ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.7

Data Mining on Parallel Database Systems

Karthikeyan .R[1], Dr.T.Geetha[2], Soundarya .R[3], Suguna .M[4], Sugitha .C[5]
Assistant Professor[1, 2], PG Scholar[3, 4, 5]
Dept of MCA
Gnanamani college of Technology, Namakkal, India

Abstract:
Association rule mining is the most popular technique in data mining. Mining association rules is a prototypical problem as the data are being generated and stored every day in corporate computer database systems. To manage this knowledge, rules have to be pruned and grouped, so that only reasonable numbers of rules have to be inspected and analyzed. In this paper we compare the standard association rule algorithms with the proposed Scaled Association Rules algorithm and AIREP algorithm. All these algorithms are compared according to the various factors like Type of dataset, support counting, rule generation, candidate generation, computational complexity and other factors .The conclusions drawn are based on the efficiency, performance, accuracy and scalability parameters of the algorithms.

Keywords: Association rule, Data Mining, Multidimensional dataset, Pruning, Frequent itemset. Introduction

I. INTRODUCTION

Mining association rules is particularly useful for discovering relationships among items from large databases [10]. A standard association rule is a rule of the form X→ Y which says that if X is true of an instance in a database, so is Y true of the same instance, with a certain level of significance as measured by two indicators, support and confidence. The goal of standard association rule mining is to output all rules whose support and confidence are respectively above some given support and coverage thresholds. These rules encapsulate the relational associations between selected attributes in the database, for instance, coke → potato chips: 0.02 support; 0.70 coverage denotes that in the database, 70% of the people who buy coke also buy potato chips, and these buyers constitute 2% of the database. This rule signifies a positive (directional) relationship between buyers of coke and potato chips [19]. The mining process of association rules can be divided into two steps.

1. Frequent Itemset Generation: generate all sets of items that have support greater than a certain threshold, called minsupport.

2. Association Rule Generation: from the frequent itemsets, generate all association rules that have confidence greater than a certain threshold called minconfidence. Apriori is a renowned algorithm for association rule mining primarily because of its effectiveness in knowledge discovery However, there are two bottlenecks in the Apriori algorithm. The purpose of the association rules is to find correlations between the different processes of any application. Knowing the associations between these processes, it helps to take decisions and to use the process methods effectively. The various association rule mining algorithms were used to different applications to determine interesting frequent patterns. One of the association rule mining algorithm such as Apriori algorithm used the property of support and confidence to generate frequent patterns. Another measure is Predictive Accuracy; it is an indicator of a rule's accuracy in future over unseen data. Confidence of a rule is the ratio of the correct predictions over all records for which a prediction is made but it is measured with respect to the database that is used for training. This confidence on the training data is only an estimate of the rule's accuracy in the future, and since the user searches the space of association rules to maximize the confidence, the estimate is optimistically biased (Scheffer 2001). Thus, the measure predictive accuracy is introduced. It gives for an association rule its probability of a correct prediction (Srikant 1999) with respect to the process underlying the database. The paper consists of 6 sections as follows. We introduce description of some works in the literature concerning the improvement of association rule algorithms in Section 2. Section 3 parameters on which the algorithms are compared. Section 4 gives the experimental study. The conclusion and future scope are presented in sections 5 and 6 respectively.

2. WHICH ALGORITHMS ARE COMPARED?

In this section we describe the software implementations of the association rule algorithms used in our experiments. The four algorithms evaluated were Apriori, FP-growth, Scaled association rule algorithm and AIREP algorithm. We provide references to articles describing the details of the algorithm when available and also specify the algorithms' parameter settings used in our experiments (if any).We started the experiments several months ago and published preliminary results to the authors of the algorithms. Several authors provided us with an updated version of their code to fix bugs and/or improve the performance. We reran our experiments with the new versions and noted below when updated versions

The reasons for comparing these algorithms are:

- Flexibility
- Popularity
- Applicability
- Types of dataset used

2.1 Aprori algorithm:

The apriori algorithm [1] is one of the earliest algorithms for mining association rules and has became the standard approach in this area. The search for association rules is guided by two parameters: support and confidence. Apriori returns an association rule if its support and confidence values are above user defined threshold values. The output is ordered by confidence. If several rules have the same confidence, then they are ordered by support. Thus apriori favors more confident rules and characterises these rules as more interesting. The apriori Mining process is composed of two major steps. The first one (generating frequent item sets) was discussed briefly in the last section. This step can be seen as support based pruning, because only item sets with at least minimum support were considered. The second step is the generation of rules out of the frequent item sets. In this step confidence based pruning is applied.Rule discovery is straightforward. For every frequent item set f and every non-empty subset s of f, apriori outputs a rule of the form s ⇒ (f − s) if and only if the confidence of that rule is above the user specified threshold. e task of discovering association rules consists in finding all the association rules having a minimum support minsup and a minimum condence minconf . The task of discovering association rules consists in finding all the association rules having a minimum support minsup and a minimum confidence minconf. Apriori is Christian Borgelt's implementation of the well-known Apriori association rule algorithm [1][2]. The source code in C for this implementation is available under the "GNU Lesser General Public License" from http://fuzzy.cs.unimagdeburg.de/~borgelt/. Apriori takes tran sactional data in the form of one row for each pair of transaction and item identifiers. It first generates frequent itemsets and then creates association rules from these itemsets. It can generate both association rules and frequent itemsets. Apriori supports many different configuration settings. In our experiments, we used the percentage of transactions that satisfy both the LHS and the RHS of a rule as the support. We also specified that Apriori should load the entire dataset into memory rather than making multiple database scans. The running Apriori using multiple database scans would be significantly slower. Apriori is the first algorithm to use Apriori-gen for candidate generation. As mentioned in the previous paragraph, Apriori-gen is separate from the counting step that determines the frequency of each current candidate. This means that each pass of Apriori consists of a call to Apriori-gen togenerate all candidates of a given size (size k in pass k) and a counting phase that determines the support for all these candidates. Each counting phase scans the entire database. Upon reading a transaction T in the counting phase of pass k, Apriori has to determine all the kcandidates supported by T and increment the support counters associated with thesecandidates. In order to perform this operation efficiently, Apriori stores candidate item-sets in a tree. The actual item-sets are stored in the leaves of the tree, and edges are labeled with items. To find the proper location for a candidate, starting from the root, traverse the edge with the first item in the set. Reaching an internal node, choose the edge labeled with the the next item in the set, until a leaf is reached. The path to locate set is marked with thickened arrows in the figure. Note that by virtue of ordering the items, each set has its unique place in the tree. The smallest items in a set that are used along the path to the leaf need not be stored. Inserting item-sets into the tree can cause a leaf node to overflow, in which case it is split and the tree grows. To count all candidates for transaction T, all leaves that could contain a candidate have to be searched, and to reach all these leaves, Apriori tries all possible combinations of the items in T as paths to a leaf. Once a leaf with a set of candidates is located in this fashion it remains to be checked which are actually supported by the transaction. As far as the implementation is concerned, this test for set inclusion can be optimized by storing the item-sets as bitmaps, one bit for each item. As observed in [1], these bitmaps can become quite large for many items (128 Bytes for 1000 items) and cause considerable overhead. Internal nodes are implemented as hash tables to allow fast selection of the next node. To reach the leaf for a set, start with the root and hash on the first item of the set. Reaching the next internal node, hash on the second item and so on until a leaf is found. Item-lists The major problem for Apriori (and for AIS as well) is that it always has to read the entire database in every pass, although many items and many transactions are no longer needed in later passes of the algorithm. In particular, the items that are not frequent and the transactions that contain less items than the current candidates are not necessary. Removing them would obviate the expensive effort to try to count item-sets that cannot possibly be candidates. Apriori does not include these optimizations, moreover they would be hard to add to Apriori (andAIS as well). The reason stems from the item-list data representation used by both algorithms. At before, transactions are stored as a sequence of sorted item-lists in this representation. While item- lists are the most common representation and the one that is usually assumed as input format, they make it difficult to remove unnecessary parts of the data. Let's assume we want to remove all items that are not part of any frequent set. Unfortunately, the knowledge of which items to keep and which to discard is only available and applicable after scanning the database to count the support for the candidates. Therefore, we can eliminate items only in the subsequent pass over the data, that is they have to be read once more, although this is not really necessary. As we will see later, the other two representations remove these items instantly, which leads to much smaller data sizes in later passes; unfortunately this is not the case for early passes, where the volume of intermediate data representations can exceed the original data size. The advantage of item-lists is therefore that the size of the data does not grow in the course of the algorithms.

2.2 FP Growth algorithm :

The FPGrowth method constructs FP-tree which is a highly compact form of transaction database. Thus both the size and the cost of computation of conditional pattern bases, which corresponds roughly to the compact form of projected transaction databases, are substantially reduced. Hence, FPGrowth mines frequent itemsets by (1) constructing highly compact FPtrees which share numerous "projected" transactions and hide (or carry) numerous frequent patterns, and (2) applying progressive pattern growth of frequent 1-itemsets which avoids the generation of any potential combinations of candidate itemsets implicitly or explicitly, whereas Apriori must generate more number of candidate itemsets for each projected database. Therefore, FPGrowth is more efficient and more scalable than Apriori, especially when the number of frequent itemsets becomes really large. FP-growth is an algorithm for generating frequent itemsets for association rules from Jiawei Han's research group at Simon Fraser University. It generates all frequent itemsets satisfying a given minimum support by growing a frequent pattern tree

structure that stores compressed information about the frequent patterns. In this way, FP-growth can avoid repeated database scans and also avoid the generation of a large number of candidate itemsets [4]. The FP tree algorithm addresses these issues and scans the data in a depth-first way. The data is only scanned twice. In the first scan, the frequent items (or 1-itemsets) are determined. The data items are then ordered based on their frequency and the infrequent items are removed. In the second scan, the data base is mapped onto a tree structure. The FP tree does never break a long pattern into smaller patterns the way the Apriori algorithm does. Long patterns can be directly retrieved from the FP tree. The FP tree also contains the full relevant information about the data base. It is compact, as all infrequent items are removed and the highly frequent items share nodes in the tree. The number of nodes is never less than the size of the data base measured in the sum of the sizes of the records but there is anecdotal evidence that compression rates can be over 100. FP-Growth: allows frequent itemset discovery without candidate itemset generation. Two step approach:

Step 1: Build a compact data structure called the FP-tree.I Built using 2 passes over the data-set.
Step 2: Extracts frequent itemsets directly from the FP-tree Traversal through FP-Tree

2.3 Scaled Association Rules algorithm
middle point of the separation region. This procedure was applied for creating intervals of values for every one of the attributes in order to generate the association rules.

Step 2: Candidate Generation
Given Lk -1, the set of all frequent -itemsets, the candidate generation procedure must return a superset of the set of all frequent k-itemsets. The k-means clustering helps in finding the appropriate and definite cluster with partitioning. This procedure has three parts:
1. Join Phase. Lk - 1 is joined with itself, the join condition being that the lexicographically ordered first k – 2 items are the same, and that the attributes of the last two items are different.

2. Subset Prune Phase. In this phase all itemsets from the join result which have some (k – 1)-subset that is not in Lk-1 are deleted.

3. Interest Prune Phase. If the user specifies an interest level and wants only itemsets whose support and confidence is greater than expected, the interest measure is used to prune the candidates further.
4.
Step 3: Counting Support of Candidates.
In the process of counting support of candidates when we make a pass, we read one record at a time and increment the support count of candidates supported by the record. Thus, given a set of candidate itemsets C and a record t, we need to find all itemsets in C that are supported by t. We partition candidates into groups such that candidates in each group have the same attributes and the same values for their categorical attributes.

Step 4: Generating Rules. We use the frequent itemsets to generate association rules. The general idea is that RTYZ and RT are frequent itemsets, then we can determine if the rule RT□ YZ holds by computing the ratio conf = support (RTYZ)/support (YZ) . If conf >= supconf, then the rule will have minimum support because RTYZ is frequent. The clusters are created with a weight for the output. This is a supervised way of producing the most suitable clusters for the prediction of the output variables, which appear in the consequent part of the rules generation.

2.4 AIREP algorithm:

Step 1: Input Phase:
The distribution of attribute values in the clusters was used for making the discretization according to the following procedure:

1. The number of intervals for each attribute is the same of the number of clusters where m is the mean value of the attribute in the in the clusters.

2. When two adjacent intervals overlap, the cut point (superior boundary of the first and inferior boundary of the next) is placed in the middle point of the overlapping region. These intervals are merged into a unique interval 3. When two adjacent intervals are separated, the cut point is placed in the
We define ,C_k as a candidate itemset of size k ,Z_k as a frequent itemset of size k, An AIREP algorithm is

1) Find frequent set L_{k-1}
2) Join step: C_k is generated by joining L_{k-1} with itself (cartesian product L_{k-1} x L_{k-1})
3) Prune step: Use the Incremental Reduced Error pruning to generate scalable single rule.
4) Frequent set L_k has been achieved.

The proposed AIREP (Aprori Incremental Reduced Error Pruning) pseudo code:
AIREP (T, ụ)
Z1 □ large multidimensional itemsets that appear in more than
Of large item set ụ transactions K □
2

```
        While ( Z_{k-1} ≠ 0 )
        Ck □ Generate (Z_{k-1} )   // join and prune step
        // using IREP
        procedure I-REP (Examples, SplitRatio)
        Theory =  ;
While Positive (Examples) ≠ ;
        Clause =  ;
Split Examples (Split Ratio, Examples, Growing Set, Pruning

Set)
        Cover = Growing Set
While Negative (Cover) ≠ ;
        Clause = Clause  Find Literal (Clause; Cover)
        Cover = Cover (Clause, Cover)
loop
        New Clause =  Best Simplification  (Clause,
Pruning Set)
if                    Accuracy(New Clause, PruningSet)
Accuracy  (Clause,  PruningSet)
        exit loop
        Clause = NewClause

if Accuracy(Clause, PruningSet)<=Accuracy(fail,PruningSet)
```

exit while
Theory = Theory Clause
Examples = Examples -Cover
return (Theory)
// end of IREP
//frequent set generation
for transaction t € Z
Ck □ Subset(C_k,t) for candidates c € Ct
count[c] =count[c + 1] Z_k □
{ c € C_k| count[c] >= e} k □ k+1
return Z_k

Figure 1: Pseudocode of proposed AIREP algorithm

The basic idea of Incremental Reduced Error Pruning (IREP) is that instead of first growing a complete concept description and pruning it thereafter, each individual clause will be pruned right after it has been generated. This ensures that the algorithm can remove the training examples that are covered by the pruned clause before subsequent clauses are learned thereby.

3. How algorithms are compared?

The four algorithms are compared on the following factors:

- Candidate Generation Support Counting
- Frequent itemset generation Computational Complexity Rule generation

Type of dataset used

The aim of the work is to obtain an associative model that allows studying the influence of the input variables related to the project management policy on the output variables related to the software product and the software process. The rules generated were created with a weight for the output variables three times greater than for input attributes. This is a supervised way of producing the most suitable clusters for the prediction of the output variables, which appear in the consequent part of the rules

Table.1.Comparison of factors affecting the algorithms

	Size of dataset	type of rules generated	Type of dataset
Aprori	small & large dataset	Minimum support	One dimensional
FP-Growth	Small,large dataset	High confidence	One dimensional
Scaled Association rules algorithm	Large quantitative dataset	High suppor and confidence	Multidimensional
AIREP algorithm	Large ,small quantitative dataset	High support and confidence	multidimensional

4. HOW ALGORITHMS ARE IMPLEMENTED?

Each of the four algorithms described in Section 3 was tested on the four datasets described in Section 4. The performance measure was the execution time (seconds) of the algorithms on the datasets with the following minimum support settings 5.00%,0.80%, 0.60%, 0.70%, 0.20%, 0.50%, 0.08%, 0.06%, 0.04%,0.02%, and 0.01%. The minimum confidence was always set to zero. That is, we required no minimum confidence for the generated association rules. Since some of the algorithms could only generate frequent itemsets, and some others could directly generate association rules, we measured the execution time for both creating the frequent itemsets and for creating the association rules whenever possible. Note that time for generating the association rules includes the computation for generating the frequent itemsets.

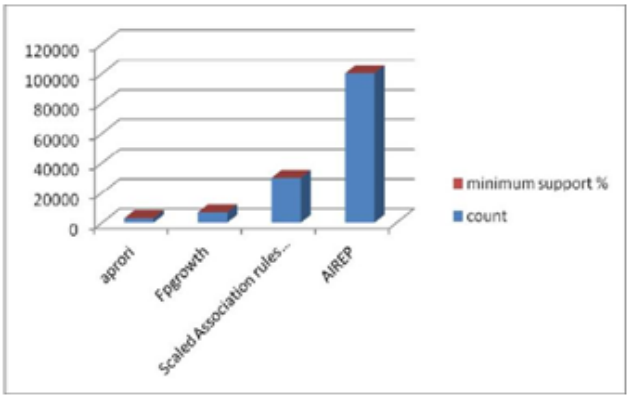

Figure .1. minimum support and count comparison

The experimental study is carried on dynamic simulation environment CBA which is a data mining tool. Its main algorithm was presented as a plenary paper "Integrating Classification and Association Rule Mining" in the 4th International Conference on Knowledge Discovery and Data Mining. However, it turns out that it is more powerful than simply producing an accurate classifier for prediction. It can also be used for mining various forms of association rules, and for text categorization or classification. This environment manages data from real projects developed in local companies and simulates different scenarios. It works with more than 18 input parameters and more than 10 output variables and generates 1569091 rules. The number of records generated for this work is 400 and the variables used are sepal length, sepal width from the Iris dataset. support of 0.40%, However, with a minimum support of 0.01% they generated 303,610 and 283,397 closed frequent items respectively, a difference of 20,213. Therefore, one or both of these implementations seems to generate incorrect closed frequent itemsets in some cases as shown in figure 5.

5. CONCLUSION

We studied the algorithms for mining quantitative association rules. Our study showed that the algorithm scales linearly with the number of records. In addition, the proposed method avoids three of the main drawbacks presented by the rule mining algorithms: production of a high number of rules, discovery of uninteresting patterns and low performance. The results show that the association rule algorithms that we evaluated perform differently on our real-world datasets than they do on the artificial dataset. The performance improvements reported by previous authors can be seen on the artificial dataset, but some of these gains do not carry over to the real datasets, indicating that these algorithms overfit the artificial dataset. The primary reason for this seems to be that the artificial dataset has very different

characteristics, suggesting the need for researchers to improve the artificial datasets used for association rule research or use more real-world datasets.

6.FUTURE SCOPE

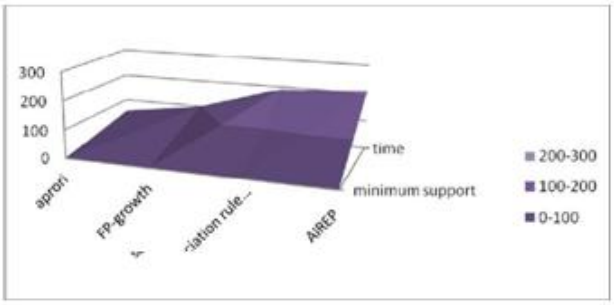

Figure.2. Rule generation comparison

We also found that the choice of algorithm only matters at support levels that generate more rules than would be useful in practice. For a support level that generates a small enough number of rules that a human could understand, Apriori finishes on all datasets in under a minute, so performance improvements are not very interesting. Even for support levels that generate around 2,000,000 rules, which is far more than humans can handle and is typically sufficient for prediction purposes, Apriori finishes processing in less than 10 minutes. Beyond this level of support, the number of frequent itemsets and association rules grows extremely quickly on the real-world datasets, and most algorithms quickly run out of either memory or reasonable disk space. Scaled association rule algorithm and AIREP generated 4,000,000 and 6000,000 respectively.

Table.1.Quality comparison of the algoritms

	QUALITY			
No of rules	**Aprori**	**FP-growth**	**Scaled Association rules**	**AIREP algorithm**
68	34	456	789	5789
2345	689	790	8457	890
34567	794	677	34788	7890
890089	800	654	677899	67890

Scaled association rule algorithms and AIREP algorithm terms of the number of closed frequent itemsets in some experiments, and the difference is large in some cases. For example, for the IBM-Artificial dataset, with a minimum. This paper was intented to compare between the standard association rule algorithms with the proposed implemented algorithms. As a future work, comparisons can be made according to different factors other than those considered in the paper. We can use normalized data or non-normalised data with different methods of generating rules.

7. REFERENCES

[1].J. P. Bigus., "Data Mining with Neural Networks", McGraw-Hill, 1996

[2].R.Karthikeyan," Improved Apriori Algorithm for Mining Rules" in the International Journal of Advanced Research in biology Engineering science and Technology Volume 11, Issue 4, April 2016, Page No:71-77.

[3].T. M. Mitchell., "Machine Learning", McGraw-Hill, 1997.

[4].R.Karthikeyan, Dr.T.Geetha "Honeypots for Network Security", International journal for Research & Development in Technology. Volume 7.Issue 2, Jan 2017, Page No.:62-66 ISSN:2349-3585

[5].R.Karthikeyan, "A Survey on Position Based Routing in Mobile Adhoc Networks" in the international journal of P2P Network Trends and Technology, Volume 3 Issue 7 2013, ISSN:2249-2615, Page No.:81-88

[6].R.Karthikeyan, "A Survey on Sensor Networks" in the International Journal for Research & Development in Technology Volume 7, Issue 1, Jan 2017, Page No:71-77

[7].R.Karthikeyan, Dr.T.Geetha "Web Based Honeypots Network", in the International journal for Research & Development in Technology. Volume 7.Issue 2, Jan 2017, Page No.:67-73 ISSN: 2349-3585.

[8].Sousa, M.S. Mattoso, M.L.Q. Ebecken, N.F.F. "Data Mining on Parallel Database Systems" Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), Special Session on Parallel Data Warehousing, CSREA Press, Las Vegas, E.U.A., Pp.1147-1154, July 1998.

[9].R.Karthikeyan,Dr.T.Geetha,"A Simple Transmit Diversity Technique for Wireless Communication", in the International journal for Engineering and Techniques. Volume 3. Issue 1, Feb 2017, Page No.:56-61 ISSN:2395-1303.

[10].R.Karthikeyan,Dr.T.Geetha "Strategy of Trible – E on Solving Trojan Defense in Cyber Crime Cases", International journal for Research & Development in Technology. Volume 7.Issue 1 ,Jan 2017,Page No.:167-171.

[11]. R.Karthikeyan,Dr.T.Geetha" Advanced Honey Pot Architecture for Network Threats Quantification" in the international journal of Engineering and Techniques, Volume 3 Issue 2, March 2017, ISSN:2395-1303, PP No.:92-96.

[12].R.Karthikeyan, Dr.T.Geetha" Estimating Driving Behavior by a smart phone" in the international journal of Engineering and Techniques, Volume 3 Issue 2, March 2017, ISSN:2395-1303,PP No.:84-91.

[13].R.Karthikeyan, Dr.T.Geetha" SAMI: Service- Based Arbitrated Multi-Tier Infrastructure for Cloud Computing" in the international journal for Research & Development in Technology, Volume 7 Issue 2, Jan 2017, ISSN (0):2349-3585, Pg.no:98-102

[14].R.Karthikeyan, Dr.T.Geetha "FLIP-OFDM for Optical Wireless Communications" in the international journal of Engineering and Techniques, Volume 3 Issue 1, Jan - Feb 2017, ISSN:2395-1303,PP No.:115-120.

[15].R.Karthikeyan, Dr.T.Geetha" Application Optimization in Mobile Cloud Computing" in the international journal of Engineering and Techniques, Volume 3 Issue 1, Jan - Feb 2017, ISSN:2395-1303,PP No.:121-125.

[16].R.Karthikeyan, Dr.T.Geetha" The Sybil Attack" in the international journal of Engineering and Techniques, Volume 3 Issue 3, May - Jun 2017, ISSN:2395-1303,PP No.:121-125.

[17].R.Karthikeyan, Dr.T.Geetha" Securing WMN Using Hybrid Honeypot System" in the international journal of Engineering and Techniques, Volume 3 Issue 3, May - Jun 2017, ISSN:2395-1303,PP No.:121-125.

[18].R.Karthikeyan, Dr.T.Geetha" Automated Predictive big data analytics using Ontology based Semantics" in the international journal of Engineering and Techniques, Volume 3 Issue 3, May – Jun 2017, ISSN:2395-1303,PP No.:77-81.

[19].R.Karthikeyan, Dr.T.Geetha" A Survey of logical Models for OLAP databases" in the international journal of Engineering and Techniques, Volume 3 Issue 3, May - Jun 2017, ISSN:2395-1303,PP No.:171-181

[20].Fayyad U, "Data Mining and Knowledge Discovery in Databases: Implications from scientific databases," In Proc. of the 9th Int. Conf. on Scientific.

[21].R.Karthikeyan, Dr.T.Geetha" A Client Solution for Mitigating Cross Site Scripting Attacks" in the international journal of Engineering Science & Computing, Volume7,Issue6, June 2017, ISSN(0):2361-3361,PP No.:13063-13067.

[22].R.Karthikeyan, Dr.T.Geetha" A Condensation Based Approach to Privacy Preserving Data Mining" in the international journal of Engineering Science & Computing, Volume7,Issue6, June 2017, ISSN(0):2361-3361,PP No.:13185-13189.

[23].R.Karthikeyan, Dr.T.Geetha" Biometric for Mobile Security" in the international journal of Engineering Science & Computing, Volume7,Issue6, June 2017, ISSN(0):2361-3361,PP No.:13552-13555.

[24].Tsau Young Lin, "Sampling in association rule mining", Conference on Data mining and knowledge discovery: Theory, Tools, and Technology VI, vol. 5433, pp.: 161-167, 2004.

[25].Klaus Julisch," Data Mining for Intrusion Detection -A Critical Review" in proc. of IBM Research on application of Data Mining in Computer security, Chapter 1 , 2002.

[26].Jeffrey W. Seifert, "Data Mining: An Overview", in proceedings of CRS Report for Congress, 2004.

[27].Coenen F, Leng P, Goulbourne, G., "Tree Structures for Mining Association Rules," In Journal of Data Mining and Knowledge Discovery, Vol. 15, pp. 391-398, 2004.