selected algorithms.
- Determine algorithm suitability for various temporal data and applications.
- Identify strengths and weaknesses in pattern detection accuracy, scalability, and efficiency.
- Contribute to advancing periodic pattern mining research and algorithm design improvements.

Methodology:
To accomplish the research objectives, the following methodology is employed:

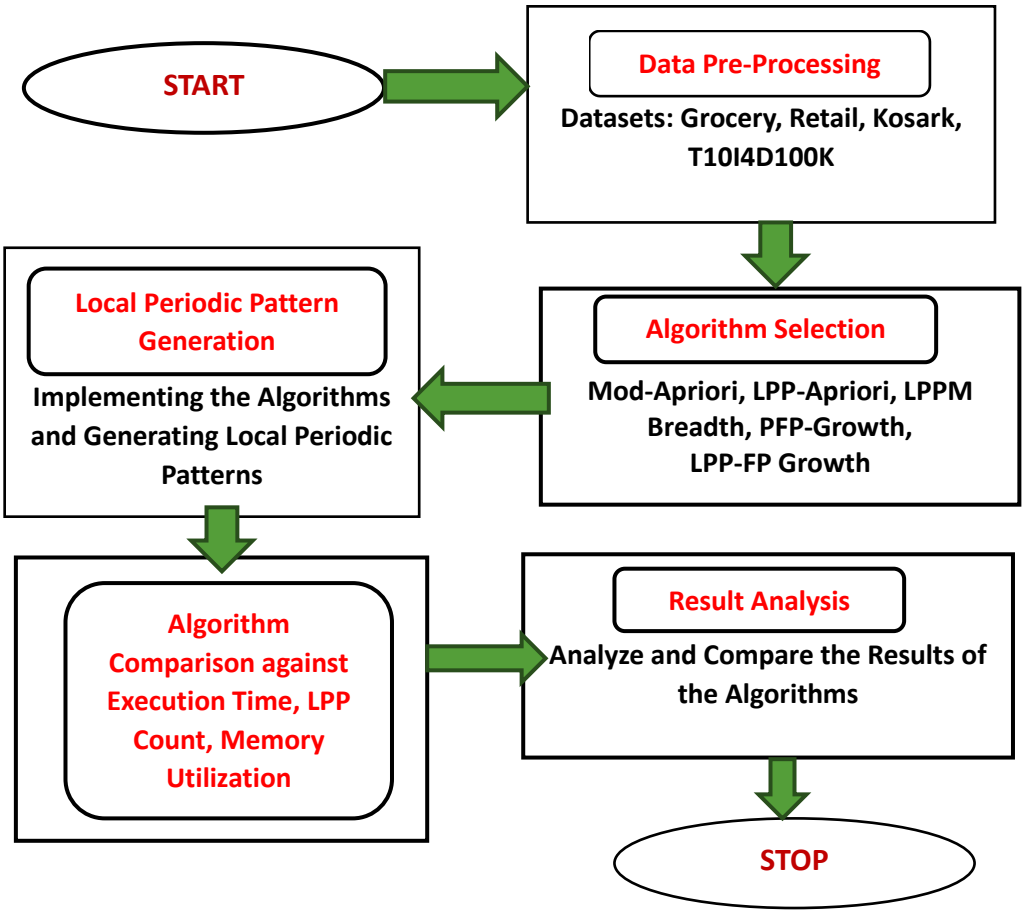

Fig 1.0 Proposed Methodology

By following this methodology, in this research paper we contribute the review of related work in the section2. In section 3 we present the description of our previously proposed algorithms. Section 4 follows the Results and discussion on comparative analysis of the algorithms. Section 5 concludes the research paper with the scope of future research directions.

2. Related Work

In the field of periodic pattern mining on temporal datasets, several studies have been conducted to compare and evaluate different algorithms. These works have contributed to the understanding of algorithmic performance, scalability, and applicability in various domains. The following is a review of some relevant related works:

Temporal data mining, as explored by Antunes and Oliviera (2001)[1], represents a significant expansion of traditional data mining techniques. By incorporating the temporal aspect, this approach allows for the extraction of more intriguing patterns that are influenced by time. Within temporal data mining, there are two main directions, as highlighted by Roddick and Spillopoulou (1999) [2]. The first direction focuses on uncovering causal relationships among events that are oriented in time. These events are arranged in sequences, where the cause of an event always precedes the event itself. The second direction involves identifying similar patterns within the same time sequence or across different time sequences.

An important expansion of the mining problem involves incorporating a temporal aspect. When a transaction occurs, its timestamp is automatically recorded. In datasets of this nature, certain patterns may exist that are specific to particular time periods rather than spanning the entire dataset's duration. Such patterns can provide valuable insights into customer behaviour. In their work (Ale and Rossi, 2000)[3], Ale and Rossi propose a method for extracting association rules that hold within the lifespan of a given item set, rather than the

lifespan of the entire dataset. However, this method does not take into account the time intervals between successive transactions involving an item set. It is not designed to identify patterns with a periodic nature.

In [15], a novel method was presented by the author to detect periodicities based on calendar intervals (annual, monthly, and daily) and interval-based temporal patterns.

In research [4], Fourier-Viger expanded the traditional problem of mining periodic patterns to discover common periodic patterns across multiple sequences. Two new measures, the standard deviation of periods and the sequence periodic ratio, were introduced. Two algorithms, MPFPS BFS and MPFPS DFS, were proposed to efficiently mine these patterns using breadth-first search and depth-first search, respectively.

While traditional techniques for frequent itemset mining are valuable, they are limited in extracting periodically appearing patterns in databases. However, in domains like market basket analysis, understanding the periodic behavior of items provides important insights for marketing strategies and inventory management. Tanbeer et al. [5] introduced the concept of Periodic Frequent Patterns (PFPs) to address this issue in transactional databases. They utilized a compact tree-based data structure called PF-tree and a pattern growth mining technique to extract complete sets of periodic frequent patterns based on user-defined periodicity and support thresholds.

In another study [6], Philippe Fournier-Viger proposed a solution to the limitations of periodic pattern mining algorithms that discard patterns with single periods exceeding a user-defined threshold. They introduced three measures: minimum periodicity, maximum periodicity, and average periodicity. These measures offered more flexibility, and an efficient algorithm named PFPM (Periodic Frequent Pattern Miner) was proposed to discover all frequent periodic patterns based on these measures. However, this approach couldn't identify patterns that were periodic only during specific time intervals or sold during specific non-predefined periods, which is common in real-life scenarios.

Adhikari [13] incorporated transaction frequency (TF) and database frequency (DF) concepts to mine locally frequent item sets and two types of periodic patterns (cyclic and acyclic) in a two-phase algorithm.

While the occurrence frequency of patterns is considered important in many applications, the temporal regularity of patterns can provide additional valuable insights. Rashid et al. [14] proposed an alternative definition of PFPs that considers the variance of interval time between pattern occurrences. They used a pattern-growth approach with user-defined minimum support and maximum variance thresholds to find regularly frequent patterns. However, this method may generate uninteresting patterns and is influenced by outlier periods.

Esther Galbrun [17] proposed an approach for mining periodic patterns from event logs using a Minimum Description Length (MDL) criterion.

Titarenko [16] presented a method for fast implementation of pattern mining algorithms with time stamp uncertainties and temporal constraints, providing an integrated approach for efficient code generation.

A study [7] introduced HOVA-FPPM, an approach based on the Apriori method with hashed occurrence vectors, to find flexible periodic patterns. However, this method requires multiple dataset scans to discover the periodic patterns.

In [8], the problem of efficiently identifying patterns with interesting behavior, such as regularly repeating occurrences within a time interval, was addressed. The author extended an existing approach derived from frequent pattern mining to operate without user-specified periodicity. The proposed algorithms could identify time intervals, periodicity, and frequency of occurrence of all periodically occurring patterns within a user-defined tolerance. However, this method discovered more fragmented patterns and was more susceptible to noise. The issue of interestingness was not addressed.

Literature [18-20] discussed the extraction of gradual patterns from temporal sequences using periodic patterns and from ordered datasets under temporal constraints.

While many algorithms exist for identifying periodic frequent patterns, most assume that the periodic behavior remains consistent over time. Philippe Fournier [9] proposed a method to discover a novel type of periodic pattern in a sequence of events or transactions, called Local Periodic Patterns (LPPs) which are patterns (sets of events) that have a periodic behaviour in some non-predefined time-intervals. Two novel measures Max-So-Perd. (Maximum Spillover Period) which allow detecting time-intervals of variable lengths where a pattern is continuously periodic, while the Min-Dur. (Minimum Duration) ensures that those time intervals have a minimum duration. These measures are used to assess the periodicity and frequency of patterns in time intervals. To discover all LPPs, the paper proposed an efficient algorithm, named LPP-Growth. It respectively adopts a pattern-growth approach by extending the FP-Growth algorithm.

Based on the literature review provided, it can be inferred that considering varying time intervals can result in the extraction of different frequent patterns, which can yield valuable insights. In this research, our focus is on identifying valid time intervals in which frequent patterns exist and determining their periodicity.

Our study involves comparing the efficiency of novel techniques for mining local periodic patterns and their associated time intervals from temporal transactional datasets.

To initiate our research, we built upon the considerations presented in a base paper [10]. Inspired by this work, we introduced a new notation called TC (Time Cube) to represent time hierarchies in the mining process. Subsequently, we implemented this novel algorithm called Modified-Apriori, which incorporates the Apriori property while introducing two new threshold values: support and density. This algorithm successfully discovers frequent itemsets along with their corresponding time intervals, merging those with neighbouring time intervals. We evaluated the algorithm's performance through experiments conducted on synthetic datasets, and during this evaluation, we identified a major limitation: the algorithm requires the user to specify the periods (time intervals) as a user-defined parameter.

To overcome this limitation, we extended our research and proposed a novel algorithm called LPP-Apriori in our previous research paper [11]. LPP-Apriori is capable of extracting various types of periodic patterns present in temporal datasets without the need for users to predefine the periods. In this method, we treated timestamps as a hierarchical data structure and extracted periodic patterns along with a list of time intervals in which they appear frequently or periodically. To determine the count of periodic patterns in different time intervals, we utilized a set operation known as Set-Superimposition, proposed by Baruah (1999), for string periods associated with itemsets. However, it was observed that this algorithm also had limitations. The assessment of periodicity relied on a strict measure called Maximum Periodicity, which assumed that a pattern must have a periodicity below a threshold value to be considered periodic. This strict measure posed a problem as real-life data often exhibits variability between periods within a pattern. Consequently, if a pattern had a single period length exceeding the Maximum threshold, it would be discarded. Thus, this method was overly strict and unable to detect patterns that are periodic only within specific time intervals rather than the entire dataset.

To address these limitations, a novel technique is required to identify local periodic patterns that consider timestamps and exhibit flexibility by finding periodic patterns within time intervals. Three challenges need to be tackled:

1) Avoiding the use of a strict definition of periodicity to discover local periodic patterns, necessitating the design of novel measures that consider the changing periodic behaviour of patterns.
2) Identifying non-periodic time intervals where patterns exhibit periodicity, requiring the identification of starting and ending points for these time intervals.
3) Efficiently finding desired patterns while minimizing the consideration of a large number of candidate patterns. Effective search space pruning techniques are essential to exclude unpromising patterns and non-periodic time intervals, as considering a large number of candidates can lead to lengthy execution times and excessive memory consumption.

To overcome these challenges, we expanded our previous method, the LPP-Apriori Algorithm, and introduced a novel technique called LPP-FP Growth Algorithm in our earlier research paper [12]. This new technique, motivated by [34], aims to identify local periodic patterns that exhibit periodic behaviour within non-predefined time intervals. The LPP-FP Growth Algorithm introduces two novel measures for assessing periodicity and frequency in time intervals:

Maximum Spillover Period (Max-So-Perd), which detects time intervals of varying lengths where a pattern exhibits periodic behaviour that may vary.

Minimum Duration (Min-Dur), a threshold ensuring that these time intervals have a minimum duration.

In the following section, we provide a brief description of our earlier proposed Algorithms.

3. Periodic Pattern Mining Algorithms

3.1 Modified Apriori:

MazaherGhorbani and Masoud Abessi, in [17] proposed a novel technique for mining frequent itemsets from temporal data. The authors proposed an efficient algorithm that focuses on discovering frequent patterns along with their associated time intervals within transactional databases.

The approach begins by introducing time cubes (TC) as a new notation to incorporate time hierarchies into the mining process. Subsequently, the authors developed an algorithm based on two thresholds: support and density, the latter being a novel addition. Frequent itemsets are identified, and those with neighbouring time intervals belonging to the same frequent itemsets are merged. This technique assumes that patterns can exist in some or all time intervals. To achieve this, a time cube analysis of frequent patterns is conducted. The entire dataset is partitioned into various time cubes, such as (hour, day, month), (day, month, year), etc., and the

Apriori algorithm is applied to these time cube data. The concept of temporal support value is utilized. The minimum support threshold is employed to assess the frequent itemsets. However, due to uneven distribution of records within time intervals, some occasions may have very few records, resulting in potentially invalid discovered patterns lacking sufficient evidence to demonstrate their validity throughout the entire time interval. This issue leads to an overestimation problem. To address these challenges, an additional threshold called Density is proposed. Density not only ensures the validity of the patterns but also eliminates time intervals with few transactions, thereby mitigating the overestimation of timespans.

Nevertheless, the algorithm described in the paper has certain limitations. The processing time is significant as the itemsets need to be mined for each time cube. Additionally, one limitation of the algorithm is that the time intervals are user-defined, potentially restricting its flexibility and adaptability.

3.2 LPP Apriori:

LPP Apriori, an improved version of the Modified Apriori algorithm, was developed to mine periodic patterns in temporal datasets. In a previous research paper [11], we introduced LPP Apriori to overcome the limitations of the Modified Apriori algorithm. Our approach leverages timestamp data to identify frequent itemsets and their corresponding time intervals.

In this method, we treat time as a hierarchical structure and dynamically extract local periodic patterns by considering time intervals without requiring explicit user input. During the scanning process, the algorithm automatically identifies time intervals where patterns exhibit frequency. By utilizing the Spillover Period and Maximum Spillover Period of itemsets, the algorithm constructs time intervals for periodic patterns. These intervals are pruned based on user-defined thresholds for Minimum Duration and Minimum Support. The algorithm generates candidate periodic patterns iteratively and prunes infrequent ones according to the specified thresholds.

While LPP Apriori offers a straightforward and intuitive approach, scalability can be a concern when dealing with large datasets since multiple scans are required to generate and validate candidate patterns. Furthermore, a major limitation of LPP Apriori lies in its strict evaluation of periodicity using the Maximum Periodicity measure. This criterion assumes that a pattern must have a periodicity below the given threshold to be considered periodic. However, real-life data often exhibits variability between periods. Unfortunately, if a pattern has a single period exceeding the Maximum threshold, it is discarded. Consequently, this strict measure fails to capture patterns that are periodic only within specific time intervals rather than spanning the entire dataset.

To address these limitations, we have extended our research with a novel technique called LPP-FP Growth, which is discussed in the subsequent section.

3.3 LPP FP Growth

Tree-based algorithms such as LPPM Breadth, LPPM Depth, and LPP-FP Growth offer effective solutions for discovering periodic patterns in temporal datasets. These algorithms leverage tree structures to efficiently capture and extract recurring patterns. In this section, we present a detailed description of our proposed LPP-FP Growth algorithm [12].

The LPP-FP Growth algorithm utilizes three important parameters: Max-Perd, Max-So-Perd, and Min-Dur. The Max-Perd parameter allows users to specify the maximum expected time between consecutive occurrences of periodic patterns. With the Max-So-Perd parameter, patterns can temporarily exceed the Max-Perd threshold if the cumulative spillover (surplus) remains below the Max-So-Perd value. This parameter enhances flexibility compared to traditional periodic frequent pattern mining algorithms that rely solely on the maximum periodicity constraint. Lastly, the Min-Dur parameter defines the minimum length of time intervals and is utilized to discard short intervals. It is crucial to set these parameters appropriately, as they depend on the dataset characteristics and user preferences.

The LPP-FP Growth algorithm operates in two steps:

(i) It first compresses the input database into a tree structure called the time-interval periodic frequent tree (LPP-tree).

(ii) It then recursively mines the LPP-tree to discover all Local Periodic Patterns (LPPs).

Inspired by the tree-based pattern-growth approach of the FP-Growth algorithm, the LPP-FP Growth algorithm has been specifically adapted for the task of mining LPPs.

3.3.1 LPP-Tree Structure

The algorithm consists of two main components: a prefix-tree and an LPP-list. The LPP-list contains entries with two fields: an Item Name and a Periodic Time-Interval List (PTL). The prefix-tree, known as the LPP-tree, draws inspiration from the structure of the FP-tree used in FP-Growth for transaction storage. Similar to the FP-tree, the LPP-tree organizes transactions as paths within a tree, where each node represents an item.

However, there is a significant distinction between the FP-tree and the LPP-tree. In the LPP-tree, nodes explicitly store occurrence information about items in transactions to efficiently calculate the PTL of patterns. Specifically, a timestamp list known as the TS-list is stored in the last node of every transaction within the LPP-tree. As a result, the LPP-tree maintains two types of nodes: Ordinary Nodes and Tail Nodes.

3.3.2 Mining LPP-Tree

Once the LPP-tree has been constructed, the proposed algorithm no longer requires scanning the original database to discover Local Periodic Patterns (LPPs). This is because all the crucial information for LPP mining is stored within the LPP-tree. To identify LPPs, our algorithm utilizes a depth-first search approach, similar to the FP-Growth algorithm, to explore the itemset search space. The exploration begins with LPPs that consist of a single item, which are stored in the LPP-list.

The algorithm performs a depth-first search exploration, taking as input the initial LPP-tree (T), user-defined thresholds such as Maximum Period, Maximum Spillover Period, and Minimum Duration, and an itemset (α) for LPP identification. By pruning itemsets that lack periodic time-intervals, the algorithm can uncover all LPPs while minimizing unnecessary search efforts.

4. Implementation &Experimental Setup

In this section, we discuss the implementation of our proposed Algorithms LPP-Apriori, LPP-FP Growth and existing algorithm in research literature Modified Apriori and LPPM-Breadth. Here we evaluate the performance of the algorithm in terms of execution time, local pattern count, and memory utilization. The performance of our proposed algorithm is compared with earlier local periodic pattern algorithms based on Apriori based Approach and Tree based Approach. These proposed algorithms are implemented in Python 3.10.2 on Intel® Core™ i5-10210U CPU @1.60 GHz on Windows 11 operating system. We have collected the data from FIMI-Frequent Itemset Mining Dataset Repository (http://fimi.uantwerpen.be/data/)& from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Online+Retail+II).

We conducted the experiments on different datasets including Grocery, Kosark, Retail, and T10I4D100K.

4.1 Description of Data sets

1. Grocery

This is the groceries data with the list of items bought by customers. From the left side is the number of items in a basket then Item 1, 2, 3, etc. stands for list of the item. The dataset contains 10027 transactions by customers shopping for groceries. The data contains 169 unique items. The data is suitable to do data mining for market basket analysis which has multiple variables. We have taken subset of this data fromhttps://www.kaggle.com/datasets/irfanasrullah/groceriesweb page.

2. Kosark

This dataset was provided by Ferenc Bodon and contains (anonymized) click-stream data of a Hungarian On-Line New Portal. This is a very large dataset containing 990 000 sequences of click-stream data. The dataset was converted in SPMF format using the original data from: http://fimi.ua.ac.be/data/. We have downloaded a subset of this data from https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php web page.

3. Retail

This dataset was donated by Tom Brijs and contains the (anonymized) retail market Basket data from an anonymous Belgian Retail Store. The data is collected over three non-consecutive periods. The total amount of receipts being collected equals 88,162. The average number of distinct items (i.e. different products) purchased per shopping visit equals 13 and most customers buy between 7 and 11 items per shopping visit.

4. T10I4D100K

This dataset was generated using the generator from the IBM Almaden Quest research group. It is available on the http://fimi.uantwerpen.be/data/ web page.

The characteristics of the above dataset(s) after formatting into a temporal dataset are shown in the following table.

Table 1 Characteristics of Dataset(s)

SNO	DATASET	NO. OF TRANSACTIONS	NO. OF ITEMS	MIN. TRANS. SIZE	MAX. TRANS. SIZE
1	Grocery	10027	86	1	11
2	Kosark	44953	18287	1	699
3	Retail	88162	16470	1	76
4	T10I4D100K	100000	870	1	29

4.2Incorporating Temporal Features in datasets

As the dataset is non-temporal so it cannot be directly used. We have incorporated the temporal features in the dataset so that it can be a temporal dataset and can be used by our proposed Algorithm. We developed a program for this. The program takes starting date and time (Timestamp), the number of Transactions or records in the dataset, and synthetic data as inputs. It then generates the series of Timestamps equal to the number of transactions and merges the generated Timestamps with the dataset as the first column entry in the dataset. Dataset features such as the number of items, number of transactions, average transaction length, and time span of the dataset can be set by the user. The Algorithms for pre-processing the data is given below:

To pre-process data by converting non-temporal data into temporal or timestamp data and handling missing values, has been done with the following steps:

1. Identify non-temporal data: Analyze the dataset and identify columns that contain non-temporal data but can be converted into temporal or timestamp data. For example, columns containing dates or time information represented in a non-temporal format.

2. Convert non-temporal data to temporal format: For each identified column, apply appropriate data conversion techniques to convert the non-temporal data into a temporal or timestamp format. This has been done by parsing and reformatting the values using Python's datetime module or specific date/time libraries.

3. Handle missing values: Missing values are common in datasets and can impact the quality of data analysis. Considered the following technique to handle missing values:

Imputation: Since the missing values are significant or removing them would result in data loss, we have considered imputing or filling in the missing values with Zero or NaN Values.

4. Fix the Number of Columns:

To fix the number of columns in a text file using Python, you can follow these steps:

Step1. Read the text file.

Step2. Split each line of the file by the delimiter (such as a comma or tab) to obtain the individual columns.

Step3. Check the number of columns in each line.

Step4. If the number of columns is less than the desired number, append empty values to make up the difference.

Step5. If the number of columns is greater than the desired number, truncate the list of columns to the desired length.

Step6. Join the columns back together using the delimiter.

Step7. Write the modified lines to a new file or overwrite the original file.

5. Update or save the pre-processed data

Finally, we have updated the original dataset with the pre-processed values or save the pre-processed data to a new file for further analysis.

We have implemented the proposed Algorithm Mod-Apriori in Python 3.10.2 on the windows 10 Operating System. The executive summary and the experimental results of the Proposed Algorithm implemented on different formatted temporal data sets are discussed in the following section.

4.3 Implementation Results

We have implemented the proposed Algorithm Mod-Apriori in python 3.10.2 on the Windows 11 Operating System. The executive summary and the experimental results of the Proposed Algorithm implemented on different formatted temporal data sets are discussed in this section. The frequent patterns of Grocery Dataset along with timestamp intervals received after implementing the LPP-Apriori Algorithm is shown below as an example. The Generated Frequent Patterns with Time intervals are:

Frequent One Itemset
Item Starting Date Ending Date Item Count Item Support
[{'bread'}, '09-10-2000 14:00:00', '14-10-2000 04:00:00', 11, 0.2]
[{'citrus'}, '19-02-2000 00:00:00', '25-02-2000 14:00:00', 16, 0.2]
[{'fruit'}, '01-01-2000 00:00:00', '15-04-2002 12:00:00', 2131, 0.21]
[{'milk'}, '01-01-2000 04:00:00', '15-04-2002 02:00:00', 2750, 0.27]
[{'other'}, '04-11-2001 00:00:00', '15-04-2002 12:00:00', 385, 0.2]
[{'pip'}, '22-10-2000 20:00:00', '25-10-2000 20:00:00', 8, 0.22]
[{'rolls/buns'}, '01-01-2000 12:00:00', '18-06-2000 22:00:00', 411, 0.2]
[{'rolls/buns'}, '22-06-2000 00:00:00', '01-09-2000 18:00:00', 185, 0.21]
[{'rolls/buns'}, '16-08-2001 14:00:00', '16-01-2002 14:00:00', 371, 0.2]
[{'vegetables'}, '01-01-2000 08:00:00', '15-04-2002 12:00:00', 2898, 0.29]
[{'water'}, '13-06-2001 04:00:00', '21-06-2001 22:00:00', 23, 0.22]
[{'white'}, '10-10-2000 00:00:00', '13-10-2000 22:00:00', 11, 0.23]
[{'whole'}, '01-01-2000 04:00:00', '19-09-2000 14:00:00', 796, 0.25]
[{'whole'}, '23-09-2000 08:00:00', '15-04-2002 02:00:00', 1717, 0.25]
Frequent-TWO itemset
Item Starting Date Ending Date Item Count Item Support
[{'fruit', 'citrus'}, '19-02-2000 00:00:00', '25-02-2000 14:00:00', 16, 0.2]
[{'fruit', 'pip'}, '22-10-2000 20:00:00', '25-10-2000 20:00:00', 8, 0.22]
[{'milk', 'other'}, '01-10-2001 20:00:00', '07-10-2001 10:00:00', 15, 0.22]
[{'milk', 'whole'}, '01-01-2000 04:00:00', '19-09-2000 14:00:00', 796, 0.25]
[{'milk', 'whole'}, '23-09-2000 08:00:00', '15-04-2002 02:00:00', 1717, 0.25]
[{'vegetables', 'other'}, '04-11-2001 00:00:00', '15-04-2002 12:00:00', 385, 0.2]
[{'whole', 'other'}, '01-10-2001 20:00:00', '07-10-2001 10:00:00', 14, 0.21]
[{'vegetables', 'whole'}, '03-09-2000 00:00:00', '19-09-2000 14:00:00', 41, 0.2]
[{'vegetables', 'whole'}, '21-07-2001 08:00:00', '08-08-2001 06:00:00', 44, 0.2]
[{'vegetables', 'whole'}, '08-09-2001 06:00:00', '12-09-2001 18:00:00', 11, 0.2]
Frequent-THREE itemset
Item Starting Date Ending Date Item Count Item Support
[{'milk', 'whole', 'other'}, '01-10-2001 20:00:00', '07-10-2001 10:00:00', 14, 0.21]
[{'vegetables', 'whole', 'other'}, '01-10-2001 20:00:00', '07-10-2001 10:00:00', 14, 0.21]

We have received the same type of results after implementing the LPP-Apriori, Mod-Apriori, LPP-FP Growth algorithms on different datasets as shown above. We have not included them due to space problem.

5. Results and Discussion

In this section, we delve into the analysis of performance for the proposed algorithms, namely LPP-Apriori and LPP-FP Growth, as well as other Python-implemented algorithms including Mod-Apriori, PFP Growth, and LPP Breadth. Our analysis focuses on two key parameters: Execution time and the generation of Local Periodic Patterns. The objective of our research revolves around generating Frequent Itemsets within specific time intervals. We aim to explore how these item sets manifest across different temporal intervals, highlighting their detection limitations in non-temporal datasets.

In the second experiment, we conduct a comparative assessment of the algorithm's execution performance against our previously developed Periodic Pattern Algorithm. The algorithms are executed on each dataset while varying parameter values. The purpose is to compare algorithm performance in diverse scenarios and observe the impact of parameter settings on their effectiveness. Generally, as the values of Max-So-Per and Max-Perd increase, and Min-Dur decreases, the search space is expected to expand, potentially uncovering more patterns and resulting in longer runtimes. However, it should be noted that the performance of the same

algorithm can vary significantly across different datasets due to varying dataset characteristics, even with identical parameter values.

5.1 Execution Time

In this section, we present a comparative analysis of the runtime performance for the proposed algorithms LPP-Apriori and LPP-FP Growth, as well as the implemented algorithms Modified-Apriori, PFP-Growth, and LPP-Breadth. The table below displays the execution time taken by each algorithm, measured in seconds.

Table 2 Comparative Table of Execution Time

Dataset	Execution Time (Sec.)				
	Mod-Apriori	LPP-Apriori	PFP-Growth	LPP Breadth	LPP-FP Growth
Grocery	46.25	51.57	50	54	47
Kosark	407.5	894.98	680	354	421
Retail	912.5	1755	1400	936	751
T10I4D	452.7	339	185	127	109

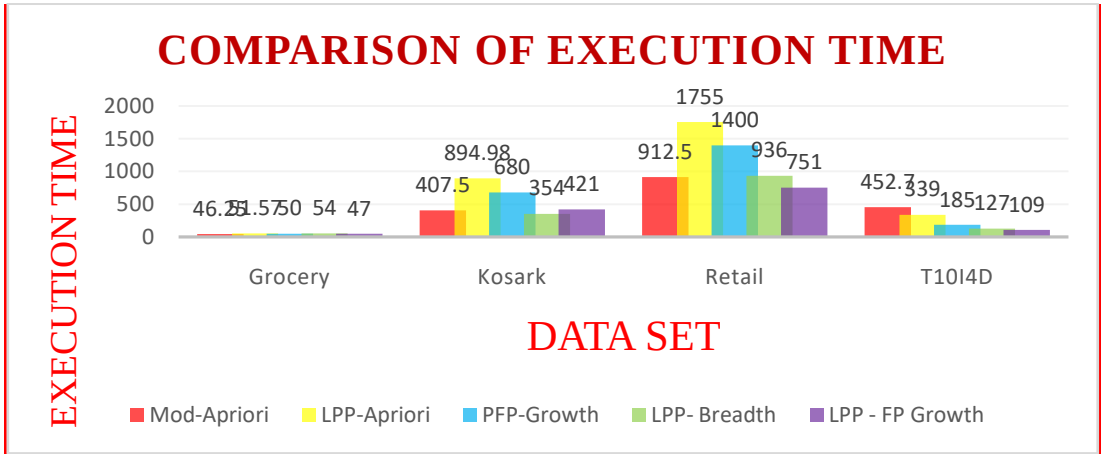

Fig2 Comparison of Execution Time

Based on the comparative graph above, it is evident that the LPP-FP Growth Algorithm exhibits shorter execution times compared to LPP-Apriori, Mod-Apriori, PFP-Growth, and LPP-Breadth Algorithms. The LPP-FP Growth algorithm demonstrates superior speed compared to the other algorithms. The graph reveals that it takes 751 seconds for execution on the Retail dataset, which is smaller in size compared to the T10I4D dataset where it takes 109 seconds. This disparity can be attributed to the Retail dataset generating a higher number of periodic patterns and being denser in nature. Therefore, we can conclude that the LPP-FP Growth algorithm excels in runtime efficiency compared to the other algorithms. The objective of improving the algorithm's runtime efficiency has been successfully achieved through the implementation of LPP-FP Growth.

5.2 Memory Usage

In the third experiment, we compared the peak memory consumed by our proposed algorithm LPP-FP Growth, LPP Apriori and earlier algorithms Mod-Apriori, PFP-Growth, LPP Breadth. The statistics of these algorithms is shown in the following table.

Table 3Comparative Table of Memory Usage

Dataset	Memory Usage				
	Mod-Apriori	LPP-Apriori	PFP-Growth	LPP Breadth	LPP-FP Growth
Grocery	59	115	165	162	147
Kosark	297	629	345	321	282
Retail	602	1342	363	480	560
T10I4D	642	1432	456	392	125

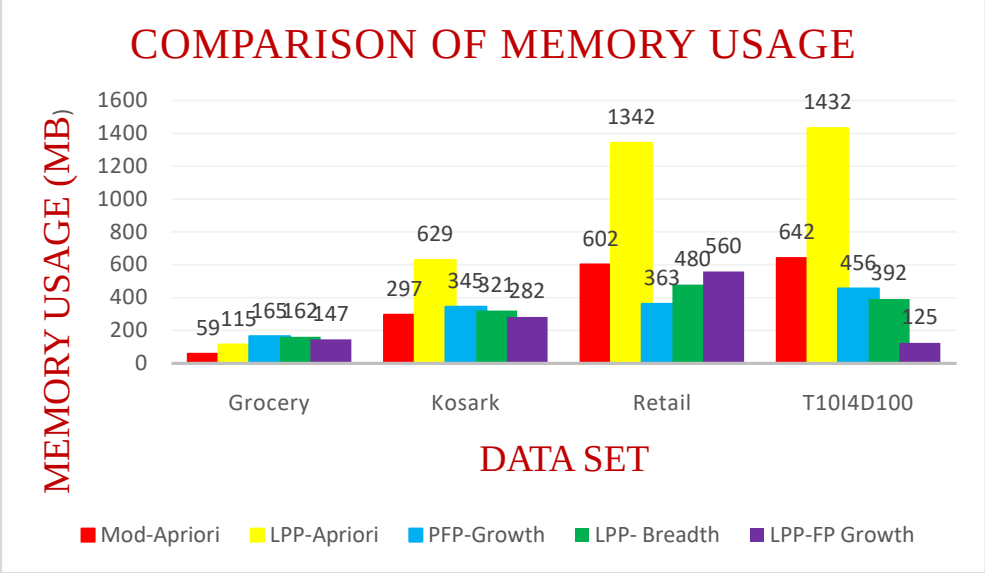

Fig3 Comparative Graph of Memory Usage

From the above Figure it is observed that our proposed Algorithm LPP-FP Growth consumes less memory than the other Algorithms. So we can conclude that the LPP-FP Growth is memory efficient than other Algorithms. The reason behind this is it uses a compact tree structure for representing the database, and timestamps are only saved in tail nodes.

5.3 LPP Count

In this we discuss the comparative results of our proposed algorithms LPP-Apriori, LPP-FP-Growth and implemented algorithms Mod-Apriori,PFP-Growth and LPP Breadth for the Local Periodic Patterns generated by the algorithms. The following table shows the LPP-count of algorithms on various datasets.

Table 4Comparative table of LPP Count

Dataset	LPP Count				
	Mod-Apriori	LPP-Apriori	PFP-Growth	LPP Breadth	LPP-FP Growth
Grocery	16	26	58	65	73
Kosark	12	38	24	38	41
Retail	23	13	84	89	94
T10I4D	12	22	46	52	55

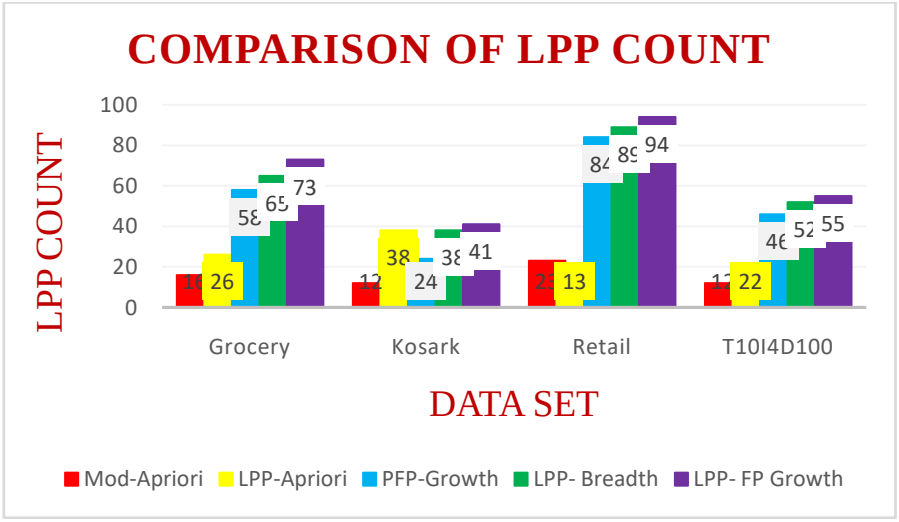

Fig4 Comparative Graph of LPP Count

Based on the comparative chart provided, it can be observed that the LPP-FP Growth Algorithm outperforms Mod-Apriori, LPP-Apriori, LPP-Breadth, and PFP Algorithms in terms of generating a greater number of Local Periodic Patterns. This indicates that the LPP-FP Growth Algorithm effectively prunes uninteresting patterns and focuses on detecting the recurring and relevant local patterns.

Through the aforementioned comparisons, it is evident that the LPP-FP Growth Algorithm excels in various aspects, including Execution time, Memory usage, and the generation of meaningful Local Periodic Patterns. The comparative results demonstrate that the LPP-FP Growth algorithm utilizes less memory, produces a higher number of LPPs, and exhibits faster runtime performance compared to the other algorithms, namely LPP-Apriori, Mod-Apriori, LPP-Breadth, and PFP-Growth algorithms.

5.4 Pattern Count Analysis of LPP FP Growth

In this section we analysed the LPP Count of LPP FP Growth Algorithm on the Grocery Dataset by varying the parameters Maximum Period, Maximum Spill Over Period, and Minimum Duration. The results obtained are shown in the following table.

Table 5 LPP Count Analysis

Max-Perd	Max.Spillover Perd.	Min.Dur	LPP count	Execution Timein Sec.	Memory Usage In MB
3	3	7	12	67.88	150.21
3	3	10	6	63.9	148.28
5	5	7	57	28.45	160.43
5	5	10	41	28.57	156.07

From the above table, it is observed that by increasing the Maximum Period and Maximum Spillover Period and decreasing the Minimum duration the LPP Count is increased, i.e. we can generate more number of Local Periodic Patterns by varying these parameters. The impact of these measures on the number of LPPs discovered is comparable to their effect on the runtime of the algorithms. This is due to the fact that modifying these parameters affects the size of the search space, which tends to increase or decrease in proportion to the number of patterns.

5.5 Pattern Analysis

In order to evaluate the effectiveness of the proposed LPP Apriori Algorithm, an analysis of patterns discovered in the Grocery dataset, consisting of customer transactions, was conducted. The objective of this experiment was to determine whether the algorithm identifies interesting patterns that offer insights into customer shopping behavior. By adjusting the parameters, it was discovered that setting Max.Perd to 2 days, MaxSoPerd to 2 days, and Min.Dur to 10 days resulted in the identification of several noteworthy patterns. A selection of these patterns is presented in the following table.

Table 6 Pattern Analysis

[{'vegetables', 'whole-Milk'}, '03-09-2000 00:00:00', '19-09-2000 14:00:00', 41, 0.2]
[{'vegetables', 'whole-Milk'}, '21-07-2001 08:00:00', '08-08-2001 06:00:00', 44, 0.2]
[{'vegetables', 'whole-Milk'}, '08-09-2001 06:00:00', '12-09-2001 18:00:00', 11, 0.2]

As depicted above, the first pattern highlights the periodic sale of Vegetable and Milk products occurring during specific time intervals: from '03-09-2000 00:00:00' to '19-09-2000 14:00:00', from '21-07-2001 08:00:00' to '08-08-2001 06:00:00', and from '08-09-2001 06:00:00' to '12-09-2001 18:00:00'. This pattern reflects the purchasing behavior of customers during these time periods. It is noteworthy that this pattern exhibits periodicity only for a short duration within the entire year and repeats within the same time intervals across multiple years (2000 and 2001). Consequently, traditional periodic frequent pattern mining algorithms, which primarily focus on identifying patterns that are periodic throughout the entire year, would disregard this pattern. Moreover, it is important to note that this pattern encompasses multiple distinct periodic time intervals within the same year.

Overall, it has been noted that the algorithms exhibit satisfactory performance and possess the ability to identify intriguing patterns within real-world data that traditional models, which primarily emphasize periodicity throughout the entire database, fail to uncover. The suggested algorithms have the capability to unveil patterns within time intervals that are not predetermined, with each algorithm demonstrating superior performance under

specific circumstances. There are various potential avenues for future research, including exploring alternative forms of localized periodic patterns, accommodating streaming data, and devising techniques for automated parameter selection.

6. Conclusion

This study presents an innovative comparative analysis of Local Periodic Pattern Mining Algorithms, specifically Modified Apriori, LPP-Apriori, PFP-Growth, LPP-Breadth, and LPP-FP Growth. The assessment of these algorithms is conducted using three real-world datasets and one synthetic dataset. The evaluation focuses on three key aspects: Execution time, Memory Utilization, and the generation of Local Period Patterns. The results reveal that the LPP-FP Growth algorithm outperforms other periodic pattern mining algorithms in terms of Execution time, Memory Consumption, and the generation of local periodic patterns. The experimental evaluation across various datasets demonstrates the efficiency of the LPP-FP Growth algorithm, highlighting its ability to discover valuable patterns that traditional periodic pattern mining algorithms fail to identify.

In future research, there would be intriguing to adapt the suggested algorithms for implementation within big data frameworks, incorporating technologies like GPUs, multi-threading, and other high-performance computing approaches. Such adaptations would enable the processing of larger databases in shorter time frames. Another intriguing avenue for exploration involves extending the proposed model to handle non-static databases, such as incremental databases and data streams. Furthermore, there is the possibility of extending the model to accommodate other types of patterns, such as sequential patterns and rules, or designing automated methods for parameter selection.

References:

[1]. C. Antunes, Arlindo L. Oliveira ,Temporal Data Mining: an overview, Proceedings Antunes 2001 Temporal DM, Published 2001, Computer Science

[2]. John F Roddick, Myra Spiliopoulou, Temporal data mining: Survey and Issues, School of Computer and Information Science, University of South Australia ISBN: ACRC-99-007, January 1999

[3]. J. M. Ale and G. H. Rossi, “An approach to discovering temporal association rules,” Proc. ACM Symp. Appl. Comp.-vol. 1, 2000, pp. 294–300.

[4]. Philippe Fournier-Viger, Zhitian Li, Efficient algorithms to identify periodic patterns in multiple sequences, ©2019 Elsevier Inc.

[5]. Syed KhairuzzamanTanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee, T. Discovering Periodic-Frequent Patterns in Transactional Databases, PAKDD 2009, LNAI 5476, pp. 242–253, 2009.c Springer-Verlag Berlin Heidelberg 2009.

[6]. [6] Philippe Fournier-Viger, PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures,Perspectives in Science, April 16, 2016.

[7]. M. Javed ,Nawaz , K. Khan, HOVA-FPPM: Flexible Periodic Pattern Mining in Time Series Databases Using Hashed Occurrence Vectors and Apriori Approach Scientific Programming (IF 1.672) Pub Date: 2021-01-04 , DOI:10.1155/2021/8841188.

[8]. A. Krzywicki, A. Mahidadia and M. Bain, "Discovering Periodicity in Locally Repeating Patterns," 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA), Shenzhen, China, 2022, pp. 1-10, doi: 10.1109/DSAA54385.2022.10032435.

[9]. Philippe Fournier-Viger, Peng Yang, Rage Uday Kiran, Maria Luna, Mining Local Periodic Patterns in a Discrete Sequence, Journal of LATEX Templates September 20, 2020.

[10]. MazaherGhorbani and Masoud Abessi, "A New Methodology for Mining Frequent Itemsets on Temporal Data", in IEEE Transactions on Engineering Management, Vol. 64, Issue. 4, pp. 566 - 573, Nov 2017.

[11]. Md. Ghouse Mohiuddin, Dr. D.L.Srinivasa Reddy, A Novel Technique for Temporal Frequent Itemset Mining, IJIRT 156966 INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY, © October 2022 | IJIRT | Volume 9 Issue 5 | ISSN: 2349-6002.

[12]. Md. Ghouse Mohiuddin, Dr.D.L.Srinivasa Reddy, Discovery of Local Periodic Patterns in Temporal Data: A Novel Approach, The International journal of analytical and experimental modal analysisVolume XV, Issue I, January/2023ISSN NO: 0886-9367Page

[13]. Jhimli Adhikari, Mining Periodic Patterns from Non-binary Transactions, Journal of Intelligent Computing Volume 9 Number 4 December 2018.

[14]. Md. Mamunur Rashid, Md. Rezaul Karim, Byeong-Soo Jeong, Efficient Mining Regularly Frequent Patterns in Transactional Databases, Database Systems for Advanced Applications, 2012, Volume 7238, ISBN : 978-3-642-29037-4

[15]. P. Fournier-Viger, Z. Li, J. C. Lin, R. U. Kiran, H. Fujita, Efficient algorithms to identify periodic patterns in multiple sequences, Inf. Sci. 489 (2019) 205–226.
[16]. Sofya S. Titarenko, Valeriy N. Titarenko, Georgios Aivaliotis and Jan Palczewski, Fast implementation of pattern mining algorithms with time stamp uncertainties and temporal constraints, Titarenko et al. J Big Data (2019) 6:37, Springer. https://doi.org/10.1186/s40537-019-0200-9
[17]. Esther Galbrun, Cellier ,Tatti1,Termier, and Bruno Cremilleux, Mining Periodic Patterns with a MDL Criterion, ECML PKDD, 2018, pp. 535–551.
[18]. Jerry Lonlac, Yannick Miras, Aude Beauger, An Approach for Extracting Frequent (Closed) Gradual Patterns Under Temporal Constraint July 2018, DOI:10.1109/FUZZ-IEEE.2018.8491665, Conference: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
[19]. Mining, Jerry Lonlaca, Arnaud Donieca, Stephane Lecoeuche, Extracting Seasonal Gradual Patterns from Temporal Sequence Data Using Periodic Patterns, Elsevier October 21, 2020.
[20]. J. Lonlac, E. MephuNguifo, A novel algorithm for searching frequent gradual patterns from an ordered data set, Intelligent Data Analysis, Volume 24, Issue 5, 2020 pp 1029–1042.
[21]. Kiran R.U., Veena P., Zettsu K., Efficient Discovery of Partial Periodic Patterns in Large Temporal Databases,Electronics 2022, 11, 1523.
[22]. Johnny, F. A., Bennett, E. O. Sako, D. J. S., A Model For Exploration Of Periodic Patterns In A Database, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 11, Issue 1, January 2022.
[23]. YalingXun , L. Wang , H.Yang Jiang Hui, Mining relevant partial periodic pattern of multi-source time series data, Information Sciences, (Elsevier) Volume 615, November 2022, Pages 638-656
[24]. FIMI-Frequent Itemset Mining Dataset Repository (http://fimi.uantwerpen.be/data/)
[25]. UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Online+Retail+II).
[26]. https://www.kaggle.com/datasets/irfanasrullah/groceries

