Using Data Mining Techniques to Diagnosis of the Covid-19 Effects on the Hospital Readmission

Yahya Zakur[1*], Seyed Bagher Mirashrafi[2] *and Laith* Flaih[3]

[1]University of Mazandaran, Faculty of Mathematical Sciences, 4741613534 Babolsar, Iran
[2]University of Mazandaran, Faculty of Mathematical Sciences, 4741613534 Babolsar, Iran
[3]Cihan University-Erbil, Department of Computer Sciences, 44001 Erbil, Iraq

Abstract. The COVID-19 pandemic led to a substantial increase in the volume, diversity, and output pace of healthcare data. Countries depended on traditional methods to monitor diseases and public health to manage the epidemic, while advanced technology such as artificial intelligence and computation enabled efficient data processing. That datasets are usually enormous, growing exponentially, and comprise a collection of complicated item sets. To extract big, complicated itemsets, robust, straightforward, and computationally efficient techniques are crucial. Based on concepts from computer science, machine learning, and data mining, the Apriori method is a viable approach for supporting the values of database items in this study. There are two distinct implementation methods for Apiori: low confidence and support (Apiori algorithm) and the Apriori property algorithm. In conclusion, the performance of the Apriori property algorithm was superior to that of the traditional Apriori algorithm.

1 Introduction

Patients develop hospital infections, sometimes known as hospital-acquired infections, upon admission. In developed nations, approximately 5% to 10% of patients contract nosocomial infections, whereas the frequency in developing nations approaches 25%. Infections contracted in hospitals can lead to unanticipated morbidity and mortality, in addition to increased healthcare expenses [1]. Also as an illustration, hospital-acquired infections affect millions of patients in the United States every year, resulting in the expenditure of billions of dollars, and are responsible for around 50% of all severe hospital problems [2]. If infected people are not swiftly discovered and treated, other patients or healthcare personnel may become sick. As a consequence of this, proactive surveillance at the hospital and sub-hospital levels are essential to detect outbreaks and growing resistance at an earlier stage. Traditionally, nosocomial infection surveillance has relied on in-person visits to individual wards, examinations of patient charts, and written reports of microbiologic data. In most cases, analysis is performed by either compiling data that has been handwritten or entering data into a database on a computer. Traditional methods of infection management take a long time, require a lot of labor, and are ineffective, which makes them difficult to use in quantitative investigations and when dealing with the increasing complexity of antibiotic resistance. The potential usefulness of computer-based monitoring has received a lot of attention in recent years, and numerous recent publications have detailed useful computer applications for disease prevention and control [3]. On the other hand, extensive hospital data

* Corresponding author: yahyazakur92@gmail.com

analysis requires significant time and resources. Consequently, these data are underutilized, and the patterns they contain remain unidentified [4]. Also, it is challenging to locate a significant number of workable rules because the dataset is so huge. Consequently, efficient algorithms are necessary to explore the search space and only investigate the significant and interesting frequencies of all criteria for data trend identification. Thus, approaches such as machine learning and data mining must be utilized. With hospital readmission data mining being a field of applied artificial intelligence that permits the extraction of valuable knowledge from vast amounts of data, it is possible to extract useful information from vast amounts of data. Consequently, data mining is becoming a greater concern for this reason [5]. A further point to consider is that an increase in readmission research is being driven by the increased interest in hospital readmission rates. For example, determining the variables or predictors of risk that led to readmission and predicting the risks of readmission through the use of statistics, machine learning, and data mining [6]. The Apriori algorithm, which was developed by [7]. In this study will be using a comparative research utilizing two different approaches. The first approach is the traditional one. As a second technique of comparison, we will also be employing Apriori, which is a method that is termed Apriori property which was developed by [8].

2 Literature Review of Hospital Readmission for Covid 19 Pandemic

2019 marked the commencement of the COVID-19 pandemic, which has been responsible for the deaths of millions of people all across the globe [9]. During this time, the impact of the ongoing economic crisis was noted severely in the field of healthcare. The purpose of this study was to assess the utilization of medical specialties one to twelve months and thirteen to twenty-four months after hospital release in COVID-19 survivors who were either critically ill or not critically ill. Both the COVID-19 survivors who were readmitted for follow-up care after being released from the hospital (89 individuals) and the COVID-19 survivors who were not readmitted for follow-up care after being released from the hospital (205 people) were followed up on in this retrospective study. An analysis was done on the data about hospital readmission after survival. A staggering 98% of patients made it through the ordeal alive. Even though readmissions to the hospital were 34%. Some of the patients continued to struggle with issues related to their neurological health, respiratory health, chronic weariness, and cardiovascular health [10]. Because there were fewer resources available during the crisis, the demand for medical care rose to a higher priority. covid19. One of the most important leading indicators of health quality improvement is the quality of service that is being improved in the health sector [11]. To effectively limit hospital infections, it is required to employ data mining and machine learning-based approaches and algorithms. Additionally, it is necessary to identify the factors that contribute to a spike in hospital readmission instances [12].

3 The Methodology

The process of "Association Rule Mining is a type of data mining" that is based on a set of rules and is used to unearth intriguing connections between various items in enormous datasets. This technique is used to find interesting correlations between items in massive datasets. The approach begins with item sets that are often observed to discover the rules. These rules are offered in the form of sentences that follow the format "if → then" and they offer information regarding the likelihood of item linkage. In market-basket analysis, it is

helpful to have the knowledge that if a consumer purchases item (X), then that consumer is likely to purchase item (Y). It was the very first time that fact had been brought up [13].

3.1 Apriori Algorithm

According to [14], the Apriori is a technique for "mining association rules" that is capable of producing accurate results while using minimal time and resources. The Apriori methodology enables the application of prior knowledge of characteristics of itemsets that occur frequently. A level-wise survey is an iterative method that Apriori employs. This approach uses k-itemsets to explore "(k+1)-itemsets". After that, the set of frequent "(1-itemsets)" is found by searching the database, increasing the count of each item, and selecting the items that meet both the "min- sup and min_ conf conditions". This completes the process. The first thing that needs to be done is to search through the database while concurrently increasing the count for each item to find the collection of unique one-item sets. And choose those things that meet both the "min- sup" criteria and the "min_ conf" qualifiers. "The set resulting is L_1. Next, L_1 is used for finding L_2, the set of frequent (2-itemsets), which is used for finding L_3, and so on until no more frequent k-itemsets were also found. Each (L_k) requires a full database scan to be generated" [15], see "**FIGURE 1,**".

1- $L_1 = \{large\ 1\text{-}itemsets\}$;
2- for $\{k = 2; L_{k-1} \neq \emptyset; k{+}{+}\}$ do begin
3- $C_k = apriori\text{-}gen(L_k\text{-}1)$; // new candidates
4- For all transactions $t \in D$ do begin
5- $C_t = subset\ (C_k, t)$; // Candidate Contend in t
6- for all candidates $c \in C_t$ do
7- c.count++;
8- finish
9- $L_k = \{C \in C_k\ 1\ c.count \geq minsup\}$
10- end
11- $Ans. = \cup_k L_k$;

Fig. 1. The "Apriori algorithm construction".

The first phase of the algorithm is just a straightforward count of the number of occurrences of each item, therefore this is the only thing that is done in this phase (l-itemsets). The following pass, which will be referred to as pass k, is divided into two stages. The first thing that happens is that the enormous itemsets L_{k-i} that were produced in the previous pass (k-1) are put to use to generate the candidate itemsets C_k, by making use of the Apriori-gen function. Following this step, the database is reviewed, and a tally of the number of supports in favour of each candidate in C_k is performed. Efficient computation needs to have a method that accurately identifies "the candidates in C_k: that was involved in a specific transaction t. L_k": A collection of huge k-items, specifically those with min- sup. And each member of this set has two parts: Itemset and support count." C_k: is a set of candidates (k-itemsets)". Each individual in this collection possesses two fields, which are as follows: (itemset & sup. count). Apriori Candidate Generation as its input, "the apriori-gen function accepts L_{k-1}, which is the collection of all huge "(k - 1)-itemsets". It gives you a superset of the set of all large "(k - 1)-itemsets" as its return value. The function works as follows [16], in the join step, join L_{k-1} with L_{k-1}: Insert into C_k. Select P.item $_1$ = q.item $_2$ …., p.item $_{K-1}$,q.item $_{K-1}$ From the L_{k-1} p, L_{k-1} q ". Where the "p.item $_1$ = q.item $_1$ …. p.item $_{K-2}$ = q.item $_{K-2}$, p.item $_{K-2}$ less than q.item $_{K-2}$". Also, in the step of prune, "all itemsets c ∈ C_k such that some (k-1)" subset of c is not in the L $_{k-1}$: "For all the itemsets c ∈ C_k do. For all the (k-l)-subsets s of c do". "If (s ∉ Lk -1) then delete c from C_k. Example: Let L_3 : {{1 2 3},{1 2 4}, {1 3 4}, {1 3 5}, {2 3 4}}". After the step of join, "C_4 will be: {{1 2 3 4}, {1 3 4 5}}. Then, the step of pruning will delete the itemset {1 3 4 5}" because it is not existing in L_3. "Then be left with

the set {1 2 3 4} in C_4 only. Compare this candidate generation with the one used in the AIS [17], and SETM" [18], algorithms. In pass k of these algorithms, a database transaction t is read and it is specified which of the huge itemsets in L_{k-i} are exist in t. Each one of these enormous item sets I is then augmented with all of those enormous terms that are found in t and come later in the lexicographic ordering than any of the terms found in 1. Referring to the previous example, consider the transaction {1 2 3 4 5}. In the fourth pass, "AIS" and "SETM" will be creating two candidates, "{1 2 3 4} and {1 2 3 5}, "by extending the massive itemset {1 2 3}". In the same approach, the other huge item sets that are found in L_3 will be expanded upon in order to create an extra three prospective item sets. This will bring the total number of candidates up for consideration in the fourth pass to five. "The apriori, on the other hand, only generates and counts a single itemset, written as "{1 2 3 4}", because it includes an a priori declaring that the other sets cannot theoretically have the minimum support. This implies that only the "{1 2 3 4}" is generated and counted by the apriori".

3.2 Apriori Property Algorithm

According to [19], "Apriori property" means: any subset of a frequent itemsets must be frequent. The conventional "apriori algorithm uses prior information the Apriori property" of frequently occurring k-itemsets to reduce the search space in preparation for the mining of "(k+1)-itemsets" [20]. It starts by finding the frequent "(1-itemsets L_1), L_1 used for detecting L_2, L_2 " used for discovering L_3, and so on. "The creation of L_k is a (two-step)" process comprising a join and a pruning step. "The joining step is detecting frequent (k-itemsets) L_k from the frequent (k-1)-itemsets (L_{k-1}) by connecting (L_{k-1})" to itself. "This results in the (k-itemsets) to which the "apriori property is applied, to generate (k-itemsets) C_k, which may or may not be frequent. During the pruning phase, the transactions database is searched to determine which of the candidate item sets are utilized most frequently". The results of this investigation are then added to the set (L_k). "The joining step joins itemsets "l_1, l_2, … l_{k-1} of L_{k-1} to each other, where two itemsets l_I, l_J of L_{k-1} are joinable if their first (k-2 items)" are prevalent (i.e., "($l_I[1] = l_J[1]$)^ ($l_I[2] = l_J[2]$)^ "…" ^ ($l_I[k-3] = l_J[k-3]$)^ ($l_I[k-2] = l_J[k-2]$)^ ($l_I[k-1] < l_J[k-1]$)", where "$l_i[j]$ is the j^{th} item in itemset l_i)". For showing the working of the classical Apriori algorithm, let the database (D), "of transactions, be exemplified in "**TABLE 1,**" below, and the minimum support count is 3".

Table 1. The transactions in apriori property.

TID	Items
t1	"a, b, c, e"
t2	"b,c"
t3	"a, b, d"
t4	"a, c"
t5	"b, c"
t6	"a,b,c"
t7	"a, b, c, e"
t8	"a,b,e"

The steps required for making the frequent itemsets are shown below, see "**TABLE 2,**" "Check database D to count each candidate 1-itemset (C_1). Comparing (C_1) itemsets with min- support, and generating the (L_1). Check D for getting candidate (1-itemset) counts".

Table 2. The frequent items.

C_1 interest	Support

{a}	6
{b}	7
{c}	6
{e}	1
{d}	3

Compare the C_1 support count with the minimum support, as in "**TABLE 3,**":

Table 3. The Comparaisons of supports of C_1.

C_1 interest	**Support**
{a}	6
{b}	7
{c}	6
{e}	1

Joining L_1 with itself and using the "Apriori property for generating the candidate (2-itemsets) C_2, (C_1 interest: "{a, b}", "{a, c}", "{a, e}", "{b, c}", "{a, e}", "{c, e}"). Scanning the database D to count of each (C_2)". Comparing (C_2) itemsets with the min-sup. and generate L_2. Joining (L_1, L_1) then apply Apriori [21].
Scanning [22], D to get candidate as in "**TABLE 4,**":

Table 4. Scanning of D.

C_2 Itemset	**Support**
"{a,b}"	"5"
"{a,c}"	"4"
"{a,e}"	"3"
"{b,c}"	"5"
"{b,e}"	"3"
"{c,e}"	"2"

Assess the C_2 support count with the lowest support, as in "**TABLE 5,**":

Table 5. The comparaison step of C_2.

L_2 Itemset	**Support**
"{a,b}"	"5"
"{a,c}"	"4"
"{a,e}"	"3"

“{b,c}”	"5"
“{b,e}”	"3"

Joining L_2 with itself and using the “Apriori property for generating the candidate (2-itemsets) C_3, (C1 interest:{a,b,c},{a,b,e}). Scanning the database (*D*) to count of each C_3”. Comparing C_3 itemsets with the min-sup. and generating the L_3. Joining (L_2 L_2) and applying Apriori. Scan the (D) for getting the candidate (3-itemsets) counts, see “**TABLE 6,**”.

Table 6. The scanning D.

L_2 Itemset	**Support**
“{a,b,c}”	"3"
“{a,b,c}”	"3"

Comparing the amount of support for (C_3) with the required amount, as shown in “**TABLE 7,**”.

Table 7. Comparing C_3 support with min-support.

L_3 Itemset	**Support**
“{a,b,c}”	"3"
“{a,b,c}”	"3"

“Determining all of the (2-itemset) subsequences of C_3”.
“t_1 (a, b, c): {a b} {a c} {b c}”
“t_1 (a, b, e): {a b} {a e} {b e}”
“t_1 (a, c, e): {a c} {a e} {c e}”
“t_1 (b, c, e): {b c} {b e} {c e}”
“t_3 (a, b, d): {a b} {a d} {b d}”
“t_6 (a, b, c): {a b} {a c} {b c}”
“t_7 (a, b, c): {a b} {a c} {b c}”
“t_7 (a, b, e): {a b} {a e} {b e}”
“t_7 (a, c, e): {a c} {a e} {c e}”
“t_7 (b, c, e): {b c} {b e} {c e}”
“t_8 (a, b, e): {a b} {a e} {b e}”
The following procedures are repeated over and over in the algorithm: Finding the potential candidate "(k-itemsets), C_k. "Check the database for detecting the support count for every itemset in "C_k. "Comparing the support count for each itemset in "(C_k) "with minimum support count, [23], for detecting the frequent itemsets "L_k. "Joining L_k to itself and applying the “Apriori property” for detecting the “C_{k+1}. Let”, the join between the L_2 with itself, which generates an initial candidate “(3-itemsets) C_3”. Joining L_2 with itself, “where *L_2 = {{ab}, {ac}, {ae}, {b c}, {be}}”*. Initial candidate (3-itemsets) “*C_3: {{a b c},{a b e},{a c e},{b c e}}”*. Select the (2-itemset) subsequences:
“(a b c): {a b}, {a c}, {b c}”
“(a b e): {a b}, {a e}, {b e}”
“(a c e): {a c}, {a e}, {c e}”
“(b c e): {b c}, {b e}, {c e}”
It is possible to use the "Apriori property "to exclude from consideration any members of C3 that contain one or more of the two-item subsequences that are absent from the “L_2 (i.e., in L_{k-1})”, i.e., whose support counts are lower than the minimum support count. To accomplish

this, the support counts of these members are compared to the support count required to meet the minimum. This leaves candidate "(3-itemset) subsequences ""C_3 = {{a b c} {a b e}}". Check to see if the item sets in C3 are repeated by searching the database for all "(3- itemset)" subsequences and counting the number of apparitions of each C_3 itemsets. "{a b c}: t1, t6, t7 (3 occurrences) and {a b e}: t1, t7, t8 (three occurrences)". Both of the candidates "(3-itemsets)" listed above clearly "meet the minimum support count of (3) in this case and are thus included in L_3 ". [24], noted that it should be emphasized that the primary benefit of the Apriori attribute is that this results in a decrease in the total number of candidate item sets, which in turn results in a reduction in the overall complexity of the method in terms of both space and time. However, there is a possibility that the main memory cannot hold the large number of potential item sets that are formed.

4 The Results

In this section, will be using the dataset about hospital readmission to test the two models and making an analysis of those datasets for knowing the important indicators that effecting on hospital infection, the datasets we will utilize for the two model's empirical evaluation account for 10161 patients and by using the jupyter environment and doing coding for those two algorithms -Apriori and property Apriori algorithms- then the results will be as below in "**TABLE 8,**":

Table 8. The averages of support, confidence, lift, and performance of algorithms.

The Algorithms	Average of Support	Average of confidence	Average of Lift	The performance
Apriori	0.005905	0.589423	6.713469	Slow
Apriori property	0.16996	0.775362	6.137681	Fast

Here also some Figures represented the averages of various indicators in different algorithms see "**FIGURE 2,**", "**FIGURE 3,**" and **"Figure 4,":**

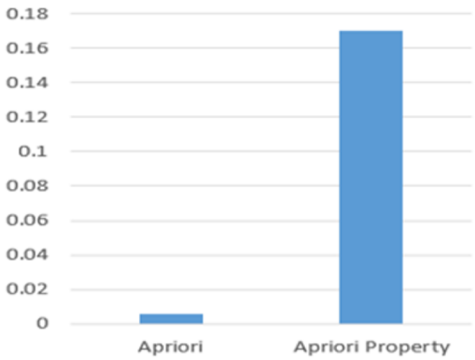

Fig. 2. The averages of support.

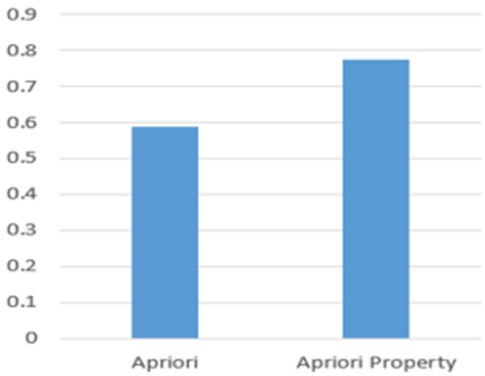

Fig. 3. The averages of confidence.

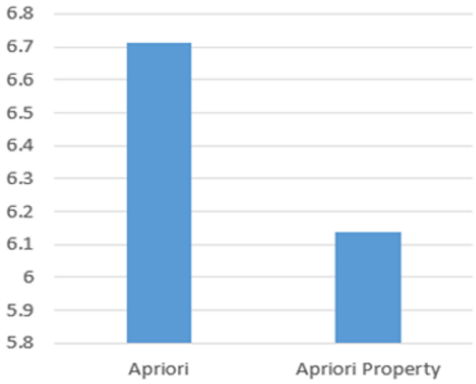

Fig. 4. The averages of lift between the algorithms.

Also, there are some of represented of important factors of hospital infection as below in "**FIGURE 5,**":

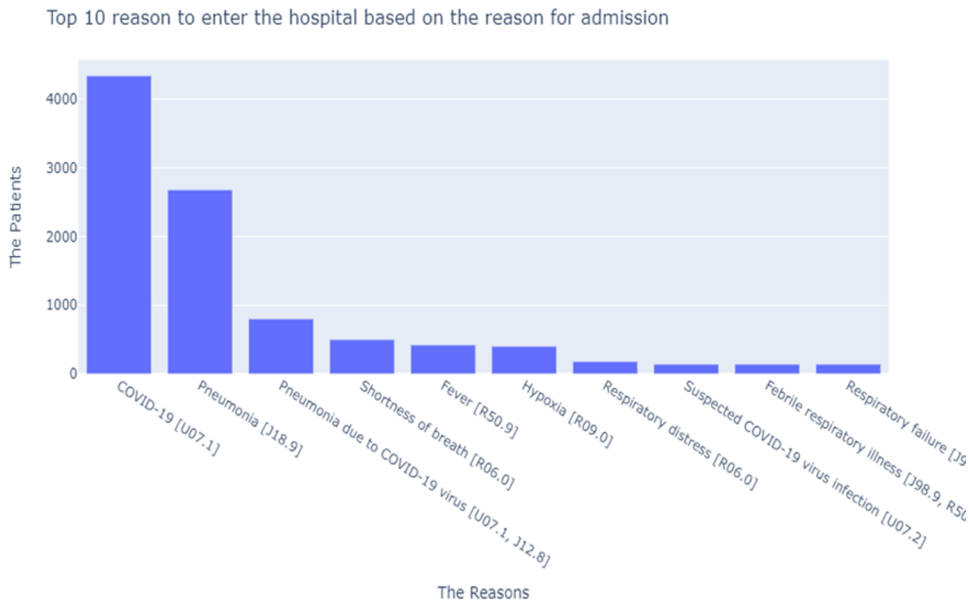

FIG. 5. The top reasons for hospital readmission.

5 Conclusion

In terms of confidence, support, and execution time, the Apriori property algorithm was superior to the Apriori algorithm, as disclosed in the findings section. The Apriori algorithm performed better than the Apriori property in the lift indicator. This makes it possible to utilize the property Apriori approach and apply it to an additional large healthcare system dataset. Infections in hospitals, for instance, might assist doctors in making appropriate decisions regarding patient situations and reducing infection rates in hospitals. Professionals and doctors can make the correct option regarding this crucial issue based on the discovery of hidden patterns that are intriguing. The factors of hospital infection revealed that covid-19 was the primary cause of the hospital's heavy workload; for instance, there were (4340) patients infected with covid-19, and they accounted for the majority of hospital admissions. Another crucial aspect was the kind of admission, with 9300 patients admitted to the ward and (860) admitted to the intensive care unit; these results could lead to a reduction in hospital infections in the future. Also, in future work, it will be beneficial to employ a larger dataset using the same algorithms or another's methods for comparison research or algorithm improvement.

References

1. F.M. Uzoka, M. Gift, K. Attai, B.A. Akinnuwesi, S.V. Mlay, P. Zeh, A. Kiirya, C. Muhumuza, J.N. Bukenya, S. Fashoto, D. Asuquo, *Tackling Occupational and Nosocomial Infection using Vitex-Medical Assistant Tool*. In 2022 IST-Africa Conference (IST-Africa). IEEE, pp. 1-9. (2022).
2. Bearman, G., Morgan, D. J., Murthy, R. K., & Hota, S. (Eds.). (2022). *Infection prevention: new perspectives and controversies*. Springer Nature.
3. I.D. Kocakoç, G.B. Türkölmez, *Using Data Mining Techniques for Designing Patient-Friendly Hospitals*. In Advances in Econometrics, Operational Research, Data Science and Actuarial Studies: Techniques and Theories. Cham: Springer International Publishing, pp. 321-343. (2022).
4. K. Batko, A. Ślęzak, *The use of Big Data Analytics in healthcare*. Journal of big Data, 9(1), pp.3. (2022).
5. T.T. Inan, M.B.R. Samia, I.T. Tulin, M.N. Islam, *A Decision Support Model to Predict ICU Readmission through Data Mining Approach*. In PACIS, pp. 218. (2018).
6. A. Ribeiro, F. Portela, M. Santos, A. Abelha, J. Machado, F. Rua, *Patients' admissions in intensive care units: a clustering overview*. Information, 8(1), pp.23. (2017).
7. E.T. Nekerow, D. Yakob, D. Teshome, *Data mining based medical intelligent system for chronic kidney disease diagnosis and treatment in the Oromo language*. International Journal of Intelligent Systems and Applications in Engineering, 10(2), pp.232-241. (2022).
8. R. Bhavani, G.S. Sadasivam, *A novel feature selection based on apriori property and correlation analysis for protein sequence classification using mapreduce*. International Journal of Data Mining and Bioinformatics, 17(3), pp.255-265. (2017).
9. S. Aslani, J. Jacob, *Utilisation of deep learning for COVID-19 diagnosis*. Clinical Radiology, 78(2), pp.150-157. (2023).
10. B. Musheyev, M.S. Boparai, R. Kimura, R. Janowicz, S. Pamlanye, W. Hou, T.Q. Duong, *Longitudinal medical subspecialty follow-up of critically and non-critically ill hospitalized COVID-19 survivors up to 24 months after discharge*. Internal and Emergency Medicine, 18(2), pp.477-486. (2023).

11. A. Raftarai, R.R. Mahounaki, M. Harouni, M. Karimi, S.K. Olghoran, *Predictive models of hospital readmission rate using the improved AdaBoost in COVID-19*. In Intelligent Computing Applications for COVID-19 (pp. 67-86). CRC Press. (2021).
12. B. Davazdahemami, D. Zolbanin, D. Delen, *An explanatory machine learning framework for studying pandemics: The case of COVID-19 emergency department readmissions*. Decision Support Systems, 161, pp.113730. (2022).
13. C.H. Yu, *Experimental implementation of quantum algorithm for association rules mining*. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 12(3), pp.676-684. (2022).
14. M.S. Krishnan, A.S. Nair, J. Sebastian, *Comparative analysis of apriori and ECLAT algorithm for frequent itemset data mining*. In Ubiquitous Intelligent Systems: Proceedings of ICUIS 2021, Springer Singapore, pp. 489-497. (2022).
15. R. Agrawal, C. Faloutsos, A. Swami, *Efficient similarity search in sequence databases*. In Foundations of Data Organization and Algorithms: 4th International Conference, FODO'93 Chicago, Illinois, USA. Springer Berlin Heidelberg, October 13–15, 1993 Proceedings 4, pp. 69-84. (1993).
16. C. Wang, X. Zheng, *Application of improved time series Apriori algorithm by frequent itemsets in association rule data mining based on temporal constraint*. Evolutionary Intelligence, 13(1), pp.39-49. (2020).
17. J.K. Chahal, *Finding Association Rules in Medical Datasets*. International Journal of Scientific Research in Science and Technology, 2(3), pp.167-170. (2019).
18. A. ÇELİK, *Using apriori data mining method in COVID-19 diagnosis*. Journal of Engineering Technology and Applied Sciences, 5(3), pp.121-131. (2020).
19. D. Dhinakaran, P.J. Prathap, *Protection of data privacy from vulnerability using two-fish technique with Apriori algorithm in data mining*. The Journal of Supercomputing, 78(16), pp.17559-17593. (2022).
20. S. Raj, D. Ramesh, M. Sreenu, K.K. Sethi, *EAFIM: efficient apriori-based frequent itemset mining algorithm on Spark for big transactional data*. Knowledge and Information Systems, 62, pp.3565-3583. (2020).
21. S. Anas, N. Rumui, A. Roy, P.H. Saputro, *Comparison of apriori algorithm and fp-growth in managing store transaction data*. International Journal of Computer and Information System (IJCIS), 3(4), pp.158-162. (2022).
22. N. Karimtabar, M.J.S. Fard, *Finding Frequent Items: A Novel Method for Improving the Apriori Algorithm.* Computer Science, 23(2). (2022).
23. M. Dehghani, A. Kamandi, M. Shabankhah, A. Moeini, *Toward a distinguishing approach for improving the apriori algorithm*. In 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE). IEEE, pp. 309-314). (2019).
24. M. Seyfi, Y. Xu, *H-DAC: discriminative associative classification in data streams*. Soft Computing, 27(2), pp.953-971. (2023).