Comparison of Transaction Reduction and Dynamic Itemset Counting Implementations of the Apriori algorithm in Python 3.8

RAYNE CARL B. PARCON, JAMES EUGENE L. SETIAS, MARY THERESE M. HOLIPAS, and EISEN ED BRIONES

Philippine Science High School Western Visayas Campus - Department of Science and Technology (DOST-PSHS WVC), Brgy. Bito-on, Jaro, Iloilo City 5000, Philippines

Article Info

Submitted: Jun 10, 2022
Approved: Aug 24, 2022
Published: Aug 27, 2022

Keywords:
Big-O notation
Time complexity
Running time
Frequent itemsets
Level-wise algorithm

Abstract

The Apriori algorithm is a level-wise algorithm designed to find frequent itemsets. Various improvements called implementations were made to improve efficiency. These are classified into five categories based on how the algorithm processes data. Existing studies focus on the comparison of the classical Apriori with a different implementation. With dynamic itemset counting (DIC) under dataset partitioning algorithm and transaction reduction under incremental update algorithm, a comparison of the running time between algorithms from different categories might not be accurate; thus, the time complexities in Big-O notation were solved. Both algorithms had the same time complexity, $O(n^2)$; thus, a comparison of the running times was made. The running times were recorded after processing three datasets of varying size and complexity. It was determined that DIC is faster by 8434% while having the same accuracy. Both algorithms slowed down at large and complex datasets; however, the DIC was still faster.

Introduction. - An algorithm is a set of specific rules or instructions typically executed in a computer to solve problems or perform tasks. [1] Different obstacles or tasks also come in different ways on how to tackle them. Since various factors can affect the outcome of an algorithm, it is essential to know how efficiently they can perform a task. Evaluating the time complexity of an algorithm is a way to see how long an algorithm will run as a function of the length of the input used.

This study focused on the Apriori algorithm, a recognized level-wise algorithm commonly used for Association Rule Mining (ARM). However, datasets collected from transactions have significantly increased in size compared to datasets ten years ago. This created a dilemma for the Apriori algorithm, as it repeatedly scans the whole transactional dataset, which increases computational cost. This dilemma has kept attracting researchers for years by developing and applying different techniques to improve its efficiency [2].

Several studies recommended methods to cope with this dilemma, each having advantages and drawbacks. Some of these methods reduced the time consumed in itemset generation [3]. Another technique reduced the memory space used, simplified support counting, and the generation of candidate itemsets via logical bitwise operations [4]. For more significant applications, algorithms that can enhance the algorithm's ability to scan and work with large data sets and extract information from the dataset are in high demand [2]. This led to a more efficient Apriori Algorithm by minimizing the number of dataset scans and limiting the input and output costs [2].

Data Mining requires expertise as well as a systematic approach. Both are especially lacking in many small and medium-sized enterprises [5]. The Apriori algorithm is an algorithm viable for SMEs [6]. However, there are different implementations of it, thus the need to determine which is faster in processing datasets. Because there is a lack of comparison between algorithm implementations on how efficient and accurate they would be in processing large datasets within a specified time. By comparing the different algorithm implementations, users will be able to identify which Apriori algorithm implementation is best suited for the size and complexity of their datasets. Additionally, studies such as Castro et al. [7] have not yet identified which Apriori implementations are the best at handling datasets that vary in size and complexity. As the Apriori algorithm can be used in market basket analysis to determine the trend in what the customers are purchasing [8], businesses can use the Apriori algorithm to make better decisions in the pricing of their products or product bundles to improve sales. Due to better business decisions, the customers and the business owners alike would be able to save money.

The main limitation of the Apriori algorithm is the costly wasting of time to hold vast numbers of frequent candidate sets with frequent itemsets, which causes the algorithm to scan the datasets repeatedly [3]. According to Brin et al. [9], one of the dynamic itemset counting (DIC)'s weaknesses would be that it does not do well with homogenous data; however, it improves when the order of transactions

How to cite this article:
CSE: Parcon RC, Setias JE, Holipas MT. 2022. Comparison of Transaction Reduction and Dynamic Itemset Counting Implementations of the Apriori Algorithm in Python. Publiscience. *5*(1): 44–49.
APA: Parcon, R.C., Setias, J.E., & Holipas, M.T. (2022). Comparison of Transaction Reduction and Dynamic Itemset Counting Implementations of the Apriori Algorithm in Python. *Publiscience*, *5*(1), 44–49.

For supplementary data, contact: publiscience@wvc.pshs.edu.ph.

is randomized. However, the transaction reduction implementation (TRI) tends to have several dataset scans when processing the transactions [10]. Thus, this research aims to know which of the two is faster in processing datasets.

These implementations were selected because they are two of the commonly used association rules mining algorithms: DIC belongs to the Data Set Partitioning Algorithm, and TRI belongs to Incremental Update Algorithm [11]. The computed time complexity would be used to know which is more efficient by interpreting the algorithms of the implementations. The time complexity is generally an equation, which outputs the time it takes to complete its tasks given the number of transactions and the computer's components. With this, both algorithms can be compared, even with different setup specifications. Additionally, researchers can explain why one algorithm is faster than the other using time complexities.

Evaluating the time complexity of an algorithm is a way to see how long an algorithm will run as a function of the length of the input used. It can be written in many ways, such as an algebraic expression. However, the most common type of expressing the time complexity of an algorithm is through Big O notation to represent the worst-case analysis which uses the worst performance of any input of a given size to summarize its performance [12].

The Big O notation was utilized in this study in generating the time complexities of both implementations. The Big O notation is the most common metric for solving time complexity, and it uses the steps needed to complete a task to generate its execution time. It is denoted by the notation of O(n), where O is the order of growth and n is the length of the input. It expresses the run time of an algorithm in terms of how quickly it grows relative to the input 'n' by defining the N number of operations that are done in it.

This study may serve as a reference for future researchers that wish to develop an improved implementation for the algorithm. The methods can also become a basis for those who want to compare other implementations of the Apriori algorithm. The results from this study may provide information in serving as a comparison to an improved algorithm that focuses on the data processing speed.

Methods. - The study aims to determine which of the two algorithms to be tested would be faster and more accurate. The procedures to be used in this study could serve as a basis for how other researchers would test other algorithms under the Apriori algorithm. To compare the algorithms, the time complexity of both codes was calculated using Big-O notation which represents the upper bound of the running time of an algorithm.

To give support on which of the two algorithms is faster, both algorithms were tested by gathering the time it takes for both algorithms to run in three different datasets. The factors that were kept constant are the number of background tasks and the hardware of the computer to be used since CPU and memory speed and utilization are the factors that greatly affect the performance of a computer [13]. Calculation of the accuracy of the algorithms on different datasets and the running of tabs aside from the ones needed for the data gathering were recommended by the study of Tank [14].

The hardware used for data gathering was one (1) desktop computer with the following specifications: A desktop computer with Ryzen 5 3600 (Base clock: 3.6 GHz, Boost Clock: 4.2GHz), 16 GB DDR4 RAM, and a Sapphire Pulse RX 5500 XT GPU. The two (2) data sets to be used were sourced from Kaggle and one (1) from the UCI Machine Learning Repository. The datasets contain different numbers of transactions and unique items. A customer transaction dataset containing the items bought by the customers, a spambase dataset containing spam and non-spam emails, and a mall customer segmentation dataset that contains customer information.

The Python source codes for the different implementations were gathered from GitHub and were modified to output the execution time of the algorithms and the frequency of frequent itemsets.

For data collection, the process has three trials for each of the three (3) datasets which were used in previous research by Delos Arcos & Hernández [2] and Saabith et al. [15]. Each trial measured the amount of time it takes an implementation to process the entire data set. Additionally, the Process Hacker 2 application will be used to make sure that the CPU and GPU utilization rate is constant all throughout the data gathering process. The codes to be used were checked and verified by our consultant, Dr. Rosslin Robles.

The data gathering process was conducted in the residences of the researchers as it is the most accessible place to put a desktop without any hassle or permit needed to utilize it. The codes for the algorithms (transaction reduction implementation and dynamic itemset counting implementation) were taken from GitHub. The code was set up in Python and had a runtime counter added to it. The data sets used were sourced from Kaggle, which contained different amounts of transactions and unique items.

Sourcing and Checking of Algorithms and Implementing the Time Library. Algorithms were first searched in GitHub. Since Python is the language used by the researchers, the search filter only caters to Python codes. After narrowing it down to a couple of codes, the researchers chose an algorithm then tested them out to see if the algorithm had any errors. If the tested algorithm has no errors, that would be the code to be used for data gathering. For both algorithms to output the time of each run, the time library needs to be implemented. To implement this, the snippet "import time" was added at the start of the code so that the code would have access to the said library. After implementing the time function in the codes of the algorithms, the implemented codes were checked by an external consultant, Doctor Rosslin Robles. If the codes are to be judged by our consultant as ready for data gathering, the said algorithms would be used for data gathering.

Calculation of the Time Complexity of the Algorithms. The time complexity of the algorithms was solved using the method of Vaz et al. [16]. This study made use of the Big-O notation for the time complexity as it is used to get the upper bound or the worst-case scenario for the running time of the algorithms. The algorithm was evaluated line-by-line by assigning

values such as 1, x, x^2, log n, etc. to each line or loop. The lines of code were grouped by the function they are under. After assigning the values of each line and loop of the code, it was grouped according to which function they are a part of. The sums of each were solved, this resulted in an algebraic expression. After getting the sums of each group, it was all combined to get the time complexity of the algorithm. In identifying the time complexity of the algorithm using Big-O notation, the resulting algebraic expression's term with the highest power was identified. Then, by disregarding the coefficient of the mentioned term, that was the time complexity of the algorithm. For example, the resulting algebraic expression is $6x^2 + 2x + 1$, which means that the time complexity of the algorithm is O(n^2).

Cleaning of the Data Sets. After opening the excel file, the researchers need to determine what data the algorithms could process. The data that was not determined to be processed by the algorithm, such as the personal information of people, were removed. Data that is usually determined to be removed are labels. Then, proceed to the other excel files and repeat the process. Then, the cleaned excel files and algorithms were checked by the previously mentioned external consultant.

Preparation of Parameters and Execution of Algorithms. Three trials were performed for each algorithm and the average running time of the three trials were computed. To ensure that the results were not affected by other factors, unrelated background tasks were closed. This action kept the CPU and RAM utilization available for use constant. Dataset number 1 was processed using the TRI and DIC for three trials, followed by datasets 2 and 3. After every run, the runtime and rules were recorded into an excel file.

Calculating the Accuracy of the Algorithms. The output after every run, excluding the processing time, was noted. Then, the frequency of each itemset in the dataset was identified. From this, the confidence levels of each rule were calculated. The highest confidence value that was calculated will be the accuracy of the algorithm for that run. The three runs' accuracies will then be averaged.

Data Analysis. After all the values were recorded to an Excel sheet, the analysis of data took place. The first process was to analyze the time complexities of the transaction reduction and dynamic itemset counting algorithms. This portion of the study is a modified version of Delos Arcos and Hernandez [2] and Tank [14] methods, wherein the mean of the three trials of each algorithm will first be solved. The mean values of the TRI and DIC were then used to solve the percentage of performance difference. Lastly, the processed values were graphed. For accuracy, the confidence levels of each rule or output were first determined. The highest confidence value out of all the rules is the accuracy of the algorithm for that run. Except for the worst-case analysis, these steps were done on Microsoft Excel as all the tools needed for the computations are present in said application. The worst-case analysis measures the resources required by the algorithm given the input size n. Examples of the resources are the algorithm's running time and the memory used to run the algorithm.

Computation of parameters from the raw data. This is done so that the succeeding step of the data analysis would be easier to do. The mean of the 3 trials was solved using the equation:

$$\frac{V1+V2+V3}{3}$$

where V1, V2, and V3 are the values of 3 trials and divided by 3. Then, the percentage of performance difference of the algorithms in the 3 datasets were solved using the equation:

$$AD = \frac{1}{VD}$$
$$AT = \frac{1}{VT}$$
$$\left(\left|\frac{AD}{AT}\right| - 1\right) \times 100$$

where VT is the average run time by the transaction reduction algorithm, while VD is the average run time by the dynamic itemset counting algorithm, and AT and AD are the reciprocals of VT and VD, respectively.

The accuracy of each was solved using the equation:

$$\frac{\sigma(X \cup Y)}{\sigma(X)}$$

where (X) is the frequency an itemset, X, appears in the dataset, while (XY) is the frequency itemset Y appears in the same transaction as itemset X. Then, the confidence of each of the rules was to be multiplied by 100 so that it would be presented in percent form.

Safety Procedure. Safety precautions during the data gathering process were observed to avoid possible harm. The researchers have identified three hazards: electrical outlets, ungrounded components, and spending too much time in front of a computer. To mitigate the identified hazards, having water near the outlets was strictly avoided as well as tampering with the hardware, unless knowledgeable enough of the surroundings and components. Breaks were also taken from time to time to prevent asthenopia or eye strains.

Results and Discussion. - The researchers first identified the number of transactions for each dataset which were gathered from their respective repositories. The number of transactions [4] and the number of distinct items [14] of the algorithms were also gathered. The number of distinct items was available on the respective repositories of the datasets and verified using the remove duplicates option in Microsoft Excel. Lastly, the average transaction length was calculated by determining the average number of characters per transaction.

Table 1 shows that D1 consists of the lengthiest transactions, while D3 has the shortest transactions. Additionally, D3 is the largest dataset while having the most number of distinct items. This means that D3 is the most diverse or randomized data among the three datasets. Then, D2 has the lowest amount of transactions and distinct items. D2 is also a small

dataset, according to the definition of Althnian et al. [17] of a small data set having 18 to 1030 transactions. However, dataset sizes are not explicitly defined [17], thus the two other datasets will be defined relative to D2 or the small dataset. As shown in Table 1, D3 has the most number of transactions and therefore, the largest dataset. Since D1's transaction count is in between the smallest and largest dataset of this study, it is the medium-sized dataset. This means that D1 (Groceries) is the most complex dataset among the three, while D3 (Spambase) is the least complex. Then, D2 (Mall Customer) is the least randomized dataset, while D3 (Spambase) is the most randomized dataset used in this study.

Table 1. The table shows the properties of the datasets used in the study.

Data Set	Number of Transactions	Distinct Items	Average Length of Transactions Length (number of characters)
D1 (Groceries)	43,349	169	10.548
D2 (Mall Customer)	800.00	102	2.798
D3 (Spam Base)	266,858	4499	1.617

In Table 2, D3 resulted in the highest percentage of performance difference among the two algorithms. As D3 is the most randomized data among the three datasets, it coincides with the findings of Brin et al. [9] that dynamic itemset counting's performance improves when the transactions are randomized. Out of the three parameters from Table 1, the running time of transaction reduction was greatly affected by the average transaction length of the dataset, as it had the highest running time in D1.

Table 2. The table shows the average processing times of transaction reduction and dynamic itemset counting on three data sets

Data Set	Transaction Reduction (seconds)	Dynamic Itemset Counting (seconds)	Percentage of Performance Difference (average running time of DIC vs. TRI)
D1	1165.51032	36.99194	3051%
D2	0.62623	0.06089	929%
D3	755.92940	3.52844	21324%

Fig. 1 shows the comparison in the percentage of performance difference of the two algorithms' average running times in the different datasets. The average percentage of performance difference of both algorithms is 8434 percent for the three datasets derived where D3 has the highest percentage of performance difference, and D2 resulted in the lowest difference in average processing time of both algorithms. So, the dynamic itemset counting implementation has an advantage on large and less complex data sets but gradually loses it on smaller and more complex data sets.

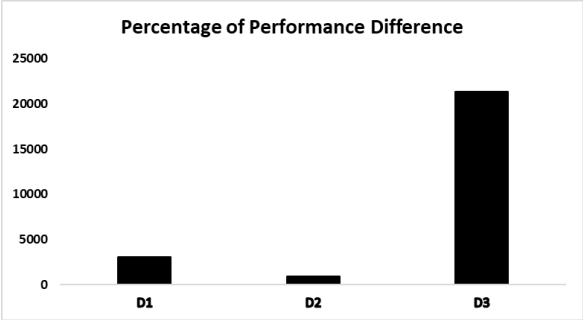

Figure 1. The bar graph shows the percentage of performance difference in each dataset.

Additionally, the accuracy of both implementations in all three datasets were the same, as well. In D1 (Groceries) both algorithms achieved 84.89%, 89.29% accuracy in D2 (Mall Customers), and 87.73% accuracy in D3 (Spambase).

Table 3. The table shows the summary of the accuracy of transaction reduction and dynamic itemset counting on three datasets.

Data Set	Transaction Reduction (%)	Dynamic Itemset Counting (%)
D1	84.89	84.89
D2	89.29	89.29
D3	87.73	87.73

As both algorithms have a time complexity of O (n^2), this means that the efficiency of both algorithms is the same. Thus, the processing times of both algorithms should be close. However, the transaction reduction implementation (TRI) had an average processing time of 640.5 seconds on all three datasets, while the dynamic itemset counting (DIC) had an average processing time of 13.5 seconds. This means that the DIC and TRI have an average percentage of performance difference (PPD) of 8434% across all three datasets.

The performance of the two algorithms had some discrepancies in varying dataset sizes and complexities. In D1 (Groceries), there was a 3051% percentage of performance difference (PPD), 928% PPD in D2 (Mall Customers), and 21324% PPD in D3 (Spambase). This is supported by the study of Brin et al. [9], which stated that DIC's weakness is that it does not do well with homogenous data; however, it improves when the order of transaction is randomized. percentage of performance difference.

As both algorithms have the same accuracy in determining the frequent itemset, this means that both algorithms have the same reliability level in the association rules they output. However, the classical Apriori has been tested to have an accuracy of between 98% [18] to 99% [19]. The decrease in accuracy can be attributed to the support value set by

the researchers, 5, which has been stated by Tank [14] to be one of the factors that affect the accuracy of the Apriori algorithm and its implementations.

Even though both algorithms have the same time complexity of O (n^2), their processing speeds have a large gap. This may be due to the fact that the Big-O notation is only used to weigh the computing cost of an algorithm in the worst-case scenario. Studies such as Delos Arcos & Hernandez [2] have opted to use an algebraic equation approach in determining the time complexity equations of algorithms. This is done so that every factor such as the number of loops, argument, and equation will be taken into consideration in the time complexity. The Big-O notation is only a good option in algorithms which have a small number of loops, as the Big-O notation does not take into account the number of loops, arguments, and equations. It only takes into account the value of the highest level loop of the entire algorithm. Other formats of describing an algorithm's time complexity may be used as the Big-O notation only evaluates algorithms in the worst-case scenario.

As DIC had a shorter average running time than TRI in all datasets, this means that DIC is more efficient in processing data. DIC had an advantage, especially in D3 (Spambase dataset). This may be because its transaction reduction implementation (TRI) tends to have more steps in scanning and data generation, which increases its processing time. This is supported by the study of Zymbler [4], which stated that DIC has good potential in parallelization. This means that on setups with multi-core CPUs or GPUs, DIC can take advantage of the extra cores to boost its performance. Additionally, the additional scans that TRI does when processing transactions increases its processing times [10]. Setups with multi-core CPUs or GPUs may want to use the DIC in processing data as it will significantly make processing faster. However, a different integrated development environment (IDE) can also be used other than PyCharm community version 2020.2.1 as PyCharm does not fully utilize the computer's available computing resources.

The biggest factor which affects the processing speed of DIC lies in the average length of transactions in the dataset, followed by its size or the number of transactions, and then the number of distinct items in the dataset. Thus, DIC thrives in datasets which have short transaction lengths and are more randomized such as the D3 (Spambase). This is supported by the study of Brin et al. [9], which stated that DIC's weakness is that it does not do well with homogenous data; however, it improves when the order of transaction is randomized. Additional variations in dataset sizes and complexities should be tested, especially in large datasets with lengthy and homogenous transactions, to determine if the advantage of DIC in randomized data will outweigh its disadvantage in large datasets with lengthy transactions.

Limitations. The limitation of this research study is the time complexity format used as the Big-O notation only evaluates algorithms in the worst case scenario. Another is the IDE (Integrated Development Environment) used, which is PyCharm. PyCharm does not fully utilize all the computing resources available to make processing faster. There is also a lack of variation in the datasets used. Lastly, the method of measuring the accuracy of the algorithm as the highest confidence value is not a good representation of the accuracy of the algorithms.

Conclusion. - The results showed that the dynamic itemset counting implementation was able to determine the frequent itemsets with the same accuracy, but faster than the transaction reduction implementation.

Recommendations. - It is recommended by the researchers to use algebraic equations over the Big-O notation to express the time complexity of the algorithms as it would be a more comprehensive representation of the algorithm. Additionally, the Integrated Development Environment (IDE) to be used will be different, as PyCharm Community Edition 2020.2.1 does not utilize all the available resources, such as the CPU and GPU. Another method to measure the accuracy of the algorithm that may be used as the highest confidence value of the output, or rules, is not a good representation of the accuracy of the algorithm. Other variations in dataset sizes and complexities may also be used to identify if an algorithm has an undiscovered pattern in processing data. Lastly, researchers may also opt to compare other commonly used implementations of the Apriori algorithm to determine which implementation is the best for different dataset sizes and complexities.

Acknowledgment. - The researchers would like to extend their gratitude to Dr. Rosslin Robles, for confirming the authenticity of the algorithms and datasets used in this study. The same appreciation is extended to Sir Joevic Pecate for verifying the validity of the determined time complexities of the algorithms.

References

[1] Moschovakis YN. 2001. What Is an Algorithm?. Berlin (DE). Springer-Verlag Berlin Heidelberg. doi: 10.1007/978-3-642-56478-9_46

[2] Delos Arcos JR, Hernandez AA. 2019. Efficient Apriori algorithm using enhanced Transaction Reduction approach. 2019 IEEE [Internet]. [cited 2020 Oct 2]. doi: 10.1109/TSSA48701.2019.8985482.

[3] Al-Maolegi M, Arkok B. 2014. An improved Apriori Algorithm for association rules. IJNLC. [Internet]. 2018. [cited 2020 Oct 24]; 3(1). doi: 10.5121/ijnlc.2014.3103.

[4] Zymbler M. 2017. Accelerating Dynamic Itemset Counting on Intel Many-core System. 2017 MIPRO [Internet]. [cited 2020 Oct 2]. doi: 10.23919/MIPRO.2017.7973631.

[5] Härting R-C, Sprengel A. 2019. Cost-benefit considerations for Data Analytics - An SME-Oriented Framework enhanced by a Management Perspective and the Process of Idea Generation. Procedia Computer Science. 159:1537–1546. doi: 10.1016/j.procs.2019.09.324.

[6] Silva J, Varela N, Borrero López LA, Rojas Millán RH. 2019. Association Rules Extraction for Customer Segmentation in the SMEs Sector Using the Apriori Algorithm. Procedia

Computer Science. 151:1207–1212. doi: 10.1016/j.procs.2019.04.173.

[7] Castro EPS, Maia TD, Pereira MR, Esmin AAA, Pereira DA. 2004. Review and comparison of Apriori algorithm implementations on MapReduce and Spark. Knowl. Eng. Rev. 0(0): 1-24. doi: 10.1017/S0269888918000127.

[8] Mustakim, Herianda DM, Ilham A, Daeng GS A, Laumal FE, Kurniasih N, Iskandar A, Manulangga G, Indra Iswara IBAI, Rahim R. 2018. Market Basket Analysis Using Apriori and FP-Growth for Analysis Consumer Expenditure Patterns at Berkah Mart in Pekanbaru Riau. J. Phys.: Conf. Ser. 1114 012131. doi: 10.1088/1742-6596/1114/1/012131.

[9] Brin S, Motwani R, Ullman JD, Tsur S. 1997. Dynamic Itemset Counting and implication rules for market basket data. SIGMOD Rec. 26(2):255-264. doi:10.1145/253262.253325.

[10] Singh J, Ram H, Sodhi JS. 2013. Improving efficiency of Apriori algorithm using transaction reduction. Int J Sci Res Pub. 3(1): 1-4. Available from: https://www.ijsrp.org/research-paper-1301/ijsrp-p1397.pdf

[11] Zhong R. Wang H. 2011. Research of Commonly Used Association Rules Mining Algorithm in Data Mining. 2011. International Conference on Internet Computing and Information Services. doi: 10.1109/ICICIS.2011.63.

[12] Roughgarden T. 2020. Beyond the Worst-Case Analysis of Algorithms. Cambridge: Cambridge University Press. p.1-24. doi: 10.1017/9781108637435.002.

[13] Wang L. 2020. Analysis of Factors Affecting Computer Data Processing Speed. J. Phys.: Conf. Ser. 1648 022136. doi: 10.1088/1742-6596/1648/2/022136.

[14] Tank D. 2014. Improved Apriori algorithm for mining association rules. IJITCS. [Internet]. [cited 2020 Oct 24]. Available from: 10.5815/ijitcs.2014.07.03.

[15] Saabith S, Sundararajan E, Abu Bakar A. 2018. A parallel Apriori-transaction reduction algorithm Using Hadoop-Mapreduce in cloud. AJCSIT. [cited 2020 Nov 7]; 1(1):1-24. doi: 10.9734/AJRCOS/2018/41045.

[16] Vaz R, Shah V, Sawhney A, Deolekar R. 2017. Automated big-O analysis of algorithms. 2017 International Conference on Nascent Technologies in Engineering (ICNTE). doi: 10.1109/icnte.2017.7947882.

[17] Althnian A, AlSaeed D, Al-Baity H, Samha A, Dris AB, Alzakari N, Abou Elwafa A, Kurdi H. 2021. Impact of dataset size on classification performance: an empirical evaluation in the medical domain. Appl Sci. 2021(11): 796. doi: 10.3390/app11020796.

[18] Funcion D. 2019. Apriori algorithm application on the prevalence of computer malware. Indian J Sci Technol. [cited 2022 May 22]; 12(17). doi: 10.17485/ijst/2019/v12i17/143328.

[19] Abidin A, Othman M, Hassan A, Murdianingsih Y, Suryadi U, Muslin Z. 2021. Verifying waste disposal practice problems of rural areas in Indonesia using the Apriori algorithm. 2021 ICIC. [cited 2022 May 22]; 1-7. doi: 10.1109/ICIC54025.2021.9632987.