International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 03 | Mar -2017 www.irjet.net p-ISSN: 2395-0072

PROFITABLE ITEMSET MINING USING WEIGHTS

T.Lakshmi Surekha[1], Ch.Srilekha[2], G.Madhuri[3], Ch.Sujitha[4], G.Kusumanjali[5]

[1]Assistant Professor, Department of IT, VR Siddhartha Engineering College, Andhra Pradesh, India.
[2]Student, Department of IT, VR Siddhartha Engineering College, Andhra Pradesh, India.
[3]Student, Department of IT, VR Siddhartha Engineering College, Andhra Pradesh, India.
[4]Student, Department of IT, VR Siddhartha Engineering College, Andhra Pradesh, India.
[5]Student, Department of IT, VR Siddhartha Engineering College, Andhra Pradesh, India.

Abstract - *In recent years, a number of association rule mining algorithms like Apriori and FP- Growth were developed. But they are purely binary in nature. They do not consider quantity and profit (profit per unit). In these algorithms, two important measures viz., support count and confidence were used to generate the frequent item sets and their corresponding association rules. But in reality, these two measures are not sufficient for decision making in terms of profitability. In this a weighted frame work has been discussed by taking into account the profit (intensity of the item) and the quantity of each item in each transaction of the given dataset. Apriori and FP Growth algorithms are the best algorithms to generate frequent item sets, but they do not consider the profit as well as the quantity of items in the transactions of the database. Here we propose to new algorithms Profitable Apriori and Profitable FP Growth in our project which eliminate the disadvantages of traditional association rule mining algorithms and they also consider quantity and profit per unit. In this by incorporating the profit per unit and quantity measures we generate the most Profitable Itemsets and we compare the results obtained by Profitable Apriori and Profitable FP-Growth.*

Key Words*: *Profit, Quantity, Profitable Item sets, Profitable Apriori, Profitable FP Growth.

1. INTRODUCTION

Mining frequent patterns or Itemsets is an important issue in the field of data mining due to its wide applications. Traditional Itemset mining is, however, done based on parameters like support and confidence. The most widely used algorithms to obtain frequent Itemsets are Apriori and Frequent pattern growth. They are binary in nature. It means that they only consider whether the product is sold or not. If the product is sold, then it is considered true and else false. And these algorithms produce frequent itemsets, which only consider the occurrence of items but do not reflect any other factors, such as price or profit. Profitable Itemset Mining has recently been proposed, in which transactions are attached with weighted values according to some criteria. It is important because if support and confidence are only the parameters assumed, we may miss some of the profitable patterns.. However, the actual significance of an Itemset cannot be easily recognized if we do not consider some of the aspects like quantity and profit per each item.. The problem of Profitable Itemset mining is to find the complete set of Itemsets satisfying a minimum profit constraint in the database. When we are calculating the Profitable Itemsets we can consider minimum weight as constraint and we can ignore the support as our goal is to find the Profitable patterns. In the real world, there several applications where specific patterns and items have more importance or priority than the other patterns. Profitable Itemset mining has been suggested to find Profitable patterns by considering the profits as well as quantity of Items. The concept of Profitable Itemset mining is attractive in that profitable patterns are discovered. We can use the term, Profitable Itemset to represent a set of profitable items. The Itemsets we get are frequent profitable itemsets as well as infrequent profitable itemsets.

1.1 BASIC CONCEPTS

Itemset mining helps us to find the frequent patterns or itemsets . The two most widely used algorithms are Apriori and FP Growth. These two algorithms are binary in nature. They concerned about whether the product is sold or not. The measures considered by these algorithms are support and confidence. But in reality they are not sufficient for decision making in the large organizations. So In this framework we consider two measures named Quantity and Profit. By using both the parameters we calculate Weight. Consider the following two transactions:
T1: {20 Buns, 5 Chocolates}
T2: {1 Bun, 1 Chocolate}
In the support-confidence frame work the above two transactions are considered to be the same, since the quantity of an item is not taken into account. But in reality, it is quite clear that the transaction T1 gives more profit than the transaction T2. Thus to make efficient marketing we take in to account the quantity of each item in each transaction. In addition we also consider the intensity of each item, which is represented using profit per item p.
Consider the following two transactions:
T3: {10 Buns, 1 Chocolate}
T4: {2 Buns, 3 Chocolates}
In reality the quantity sold in transaction T3 is greater than transaction T4, but the amount of profit gained by selling a chocolate (Say Dairy milk) is 10 times that of a Bun. So, the profit is also given priority represented by p. “p” may represent the retail price / profit per unit of an item.

1.2 PROBLEM DEFINITION

In this paper taking into account the profit / intensity of the item and the quantity of each item in each transaction of the given database, we propose two algorithms Profitable Apriori and Profitable FP Growth. These algorithms take into account two parameters Quantity and Profit. In this by incorporating the profit per item and quantity we generate Profitable Apriori and Profitable FP Growth. In this paper, the notations corresponding to the new structure is given below.
Universal Itemset = I
I = { i1 :q1*p1 , i2 : q2*p2 , . . . , im :qm* pm } ,
{ i1 , i2 , . . . , im } represents the items purchased,
{q1,q2,q3.....qm} represents quantity of purchase,
{p1,p2,p3.....pm} represents their respective profits.
The ith transaction of the database D is of the form
Ti={wi_r,....}
r represents item number,
w = q*p.

2. PROPOSED WORK

In this paper we propose two algorithms Profitable Apriori and Profitable FP Growth. These two algorithms consider quantity as well as profit values per each unit and calculate weights. And the results of both the algorithms are compared.

2.1 PROFITABLE APRIORI ALGORITHM

This algorithm is based on one of the traditional Itemset mining algorithms namely Apriori. Here we propose an algorithm Profitable Apriori, which extends the Apriori algorithm by incorporating the weight and quantity measures and generates Profitable Itemsets**.** The rules are filtered based on a new measure called Minimum Profit, and then prioritized. Profitable Apriori develops an interface between the itemsets and customer characteristics for efficient target marketing.

2.1.1 ALGORITHM

It involves two steps.

• Join Step: Ck is generated by joining Lk-1with itself

• Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

1. *C*1 = {1- item candidate sets};
2. Calculate the support count and the weight of each candidate set in *C*1
3. *L*1 = {1- item frequent sets };
4. *M*1 = {1- item semi-frequent sets};
5. **for** (*k* = 2; *L*k-1 !=∅; *k*++) **do begin**
6. *C*k = candidates generated from *L*k-1; i.e., C k = *L*k-1 *X L*k- 1 *;*
7. **for each** transaction *t* in database D do
8. increment the count of all candidates in *C*k that are contained in *t;*
9. Calculate the support count and the weight of each candidate set in *C*k;
10. **end**
11. Lk = {k-item freq.sets};
12. *M*k = { k-item semi-frequent sets};
13. Arrange all freq.sets in Lk in the descending order of their weights.
14. Arrange all candidate sets in Mk in the descending order of their weights.
15. **end**
16. **return** ∪k *L*k and ∪k Mk

2.2 PROFITABLE FP GROWTH

Profitable FP-Growth algorithm is based on FP-Growth algorithm in data mining. Generally FP-Growth algorithm is binary in nature. It doesn't consider quantity and weight per unit in a transaction. So In our algorithm we consider quantities and profits per items in a transaction.

2.2.1 ALGORITHM

Input:
1. D, a Transactional database that also
Includes quantity of items purchased.
2. Min_sup, the minimum support count threshold.
3. Profit table that displays profit earned by each
Item.
Output:
The complete set of profitable patterns.
Method:

Step1:

1. Scan the database and develop a new table by multiplying profit of each item with their corresponding quantity, represented as weight.
2. Calculate the frequency for each item i.e., sum of Weights of each item in all the transactions.
3. Discard infrequent items.
4. Sort frequent items in decreasing order based on their support and a new database table is generated.
Use this order when building the FP-Tree, so common prefixes can be shared.

Step2:

Nodes correspond to items and have a counter
1. FP-Growth reads the first transaction at a time and maps it to a path.
2. Fixed order is used, so paths can overlap when transactions share items (when they have the same prefix). In this case, counters are incremented by their corresponding weights.
3. Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines). The more paths that overlap, the higher the compression. FP-tree may fit in memory.
4. Frequent itemsets extracted from the FPTree.

3. RESULTS AND OBSERVATIONS

Quantity dataset:

DAIRY DATASET (QUANTITIES)

TID	MILK(lts)	CURD(gms)	BASUNDI(gms)	FRESH CREAM(gms)	BUTTERMILK(lts)	KOVA(gms)
1	1	0	1	0	0	0
2	0	0	2	0	1	1
3	1	1	0	2	1	0
4	0	0	0	1	0	2
5	2	2	3	0	2	2
6	0	3	2	2	0	0
7	3	0	0	0	3	0
8	0	1	2	1	0	3
9	0	2	0	0	0	0
10	1	0	1	0	0	2
11	0	3	2	2	1	1
12	2	0	0	1	2	0
13	0	2	3	0	0	2
14	3	0	2	3	3	0
15	0	1	0	0	2	1
16	0	0	0	2	0	2
17	2	2	1	0	0	0
18	0	2	0	0	1	3
19	1	1	2	1	2	0
20	0	0	0	1	0	2
21	0	0	3	0	0	1
22	2	3	0	0	3	0

Fig 1: Quantity values of items.

Profit dataset:

DAIRY DATASET(PROFITS)

ITEM ID	ITEM	PROFIT/UNIT
1	MILK	5
2	CURD	2
3	BASUNDI	1
4	FRESH CREAM	4
5	BUTTER MILK	3
6	KOVA	4

Fig 2: Profit values of items.

The quantities and profits are multiplied in order to get weights. The weights (total profits) are obtained.

Weighted dataset:

Problems | Javadoc | Declaration | Console

<terminated> Weights1 [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\j

```
The Weights table is
I1      I2      I3      I4      I5      I6
5       0       1       0       0       0
0       0       2       0       3       4
5       2       0       8       3       0
0       0       0       4       0       8
10      4       3       0       6       8
0       6       2       8       0       0
15      0       0       0       9       0
0       2       2       4       0       12
0       4       0       0       0       0
5       0       1       0       0       8
0       6       2       8       3       4
10      0       0       4       6       0
0       4       3       0       0       8
15      0       2       12      9       0
0       2       0       0       6       4
0       0       0       8       0       8
10      4       1       0       0       0
0       4       0       0       3       12
5       2       2       4       6       0
0       0       0       4       0       8
0       0       3       0       0       4
10      6       0       0       9       0
0       0       2       0       6       0
15      4       0       8       0       12
0       0       3       0       0       8
10      2       3       12      3       0
0       0       0       0       0       8
```

Fig 3: Weights of items.

Input Data Format:

input.txt

```
1 3:6:5 1
3 5 6:9:2 3 4
1 2 4 5:18:5 2 8 3
4 6:12:4 8
1 2 3 5 6:31:10 4 3 6 8
2 3 4:16:6 2 8
1 5:24:15 9
2 3 4 6:20:2 2 4 12
2:4:4
1 3 6:14:5 1 8
2 3 4 5 6:23:6 2 8 3 4
1 4 5:20:10 4 6
2 3 6:15:4 3 8
1 3 4 5:38:15 2 12 9
2 5 6:12:2 6 4
4 6:16:8 8
1 2 3:15:10 4 1
2 5 6:19:4 3 12
1 2 3 4 5:19:5 2 2 4 6
4 6:12:4 8
3 6:7:3 4
1 2 5:25:10 6 9
3 5:8:2 6
1 2 4 6:39:15 4 8 12
3 6:11:3 8
1 2 3 4 5:30:10 2 3 12 3
6:8:8
1 2 3 4 5 6:46:15 4 1 8 6 12
3 4 5 6:29:2 12 3 12
2 6:14:2 12
```

Fig 4: Input data format

The Input for the program is given in the above format. For example consider the first transaction.

1 1 3:6:5 1

This means that there are two items in the transaction. They are I1(Item 1) and I3(Item 3). And the corresponding profits obtained by purchasing those items are given after the second colon. Those values are 5 and 1. So the profit obtained by I1 is 5 and I2 is 1. The total profit of the transaction is taken in the middle i.e In between the colons.

Patterns obtained for Profitable Apriori:

ProfitableApriori.java | output.txt

```
1 PATTERN   : 1  TOTAL PROFIT: 530
2 PATTERN   : 4  TOTAL PROFIT: 392
3 PATTERN   : 5  TOTAL PROFIT: 342
4 PATTERN   : 6  TOTAL PROFIT: 508
5 PATTERN   : 1 2  TOTAL PROFIT: 308
6 PATTERN   : 1 3  TOTAL PROFIT: 427
7 PATTERN   : 1 4  TOTAL PROFIT: 477
8 PATTERN   : 1 5  TOTAL PROFIT: 601
9 PATTERN   : 1 6  TOTAL PROFIT: 530
10 PATTERN   : 2 4  TOTAL PROFIT: 328
11 PATTERN   : 2 6  TOTAL PROFIT: 380
12 PATTERN   : 3 5  TOTAL PROFIT: 301
13 PATTERN   : 3 6  TOTAL PROFIT: 392
14 PATTERN   : 4 5  TOTAL PROFIT: 463
15 PATTERN   : 4 6  TOTAL PROFIT: 524
16 PATTERN   : 5 6  TOTAL PROFIT: 476
17 PATTERN   : 1 3 4  TOTAL PROFIT: 371
18 PATTERN   : 1 3 5  TOTAL PROFIT: 520
19 PATTERN   : 1 3 6  TOTAL PROFIT: 484
20 PATTERN   : 1 4 5  TOTAL PROFIT: 511
```

Fig 5: Profitable Patterns(Profitable Apriori)

The above figure is the output for Profitable Apriori algorithm. The output in the format

1 **PATTERN : 1 TOTAL PROFIT :530**

This means that the profit of the Item 1 in the entire database is 530.

5 **PATTERN : 1 2 TOTAL PROFIT :308**

This means that the profit of the Items 1 and 5 in the entire database is 308.

Patterns obtained for Profitable FP Growth:

ProfitableApriori.java | output.txt

```
15 PATTERN   : 4 6  TOTAL PROFIT: 524
16 PATTERN   : 5 6  TOTAL PROFIT: 476
17 PATTERN   : 1 3 4  TOTAL PROFIT: 371
18 PATTERN   : 1 3 5  TOTAL PROFIT: 520
19 PATTERN   : 1 3 6  TOTAL PROFIT: 484
20 PATTERN   : 1 4 5  TOTAL PROFIT: 511
21 PATTERN   : 1 4 6  TOTAL PROFIT: 395
22 PATTERN   : 1 5 6  TOTAL PROFIT: 530
23 PATTERN   : 2 4 6  TOTAL PROFIT: 370
24 PATTERN   : 3 4 5  TOTAL PROFIT: 397
25 PATTERN   : 3 4 6  TOTAL PROFIT: 372
26 PATTERN   : 3 5 6  TOTAL PROFIT: 432
27 PATTERN   : 4 5 6  TOTAL PROFIT: 491
28 PATTERN   : 1 3 4 5  TOTAL PROFIT: 481
29 PATTERN   : 1 3 4 6  TOTAL PROFIT: 357
30 PATTERN   : 1 3 5 6  TOTAL PROFIT: 548
31 PATTERN   : 1 4 5 6  TOTAL PROFIT: 417
32 PATTERN   : 3 4 5 6  TOTAL PROFIT: 445
33 PATTERN   : 1 3 4 5 6  TOTAL PROFIT: 441
34
```

Fig 6: Profitable Patterns (Profitable FP Growth)

Profitable FP Growth and Profitable Apriori gives the same patterns.

3.1 STATISTICS

Profitable apriori statistics

Problems | Javadoc | Declaration | Console

```
<terminated> ProfitableApriori [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (16-Mar-
============ PROFITABLE APRIORI STATISTICS ============

 TOTAL TIME TAKEN ~ 31 ms
 MEMORY USAGE ~ 1.280242919921875MB
 CANDITATES COUNT : 52
 PROFITABLE PATTERNS COUNT : 33

================================================
```

Fig 7: Statistics (Profitable Apriori)

So, the above figure shows the statistics of Profitable Apriori algorithm. The total time taken by the algorithm to produce the results is 31ms. The memory used by the algorithm to store the patterns in the text file is 1.28MB. The number of patterns produced by the algorithm which satisfy the minimum weight constraint are 33.

Profitable FP Growth statistics

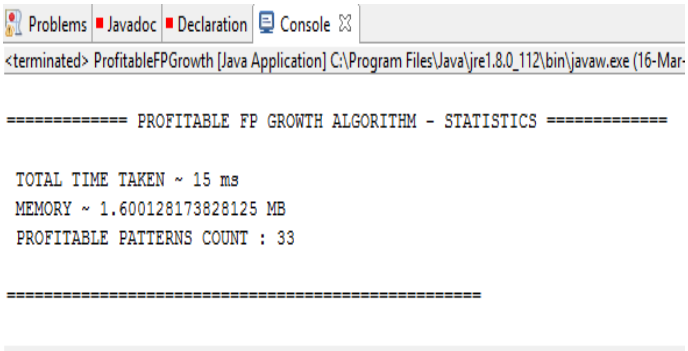

Fig 8: Statistics (Profitable FP Growth)

So, the above figure shows the statistics of Profitable FP Growth algorithm. The total time taken by the algorithm to produce the results is 15ms. The memory used by the algorithm to store the patterns in the text file is 1.60MB. The number of patterns produced by the algorithm which satisfy the minimum weight constraint are 33.

4. OUTPUT ANALYSIS

The results of both the algorithms are compared using charts in Microsoft Excel.

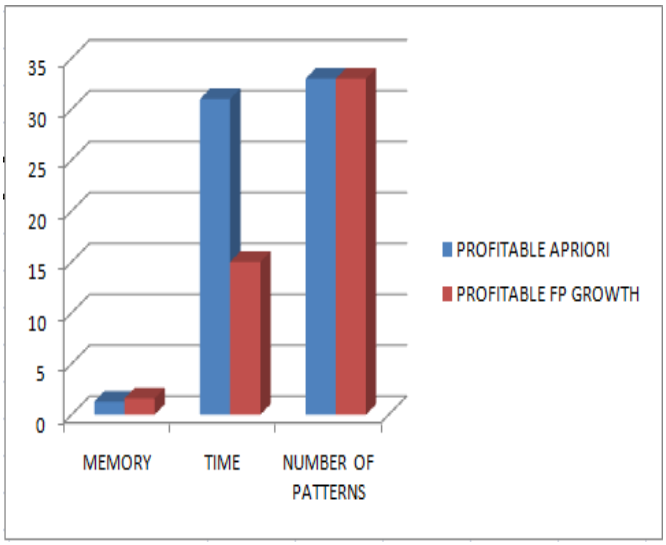

Fig 9: Comparative result

The memory taken by the Profitable FP Growth is more than Profitable Apriori since FP Tree takes more space. The time taken by the Profitable Apriori is more since candidate keys are generated in Profitable Apriori. The number of patterns produced by both the algorithms are same and similar.

5. CONCLUSIONS

The profit and quantity factor given to each item in each transaction is used to find the weight of each frequent set. The weights are obtained by multiplying profits and quantities. The marketers are more interested in the profit than the frequency of a set. So, we obtained profitable Patterns using Profitable Apriori and Profitable FP-Growth algorithms. Results of both the algorithms are compared based on the parameters Memory usage, Time taken to produce the frequent patterns and the number of frequent patterns.

REFERENCES

[1] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases", Proc. 1993 ACM SIGMOD,Washington, DC, pp. 207–216, May 1993

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. 20th International Conference on Very Large Databases (VLDB"94), Santiago, Chile, pp. 487–499, Sept. 1994

[3] J. Han and Micheline Kamber, "Data Mining – Concepts and Techniques", Morgan Kaufmann Publishers, 2001

[4] Robert J. Hilderman, Colin L. Carter, Howard J. Hamilton, And Nick Cercone, "Mining Association Rules From Market Basket Data Using Share Measures and characterized Itemsets"

[5] Feng Tao, Fionn Murtagh, Mohsen Farid, "Weighted Association Rule Mining using Weight ed Support and Significance framework"

[6] Wang, B-Y., Zhang, S-M.: A Mining Algorithm for Fuzzy Weighted Association Rules. In: IEEE Conference on Machine Learning and Cybernetics, 4, pp. 2495--2499 (2003).

[7] Unil Yun, John J. Leggett, 'WFIM: Weighted Frequent Itemset Mining with a weight range and a minimum weight" SDM`05, April 2005.

[8] J.Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.

[9] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, Jan., 2008.

[10] Y. Liu, W. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.

[11] B.-E. Shie, V. S. Tseng, and P. S. Yu. Online mining of temporal maximal utility itemsets from data streams. In Proc. of the 25th Annual ACM Symposium on Applied Computing, Switzerland, Mar., 2010.

[12] R.Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[13] K. Elissa, "Title of paper if known," unpublished.