ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 7, July 2019

# Detection of the Frequent Patterns in the Data Science Environment

**K. G. Maheswari**

Assistant Professor (Senior), Department of IT, Institute of Road and Transport Technology, Anna University, Erode, Tamil Nadu, India

**ABSTRACT:** Today, cloud computing has been coined as sophisticated on demand computing services for organization and individual. Because of this new technology growth, large amount of Big data are generated in different types of area like hospital, share market, social network, retail on line business etc. With the increase in Big data, automatically increase the storing large set of data online. In cloud environment data are analysed for classification, clustering, predication, life time value and forecasting. Sorting of these big data, and find frequent itemsets are the big challenging for data scientist in cloud environment. There are several algorithms are in data mining, for finding a sequential pattern in massive datasets. In this paper, Eclat algorithm is used for identifying and projecting the frequent item sets in an effective manner. The focus of the research work is to determine the frequent item set in data science. The performance of the algorithm is compare to existing algorithm and the proposed algorithm gives the better performance

**KEYWORDS** – Apriori Algorithm, Eclat algorithm, frequent items, sequential pattern, data science

## I. INTRODUCTION

Today's world , Data science play an important role in cloud computing technology because of huge amount of data generated by different area like organization , medical and product etc. These data are Analyed for different purpose like the prediction product, health care, predicting life time value, demand of forecasting etc. So to maintain the data science is important work and one of the most challenging area of data science is to find the frequent item sets present in the data ware house (or) from massive datasets. In traditional data analysis, focus on between data and human but today massive interconnection processing between data and machines. This will lead the two effects

- Based on the data machine can take own decision making
- Correct conclusions between human to computer

Data Mining is the most important and widely used method for the discovery and investigation of large quantity of data, in to obtain valid, useful and intelligent patterns hidden in database. The real world database like retail sales, production, finance, banking, population study etc., have huge amount of record, but it is very difficult to get useful information because of lack of tools to use this known information. The exponential growth of real time data, new techniques and tools are needed that could intelligently transform data into useful information. This is achieved by discovering the valuable patterns, transforming data and knowledge presentation etc. Data mining is a one the technique to analyzing data from different presectives and gives the useful information. There different types of data mining techniques like clustering, classification and association etc. One of the specific data mining tasks is frequent pattern mining. The objective of frequent pattern mining is to discovery all the sequential patterns with minimum user specified support (ie) the pattern is supported by number of data sequences. Another important data mining technique is association rule. It is used in the analysis the real time application like customer behavior in banking system, retail trade etc. The goal of association rule mining is using with association rules finds the minimum support (item set) and confidence from the known database (Frequent item set). For example let I = {A1, A2, A3…..m} be an itemset. Let D, the task relevant data, each transaction T is a non empty itemset such that $T \leq C$ I. Let A be a set of items. T the transaction is said to contain A if $A \subseteq T$. An association rule is A => B, where $A \subset I$ , $B \subset I$, $A \neq \phi$,$B \neq \phi$□ , $A \cap B = \phi$.□The rule A => B holds in transaction set D with support s and confidence C in the transaction set D.

$$\text{Support } (A \Rightarrow B) = p\,(A \cap B) \qquad (1)$$

$$\text{Confidence } (A \Rightarrow B) = P\,(B|\,A) \qquad (2)$$

The occurrence frequency of an item set in the number of truncations in the item set. Then I is a frequent itemset. The set of frequent k – itemset is commonly denoted by $L_{k.}$

$$\text{Confidence } (A \Rightarrow B) = P\,(B|\,A) = \frac{support(A \cup B)}{Support(A)} = \frac{support\_count(A \cup B)}{Support\_count(A)} \qquad (3)$$

The equation (3) shows the confidence rule of A => B, which is derived from support _count of A and $A \cup B$.

## II. RELATED WORK

Agrawal et al [1] have analyzed various types of algorithm for frequent item set. They proposed two algorithms Apriori and Apriori Tid for frequent patterns. These combined algorithm procedure results in creation of a much smaller number of candidate item sets. Srikant et al [5] discuss the mining of generalized association rules for a huge database of transactions, which specify the association among the data items available in the transaction.

Show-Jane Yen et al., [4] proposed an algorithm based on graph to mine the frequent data items from the database and this algorithm gives association graph to specify the associations between items, and then pass through the graph to generate large item sets and sequences. The proposed algorithm used to decreasing scan time and also increasing the performance.

Wei Wei et al [7] discuss an association rule growth called AR-Growth, which all together discover frequent item set and association rules in a big database. It is analyzed the algorithm and association rules generated by the algorithm are absolute. Wanjun Yu et al [6]., developed a new algorithm called Reduced Apriori Algorithm, which minimize one redundant pruning operations of C2, so reduce the processing time and increasing efficiency.

Chaohui Liu et al.,[2] discuss about a Three-dimensional Item sets which consists of Matrix, Vectors algorithm, and broke through the Apriori bottom-up framework. It requests one pass to scan the database and but not created any candidate item sets. FUFIA (Fast Updating Frequent Item sets Algorithm) generates new frequent item sets through three-dimensional matrix.

Jia- Ling Koh et al., [3] projected algorithm which is FP-tree based Approximate Frequent Item sets to find frequent item sets from a FP-tree structure which represent recursive function for transactions which fault tolerant contain an item set P and the patterns are specify by the conditional AFP-trees of P. Maheswari KG et al.,[8] discuss the security issues in the cloud environment and they proposed new devices as part of the VM assignment give security. Maheswari KG et al.,[9] They discuss the various classification of attacks related to web application attacks. Maheswari KG et al.,[10] They discuss about the client side scripting attacks and to identify the intrusions with high flase alarms. Nalinipriya et al .,[11]in that work they discuss about the essential process for evaluation of cloud security. Nalinipriya et. al., [12] They discuss the security risks in cloud environment and they also discuss to minimize the real time attacks in cloud environment.

## III. APRIORI ALGORITHM FOR FINDING FREQUENT ITEMSETS

Apriori is an algorithm proposed by R.Agrawal and R.Srikant[1] for mining frequent itemsets using Boolean association rules. But it is the algorithm uses prior knowledge of frequent itemset properties. Apriori algorithm uses an iterative approach known as level-wise search, where k-itemsets are used to get (k+1) - itemsets. Initially a set of frequent 1-item sets is found by scanning the database for the occurrences of count of each item and collecting those items that satisfies the minimum support count.

The figure 1.1 shows the Apriori Algorithm [1], which The resulting set L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. Each Lk requires one full scan of the database to find item sets and to improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property used for reducing the search space.

Apriori property: All non-empty subsets of a frequent item set must also be frequent.

```
$L_1$ = find_frequent 1-itemsets (DB);
for (k = 2; $L_{k-1} \neq ¢$; k++)
{   $C_k$ = apriori_cangen($L_{k-1}$);
    for each transaction t € DB
    {       $C_t$ = subset($C_k$,t);
        for each candidate c € $C_t$
            c.count++;
    }
    $L_k$ = {c € $C_k$[c.count >= min_sup]}
} return L = $U_k L_k$;
procedure apriori_cangen($L_{k-1}$ : frequent(k-1)-itemsets)
  for each itemset $l_1$ € $L_{k-1}$
     for each itemset $l_2$ € $L_{k-1}$
        if ($l_1$[1] = $l_2$[1]^ $l_1$[2] = $l_2$[2]) ^...^ ($l_1$[k-2] = $l_2$[k-2]^ $l_1$[k-1] = $l_2$[k-1])          then {
            c =  Join $l_1$ and  $l_2$;
            if has_infrequent_subset(c, $L_{k-1}$) then
                delete c;
            else add c to $C_k$;
           }
    return $C_k$;
procedure has_infrequent_subset(c; $L_{k-1}$);
  for each .(k-1)-subset s of c
      if $s \in L_{k-1}$  then
          return TRUE;
 return FALSE;
```

Figure 1.1 Apriori Algorithm

There are two-steps to find the frequent itemsets: join and prune actions.

a) The join step : To find Lk a set of candidate k-item sets Ck, is created by

joining Lk-1 with itself.

b) The prune step

A scan of the database to define the count of each candidate in Ck result in the determination of Lk, all candidates having a count not less than the least support count are frequent and therefore belong to Lk. To reduce the size of Ck, the Apriori property is used as follows. Any (K-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset.

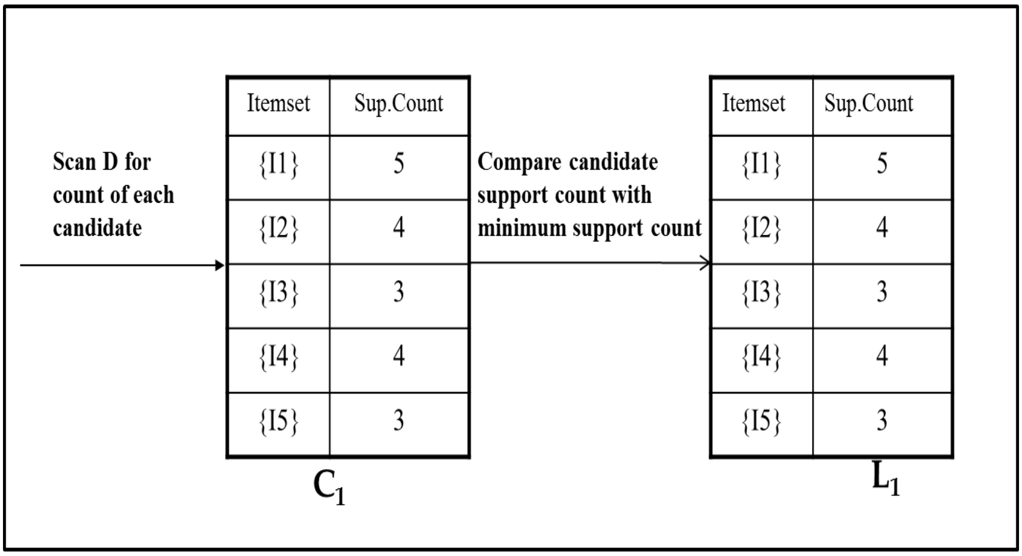
If any (K-1)-subset of a candidate k-itemset is not in Lk-1, then the candidate cannot be frequent either and so can be removed from Ck. Once we have retrieved all frequent itemsets in the database, it generates the association rule satisfied the confidence threshold (min_conf) from frequent itemset. DB is a database of transactions taken as input and output the frequent itemsets available in the database. Initially Apriori finds the frequent 1 itemsets, $L_1$ then $L_{k-1}$ is used to generate candidates $C_k$ to find $L_k$ for k>= 2. The apriori_cangen procedure generates the candidate itemsets using join procedure by joining $L_{k-1}$ with $L_{k-1}$ then pruning applies the Apriori property to eliminate the itemsets that is having subset which is not frequent. Once all of the candidates have been generated, the database is scanned, the count for each of these candidates is accumulated and all the candidates satisfying the minimum support count form the set of frequent itemsets, L. Association rules from the frequent itemsets have to be generated. The test for infrequent subsets is shown in procedure has_infrequent_subset.

| TID | List of Items |
|---|---|
| T1 | I1, I2, I4 |
| T2 | I2, I4 |
| T3 | I3, I4 |
| T4 | I1, I2, I5 |
| T5 | I1, I4 |
| T6 | I1, I3, I5 |
| T7 | I1, I2 ,I3, I5 |

Table 1.1 **Transaction Database for Apriori Algorithm**

The transaction databases D with seven transactions are shown in Table 1.1. The transaction T1 contains the itemsets of I1, I2, I4 and transaction T2 contains the itemsets of I2, I4. In this way all the seven transactions have its own itemsets.

Apriori algorithm is used for the generation of frequent 1 itemset from the databases D is shown in the figure 1.2. In the first iteration of the algorithm, the entire available item in the database is a member of the candidate 1-itemsets, C1.

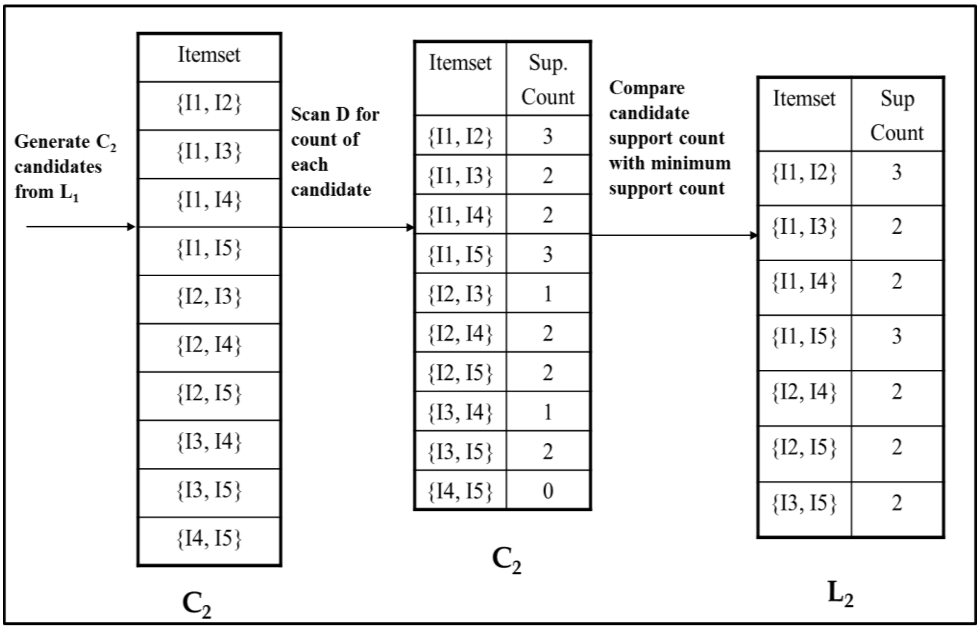


| Itemset | Sup.Count |
|---|---|
| {I1} | 5 |
| {I2} | 4 |
| {I3} | 3 |
| {I4} | 4 |
| {I5} | 3 |

| Itemset | Sup.Count |
|---|---|
| {I1} | 5 |
| {I2} | 4 |
| {I3} | 3 |
| {I4} | 4 |
| {I5} | 3 |

**Figure 1.2 Generation of Frequent 1- Itemset in Apriori Algorithm**

The algorithm counts the number of occurrences of each item by scanning all the transactions. The minimum support count is 2, frequent 1-itemsets, L1, can be determined by all of the candidates in C1 satisfy minimum support. Figure 1.3 shows how Apriori algorithm is used for the generation of frequent 2 itemset from the databases D.



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

| Itemset | Sup. Count |
|---|---|
| {I1, I2} | 3 |
| {I1, I3} | 2 |
| {I1, I4} | 2 |
| {I1, I5} | 3 |
| {I2, I3} | 1 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 1 |
| {I3, I5} | 2 |
| {I4, I5} | 0 |

| Itemset | Sup Count |
|---|---|
| {I1, I2} | 3 |
| {I1, I3} | 2 |
| {I1, I4} | 2 |
| {I1, I5} | 3 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I5} | 2 |

**Figure 1.3 Generation Frequent 2- Itemset in Apriori Algorithm**

The candidate 2 itemsets are generated by joining L1 with L1 and during pruning no candidates are removed from C2 because each subset of the candidates set is also frequent. Database is scanned for generating the support count of each candidate itemset in C2. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.

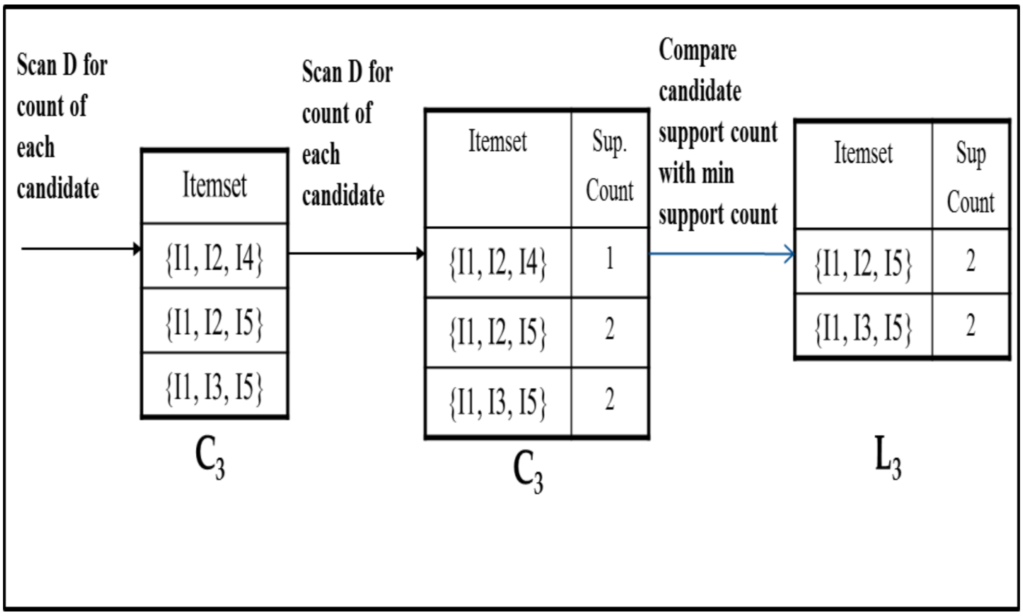


**Figure 1.4 Generation of Frequent 3- Itemset in Apriori Algorithm**

Frequent 3 itemsets from database D is generated using apriori algorithm is shown in figure 1.4.The candidate 3 itemsets are generated by joining L2 with L2 and apply pruning operation on that candidate 3 itemsets. The transactions in D are scanned to determine L3, consisting of those candidate3-itemsets in C3 having minimum support. L3 is having 2 sets of 3 item sets which satisfies the minimum support count are {I1,I2,I5} and {I1,I3,I5}. Frequent 4 itemsets from database D is generated by joining L3 with L3 to obtain candidate 4 itemsets and results in {I1, I2, I3, I5} and it is pruned because its subset is not frequent. Thus C4 = ¢ and algorithm terminates. Let I = $\{i_1,i_2...i_m\}$ be a set of items. Let D the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. An association rule is an implication of the form in equation (4)

$$A \Rightarrow B, \text{ where } A \subset I,\ B \subset I \text{ and } A \cap B = \Phi. \qquad (4)$$

The rule A=>B holds in the transaction set D with support s, where s is the percentage of transactions in D that contain AUB. This is taken to be the probability, P(AUB). The rule A=>B has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(A|B). The support of an itemset is the percentage of transactions in the DB in which the itemset appears. **A =>B figure 1.5 shows** Support percentage of transaction

Support(A => B) = P(A U B)

Support(A U B) = No. of Tuples containing both A and B

/ Total no of tuples

Figure 1.5 Support percentage of transaction

Confidence is defined as the measure of certainty or trustworthiness associated with each discovered Pattern. This signifies the purchase of item B, whenever item A is purchased. **A =>B figure 1.6 shows** Confidence percentage of transaction

Confidence(A => B) = P(B | A)

Confidence(A => B) =No. of Tuples containing both A and B / No of tuples containing A

Figure 1.6 shows Confidence percentage of transaction

Rules that satisfy both a minimum support threshold (min_sup) and a minimum confidence threshold (min_conf) are called strong rules. Table1.2 shows the strong association rule generated for the transaction

I2 ∧ I5 => I1

I3 ∧ I5 => I1

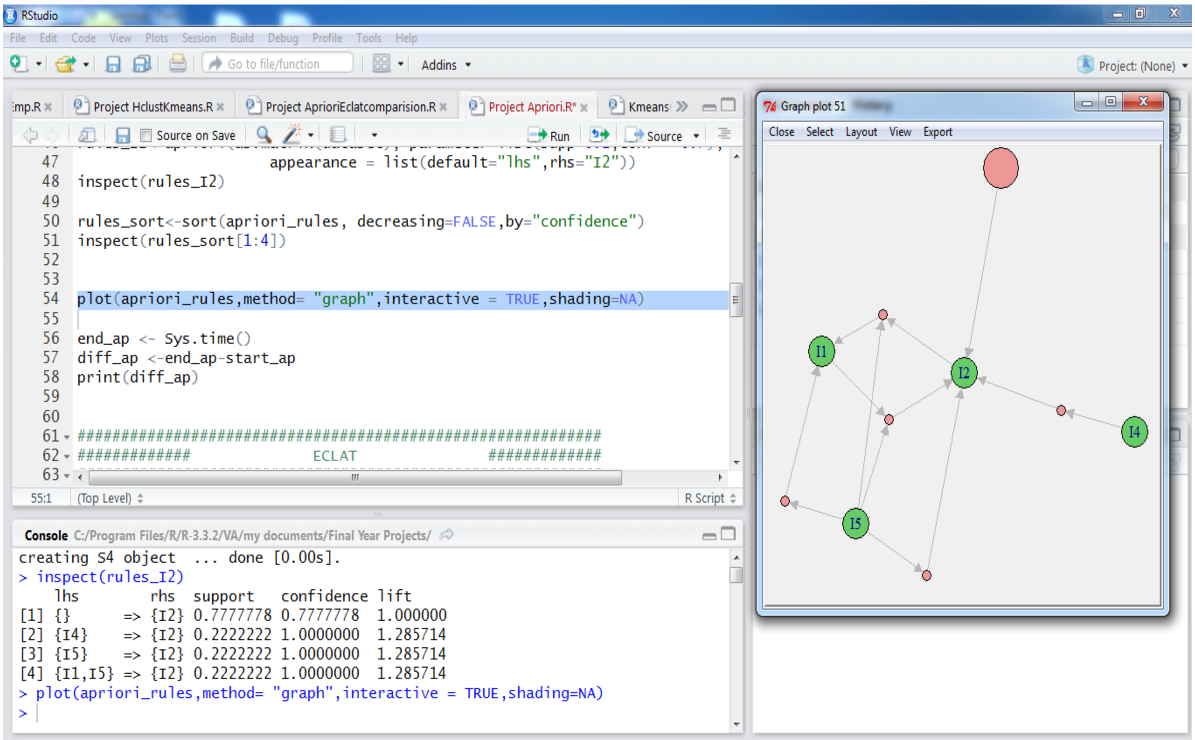
I1 ∧ I3 => I5

Table1.2 strong association rule

The above association rule specifies that in a sales environment, the customer who is getting I2 and I5 will also get I1 then the customer who is buying I3 and I5 will also buy I1 and the customer who is purchasing I1 and I3 will also purchase I5. In this logic, Apriori algorithm will generates frequent patterns and association rules among the dataitems.

### APRIORI ASSOCIATION RULE GRAPH

**The figure shows the Apriori association rule graph**



```
                      appearance = list(default="lhs",rhs="I2"))
inspect(rules_I2)

rules_sort<-sort(apriori_rules, decreasing=FALSE,by="confidence")
inspect(rules_sort[1:4])


plot(apriori_rules,method= "graph",interactive = TRUE,shading=NA)

end_ap <- Sys.time()
diff_ap <-end_ap-start_ap
print(diff_ap)


############################################################
#############             ECLAT               #############
```

Console C:/Program Files/R/R-3.3.2/VA/my documents/Final Year Projects/

```
creating S4 object  ... done [0.00s].
> inspect(rules_I2)
    lhs          rhs  support   confidence lift
[1] {}        => {I2} 0.7777778 0.7777778  1.000000
[2] {I4}      => {I2} 0.2222222 1.0000000  1.285714
[3] {I5}      => {I2} 0.2222222 1.0000000  1.285714
[4] {I1,I5} => {I2} 0.2222222 1.0000000  1.285714
> plot(apriori_rules,method= "graph",interactive = TRUE,shading=NA)
>
```

Figure 1.7 ASSOCIATION RULE GRAPH

## IV. ECLAT ALGORITHM FOR FINDING FREQUENT ITEMSETS

Zaki et al[ } was introduce éclat algorithm and Elcat means equivalence class clustering and Bottom up Lattice Traversal. Eclat algorithm is used for identifying and projecting the frequent itemsets and association rules among the data sets from large database in an effective manner. Goethals(2003) describe the éclat algorithm as follows figure 4.1

1. Get TID list for each item from DB

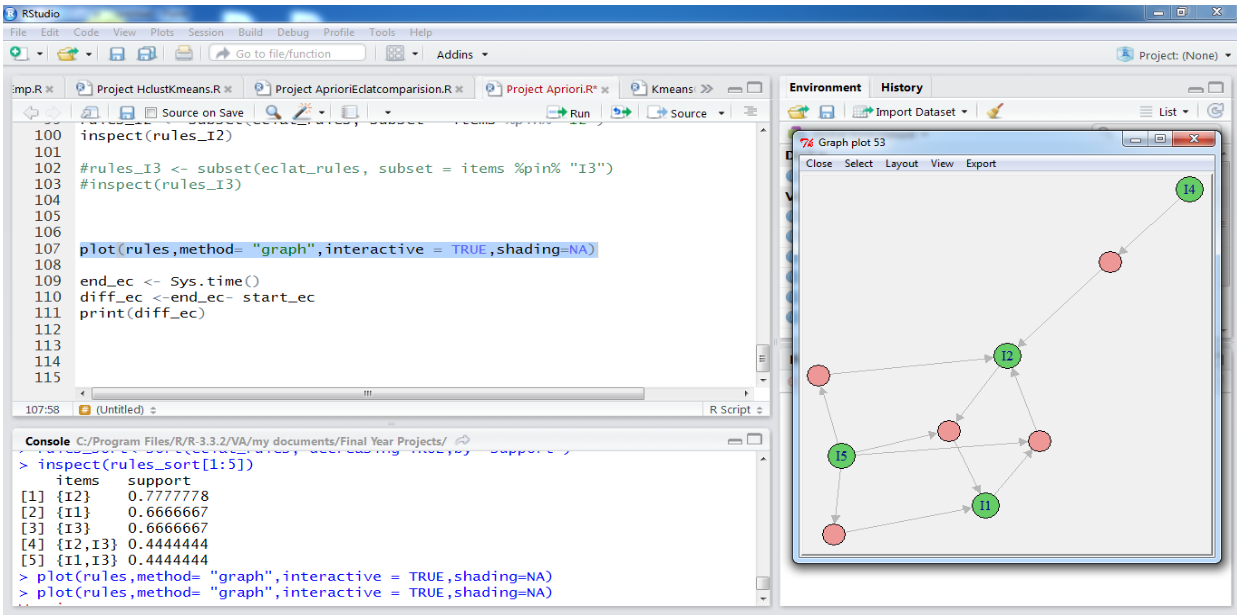
2. TID list of {a} is exactly the list of transactions containing {a}

3. Intersect the TID list of {a} with TID list of all other items, resulting in 2-itemsets – {a,b},{a,c}….

4. Then form 3 –itemsets – {a,b,c},….

The above process is repeated until no

The main difference between Eclat and apriori algorithm is Eclat have depth first search and apriori algorithm is BFS. The initial step to éclat is I value of { }, it means prefix is not required and it used to get all single _item frequent itemset. The objective of Eclat algorithm is intersecting tidsets. It is one of the important factor, because that affecting running time and memory usage of éclat. The sample dataset is given as shows in Figure 2



```
inspect(rules_I2)

#rules_I3 <- subset(eclat_rules, subset = items %pin% "I3")
#inspect(rules_I3)



plot(rules,method= "graph",interactive = TRUE,shading=NA)

end_ec <- Sys.time()
diff_ec <-end_ec- start_ec
print(diff_ec)
```

```
> inspect(rules_sort[1:5])
    items   support
[1] {I2}    0.7777778
[2] {I1}    0.6666667
[3] {I3}    0.6666667
[4] {I2,I3} 0.4444444
[5] {I1,I3} 0.4444444
> plot(rules,method= "graph",interactive = TRUE,shading=NA)
> plot(rules,method= "graph",interactive = TRUE,shading=NA)
```

Figure 4.2 sample dataset

## V. PROPOSED SYSTEM ARCHITECTURE

The architecture diagram is defined with the flow of data, which is refined and used for finding frequent patterns and association rule between the itemsets. The fig 5.1 shows the architectural diagram of this project. The first step is Data Importing and Data Preprocessing. In data importing, data have to be loaded in to the R environment for analysis.

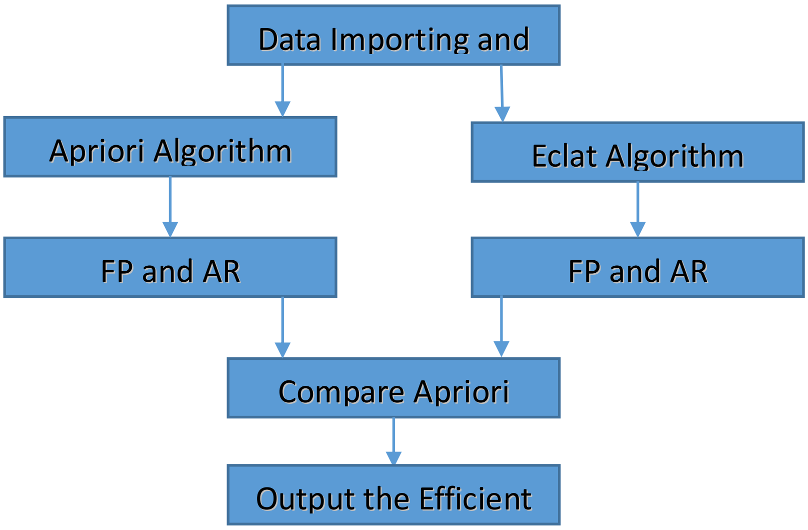


Figure 5.1 Proposed **System Architecture**

data preprocessing, the collected raw data have to be converted into understandable format. Standardization and Normalization is the technique which is used to transform the various format of data into the common format and min-max technique is used for normalization of data values. It is not necessary to hold all the attributes for doing the analysis, we can hold only the attributes which is affecting the analysis. The missing Values problem have to be solved by simple statistical techniques. The preprocessed data is given as an input to Apriori algorithm and Eclat algorithm. These algorithm generates the frequent patterns and association between the itemsets. The output is visualized using graph. Finally, the performance of Apriori and Eclat algorithm, is compared on the basis of execution time.

## VI. COMPARISON OF DATA SCIENCE ALGORITHMS

The Table 5.2 shows the performance of Apriori and Eclat algorithms are evaluated using execution time. The total number of records taken for evaluation is 10000, 50000 and 100000.

| **S.No** | **ALGORITHMS** | **TOTAL NO OF RECORDS** | | |
|---|---|---|---|---|
| | | **10000** | **50000** | **100000** |
| **1** | **APRIORI** | **4.7 sec** | **5.9 sec** | **7.4 sec** |
| **2** | **ECLAT** | **3.5 sec** | **4.4 sec** | **4.7 sec** |

**Table 5.2 Performance evaluation of Apriori and Eclat**

For 10000 records, Apriori algorithm takes 4.7 seconds and Eclat algorithm takes only 3.5 seconds. For 50000 record, Apriori algorithm takes 5.9 seconds and Eclat algorithm takes only 4.4 seconds. For 100000 records, Apriori algorithm takes 7.4 seconds and Eclat algorithm takes only 4.7 seconds.

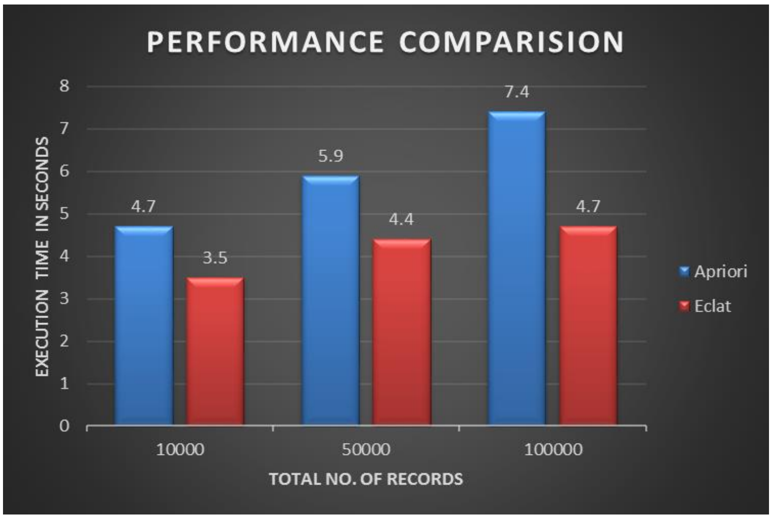


**Figure 5.6 Performance graph of Aprior and Eclat Algorithm**

The Eclat algorithm has taken less execution time for finding the frequent itemsets and association rule among the items when compared to Apriori algorithm. The performance of the algorithms has shown in graphical representation in the figure 5.6. This shows that the Eclat algorithm has a better performance over Apriori algorithm.

## VII. CONCLUSION AND FUTURE ENHANCEMENT

This work addresses the algorithms used for finding of frequent patterns and association rule from huge amount of datasets. Initially Apriori algorithm is applied to find the frequent patterns and association rule, then Eclat algorithm is used. In Eclat, it is not necessary to scan the entire database, for finding the support count. The massive experimental work was performed for evaluating and comparing Apriori and Eclat algorithm. The execution time taken by Eclat algorithm for calculating frequent patterns and association rule is very less compared to Apriori algorithm. Thus this work concludes, that Eclat algorithm consistently performs better and faster than Apriori algorithm.

The future enhancement will focus on, now it takes only for structured dataset , this can be improved by using unstructured dataset. Association rule generated by Apriori and Eclat can be optimized using genetic algorithm, so that strong rules will be obtained which improves the effectiveness of the algorithm, so that performance of the algorithm will be increased.

## REFERENCES

1. Agrawal, R., and Srikant, R. " Fast Algorithms for Mining Association Rules", Proceedings of the 20th International Conference on Very Large Databases, Santiago, Chile, 1994, pp.487-499.
2. Chaohui Liu., and Jiancheng An. "Fast Mining and Updating Frequent Itemsets", Proceedings of IEEE International Colloquium on Computing, Communication, Control, and Management, 2008, pp.365- 368.
3. Jia-Ling Koh., and Y-Lang Tu. "A Tree-based Approach for Efficiently Mining Approximate Frequent Itemsets", IEEE Transactions on Knowledge and Data Engineering, 2010, pp.25-36.
4. Show-Jane Yen., and Arbee Chen, L.P. "An Efficient Approach to Discovering Knowledge from Large Databases", Proceedings of IEEE International Conference on Parallel and Distributed Information Systems, 1996, pp.8-18.
5. Srikant, R., and Agrawal, R. "Mining Generalized Association Rules", Proceedings of the 21st International Conference on Very Large Databases, Zurich, Switzerland, 1995, pp.407-419.
6. WanjunYu., and XiaochunWang. "The Research of Improved Apriori Algorithm for Mining Association Rules", IEEE International Conference on Communication Technology, 2008, pp.513-516.
7. Wei Wei., Songnian Yu., Qiang Guo., and Wang Ding. "An Effective Algorithm for Simultaneously Mining Frequent Patterns and Association Rules", IEEE Transaction on Knowledge and Data Engineering, Vol.1, 2008, pp.190-195.
8. **Maheswari KG** & Anita R 2016 'A dynamic virtual machine security scheme against co resident attack in cloud computing' **Asian Journal of Information Technology**, vol. 15, no. 13, pp. 2116-2122, ISSN 1682-3915. (Updated Journal 2016 IF: 0.22)
9. **Maheswari KG** & Anita R 2015 "Generalization of minkowski distance metrics in mixed case analysis for web intrusion detection" in **International Journal of Soft Computing,** vol. 10, no. 4, pp. 274-278, ISSN ISSN 1816-9503.
10. **Maheswari KG** & Anita R 2015 "A dynamic tool for detection of xss attacks in a real time environment" **ARPN Journal of Engineering and Applied Science**, vol. 10, no. 10, pp. 4627- 4634, ISSN 1819-6608.
11. Nalinipriya, **K.G. Maheswari**, Balamurugan Balusamy , K. Kotteswari and Arun Kumar Sangaiah , "Availability modeling for multi-tier cloud environment" **IntellIgent AutomAtIon & Soft ComputIng,** Vol. 23, no. 3, PP. 485–492, ISSN: 1079-8587, 2017.

12. G. Nalinipriya, P.J.Varalakshmi, K.G.Maheswari, R. Anita, ‘An Extensive Survey on Co- Resident Attack in Dynamic Cloud Computing Environment” published in **International Journal of Applied Engineering Research** , ISSN 0973-4562 , Volume 11, Number 5 , pp 3019-3023 , **2016.**