Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis on Improved Pruning in Apriori Algorithm

Jovita Vani Sequeira, Zahid Ansari
Dept.of Computer Science & Engineering,
P.A College of Engineering, Mangalore, India

Abstract*— Web usage mining is the application of data mining techniques to discover usage patterns from Web data, in order to understand and better serve the needs of Web based applications. To analyze the pattern from large transactional database there are many algorithms. One of the algorithms which is very simple to use and easy to implement is the Apriori algorithm. But this apriori algorithm is time consuming algorithm during its candidate item-set generation. IP-Apriori i.e Improved Pruning in Apriori is the improved variation of Apriori algorithm which improves the pruning step of the existing apriori algorithm. This algorithm uses average support instead of minimum support in the pruning step, to generate the probabilistic item set instead of large item-set. This analysis work is on IP-Apriori algorithm on different datasets. Based on the comparison of frequent item sets generated and time consumed, its shown that IP-Apriori algorithm is better than the Apriori Algorithm.

***Keywords*— Apriori Algorithm, Support, IP-Apriori, Pruning, Frequent itemsets**

I. INTRODUCTION

Data is a raw known fact and pattern is a subset of the data. The process of finding useful data patterns from the large amount of data is known as data mining, it is also called as information discovery, information harvesting. Data mining is just a step in Knowledge Discovery in Databases (KDD) process. KDD refers to the overall process of discovering useful knowledge from data. Data mining is the application of specific algorithms for extracting patterns from data.KDD process has several steps, which are performed to extract patterns to user, such as data cleaning, data selection, data transformation, data pre-processing, data mining and pattern evaluation. Data mining involves six classes of tasks commonly-Anomaly detection, Association Rule mining, Clustering, Classification, Regression and Summarization.

Association Rule mining consists of two main steps:1)Finding the frequent itemsets that satisfies minimum support and 2)Using the frequent itemsets to generate association rules. Finding frequent itemsets is more expensive since the number of itemsets grows exponentially with the number of items.

Web usage mining consists of three phases, namely preprocessing, pattern discovery, and pattern analysis. To analyze the pattern from large transactional database there are many algorithms. Two main strategies to find frequent itemsets from the transactions are candidate generation and test approach and pattern growth approach. One of the algorithms which is very simple to use and easy to implement is the Apriori algorithm which is generation and test approach. Apriori algorithm is the one of the most popular algorithm for association rule mining. This algorithm generates the candidate itemsets and extracts frequent item-sets from large database then applies association rule.

Literature review on Apriori algorithm is given in section II,algorithms and explanations are given in section III,then results and discussions are given in section IV. And based on the result obtained during experiments,conclusion is specified.

II. LITERATURE REVIEW

A. Related Work on Association Rule Mining:

Association rule mining is a function of data mining to discover interesting relations between items in large transactional databases, which was introduced by Rakesh Agarwal[1]. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent item-sets and the minimum confidence constraint are used to form rules.

Let I=$\{i_1, i_2, i_3, \ldots \ldots i_m\}$ be a set of items. Let D be a database of transactions. A set X=$\{i_1, i_2, i_3, \ldots \ldots i_k\}$ i.e. subset of I is called item set or a K-itemset if it contains K items. An association rule is an implication of form $A \Rightarrow B$, where A and B are subsets of I, A$\cap$B=$\emptyset$. A is called antecedent where B is called consequent. Association Rule Mining mainly involves the concept of finding support and confidence.

Support(s): It's defined as the percentage/fraction of records that contain $X \Rightarrow Y$ to the total number of records in the database. Suppose the support of an item is 0.3%,it means only 0.3 percent of the transaction contain purchasing of this item.[2]

Confidence(c):It's defined as the percentage/fraction of the number of transactions that contain $X \Rightarrow Y$ to the total number of records that contain X. Suppose the confidence of the association rule $X \Rightarrow Y$ is 90%, it means that 90% of the transactions that contain X also contain Y together.[2]

B. Related Work on Apriori Algorithm:

Mohammed Al-Maolegi and others proposed Apriori algorithm to reduce the time consuming for candidates item-set generation [3]. They firstly scan all transactions to generate L1 which contains the items, their support count and Transaction ID where the items are found. And then use L1 later as a helper to generate L2, L3 ... Lk. Then to generate C2, they make a self-join L1 * L1 to construct 2-itemset C (x, y), where x and y are the items of C2. Before scanning all transaction records to count the support count of each candidate, use L1 to get the transaction IDs of the minimum support count between x and y, and thus scan for C2 only in these specific transactions. The same thing for C3, construct 3-itemset C (x, y, z), where x, y and z are the items of C3 and use L1 to get the transaction IDs of the minimum support count between x, y and z, then scan for C3 only in these specific transactions and repeat these steps until no new frequent item-sets are identified[3].

In the apriori algorithm, if any length k pattern is not frequent in the database, it's length (k+1) super-pattern can never be frequent[4].In situations with a large numbers of frequent patterns, long patterns, or quite low minimum support thresholds, an Apriori like algorithm may suffer from the following two nontrivial costs[4]:

-It is costly to handle a large number of candidate sets. For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7length 2-candidates and accumulate and test their occurrence frequencies. To discover a frequent pattern of size 100,such as $\{i_1, i_2, i_3, \ldots \ldots i_{100}\}$,it must generate $2^{100} - 2 \approx 10^{30}$candidates in total. This is the inherent cost of candidate generation, no matter what implementation technique is applied.

-Its time consuming to scan the database repeatedly and to check a large set of candidates by pattern matching, which is especially true for mining long patterns.

III. METHODOLOGY

The Apriori algorithm generate the candidate item-sets in one pass by using only the large item-sets of the preceding pass without considering the transactions of database. The basic concept used here is that every subset of a large item-set must be large. The k^{th} pass candidate item-sets with k items is generated from previous pass by joining large item-sets of k-1 items(candidate generation),and deleting those item-sets that contain any subset that is not large(pruning).This pruning process results in generation of a much smaller number of candidate item-sets. Apriori algorithm is given below:

1). $L_1 = \{\text{large 1-itemsets}\}$
2). For ($k = 2; L_{k-1} \neq \varphi; k++$) do begin
3) $C_k = \text{candidates\_generated}(L_{k-1})$;
4.) forall transactions $t \in D$ do gin
5.) $C_t = \text{subset}(C_k, t)$
6) forall candidates $c \in C_t$ do
7) c.count ++;
8) end
9) end
10) $L_k = \{ c \in C_k \mid c.\text{count} \geq \text{minsup} \}$
11) end
12.) Answer $= \bigcup_k L_k$;

Fig 1: Apriori Algorithm[5]

The first pass of the algorithm finds the item occurrences to determine the large 1-itemsets. A pass, say k, has two phases.

First, the large item-sets, say L_{k-1} founded in the $(k-1)^{th}$ pass are used to generate the candidate item-set C_k,using the apriori-gen function takes an argument L_{k-1},the set of all large(k-1)-item-set and returns a superset of the set of all large k-item-sets. The function in the join step joins one L_{k-1} with another matching L_{k-1}. And in the next prune step,we delete those item-sets c,where $C \epsilon C_k$ such that any (k-1)-subset of c is not in L_{k-1} of the previous pass.

Then,in the next phase the database is scanned for support of candidates in C_k.

The Apriori Algorithm needs user defined threshold values, i.e minimum support(sup_{min}) and minimum confidence($conf_{min}$). Sup_{min} is needed to generate the large item-set from candidate set and $Conf_{min}$ is required to generate required set of association rules from generated large item-sets. The process of creating large item-set by removing the not so required candidate sets is called pruning. So, pruning is dependent on Sup_{min} threshold provided

by user.But some important rules get pruned because of this user-defined threshold as user doesn't know that at which threshold , valuable association rules get generated.

1) Algorithm Apriori $-$ gen(L_k)
2) insert into C_k
3) select $p.item_1 . p.item_2 \ldots \ldots \ldots p.item_n , q.item_{k-1}$ from $L_{k-1} p, L_{k-1}$ q
   Where $p.item_1 = q.item_1 \ldots \ldots . p.item_{k-2} = q.item_{k-2}, p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$,
4) for all item $-$ sets $c \in C_k$ do
5) for all $(k-1) -$ subsets s of c do
6) if($s \not\exists L_{k-1}$)then
7) delete c from C_k

Fig 2 : Apriori $-$ gen $-$ function[5]

Let's explain apriori algorithm using a simple example. A database D consists of 9 transactions. Each transaction consists of items. Here user defined minimum support is taken as 2.

TABLE I: Database D

Transactions	Items
T1	I1,I2,I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Step 1:Generating itemset of 1 frequent pattern. Scan D to find the count of each candidate.

TABLE II: C1 with the minimum support count

Itemset	Support Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Now compare the candidate support count in C1 with the minimum support count. Table III , L1 is set of frequent 1 itemset satisfying minimum support.

TABLE III: L1 with support count satisfying minimum support.

Itemset	Support Count
{I1}	6
{I2}	7

{I3}	6
{I4}	2
{I5}	2

Step 2:Generating itemset of 2 frequent items. The algorithm uses L1join L1 i.e., Fig.2.

TABLE IV: Candidate itemset of 2 frequent items.

Itemset
{I1,I2}
{I1,I3}
{I1,I4}
{I1,I5}
{I2,I3}
{I2,I4}
{I2,I5}
{I3,I4}
{I3,I5}
{I4,I5}

After finding C2,scan D to find the count of each itemset in C2.

TABLE V: Support count of C2.

Itemset	Support count
{I1,I2}	4
{I1,I3}	4
{I1,I4}	1
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2
{I3,I4}	0
{I3,I5}	1
{I4,I5}	0

Compare candidate support of C2,if support count is less than the minimum support then discard the particular set from the itemset list.

TABLE VI:L2 includes itemsets which satisfies minimum support.

Itemset	Support count
{I1,I2}	4

{I1,I3}	4
{I1,I5}	2
{I2,I3}	4
{I2,I4}	2
{I2,I5}	2

Step 3:Generating 3-itemset frequent pattern. Here to generate set of candidate 3 itemset C3, should apply Apriori Property.To find C3, compute L2joinL2.

TABLE VII:Candidate itemset C3 with the 3 frequent items.

Itemset
{I1,I2,I3}
{I1,I2,I5}

Compare the support count of each candidate set in C3 with the minimum support.

TABLE VIII: L3 shows the items of C3 satisfying minimum support.

Itemset	Support Count
{I1,I2,I3}	2
{I1,I2,I5}	2

Based on Apriori Property all subsets of a frequent itemset must also be frequent. For example,2 itemset subsets of {I1,I2,I3} are {I1,I2},{I1,I3},{I2,I3}.All 2 item subsets of {I1,I2,I3} are there in L2,hence we keep {I1,I2,I3}in C3.Let's take another example {I2,I3,I4} which has subsets {I2,I3},{I2,I4},{I3,I4}.But {I3,I4} is not there in L2.Hence its violating Apriori Property. So can't include {I2, I3, I4} in C3.

Step 4:Generating 4 itemset frequent pattern. To find candidate set of 4 itemsets algorithm uses L3join L3. The join results in {I1,I2,I3,I5} this itemset is pruned because its subset {I2,I3,I5} is not frequent.Hence C4=∅.The apriori algorithm terminates here.

The algorithm proposed by Prince verma and Dinesh Kumar[6]improves the pruning procedure of Apriori Algorithm by using average support Sup_{avg} instead of minimum support(Sup_{min}),to generate probabilistic item-set. In this algorithm average support is not user defined value but it is calculated using the below formula:

$$Sup_{min} = \frac{\sum_{k=1}^{m} \left(sup(k)\right)}{w}$$

Where w is the number of items in the transactions .So this formula would lead to better item-sets and this increases the number of better association rules. IP-Apriori contains following steps[9]:

1. Calculate average support Sup_{avg} from the given database D.
   a. For item-sets with $sup \geq Sup_{avg}$ are inserted into probability item-set.
   b. For item-sets with $sup \leq Sup_{avg}$,item-sets with For item-sets with $sup \geq S_p$ (where $S_p = Sup_{avg}/1.2$) and their probability of occurrence $> 80\%$ are inserted into probability item-set, while others are pruned.

2. Repeat the above steps for all probability item-sets and then from probability item-set, association rules are produced.

IP –Apriori algorithm is as follows:

```
1)  P_1 = {probability − item − sets};
2)  for(k = 2; P_{k−1} ≠ ∅; k++)do begin
3)      C_k = IPApriori − candidate(P_{k−1})//new candidates
4)      for all transactions t ∈ D do begin
5)          C_t = subset(C_k , t);
6)          for all candiadates C ∈ C_t do
7)              c.count++;
8)      end for
9)      if(c.count ≥ avgsup)
10) P_k = {c ∈ C_k};
```


11) else
12) $S_p = \frac{Sup_{avg}}{1.2}$;
13) $c.prob = (c.count * 80)/100$
14) if$(c.count \geq S_p$ and $c.count \geq c.prob)$
15) $P_k = \{c \in C_k\}$;
16) end if
17) end
18) Answer $= U_k P_k$

Fig 3 : IP − Apriori Algorithm [6]

1) Algorithm IPApriori − candidate(P_k)
2) insert into C_k
3) select $s.item_1 . s.item_2 \ldots \ldots \ldots s.item_n, t.item_{k-1}$ from P_{k-1}s, P_{k-1} t
Where $s.item_1 = t.item_1 \ldots \ldots s.item_{k-2} = t.item_{k-2}, s.item_{k-2} = t.item_{k-2}$,
$s.item_{k-1} < t.item_{k-1}$,
4) for all item − sets $c \in C_k$ do
5) for all $(k-1)$ − subsets s of c do
6) if$(s \notin P_{k-1})$then
7) delete c from C_k

Fig 4: IP − Apriori − candidate generation function Algorithm[5]

Here in IP-Apriori-candidate algorithm candidates are generated for creating probability item-set in IP-Apriori algorithm. For example, consider the database given in table I .In this algorithm,after finding the support count i.e., table II, average of the support count is calculated. Then the itemsets with support count more than the average support count are inserted into the probability itemset.

$$Sup_{avg} = \frac{6+7+6+2+2}{5} = 4.6.$$

TABLE IX: L1 ,satisfying average support.

Itemset	Support Count
{I1}	6
{I2}	7
{I3}	6

Average support is 4.6. Hence items {I4} and {I5} with support count 2 does not satisfy the minimum support . Calculate $S_p = \frac{Sup_{avg}}{1.2} = \frac{4.6}{1.2} = 3.8333$. Again calculate the 80% of the {I4} frequent count. $(\frac{2*80}{100} = 1.6)$. Check $2 \geq 3.8333$ and $2 \geq 1.6$. This condition does not satisfy.{I4} and {I5} with frequency count 2 are deleted from the list. Now, generate the itemset of 2 frequent items. This can be done by L1joinL1.

TABLE X: C2 by L1join L1.

Itemset
{I1,I2}
{I1,I3}
{I2,I3}

After finding C2, scan the database D to find the support count.

TABLE XI: L2,probabilistic itemset of C2.

Itemset	Support Count
{I1,I2}	4
{I1,I3}	4

{I2,I3}	4

Table XI, L2 contains the support count of candidate itemsets. Find the average count $Sup_{avg} = \frac{4+4+4}{3} = 4$. Compare the support count with the average count. Since all support counts are equal to the average count, it satisfy the first condition.
Generate the C3,by joining L2.

TABLE XII: C3 found by joining L2.

Itemset
{I1,I2,I3}

Find the support count of C3 candidate itemset.

TABLE XIII: Support count of C3.

Itemset	Support count
{I1,I2,I3}	2

Average support is 2. Hence this is taken as frequent itemset. There is only one candidate set. So join operation cannot be performed. Algorithm terminates here.

IV. RESULTS AND DISCUSSIONS

Experiments are performed by taking a small part of chess and mushroom dataset. Dataset description is given below:

TABLE XIV: Dataset Description

Dataset	Transactions	Items
Chess	20	35
Mushroom	20	70

The results obtained for chess_dataset is given below in TABLE XV ,when the user defined support is 10%,most of the itemsets sre selected as frequent itemsets and time taken is very high. Same way for 20%.But when user defined support is 40% or 60%,very few frequent itemsets are generated within few seconds i.e.,some of the frequent itemsets are not selected. In the IP-Apriori algorithm,in each iteration average support changes,for that average support frequent itemset generated are based on the average of total support of items in particular dataset.Hence the frequent itemset generated are neither high nor less. Even in mushroom_dataset results obtained proves that IP-Apriori generates good and better frequent item sets i.e.,TABLE XVI.

TABLE XV: Results of chess_dataset

Support	10%	20%	40%	60%	Avg_Support
FrequentItemsets	71807	26495	3967	1535	3839
Time(seconds)	156	25	2	1	2

The results obtained for mushroom_dataset is given below:

TABLE XVI: Results of mushroom_dataset

Support	10%	20%	40%	60%	Avg_Support
FrequentItemset	101989	18799	773	79	81220
Time(seconds)	312	14	0.258	0.041	200

Comparison based on time is shown in Fig.5. Chess_dataset and mushroom_dataset takes high time when support is 10%. When support threshold is 20% or 40% or 60%,time taken is very less,because it generates very less frequent itemsets. But for average support time consumed is neither high nor less.

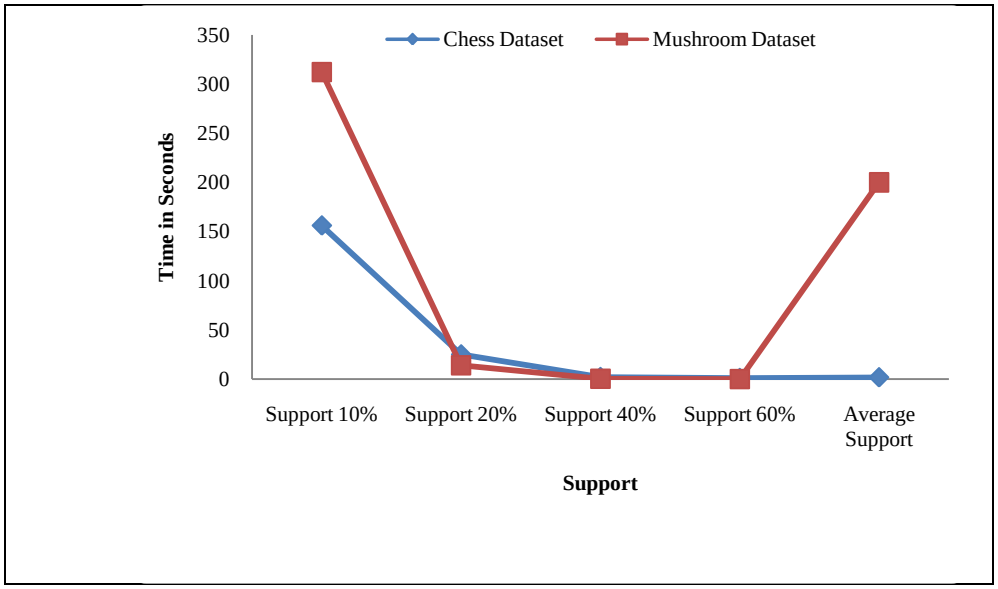

Fig 5:Comparison based on Time

Comparison based on frequent itemsets generated for support is shown in Fig.6. Chess_dataset and mushroom_dataset generates high frequent itemsets when support is 10%. When support threshold is 20% or 40% or 60%,frequent itemset generated is very less.But for average frequent itemsets generated are neither less nor high.

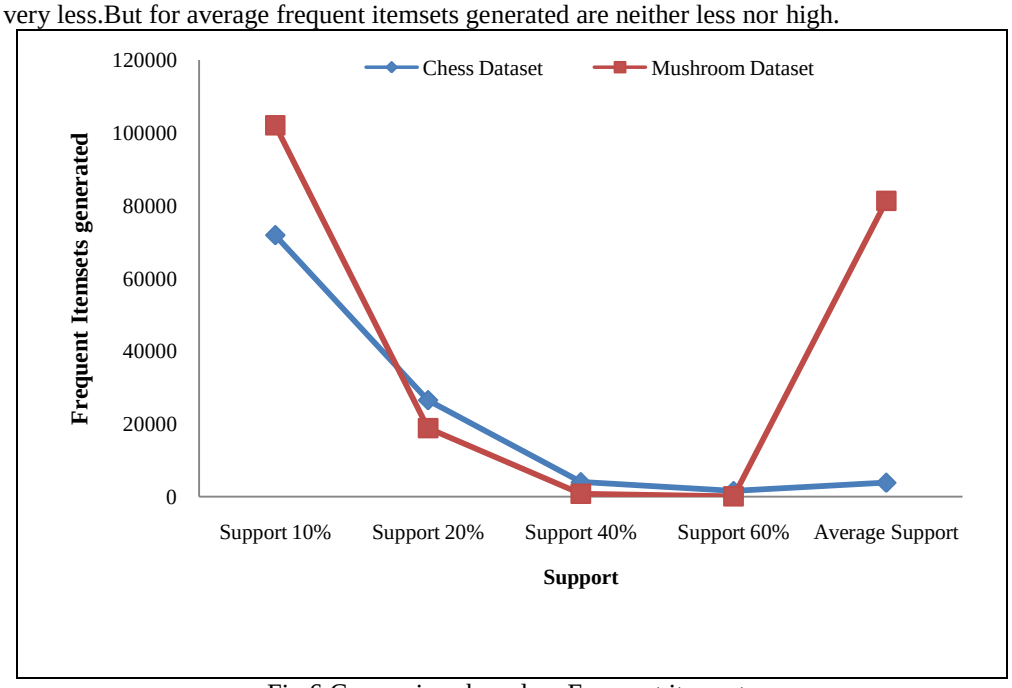

Fig 6:Comparison based on Frequent itemsets

IV. CONCLUSION

In the Apriori algorithm, user should change support threshold each and every time to get the better result. IP-Apriori requires only average support.The experiment conducted on different datasets proves that user defined support leads to a conflict results. But taking average support while pruning, generates better frequent itemsets. Even in case of time consumption, user defined support threshold needs lots of time to generate frequent itemsets. Hence Analysis on IP-Apriori shows that IP-Apriori is better than the Apriori Algorithm.

REFERENCES

[1] R.Agrawal,T.IMielinski, and A.N.Swami, "Mining Association rules between sets of items in large databases" *ACM SIGMOD International Conference on Management of Data*,Washington,D.C.,1993,pp 207-216.

[2] Kuldeep Malik,Neeraj Raheja and Puneet Garg,*"Enhanced FP-Growth Algorithm"*,ISSN(online):2230-7893,*"International Journal of Computational Engineering and Management*",Vol.12,April 2011.

[3] Mohammed Al-Maolegi, Bassam Arkok,"An Improved Apriori Algorithm For Association Rules",*International Journal on Natural Language Computing(IJNLC)* Vol.3,No.1,February 2014.

[4] Jiawei Han,Jian Pei,Yiwen Yin,Runying Mao,"Mining Frequent Patterns without candidate Generation :A Frequent-Pattern Tree Approach",Data mining and Knowledge Discovery,8, 53-87,2004.

[5] Rakesh Agarwal,Ramakrishna Srikant,"Fast Algorithms for mining association rules" *VLDB conference Santiago*,Chile,1994,PP 487-499.

[6] Prince Verma, and Dinesh Kumar,"IP-Apriori:Improved Pruning in Apriori for Association Rule Mining",*International Journal of Recent Technology and Engineering(IJRTE)*,ISSN:2277-3878,Volume-2,Issue-4,September 2013

[7] A.V.Senthil and R.S.D.Wahidabanu,"Discovery of Frequent Itemsets: Frequent Item Tree-Based Approach",ITB J.ICT Vol.1 C,No.1,2007,42-55.

[8] M.Suman,T.AnuRadha,K.Gowtham,A.Ramakrishna, *"A Frequent Pattern Mining Algorithm Based On Fp-tree Structure And Apriori Algorithm",ISSN:2248-9622,"International Journal of engineering Research and Applications",Vol*.2,Issue 1,Jan-Feb 2012,pp114-116

[9] B.Santhosh Kumar ,K.V.Rukmani , Implementation of web usage mining using Apriori and FP-Growth Algorithms, *Int.J.of Advanced Networking and Applications*,Vol.01,Issue:06Pages :400-404(2010).

[10] F.Crespo and R.Weber,"A methodology for dynamic data mining bassed on fuzzy clustering",*Fuzzy Sets and Systems*,vol.150,no:2,pp.267-284,Mar.2005.

[11] U.Fayyad,G.Piatesky-Shapiro and P.Smyth,"From Data Mining to Knowledge Discovery :an overview"*Advances in Knowledge discovery and Data Mining,MIT* Press,Cambridge,1996,MA.

[12] J. Han and M. Kamber, *"Data Mining: Concepts and Techniques"* .Morgan Kaufmann Publications 2009.

[13] Sotiris Kotsiantis, Dimitris Kanellopoulos,Association Rule Mining:A recent Overview *GESTS International Transactions on Computer Science and Engineering*, Vol.32 (1), 2006, pp. 71-82.

[14] L.K. Joshila Grace, V. Maheswari, and Dhinaharan Nagamalai (Jan 2011)" Web Log Data Analysis and Mining" in *Proc CCSIT-2011*, Springer CCIS, Vol 133, pp 459-469

