Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 2, November 2021, pp. 1084~1090
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i2.pp1084-1090

IPOC: an efficient approach for dynamic association rule generation using incremental data with updating supports

P. Naresh, R. Suguna
Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India

Article Info

Article history:

Received May 25, 2021
Revised Sep 16, 2021
Accepted Sep 21, 2021

Keywords:

Dataset
Incremental tree generation
IPOC
Nodeset

ABSTRACT

According to recent statistics, there was drastic growth in online business sector where more number of customers intends to purchase items. Due to these retailers accumulates huge volumes of data from day to day operations and engrossed in analyzing the data to watch the behavior of customers at items which strengthen the business promotions and catalog management. It reveals the customer interestingness and frequent items from large data. To carry out this there was known algorithms present which deals with static and dynamic data. Some of them are lag time and memory consuming and involves unnecessary process. This paper intents to implement an efficient incremental pre ordered coded tree (IPOC) generation for data updates and applies frequent item set generation algorithm on the tree. While incremental generation of tree, new data items will link to previous nodes in tree by increasing its support count. This removes the lagging issues in existing algorithms and does not need to mine from scratch and also reduces the time, memory consumption by the use of nodeset data structure. The results of proposed method was observed and analyzed with existing methods. The anticipated method shows improved results by means of generated items, time and memory.

Corresponding Author:

P. Naresh
Department of Computer Science and Engineering
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology
Chennai, India
Email: pannanginaresh@gmail.com

1. INTRODUCTION

Now a day there is more competition in business sector which aims to attract the customers with latest and interested items. At the same time it is required to identify the user interests on items by applying association item analysis. It will help to know the frequent items bought by different users of all transactions at a super market or store. By analyzing these data one can suggest more items to customers depending on their frequency which intern increase the sales and also it assist in maintaining a good and updated catalog of items. It is achieved by the implementations of various trendy data mining approaches at analyzing data and makes ideal decisions. Data mining is a fundamental process in knowledge mining which discovers unknown facts, needy information and interesting patterns form massive data. Frequent itemset mining [1] and association rule mining plays a crucial role on finding user interests from transactional database.

Association rule generation from frequent items from retail and real world datasets employs a vital role. Apriori and frequent pattern growth (FP-growth) are the fundamental algorithms for mining transactional dataset to discover frequent items [2]. Apriori generates candidate key at each itemsets and then

verifies the support threshold, which is a time consuming process. The other FP-growth doesn't generate candidate key but it builds a frequent pattern tree (FP-tree) for all transactional items. While linking the item with the node of tree it checks its support and occurrences at each transaction. Support and Confidence are qualitative measures for refining the transactions to know the item frequencies. High utility [3]-[5] ARM is task which performs mining by considering utilities of respective items [6], [7]:

- Support–It is the count of occurrence of a particular item I at all transactions.
- Confidence of X➔Y–It is the probability of occurrence of more than one item or item combinations in a single or all transactions.

The Table 1 contains sample transactional data with transaction id and respective items which were brought together. The Table 2 holds the information about all items and their respective supports. *Mobile➔Charger (confidence 70%)* means that 70% of the customers who bought mobile also buy a charger.

Table 1. Sample transactional data

Transactions	Items
T501	I2,I5,I6
T502	I3,I4,I6
T503	I1,I3,I5,I6
T504	I2,I4,I5
T505	I1,I3,I5,I6

Table 2. Items with support count

Item	Support
I1	2
I2	2
I3	3
I4	2
I5	4

In real time, new data generated drastically as continuous transactions [8] were performed by different customers at retail stores. Furthermore existing dataset can be updated with new information continuously which makes the dataset asymmetrical due to updates [9], and it is very difficult to analyze these kinds of datasets [10] and also increase the complexity on mining frequent items. Previous methods which deal with frequent items mining were applicable to static datasets where no changes were made further. There exist some algorithms like frequent itemset with nodeset (FIN) [11] which will increase the fast of mining process but unaware of handling with incremental or dynamic data.

Chiu *et al.* [12] implemented a tree based method for dealing with dynamic data using FCFP tree. It depends on generation of full compression tree generation. Afterward Deng [13] discovered a novel data structure called nodeset, to increase the mining accuracy. In their paper they generated pre order coded (POC) tree and pre–post order coded (PPC) tree depends on ordered transactional items to fast mining. Whenever new data added to existing, it is tough to mine accurately and sometimes earlier generated items or rules gets invalid. All this process entirely becomes worthless due to start from scratch recursively for all newly added data. To conquer these issues and to deal with dynamic dataset [14], this paper aims to implement a novel and efficient algorithm called incremental pre ordered coded tree. It works with nodeset data structure creation and applies frequent itemset mining (FIM) on the tree [15]. The forthcoming sections of this paper explores about literature survey of existing approaches related to ARM and dynamic item set mining, proposed algorithm which handles the dynamic data by the use of nodeset data structure, analysis of results (comparison with existing algorithms) and finally ends with conclusion section.

Dynamic itemset mining and association rule mining [16] has been challenging era in data mining from incremental datasets. As the data in retail industry or supermarket intends to changes, continuous mining become a complex task. So it is essential to develop a data structure mostly suitable for handling updating data or new data. It maintains information regarding added transactions dynamically [17]. This process discovers new frequent itemsets by eliminating infrequent itemsets depending on their updated support [18] thresholds. Existing methods scan the data base from scratch for each added transaction which is a time consuming and had unnecessary tasks. So finally this survey aims on identifying novel and improved algorithms for fast mining and assimilates with dynamic datasets using machine leaning approaches [19].

Sun *et al*. [1] in their paper "Incremental Frequent Itemsets Miningwith FCFP Tree" implemented a new tree based data structure for maintaining both frequent and infrequent items to overcome time wastage. They used compression technique to save space while storing itemsets. When support of items changes even that time also it showed good results.

Unil and Lee [20], wrote a paper on "Incremental mining of weighted maximal frequent itemsets from dynamic databases", discussed how to mine maximal frequent itemsets by considering item weights. Depends on itemset weights, those sets with maximum weight will be taken first as most frequent items. Deng [11], "Fast mining frequent itemsets using Nodesets", defined a new data structure named nodeset. It stores the items information in tree structures either pre order or post order of their arrival in each transaction. There after applied a mining algorithm on that tree to mine fast. They proved their proposal increases the mining speed and accuracy. Yao and Hamilton [21], in their article "Mining itemset utilities from transaction

databases", mentioned utility for each item and depends on item utility mining of itemsets and rules was done. Items with highest utility will be preferred as maximum priority and it will be considered as first itemset.

Bagui and Stanley [22], in their research article "Mining frequent itemsets from streaming transaction data using genetic algorithms", considered streaming data as input to mine and generate frequent itemsets. Streaming data [23], [24] means where data in and out takes place on a particular database. It is subjected to continuous changes respect to time. So these kinds of datasets were very difficult to mine. The authors projected a genetic algorithm which accepts data in sliding window format as input and used drift concept to pick frequent items.

Chiu *et al.* [12], "Incremental mining of closed inter-transaction itemsets over data stream sliding windows", applied a novel mechanism to find closed itemsets. They also considered streaming data to mine. From that they first mine sliding window information after that slide will move to next part of dataset. Those items which are very close relation to fulfill strong association were given highest priority in mining. They proved it is better applicable for streaming data.

Viger *et al*. [25], "FHM+: faster high utility itemset mining using length upper-bound reduction", discussed high utility mining. They defined FHM+ algorithm for fast mining depends on items utility. While taking items utilities, also focused on upper bound of all transactions to reduce infrequent items.

Deng and Lv [26], titled "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children–Parent Equivalence pruning", wrote an algorithm named PrePost+ for efficient mining process. A tree was generated with left and right child's by make use of N-list as anetwork [27], and applied pruning at left part and right part of the tree referred as equivalence pruning. It proved more efficient than existing approaches.

Han *et al*. [28], "Mining frequent patterns without candidate generation: a frequent-pattern tree approach", is a general approach for mining frequent items. It doesn't generates candidate key for each 1-item, 2- itemsets. It stores support count of all items in a table and generated an fp-tree where each node links with respected items [29]. Dawar dan Goya [30], "UP - Hist tree: An efficient data structure for mining high utility patterns from transaction databases", derived a novel tree structure named UP-Hist tree. It maintains a histogram of each transactional items and associate it with each node of the tree. The histogram allows calculation of enhanced efficacy estimates for effectual pruning of the search space.

Tseng *et al*. [31], "Efficient algorithms for mining high utility itemsets from transactional databases", implemented UP-Growth and UP-Growth+ algorithms for mining high utility itemsets. At each stage pruning was done to filter irrelevant items and low utility items [32]. Postdiffset algorithm [33] in rare pattern reduces candidates count and consumed less time for long transactional databases.

2. PROPOSED METHODOLOGY

Scanning the database is not much complex when volume of data is fixed but when it is gradually increasing then it's difficult to mine. In this regard it is needed that to update current datastructure with new data without processing from scratch. To achieve this it is mandatory to define and construct a datastructure which needs one scan for dynamic mining. Moreover the dataset should be sorted in an appropriate order to fasten the process and removes barrier on searching for random items. To conquer the mentioned problems a nodeset structure was suggested. The nodeset requires either pre order or post order of nodes in the tree**.** Here preorder was taken for implementation so it is required to generate POC tree.

Along with nodeset, FIN algorithm used to discover frequent items from scratch. It was achieved by selecting node by node in set enumeration tree, and also avoids repetitive search by using pruning at each node. By considering pre order only it is possible to select frequent items by level wise from tree. To achieve desired dynamic itemset mining the POC tree needs dynamic updations incessantly for each newly added record. It is done by maintaining a dataset that stores each and every transaction information and called as Incremental pre ordered coded (IPOC) tree.

The steps followed in generation of dynamic itemsets by addition of new data are:

Input: a dataset D.

- Generate a normal pre ordered coded (POC) tree for D.
- Append new data d1 to original data D.
- Generate IPOC tree for existing and new data (link new items info with existing nodes).
- Apply FIM algorithms on updated data to mine frequent itemsets.
- If new data d2 added then repeat step 3 and 4 to get updated items.

Output: Frequent itemsets for dynamic data. Consider a dataset D, given as shown in Table 3:

Table 3. Transaction database with ordered items of min_sup 2

Transaction ID	Items	Ordered Items
T200	I1,I6,I7	I1,I6
T201	I1,I2,I3,I5	I2,I3,I5,I1
T202	I2,I3,I5,I9	I2,I3,I5
T203	I2,I3,I5,I8	I2,I3,I5,
T204	I2,I3,I4,I5,I6	I2,I3,I5,I6

This Table 3 contains transaction id's, items respect to each transaction and ordered items derived by applying min_support(2). I4, I7, I8 and I9 are eliminated form ordered items list because, their support is less than min_support. By using the rest or items a POC tree is going to be generated as shown in Figure 1. So in next step, generate a POC tree for given data D.

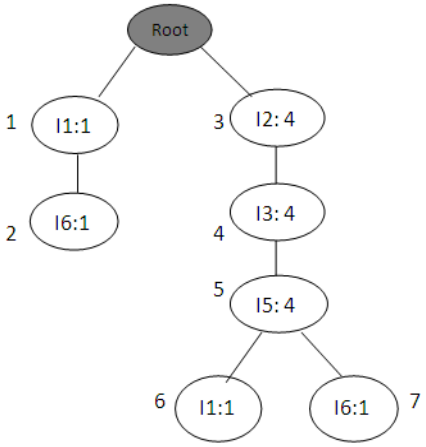


Figure 1. POC tree

A new data d1 is added to D, and then the POC will become IPOC by making update. After adding new dataset d1 to D in Table 4, some items support satisfies the min_sup criteria. So I7 and I8 will be added to existing tree and also new frequent items will be generated including new data. Depends on the frequency count of each item the items order will change in ordered items list. So it is not needed to mine from scratch (existing data). Instead of that new items count is updated in tree structure, by seeing count it is easy to mine frequent items. The following tree describes the IPOC structure.

Table 4. Ordered items of original (D) and new transaction database (d1)

	Transaction ID	Items	Ordered Items
	T200	I1,I6,I7	I1,I6,I7
	T201	I2,I3,I5	I2,I3,I5
Original Data - D	T202	I2,I3,I5,I9	I2,I3,I5
	T203	I2,I3,I5,I8	I2,I3,I5,I8
	T204	I2,I3,I4,I5,I6	I2,I3,I5,I6
New data - d1	T205	I1,I6,I7,I8	I1,I6,I8,I7
	T206	I1,I2,I3,I8	I2,I3,I1,I8
	T207	I1	I1

In the Figure 2, root node is empty node and had child's. Each node represents one item like I1, I2… along with its frequency count. Outside the each node the items preorder is represented. It shows the order in which all the transactional items were mined to obtain frequent items. Form the obtained items, it is needed to draw valid association rules which give relationship among all items and tells the analyst which items are most frequent. By using IPOC tree structure, more time needed for mining will save and mining fast, efficiency is improved. These are the most frequent association rules drawn from D and d1 with 4 items: {I1→I6→I8→I7}, {I2→I3→I5→I8}, {I2→I3→I5→I6} and {I2→I3→I1→I8}.

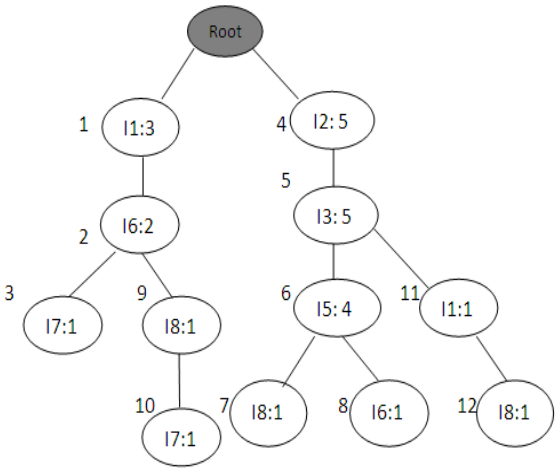


Figure 2. IPOC tree for old data (D) and new data (d1)

3. RESULTS AND DISCUSSION

Experimental background: Python 3.6.4 was used for implementing the needed algorithms. The jupyter lab used as integrated development environment (IDE) where source code is written and suitable datasets were uploaded. The datasets may filtered to give more efficient information, and here it is possible to analyze the datasets and outcomes by means of graphs and scatter plots. To achieve this it is mandatory to use built-in libraries which support additional functionalities.

Datasets used: To analyze and discover the frequent items some standard datasets were needed. In this implementation mushroom, connect, chess and online retail datasets were used. Mushroom, connect and chess datasets were downloaded from frequent itemset mining implementations (FIMI) repository and online retail dataset was taken from UCI machine learning repository [34]. The following Table 5 displays the datasets with their instances, attributes, size and no of items present. All the datasets were transactional datasets more suitable for mining of frequent items. Table 6, illustrates the comparative study (statistics) of existing and proposed algorithms by means of time and memoy consumptions for different datasets.

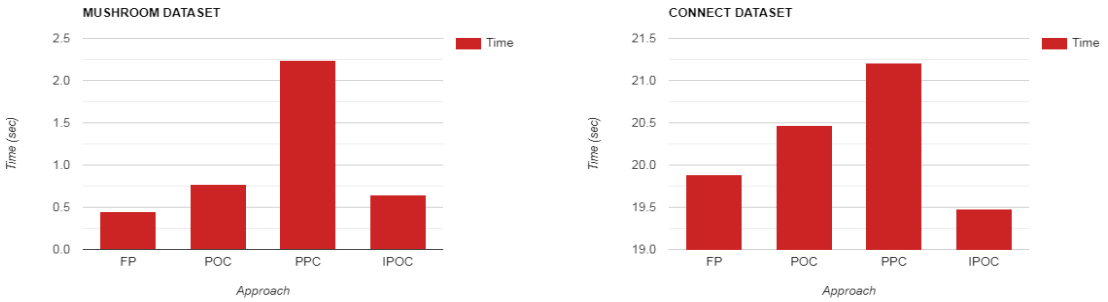
Table 5. Datasets description

Dataset	No of Instances	No of Attributes	Size	Items
Chess	3196	36	349KB	76
Mushroom	8124	22	365KB	119
Connect	67557	42	5829KB	129
Online Retail	541909	8	23160KB	2603

Table 6. Comparison of all datasets

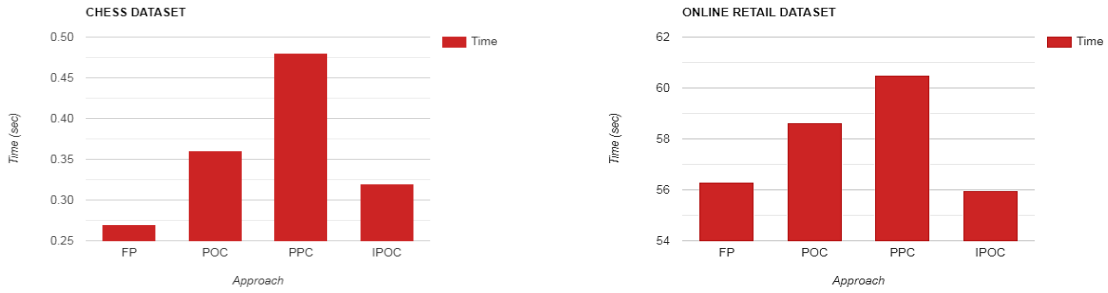
Algorithm/Dataset	Mushroom		Connect		Chess		Online retail	
	Time (sec)	Memory (MB)	Time (sec)	Memory (MB)	Time (sec)	Memory (MB)	Time (sec)	Memory (MB)
FP	0.45	27.29	19.89	47.85	0.27	17.95	56.29	76.25
sPOC	0.77	19.29	20.47	44.28	0.36	15.46	58.65	74.62
IPOC (Proposed)	2.24	18.87	21.21	53.49	0.48	15.93	61.87	83.54

The Figure 3 illustrates time taken by FP, POC, PPC and IPOC algorithms for mushroom dataset, in 3(a) and connect dataset in 3(b). The IPOC showed better performance while mining frequent items. Figure 4 illustrates time taken by FP, POC, PPC and IPOC algorithms for chess dataset, in 4(a) and retail dataset in 4(b). The IPOC showed better performance while mining frequent items.



(a) (b)

Figure 3. Time comparisons of datasets; (a) mushroom (b) connect datasets



(a) (b)

Figure 4. Time comparisons of datasets; (a) chess (b) online retail datasets

4. CONCLUSION

In this paper, finding frequent itemsets and association rule mining for various datasets were analyzed. While working with static datssets there is no issues in complexity in mining and time consumption. When dealing with dynamic datsets it is subject to continuous updates and mining become difficult as dataset varies in size and transaction count. Some previous algorithms referred in survey were unaware of dynamic data handling and some were doing unwanted tasks which deteriorate the mining process by wasting time. So those issues were handled in this paper by adopting tree structures and FIM algorithms which interns enhanced to work with dynamic database. Proposed IPOC tree with nodeset considers preorder of all transactional items and incrementally update the tree whenever new dataitems were added. FIN algorithm parallelly and continuously works on tree to mine frequent items and reduce number of candidates in the process of constructing trees. Four data sets were considered to analyze time and memory parameters of existing and proposed. The proposed approach proved reduced time consumption, took less memory and efficient for dynamic datasets.

REFERENCES

[1] J. Sun, Y. Xun, J. Zhang, and J. Li, "Incremental Frequent Itemsets Miningwith FCFP Tree," *IEEE Access*, vol. 7, pp. 136511-136524, 2019, doi: 10.1109/ACCESS.2019.2943015.

[2] P. S. Yu, and Y. Chi, "Association Rule Mining on Streams," In: *LIU L., ÖZSU M.T*, Boston, MA, 2009. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_25

[3] T. L. Dam, H. Ramampiaro, K. Nørvåg, and Q. H. Duong, "Towards efficiently mining closed high utility itemsets from incremental databases," *Knowledge-Based Systems*, vol. 165, 2019, doi: 10.1016/j.knosys.2018.11.019.

[4] U. Yun, and H. Ryang, "Incremental high utility pattern mining with static and dynamic databases," *Applied intelligence*, vol. 42, pp. 323-352, 2015, doi: 10.1007/s10489-014-0601-6.

[5] V. Goyal, S. Dawar, and A. Sureka, "High utility rare itemset mining over transaction databases," In *International Workshop on Databases in Networked Information Systems*, pp. 27-40, 2015, doi: 10.1007%2F978-3-319-16313-0_3.

[6] G. Liu, H. Lu, J. X. Yu, W. Wei, and X. Xiao, "AFOPT: An Efficient Implementation of Pattern Growth Approach," In *Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations*, vol. 80, Nov. 2003.

[7] M. Man, A. J. Julaily, S. I. A. Saany, W. A. W. A. Bakar, and M. H. Ibrahim, "Analysis study on R-Eclat algorithm in infrequent itemsets mining," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 5446-5453, 2019, doi: 10.11591/ijece.v9i6.pp5446-5453.

[8] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and S. Y. Philip, "HUOPM: High-Utility Occupancy Pattern Mining," *IEEE Transactions on Cybernetics*, vol. 50, no. 3, 2020, doi: 10.1109/TCYB.2019.2896267.

[9] H. Qiu, R. Gu, C. Yuan, and Y. Huang, "YAFIM: A parallel frequent itemset mining algorithm with spark," In: *IEEE international parallel distributed processing symposium workshops*, pp. 1664-1671, 2014, doi: 10.1109/IPDPS W.2014.185.
[10] L. K. Ramasamy, S. Kadry, Y. Nam, and M. N. Meqdad,. "Performance analysis of sentiments in Twitter dataset using SVM models," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2275-2284, 2021, doi: 10.11591/ijece.v11i3.pp2275-2284.
[11] Z. H. Deng, and S. L. Lv, "Fast mining frequent itemsets using Nodesets," *Expert Syst. Appl.*, vol. 41, no. 10, pp. 4505-4512, 2014, doi: 10.1016/j.eswa.2014.01.025.
[12] S. C. Chiu, H. F. Li, J. L. Huang, and H. H. You, "Incremental mining of closed inter-transaction itemsets over data stream sliding windows," *Journal of Information Science*, vol. 37, no. 2, pp. 208-220, 2011, doi: 10.1177/0165551511401539.
[13] Z. H. Deng, "DiffNodesets: An efficient structure for fast mining frequent itemsets," *Applied Soft Computing*, vol. 41, pp. 214-223, 2016, doi: 10.1016/j.asoc.2016.01.010.
[14] Y. Li, Z.-H. Zhang, W.-B. Chen, and F. Min, "TDUP: an approach to incremental mining of frequent itemsets with three-way-decision pattern updating," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 441-453, 2015, doi: 10.1007/s13042-015-0337-6.
[15] T. P. Hong, C.W. Lin, and Y. L. Wu, "Incrementally fast updated frequent pattern trees," *Expert sytems with Applications*, vol. 34, no. 4, pp. 2424-2435, 2008, doi: 10.1016/j.eswa.2007.04.009.
[16] K. R. Prasad, "Optimized High-Utility Itemsets Mining for Effective Association Mining," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 5, pp. 2911-2918, doi: 10.11591/ijece.v7i5.pp2911-2918.
[17] C. W. Lin, T. P. Hong, G. C. Lan, J. W. Wong, and W.-Y. Lin, "Incrementally mining high utility patterns based on pre-large concept," *Applied Intelligence*, vol. 40, no. 2, pp. 343-357, 2014, doi: 10.1007/s10489-013-0467-z.
[18] M. Shapol, J. Karwan, and Z. Subhi, "A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, pp. 552-562, 2014, doi: 10.11591/ijeecs.v22.i1.pp552-562.
[19] M. Mahmood, B. Al-Khateeb, and W. M. Alwash, "A review on neural networks approach on classifying cancers," *International Journal of Artificial Intelligence*, vol. 9, no. 2, pp. 317-326, 2020, doi: 10.11591/ijai.v9.i2.pp317-326.
[20] Y. Unil, and G. Lee, "Incremental mining of weighted maximal frequent itemsets from dynamic databases," *Expert Systems with Applications*,vol. 54, pp. 304-327, 2016, doi: 10.1016/j.eswa.2016.01.049.
[21] H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction databases," *Data and Knowledge Engineering*, vol. 59, no. 3, pp. 603-626, 2006, doi: 10.1016/j.datak.2005.10.004.
[22] S. Bagui, and P. Stanley, "Mining frequent itemsets from streaming transaction data using genetic algorithms," *Journal of Big Data*, vol. 7, no. 54, 2020, doi: 10.1186/s40537-020-00330-9.
[23] F. Nori, M. Deypir, and M. H. Sadreddini, "A sliding-window based algorithm for frequent closed itemset mining over data streams," *Journal of Systems and Software*, vol. 86, no. 3, 2013, doi: 10.1016/j.jss.2012.10.011.
[24] H. Ryang, and U. Yun, "High utility pattern mining over data streams with sliding window technique," *Expert Systems with Applications*, vol. 57, pp. 214-231, 2016, doi: 10.1016/j.eswa.2016.03.001.
[25] P. Fournier-Viger, J. C.-W. Lin, Q.-H. Duong, T.-L. Dam, "FHM+: faster high utility itemset mining using length upper-bound reduction," In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2016, pp. 115-127, doi: 10.1007/978-3-319-42007-3_11.
[26] Z. H. Deng, and S. L. Lv, "PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via Children–Parent Equivalence pruning," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5424-5432, 2015, doi: 10.1016/j.eswa.2015.03.004.
[27] A. R. Hakim, T. Djatna, and A. Febransyah, "Technology Map: A Text Mining and Network Analysis Approach," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 3, no. 1, pp. 200-208, 2016, doi: 10.11591/ijeecs.v3.i1.pp200-208.
[28] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," *Data Mining and Knowledge Discovery*, vol. 8, 2004, doi: 10.1023/B:DAMI.0000005258.31418.83.
[29] D. S. Maylawati, H. Aulawi, and M. A. Ramdhani, "Flexibility of Indonesian text pre-processing library," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no. 1, pp. 420-426, 2019, doi: 10.11591/ijeecs.v13.i1.pp420-426.
[30] S. Dawar, and V. Goya, "UP-Hist tree: An efficient data structure for mining high utility patterns from transaction databases," In *Proceedings of the 19th international database engineering & applications symposium*, 2015, pp. 56-61, doi: 10.1145/2790755.2790771.
[31] V. S, Tseng, B.-E, Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1772-1786, 2013, doi: 10.1109/TKDE.2012.59.
[32] M. Man, W. A. W. A. Bakar, M. M. A. Jalil, and J. A. Jusoh, "Postdiffset algorithm in rare pattern: An implementation via benchmark case study," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 4477-4485, 2018, doi: 10.11591/ijece.v8i6.pp.4477-4485.
[33] M. Deypir, M. H. Sadreddini, and M. Tarahomi, "An efficient sliding-window based algorithm for adaptive frequent itemset mining over data streams," *Journal of Information Science and Engineering*, vol. 29, no. 5, pp. 1001-1020, 2013.
[34] M. Man, and M. A. Jalil, "Frequent itemset mining: technique to improve eclat based algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 6, pp. 5471-5478, 2019, doi: 10.11591/ijece.v9i6.pp5471-5478.