Comparative Analysis of Different Versions of Association Rule Mining Algorithm on AWS-EC2

Ahamed Lebbe Sayeth Saabith[1(✉)], Elankovan Sundararajan[1], and Azuraliza Abu Bakar[2]

[1] Faculty of Information Science and Technology,
Centre for Software Technology and Management,
Universiti Kebangsaan Malaysia, UKM, 43600 Bangi, Selangor-DE, Malaysia
p68509@siswa.ukm.edu.my, elan@ukm.edu.my

[2] Faculty of Information Science and Technology,
Center for Artificial Intelligence and Technology,
Universiti Kebangsaan Malaysia, UKM, 43600 Bangi, Selangor-DE, Malaysia
azuraliza@ukm.edu.my

Abstract. Data mining is an essential step of knowledge discovery in databases (KDD) process by analyzing the huge amount of data from different perspectives and summarizing it into potentially valuable, valid, novel, interesting, and previously unknown information. Due to the importance of extracting knowledge from the massive data repositories, data mining is an essential components in various fields. Association rule mining (ARM), is one of the most important and well researched techniques of data mining, It aims to extract essential relationships, frequent patterns, associations among itemsets in the transaction databases or other data repositories. Many algorithm have been proposed to find the frequent itemset efficiently. In this research, we have chosen four well established frequent itemset mining methods which are Apriori, Apriori TID, Eclat, and FP-Growth to analyze their performance on cloud environment. Cloud computing is a new paradigm to analyze big data efficiently and cost effectively. In this study we analyzed the algorithms on Amazon web service (AWS) platform using elastic cloud computing (EC2) service. We thereafter compare the four algorithms based on their execution time by varying the minimum support (min_sup) values.

Keywords: KDD · ARM · Cloud computing · AWS-EC2 · Data mining

1 Introduction

Data mining is the process of extracting useful, potential, novel, understandable, concealed information from the databases which are huge, noisy, and ambiguous [1, 2]. Data mining plays a vital role in various application in the modern world such as market analysis, credit assessment, fraud detection, medical and pharma discovery, fault diagnosis in production system, insurance and healthcare, banking and finance, hazard forecasting, customer relationship management (CRM), and exploration of science [3–8].

H. Badioze Zaman et al. (Eds.): IVIC 2015, LNCS 9429, pp. 64–76, 2015.
DOI: 10.1007/978-3-319-25939-0_6

Association Rule Mining (ARM) or Frequent Itemset Mining (FIM) is one of the key areas of the data mining paradigm. Its main intention is to extract interesting relationships, patterns, associations among sets of items in the transaction database or other data repositories [9–12]. The most typical application of ARM is in market basket analysis which analyzes the purchasing behavior of customers by finding the frequent item purchased together. In addition to the many business application, it is also applicable to telecommunication networks, web log mining, market and risk management, inventory control, bio-informatics, medical diagnosis and text mining [8–13].

Recently data mining techniques, and tools are used in the cloud computing. Cloud computing is now a very powerful trend in all range of business and scientific field. It has become a great area of focus in data mining. Cloud computing offers many services to analyze, store, and manage the massive dataset such as deliver the software and hardware over the internet, data storage with efficient, reliable, and cost effective way [14, 15].

Dozens of algorithms have been proposed to find the frequent item set from transaction dataset. A very classical association rule mining algorithm is Apriori and several other algorithms have been developed based on this Apriori algorithm such as AprioriTID [12], Ecalt [16, 17], dEclat [17], FP-Growth [18], Relim [19], H-mine [19], FIN [20]. In this research, we have chosen four well established frequent itemset mining methods of Apriori, AprioriTID, Eclat, and FP-Growth performance for comparison within cloud computing environment.

The rest of this paper is organized as follows. Section 2 explains the related work in this research. Section 3 explains basic concepts of ARM and focuses on the selected ARM algorithm. Section 4 presents details about Amazon web service and Elastic cloud computing (EC2) service, Sect. 5 provides comparative analysis, whereas, Sect. 6 concludes the findings in this paper.

2 Related Work

Several study had been carried out to compare the performance among the various association rule mining algorithms [21–25]. Trivedi [26] analyzed the performance of several association rule mining algorithm and concluded that among the three algorithms compared, FP-Growth's performance is the best followed by Eclat while Apriori had the worst performance.

In a related development Garg and Kumar [27] comparatively studied the performance among Apriori, Eclat, and FP-Growth. They concluded that FP Growth is the best among the three algorithms and also scalable and the Apriori performs the worst. However, they used only one dataset for their experiment.

Similarly, Sinha and Ghosh [28] presented the comparison the performance of these same algorithms. They used only one dataset that is 'Pima' and they made several experiment by varying support count. They concluded that Eclat is better algorithm than Apriori and FP-Growth.

From the earlier studies, some researchers opined the FP-Growth algorithm is better than Apriori, AprioriTID, and Eclat based on their experimental research. On the other hand some researchers also concluded based on their research the Eclat is more efficient than Apriori, and FP-Growth.

The performance of the data mining algorithms depends on the size, generating number of candidates and frequent itemset, and density of the dataset. In this study, we choose small, medium, and dense dataset for evaluating the performance of the ARM algorithms.

3 Association Rules Mining (ARM) Algorithms

ARM is one of the key method of data mining techniques and it was introduced by Agrawal et al. in 1993. We elaborate on some generic concepts of association rules mining formally as follows.

Let $I = \left\{ i_1, i_2, \ldots, i_m \right\}$ be a set of m different literals, or items. For instance, goods such as bag, pen, and pencil for purchase in a shop are items.

X is a set of items such that $X \subseteq I$, a collection of zero or more items is called an itemset. If an itemset contains k items, it is called k-itemset. For example, a set of items for purchase from a super market is an itemset.

Let $D = \left\{ t_i, t_{i+1}, \ldots .., t_n \right\}$ is a set of transactions, where each transaction t has tid and $t - itemset\, t = (tid, t - itemset)$.

The itemset X in the transaction dataset D has a support, denoted as S, if $S\%$ transaction contains X, here we called $S = Supp\,(X)$.

$$Supp\,(X) = \frac{|\{t \in D; X \subseteq t\}|}{|D|}$$

An itemset X in a transaction database D is said to be large, or frequent itemset if its support is equal to, or greater than, the threshold minimal support (*minsup*) given by users. The *negation* of an itemset X is $\neg X$.

The support of $\neg X$ is $supp\,(\neg X) = 1 - supp(X)$.

An association rule is an implication in the form of $X \rightarrow Y, whereX, Y \subseteq IandX \cap Y = \phi$ [12].

The quality of association rule can be determined by measurements, support and confidence.

Support (S) determines how often a rule is applicable to a given dataset.

$$S\,(X \rightarrow Y) = \mathrm{Supp}\,(X \cup Y)\,/D$$

Confidence (C) determines how frequently items in Y appear in transactions that contains X,

$$C\,(X \rightarrow Y) = Supp\,(X \cup Y)\,/Supp\,(X)$$

The association rule mining task can be broken down into two sub tasks [9, 29–31].

I. Finding all of the frequent itemsets which have support above the user specified minimum support value All frequent itemset are then generated.

II. Generating all rules that have minimum confidence in the following simple way: For every frequent itemset X, and any $B \subset X$, $letA = X - B$. If the confidence of a rule *is* greater than, or equal to, the minimum confidence (or $supp\,(X)\,/supp\,(A) \geq minconf$), then it can be extracted as a valid rule.

The ARM performance typically depend on the first task. Usually, ARM generates vast number of association rules. Most of the time, it is difficult for users to understand and confirm a huge number of complex association rules. So, it is important to generate only "interesting" and "non-redundant" rules, or rules satisfying certain criteria such as easy to handle, control, understand, and increase the strength. Ever since, dozens of algorithms have been developed to find the frequent itemset and association rules in ARM. Some algorithms are more popular to find the frequent itemsets and association rules which are Apriori, Apriori-TID, FP-growth, Eclat, dEclat, Relim, H-mine, FIN, Charm, dCharm and so on. In this study, we have chosen four well established algorithm which are Apriori, Apriori-TID, FP-growth, and Eclat. We have evaluated the performance of the selected algorithms on cloud platform.

3.1 Apriori Algorithm

Apriori is classic and broadly used ARM algorithm. It uses an iterative approach called breath-first search to generate $(k - 1)$ itemsets from k item sets. The basic principle of this algorithm is that all nonempty subsets of a frequent itemset must be frequent [8, 11, 18].

The *Apriori-gen* function takes as argument L_{k-1}, the set of all large $(k - 1)$-itemsets. It returns a superset of the set of all large k-itemsets. There are two main steps in Apriori algorithm these are as follows:

- The prune step: remove the itemsets if support is less than min_sup which predefined by user value and abandon the itemset if its subset is not frequent. So, we can delete all itemsets $c \in C_k$ such that some $(k - 1)$-subset of c is not in L_{k-1}:
- The Join step: the candidates are produced by joining among the frequent item sets in level-wise way. The key drawback of this algorithm is the multiple dataset scan. So, we can join L_{k-1} with L_{k-1}.

3.2 AprioriTID

AprioriTid is a small variation on the Apriori algorithm and using Apriori-Gen function to produce candidates with some modification which does not use database for counting support after first pass, keeps a separate set C_k' which holds information: $<TID, \{X_k\}>$ where each X_k is a potentially large k-itemset in transaction TID, and if a transaction does not contain any large itemsets, it is removed from C_k' [12, 31].

3.3 FP-Growth

The FP-Growth Algorithm is an alternative way to find frequent itemsets without using candidate generations, thus improving performance. It uses a divide-and-conquer strategy.

The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the itemset association information.

In simple words, this algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. After this first step it divides the compressed database into a set of conditional databases, each one associated with one frequent pattern. Finally, each of such database is mined separately. Using this strategy, the FP-Growth reduces the search costs looking for short patterns recursively and then concatenating them in the long frequent patterns, offering good selectivity [32–34]. FP-growth is efficient and scalable for mining both long and short frequent patterns [35].

3.4 Eclat Algorithm

Eclat takes a depth-first search and adopts a vertical layout to represent databases, such that each item is represented by a set of transaction IDs (called a tidset) whose transactions contain the item. Tidset of an itemset is generated by intersecting tidsets of its items. Because of the depth-first search, it is difficult to utilize the downward closure property like in Apriori. However, using tidsets has an advantage that there is no need for counting support. The support of an itemset is the size of the tidset representing it. The main operation of Eclat is intersecting tidsets, thus the size of tidsets is one of the main factors affecting the running time and memory usage of Eclat. The bigger tidsets are, the more time and memory are needed [16, 17].

4 Cloud Platform

4.1 Amazon Web Service (AWS)

Amazon Web Service (AWS) provides a highly reliable, scalable, and low-cost infrastructure platform in the cloud. Whether indexing or analyzing large amount of business or scientific data sets, AWS offers set of big data tools and services and it is more suitable for any massive data analysis domain. There are several benefits accruable from the use of AWS including easy and securely host the user application using AWS management console, AWS services are more flexible to select the operating system, programming language, web application platform, database tools, and other useful services as user needs. It is a cost effective web service meaning that user can pay only for the computing resource usage per hourly basis and there are no long-term contracts and up-front commitment. AWS provides reliable, global secure, and scalable platform, and AWS tools can be auto scaling and elastic load balancing, so user can resize the application based on demand [36–38].

Furthermore, Amazon EC2 also provides pre-configured templates for user instances known as Amazon Machine Images (AMI). These AMI templates can include just an operating system like Windows or Linux, and can also include a wide range of components such as operating system, and pre-installed software packages. Amazon EC2 instances range start from small “micro” instances for small jobs to high performance “X-large” instances for like data warehousing [38].

5 Comparative Analysis

5.1 Dataset Details

We have chosen four different benchmark dataset which are related in frequent itemset mining and were downloaded from [39]. In specific chess, accident, and mushroom are real life dataset and t20i6D100 K which was synthetically generated by IBM generator. Table 1 describes more details of the dataset.

Table 1. Dataset details with number of transaction and their attributes

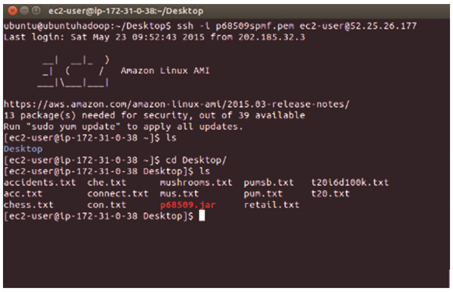
Dataset name	Total transaction	Total attributes
Chess	3196	36
Accident	340183	50
Mushroom	8124	22
t20i6D100 K	100000	26

5.2 SPMF

SPMF is an open-source data mining library for frequent pattern mining. It was developed under the GPL v3 license and written using java programming language. It has 93 data mining algorithms for sequential pattern mining, association rule mining, itemset mining, and sequential rule mining, and clustering. SPMF can be used as a standalone program with a simple user interface or from the command line [40].

5.3 AWS-EC2 Details

All experiments were performed on amazon web service cloud platform using EC2 instance type "m2-medium" that contains: Linux operating system, memory 4 GB, 2 core. Figure 1 illustrates the logon screen of EC2-m2-medium instance.



```
ec2-user@ip-172-31-0-38:~/Desktop
ubuntu@ubuntuhadoop:~/Desktop$ ssh -i p68509spmf.pem ec2-user@52.25.26.177
Last login: Sat May 23 09:52:43 2015 from 202.185.32.3

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2015.03-release-notes/
13 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-0-38 ~]$ ls
Desktop
[ec2-user@ip-172-31-0-38 ~]$ cd Desktop/
[ec2-user@ip-172-31-0-38 Desktop]$ ls
accidents.txt  che.txt      mushrooms.txt  pumsb.txt   t20i6d100k.txt
acc.txt        connect.txt  mus.txt        pum.txt     t20.txt
chess.txt      con.txt      p68509.jar     retail.txt
[ec2-user@ip-172-31-0-38 Desktop]$
```

Fig. 1. AWS-EC2 instance login screen

5.4 Results Comparison

Table 2 and Fig. 2 show the performance of the four chosen algorithms in Chess dataset with different min_sup. The results show that FP-Growth algorithm outperforms the other three algorithms. We were only able to find the results using AprioriTID algorithms until min_sup = 0.65 because of the memory constraints.

Table 2. Chess dataset comparison with execution time and frequent itemset count

Chess dataset		Total time (ms)			
Min_Sup	Frequent itemset count	Apriori	AprioriTID	FPGrowth	Eclat
0.95	77	238	258	284	133
0.9	622	381	602	292	246
0.85	2669	1171	2159	377	600
0.8	8227	2854	6288	639	1285
0.75	20993	6851	8871	802	2561
0.7	48731	15728	24234	978	5293
0.65	111239	36104	-	1211	11707
0.6	254944	97744	-	1152	23642
0.55	574998	291115	-	1843	50780
0.5	1272932	966186	-	3025	107914

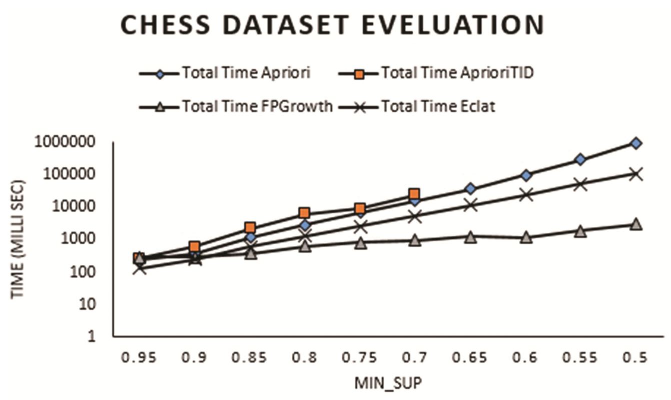


Fig. 2. Chess dataset result comparison with execution time

Table 3 and Fig. 3 show the performance of the four chosen algorithms in accidents dataset with different min_sup. The results show that FP-Growth algorithm outperforms the other three algorithms. Also, we were only able to find the results using AprioriTID algorithms until min_sup = 0.65 because of the memory constraints.

Table 3. Accidents dataset comparison with execution time and frequent itemset count

Accidents dataset		Total time (ms)			
Min_Sup	Frequent itemset count	Apriori	AprioriTID	FPGrowth	Eclat
0.95	15	479	1717	699	2285
0.9	31	554	2154	726	2424
0.85	71	756	2710	757	2553
0.8	145	1220	4578	758	2798
0.75	318	2215	9183	858	3340
0.7	553	3588	19154	876	4573
0.65	1102	6833	-	936	6065
0.6	2074	12241	-	1120	8704
0.55	3971	22513	-	1211	13483
0.5	7855	41243	-	1597	24153

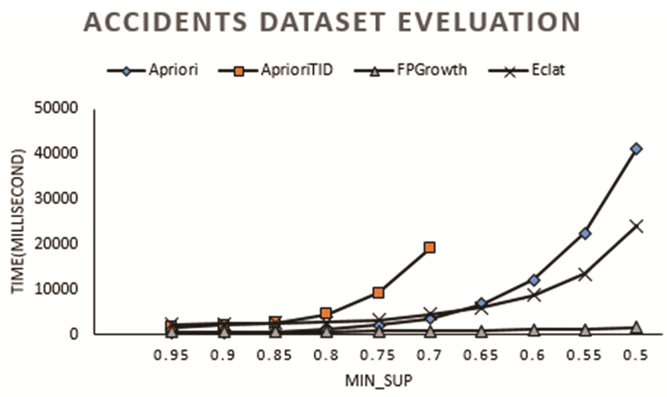


Fig. 3. Accidents dataset result comparison with execution time

Figure 4 and Table 4 show the performance of the four chosen algorithms in mushroom dataset with different min_sup. The result shows that Eclat algorithm outperforms the other three algorithms.

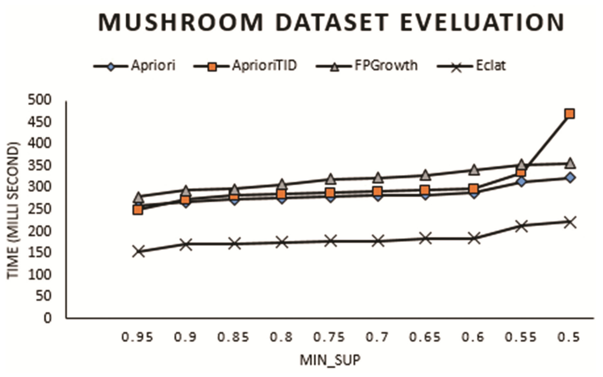


Fig. 4. Mushroom dataset result comparison with execution time

Table 4. Mushroom dataset comparison with execution time and frequent itemset count

Mushroom dataset		Total time (ms)			
Min_Sup	Frequent itemset count	Apriori	AprioriTID	FPGrowth	Eclat
0.95	7	259	250	280	154
0.9	11	268	273	295	171
0.85	15	273	283	298	173
0.8	17	276	285	309	175
0.75	25	279	289	321	178
0.7	31	282	292	324	179
0.65	39	284	295	329	185
0.6	51	289	297	342	185
0.55	115	314	335	354	213
0.5	163	323	468	356	222

Table 5 and Fig. 5 show the performance of the four chosen algorithms in t20i6D100 K dataset with different min_sup. The results show that Eclat algorithm outperforms the other three algorithms.

Table 5. t20i6D100 K dataset comparison with execution time and frequent itemset count

t20i6D100 K dataset		Total time (ms)			
Min_sup	Frequent itemset count	Apriori	AprioriTID	FPGrowth	Eclat
0.1	7	539	1442	688	2084
0.09	13	609	1668	712	2130
0.08	23	1038	1769	792	2180
0.07	34	1822	1809	891	2233
0.06	61	4160	2094	1237	2233
0.05	99	12315	2781	1452	2313
0.04	154	23703	4371	2359	2342
0.03	242	45393	7072	3981	2377
0.02	378	105189	11760	4453	2407
0.01	1523	261538	21301	6121	2520

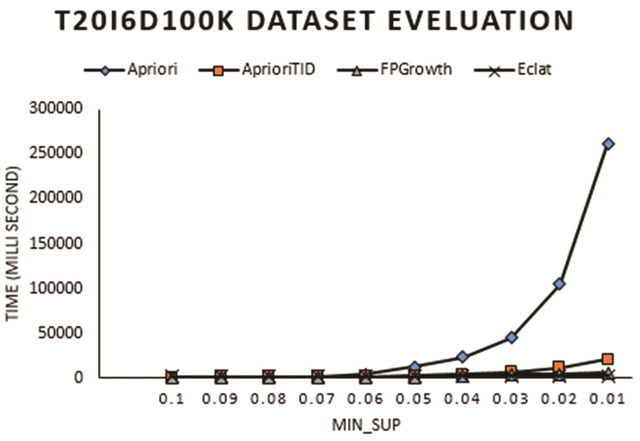


Fig. 5. t20i6D100 K dataset result comparison with execution time

6 Conclusion

In this work, four different association rule mining algorithms (Apriori, AprioriTID, FP-Growth, and Eclat) was implemented on cloud environment. We have chosen the cloud platform as the amazon web service platform and used EC2 service. We implemented four different benchmark dataset including chess, accidents, mushrooms and t20i6d100 K. We evaluated the performance of those algorithms based on their execution time by varying the min_sup values. From this study, we make the following observations and conclusion as follows:

– Cloud platform is much suitable for data mining process in the areas of efficiently, reliability, and cost effectiveness.
– During comparison Apriori requires more time to produce the frequent itemset when the min_sup values decreases. In contrast AprioriTID algorithm is not suitable for dense dataset such as chess and accidents. This is because those datasets are producing more frequent itemset when the min_sup value decreases and AprioriTID is not able to produce the results beyond particular min_sup values shown in Tables 3 and 4.
– Eclat algorithm is suitable for any dataset (small or medium or dense dataset) with compared Apriori, and AprioriTID.
– From this study the FP-Growth algorithm is more suitable for medium size and dense dataset. Tables 2 and 3 clearly show the experimental results. Tables 4 and 5 clearly express that the FP-Growth is not suitable for small size and simple dataset.
– Eclat and FP-Growth algorithms are more efficient algorithm than Apriori, and AprioriTID algorithms. Comparing these algorithms, FP-Growth is more suitable for dense and medium size dataset. It may therefore be concluded that Eclat is appropriate for medium size and less dense dataset.

Acknowledgment. We wish to thank Universiti Kebangsaan Malaysia (UKM) and Ministry of Higher Education Malaysia for supporting this work by research Grants (ERGS/1/2013/ICT07/UKM/02/3).

References

1. Tan, P.: Introduction to Data Mining, vol. 1. Pearson Addison Wesley, Boston (2007)
2. Hand, D.J.: Principles of Data Mining, vol. 30, no. 7. MIT press, Cambridge (2007)
3. Ngai, E.W.T., Xiu, L., Chau, D.C.K.: Application of data mining techniques in customer relationship management: a literature review and classification. Expert Syst. Appl. **36**(2 PART 2), 2592–2602 (2009)
4. Shaw, M.J.B.C., Subramaniam, C., Tan, G.W., Welge, M.E.: Knowledge management and data mining for marketing. Decis. Support Syst. **31**(1), 127–137 (2001)
5. Obenshain, M.K.: Application of data mining techniques to healthcare data. Infect. Control Hosp. Epidemiol. **25**(8), 690–695 (2004)
6. Antonie, M., Coman, A., Zaiane, O.R.: Application of data mining techniques for medical image classification. In: Proceedings of the Second International Workshop on Multimedia Data Mining (MDM/KDD 2001), pp. 94–101 (2001)
7. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.-N.: Web usage mining: discovery and applications of usage patterns from web data. ACM SIGKDD **1**(2), 12–23 (2000)
8. Han, J., Kamber, M.: Data Mining, Southeast Asia Edition: Concepts and Techniques. Morgan Kaufmann, Los Altos (2006)

9. Hipp, J., Güntzer, U., Nakhaeizadeh, G.: Algorithms for association rule mining - a general survey and comparison. ACM SIGKDD Explor. Newsl. **2**(1), 58–64 (2000)
10. Zhang, C., Zhang, S.: Association Rule Mining: Models and Algorithms, vol. 2307. Springer, Berlin (2002)
11. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. ACM SIGMOD Rec. **22**, 207–216 (1993)
12. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data bases, VLDB (1994)
13. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Los Altos (2005)
14. Ambulkar, B., Borkar, V.: Data mining in cloud computing. In: MPGI National Multi Conference, pp. 23–26 (2012)
15. Petre, R.S.: Data mining in cloud computing. Datab. Syst. J. **3**(3), 67–71 (2012)
16. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. **12**(3), 372–390 (2000)
17. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and data mining - KDD 2003, p. 326 (2003)
18. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. **8**(1), 53–87 (2004)
19. Borgelt, C.: Keeping things simple: finding frequent item sets by recursive elimination. In: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, pp. 66–70 (2005)
20. Deng, Z.-H., Lv, S.-L.S.: Fast mining frequent itemsets using nodesets. Expert Syst. Appl. **41**(10), 4505–4512 (2014)
21. Krishna, T.: Effectiveness of various FPM algorithms in data mining. ijcsit.org **02**(01), 01–05 (2014)
22. Patel Tushar, S., Mayur, P., Dhara, L., Jahnvi, K., Piyusha, D., Ashish, P., Reecha, P., Tushar, S.P., Mayur, P., Dhara, L.: An analytical study of various frequent itemset mining algorithms. Res. J. Comput. Inf. Technol. Sci. **1**(1), 2–5 (2013)
23. Pramod, S., Vyas, O.P.: Survey on frequent itemset mining algorithms. Int. J. Comput. Appl. **1**(5), 1–6 (2010)
24. Prithiviraj, P., Porkodi, R.: A comparative analysis of association rule mining algorithms in data mining: a study. Open J. Comput. Sci. Eng. Surv. **3**(1), 98–119 (2015)
25. Tiwari, M., Jha, M.B., Yadav, O.: Performance analysis of data mining algorithms in Weka. IOSR J. Comput. Eng. ISSN **6**, 661–2278 (2012)
26. Trivedi, M.M.: Review and analysis of various efficient frequent pattern algorithms. Int. J. Technol. Res. Eng. **2**(2), 139–143 (2014)
27. Garg, K., Kumar, D.: Comparing the performance of frequent pattern mining algorithms. Int. J. Comput. Appl. **69**(25), 21–28 (2013)
28. Sinha, G., Ghosh, S.M.: Identification of best algorithm in association rule mining based on performance. Int. J. Comput. Sci. Mob. Comput. **3**(11), 38–45 (2014)
29. Nichol, M.B., Knight, T.K., Dow, T., Wygant, G., Borok, G., Hauch, O., O'Connor, R.: Quality of anticoagulation monitoring in nonvalvular atrial fibrillation patients: comparison of anticoagulation clinic versus usual care. Ann. Pharmacother. **42**(1), 62–70 (2008)
30. Yu, L.C., Chan, C.L., Lin, C.C., Lin, I.C.: Mining association language patterns using a distributional semantic model for negative life event classification. J. Biomed. Inform. **44**(4), 509–518 (2011)

31. Zhao, Q., Bhowmick, S.S.: Association Rule Mining: a Survey. Nanyang Technological University, Singapore (2003)
32. Said, A.M., Dominic, P.D.D., Abdullah, A.B.: A comparative study of fp-growth variations. Int. J. Comput. Sci. Netw. Secur. **9**(5), 266–272 (2009)
33. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM SIGMOD Rec. **29**(2), 1–12 (2000)
34. Zaiane, O.R., El-Hajj, M., Lu, P.: Fast parallel association rule mining without candidacy generation. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 665–668 (2001)
35. Borgelt, C., Borgelt, C., Kruse, R., Kruse, R.: Induction of association rules: apriori implementation. In: 15th Conference on Computational Statistics Physica Verlag, Heidelberg, Germany 2002, vol. 1, pp. 1–6 (2002)
36. Amazon, A.W.S., Miller, F.P., Vandome, A.F., McBrewster, J.: Amazon web services, vol. 12, pp. 1–3 (November 2012). http://aws.Amaz.com/es/ec2/
37. Murty, J.: Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB. O'Reilly Media Inc, Sebastopol (2008)
38. Robinson, D.: Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services Enables You to Reach Business Goals Faster. Emereo Pty Ltd, Brisbane (2008)
39. Goethals, B.: Frequent itemset mining implementations repository (2003). http://fimi.ua.ac.be/
40. Fournier-Viger, P.: SPMF- an open-source data mining library (2003). http://www.philippe-fournier-viger.com/spmf/