IJCAT - International Journal of Computing and Technology, Volume 5, Issue 3, March 2018
ISSN (Online) : 2348-6090
www.IJCAT.org
Impact Factor: 0.835

Review of High Utility Pattern Mining Algorithms Focused On Memory Utilization

[1] Shilpa Ghode

[1] Asst. Prof. In Computer Technology Department
Kavikulguru Institute of Technology and Science, Ramtek

Abstract - Discovering interesting patterns and useful knowledge from massive data has become an important data mining task. These days, we come across a lot of things that have profit technically referred as external utility, value greater than the other item sets in the database. Utility mining is an important topic in data mining and has received extensive research in last few years. In utility mining, each item is associated with a utility that could be profit, quantity, cost or other user preferences. Objective of Utility Mining is to identify the item sets with highest utilities. High utility itemset mining is an extension to the problem of frequent pattern mining. Many algorithms have been proposed in this field in the recent years. In this paper we emphasis on an emerging area called High Utility Mining which not only considers the frequency of the itemsets but also considers the utility associated with the itemsets. In High Utility Itemset Mining the target is to identify itemsets that have utility value greater than the threshold utility value. In this paper we present a review of the various techniques and current scenario of research in mining high utility itemset also presented advantages and limitations of various techniques for High Utility Itemset Mining. We mainly focus on the D2HUP and MAHUSP approach and algorithms for high utility pattern mining with less memory utilization.

Keywords *- Data mining, Frequent Patterns, High Utility Pattern Mining, High Utility Itemsets, High Utility Mining Algorithms*

1. Introduction

Data mining is basically extracting or mining the knowledge from large amount of data. The term data mining is appropriately named as "Knowledge mining". Interestingness measures play an important role in knowledge discovery and finding interesting patterns is essential for variety of applications such as genome analysis, condition monitoring, cross marketing, and inventory prediction[11]. There are two general classification of Data Mining: Descriptive Mining and Predictive Mining. Clustering, Association Rule Discovery, Sequential Pattern Discovery are the Descriptive Mining techniques. They are used to find human-interpretable patterns that describe the data. Classification, Regression, Deviation Detection, use some variables to predict unknown or future values of other variables are Predictive Mining techniques. The important areas in research is Association Rule Mining (ARM) in data mining. It is a prominent part of Knowledge Discovery in Databases (KDD).That's why it requires more concentration to explore. Association rule mining (ARM) is a technique for discovering co-occurrences, correlations, and frequent patterns, associations among items in a set of transactions or a database. We find rules having confidence and support above user defined threshold.

In frequent pattern mining a pattern is regarded as interesting if its occurrence frequency exceeds a user-specified threshold. However, a user's interest may relate to many factors that are not necessarily expressed in terms of the occurrence frequency. Frequent pattern mining is used to discover all patterns whose supports are no less than a user-defined minimum support threshold[2].Frequent pattern mining employs the anti-monotonicity property. However discovering combinations of products with high profits relates to the unit profits and purchased quantities of products that are not considered in frequent pattern mining. But it has some important limitations when it comes to analyzing customer transactions. An important limitation is that purchase quantities are not taken into account. Thus, an item may only appear once or zero time in a transaction. Thus, if a customer has bought five breads, ten breads or twenty breads, it is viewed as the same. A second important limitation is that all items are seen as having the same importance, utility of weight. For example, if a customer buys a very expensive bottle of wine or just a piece of bread, it is seen as being equally important. Thus, frequent pattern mining may find many frequent patterns that are not interesting[22].

Utility mining has been emerged to address the limitation of frequent pattern mining by considering the user's expectation as well as the raw data. Utility itemset mining, also generally called utility pattern mining. Among utility mining problems, utility mining with the itemset framework is a difficult one, since no anti–monotonicity property holds with interestingness measure and also the downward closure property in frequent itemset mining does not hold in utility itemsets. Thus utility

mining with the itemset share framework is more challenging than the other categories of utility mining which include weighted itemset mining, objective-oriented utility based association mining and frequent pattern mining. Every item in the itemsets is associated with an additional value, called internal utility which is the quantity of the item and an external utility is attached to an item, showing its quality. The utility is a measure of how useful or profitable an item set X is. The utility of an item set X, i.e., u(X), is the sum of the utilities of item set X in all the transactions containing X. An item set X is called a high utility item set if and only if u(X) >= min_utility, where min_utility is a user-defined minimum utility threshold. Frequent item set mining follows the download closure property, if K - item set is generated, K+1 item set can be generated by considering only K-item set or in other words K+1 item set will contain only the item set present in the K-item set. This downward closure property is not satisfied, if k-item set is low utility item set K+1 can be high utility item set and vice versa. Both the monotonic and antimonotonic property is not supported by high utility item set. This issue is addressed by the overestimation method [6].The various algorithms for mining frequent patterns as well as algorithms for mining high utility patterns fall into three categories such as breadth-first search, depth first search, and hybrid search. This research paper presents a literature survey of the various approaches and algorithms for high utility patterns[11].

We mainly focus on, D2HUP algorithm, which is used for utility mining with the itemset share framework. D2HUP employs several techniques proposed for mining frequent patterns, including exploring a regular set enumeration in a reverse lexicographic order and heuristics for ordering items[11].
On other hand we have MAHUP algorithm which efficiently discovers HUSPs over a data stream with a high recall and precision. The proposed method guarantees that the memory constraint is satisfied and also all true HUSPs are maintained in the tree under certain circumstances[25].

2. Literature Survey Of High Utility Mining Algorithms

High utility mining is the challenging research area in data mining which mainly concentrates on high utility pattern growth mechanism. Several research works have been proposed to meet this challenge. The various proposed algorithms for mining high utility patterns are presented as follows.

Liu et al., [15] proposed **Pseudo Projection** algorithm which is fundamentally different from those proposed in the past. This algorithm uses two different structures such as array based and tree-based to represent projected transaction subsets and heuristically decides to build unfiltered pseudo projection to make a filtered copy according to features of the subsets. This work build tree-based pseudo projections and array-based unfiltered projections has been build for projected transaction subsets which makes algorithm both CPU time efficient and memory saving. This algorithm grows the frequent itemset tree by depth first search, where as breadth first search is used to build the upper portion of the tree if necessary. This algorithm is tested on real world datasets, such as BMS-POS, and on IBM artificial datasets. This algorithm is not only efficient on sparse and dense databases at all levels of support threshold and also highly scalable to very large databases. The disadvantage of this algorithm is, it only support minimum description code length with small number of patterns.

Han et al., [10] proposed **a Frequent Pattern Growth (FP-Growth)** algorithm for mining frequent pattern with constraints. In this work the frequent pattern tree (FP-tree) structure which is an extended prefix tree structure developed for storing crucial information about frequent patterns. The pattern fragment growth mines the complete set of frequent patterns using the FP-growth. This algorithm constructs a highly compact FP-tree and applies a pattern growth method for database scans which is usually substantially smaller than the original database by which costly database scans are saved in the subsequent mining processes. The disadvantage of this algorithm is it reduces multipass candidate generation process in the first phase by discarding isolated items to reduce the number of candidates .Also this work shrink the database scanned in each pass and it takes more computation time.

Liu et al., [16] proposed a **Two-phase** algorithm to find high utility itemsets. This algorithm efficiently prunes down the number of candidates and obtains the complete set of high utility itemsets. This work has developed with two phases. Phase one uses transaction-weighted downward closure property which is applied to add high transaction weighted utilization sets during the level wise search. In phase two, any over estimated low utility itemsets are filtered using an extra database scan. This algorithm requires fewer database scans, less memory space and less computational cost for large databases and performs very well in terms of speed and memory cost on both synthetic and real database. The main disadvantage of this algorithm is, the insufficient frequent counts for repeated candidate itemset which can lose interesting patterns

Li et al., [17] proposed an **Isolated Items Discarding Strategy (IIDS)** algorithm for utility mining. This algorithm discovered high utility itemset with less number of candidates which improve the performance of the pattern mining. This algorithm shows that itemset share mining problem can be directly converted to utility mining problem by replacing the frequent values of each items in a transaction by its total profit, i.e., multiplying the frequency value by its unit profit. In this work the share frequent set mining scans the database to calculate the share value of each itemset and removes all useless candidate itemsets and remaining candidate to generate. The direct condition generation is a level wise method and it maintains an array for each candidate during each pass. This algorithm provides an efficient way to designed critical operations by using transaction weighted downward closure. However this algorithm still suffers with the problem of level wise generation

and test problem of apriori and it requires multiple database scans.

Erwin et al., [7] proposed a **Transaction Weighted Utility (TWU)** algorithm which is based on compact utility pattern tree data structures. This work implements the parallel projection scheme to utilize the disk storage. This algorithm first identifies the TWU items from transaction database and the compressed utility pattern tree is constructed for mining complete set of high utility patterns. In this algorithm parallel projection is used to create subdivision for subsequently mining. This algorithm has anti-monotone property which is used to discover the pruning space. In this work the task of high utility itemset mining discovers all the utility which has utility higher than the user specified utility. Generation of frequent graphs results in high in memory usage and low in accuracy.

Shankar et al., [19] proposed a **Fast Utility Mining (FUM)** algorithm that finds all high utility itemset within the given utility constraint threshold. It is faster and simpler than the original UMining algorithm. This algorithm efficiently handles the duplicate itemsets. It checks whether a transaction defined by an itemset purchased in it, repeats its occurrence in a later transaction. If a later transaction also contains same itemset purchased in any of the previous transactions, then that transaction is ignored from processing and duplicate itemset are removed. This reduces the execution time of the algorithm further more. This algorithm provides absolute accuracy and proves to be extremely efficient in finding every possible high utility itemset from the transactions in the database. This algorithm executes transaction datasets exceptionally faster when more itemset are identified as high utility itemset and when the number of distinct items in the database increases.

Ahmed et al., [1] proposed **a Tree-based Incremental High Utility Pattern Mining (IHUPM)** algorithm. In this work a tree based structure called IHUP-Tree which is used to maintain the information about itemsets and their utilities. This work proposes three tree structures to perform incremental and interactive high utility pattern mining efficiently. This reduces the calculations when a minimum threshold is changed or a database is updated. The first tree structure is an incremental high utility pattern lexicographic tree (IHUPLTree) that is arranged according to an item's lexicographic order. It can capture the incremental data without any restructuring operation. The second tree structure is the incremental high utility pattern transaction frequency tree (IHUPTF-Tree) which is simple and easy to construct and handle. In this tree the items are arranged according to their transaction frequency. It does not require any restructuring operation even when the data base is incrementally updated. They have achieved the less memory consumption. The third tree structure is the incremental high utility pattern transaction weighted utilization tree (IHUPTWU-Tree). This tree is based on the transaction weighted utility value of items in descending order. This algorithm takes insufficient `memory usage and outperforms with earlier lexographical approaches.

Tseng et al., [21] proposed an efficient algorithm called as U**tility Pattern Growth Plus (UP-Growth+)** which is an improved version of utility pattern growth (UP-Growth) mining algorithm. In this work the information of high utility itemset is maintained in a special data structure named utility pattern tree (UP-Tree) and the candidate itemsets are generated with one scans of the database. The four strategies, applied in this algorithm are discarding global unpromising items (DGU), decreasing global node utilities (DGN), discarding local unpromising items (DLU), and decreasing local node utilities (DLN). By these strategies, the estimated utilities of candidates are well reduced, by discarding the utilities of the items which are impossible to be high utility or not involved in the search space. The proposed strategies not only decrease the estimated utilities of the potential high utility itemsets but also reduce the number of candidates. This algorithm outperforms substantially in terms of execution time, especially when the database contains lots of long transactions. However the operation time and search space of high-utility itemset mining can increase the high computation cost.

Liu et al., [14] proposed a H**igh Utility Itemset Miner (HUI-Miner)** for high utility itemset mining. This algorithm uses a novel structure called utility-list which is used to store both the utility information about an itemset and the heuristic information for pruning the search space. This algorithm first creates an initial utility list for itemsets of the length 1 for promising items. This algorithm constructs recursively a utility list for each itemset of the length k using a pair of utility lists for itemset of the length k-1 for mining high utility itemset, each utility list for an itemset keeps the information of indicates transaction for all of transactions containing the itemset, utility values of the item setin the transactions, and the sum of utilities of the remaining items that can be included to super itemset of the itemset inthe transactions. This algorithm first estimate the utilities of the itemsets and generate the candidate itemsets and thenby scanning the database compute the exact utilities of the itemset to generate the high utility itemset. This algorithm mines the high utility itemset without generation of the candidates and the algorithm outperforms in terms of both running time and memory consumption.

Fournier-Viger et al., [8] proposed an algorithm F**ast High Utility Miner (FHM)** which extends the high utility itemset miner (HUI-Miner) algorithm. It is a depth-first search algorithm that relies on utility-lists to calculate the exact utility of itemsets. This algorithm consists of discovering frequent itemset that is groups of itemsets appearing frequently in transactions. This work integrates a novel strategy named estimated utility co-occurrence pruning (EUCP) to reduce the number of joins operations when mining high utility itemset using the utility list data structure. The estimated utility co-occurrence pruning structure (EUCP) stores the transaction weighted utility of all itemsets. It built during the initial database scans. The memory footprint of the estimate utility co-occurrence pruning structure is small. This algorithm is up to 6 times faster than high utility itemset miner. The proposing a

novel strategy based on the analysis of item co-occurrences to reduce the number of join operations that need to be performed. An important limitation of this algorithm is it assumes that each item cannot appear more than once in each transaction and that all items have the same importance.

Junqiang Liu et al., [11] proposed an algorithm D**irect Discovery High Utility Pattern (D2HUP)** which gains the combination of high utility pattern miner and utility pattern. This algorithm mines utility itemset in share framework. The direct discovery of high utility patterns, which is an integration of the depth-first search of the reverse set enumeration tree. This algorithm addresses the scalability and efficiency issues occurred in the existing systems as it directly extracts the high utility patterns from large transactional databases. This algorithm is based on the powerful pruning approaches. The look ahead strategy tries to find the patterns in recursive enumeration and it utilizes the singleton and closure property to enhance the efficiency of dense data. The linear data structure as chain of accurate utility list is used to show the original information of utility in the unrefined data. This work helps to discover the root causes of prior algorithm which employs to maintain data structure information of original utility.

Morteza Zihaya et al.,[25] proposed a **Memory-Adaptive High Utility Sequential Pattern (MAHUP)** approach to discovering HUSPs from a dynamically-increasing data stream. This is the first piece of work to mine high utility sequential patterns over data streams in a memory adaptive manner. In this algorithm, a novel method is proposed for incrementally mining HUSPs over a data stream. This method can not only identify recent HUSPs but also HUSPs over a long period of time (i.e., since the start of the data stream). Then a novel and compact data structure is proposed, called *MAS-Tree*, to store potential HUSPs over a data stream. The tree is updated efficiently once a new potential HUSP is discovered. After that two efficient memory adaptive mechanisms are proposed to deal with the situation when the available memory is not enough to add a new potential HUSPs to *MAS-Tree*. The proposed mechanisms choose the least promising patterns to remove from the tree to guarantee that the memory constraint is satisfied and also that all true HUSPs are maintained in the tree under certain circumstances. Using *MAS-Tree* and the memory adaptive mechanisms, the algorithm called *MAHUSP*, efficiently discovers HUSPs over a data stream with a high recall and precision. The proposed method guarantees that the memory constraint is satisfied and also all true HUSPs are maintained in the tree under certain circumstances.

3. Running Time and Memory Usage

The running time by the five algorithms. For example, for T10I6D1M database with minU ¼ 0:01%, d2HUP takes 27 seconds, HUIMiner 154 UP-GROUTH + 101, IHUPM 109, and TwoPhase runs out of memory. The observations are as follows. First, d2HUP is up to 1 to 3 orders of magnitude more efficient than UP-GROUTH +, IHUPM, and TwoPhase. In particular, d2HUP is up to 6.6, 6.8, 7.8, 261, 472, and 1,502 times faster than UP-GROUTH+ on T20I6D1M, T10I6D1M, Chain-store, WebView-1, Foodmart, and Chess respectively. The reasons are **first**, the number of patterns enumerated by d2HUP is much smaller than the number of candidates generated by UP-GROUTH +, IHUPM, and TwoPhase. Thus, d2HUP even takes less time than the first phase of the latter algorithms. The latter materialize candidates and need a second phase to match each candidate with transactions in the database, which causes scalability issue when the number of candidates is large, and has efficiency issue when the database is large.

Second, d2HUP is up to 1 order of magnitude more efficient than HUIMiner [11]. Concretely, d2HUP is up to 6.3, 6.5, 16.8, 45, 49, and 875 times faster than HUIMiner on T10I6D1M, T20I6D1M, Chess, Chain-store, Foodmart, and WebView-1 respectively. The reasons are: d2HUP has stronger pruning and enumerates less patterns than HUIMiner. And d2HUP proposes the data structure, CAUL, that enables efficient computation, while HUIMiner employs inefficient join operations on a vertical data structure, which is also not scalable**.**

Due to a huge number of high utility patterns, the scalability benefit does not excel while the additional computation cost is amplified. A good choice for d2HUP is to make a materialized copy of the pseudo CAUL (f ¼ 1).

Memory usage. In the paper [11] collected the peak memory usage statistics by every algorithm during its execution except TwoPhase. For example, for T10I6D1M with minU ¼ 0:1%, the peak memory usage by d2HUP is 147 MB, and that by HUIMiner, by I UP-GROUTH +, and by IHUPM are 191, 153, and 154 MB respectively. The following is a summary.

1.Our d2HUP algorithm uses the least amount of memory because d2HUP uses CAUL that is more compact than the vertical data structure by HUIMiner, and d2HUP does not materialize candidates in memory while UP-GROUTH +, IHUPM, and TwoPhase do.

2.The memory usage by UP-GROUTH + and IHUPM are 50 percent to 2 orders, and 90 percent to 2 orders of magnitude more than d2HUP respectively. TwoPhase uses the most, and usually runs out of memory when minU is small[11].

D2HUP does not work on high utility sequential pattern mining. MAHUSP works on sequential data stream that employs memory adaptive mechanisms to use a bounded portion of memory, in order to efficiently discover HUSPs over data streams.

MAHUSP tackled the problem of memory adaptive high utility sequential pattern mining over data streams. *MAHUSP* is based on a compressed tree structure and two memory adaptive mechanisms ([*Leaf Based Memory Adaptation* (*LBMA*)] , [*Sub-Tree Based Memory Adaptation* (*SBMA*)]) that can adapt the memory usage to the available memory by pruning the least promising part of the tree when necessary[25].

4. Conclusion

A Utility mining is an apparent topic in data mining. The main focus in the field of Utility Mining is not only Frequent Itemset Mining but also the consideration of utility. Practically it has been found that the utility is of great interest in industry if considers with high utility itemsets. This research paper presents a review of various existing high utility itemset mining algorithms. The reviewed algorithms effectively mining high utility itemsets based on the various data structure and constraint techniques. This will be helpful for developing new efficient and optimize techniques for high utility itemset mining. However to discover patterns for large transactional datasets an effective high utility pattern mining algorithm is required for improving the performance and search space of high utility itemsets. As the concept of High Utility Itemset Mining has a vast opportunities to be researched, the future work will incorporate soft computing methodologies for high utility itmesets mining such as the intuitionistic fuzzy logic can be explored in the field of High Utility Itemset Mining and its memory consumption.

References

[1] Ahmed C. F., Tanbeer S. K., Jeong B.-S., and Lee Y. -K., "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 12, 2009, pp. 1708–1721.

[2] Agrawal R., Imielinski T., and Swami A., "Mining association rules between sets of items in large databases," In Special Interest Group on Knowledge Discovery in Data. Association for Computing Machinery, 1993, pp. 207–216.

[3] Anusmitha A., Renjana Ramachandran M., "Utility pattern mining: a concise and lossless representation using up growth", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, No. 7, 2015, pp. 451– 457.

[4] Chun-Wei Lin J., WenshengGan., Fournier-Viger P., and Yang L., Liu Q., Frnda J., Sevcik L., Voznak M., " High utility itemset-mining and privacy-preserving utility mining," Vol. 7, 2016, No. 11, pp. 74–80.

[5] Dawar S., Goya V. l., "UP - Hist tree: An efficient data structure for mining high utility patterns from transaction databases," In Proceedings of the 19th International Database Engineering & Applications Symposium. Association for Computing Machinery, 2015, pp. 56–61.

[6] De Bie T., "Maximum entropy models and subjective interestingness: an application to tiles in binary databases," Data Mining and Knowledge Discovery, Vol. 23, No. 3, 2011, pp. 407–446.

[7] Erwin A., Gopalan R. P and. Achuthan N. R., "Efficient mining of high utility itemsets from large datasets," In Proceeding of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2008, pp. 554–561.

[8] Fournier-Viger P., Wu C.-W., Zida S., and Tseng V.S., "Fhm: Faster high-utility itemset mining using estimated utility Cooccurrence pruning," In Proceedings of the 21th International Symposium on Methodologies for Intelligent Systems. Springer, 2014, pp.83-92.

[9] Geng L., Hamilton H.J, "Interestingness measures for data mining: A survey," Association for Computing Machinery. Vol. 38, No. 3, 2006, pp.1–9.

[10] Han J., Pei J., Yin Y., Mao R., "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," Data Mining Knowledge Discovery in Data. Vol. 8, No. 1, 2004, pp. 53–87.

[11] Junqiang Liu., Ke Wang., Benjamin., Fung C.M.,"Mining High Utility Patterns in One Phase without Generating Candidates", IEEE Transactions on Knowledge and Data Engineering, Vol. 28, No. 5, 2016, pp.1–14.

[12] Jyothi Pillai., Vyas O.P., "Overview of itemset utility mining and its applications," International Journal of Computer Applications, Vol. 5, No. 11, 2010, pp. 9 –13.

[13] Liu J., Wang K., and Fung B., "Direct discovery of high utility itemsets without candidate generation," In Proceedings of the12th International Conference. IEEE, 2012, pp. 984–989.

[14] Liu M., Qu J., "Mining high utility itemsets without Candidate generation," Conference on Information and Knowledge Management. Association for Computing Machinery, 2012, pp. 55–64.

[15] Liu J., Pan Y., Wang K., and Han J., "Mining frequent item sets by opportunistic projection," In Special Interest Group onKnowledge Discovery and Data Mining. Association for Computing Machinery, 2002, pp.229–238.

[16] Liu V., Liao W., and Choudhary A., "A fast high utility itemsets mining algorithm," in utility – Based Data Mining Workshop in Special Interest Group on Knowledge Discovery in Data. Association for Computing Machinery, 2005, pp. 253 – 262.

[17] Li Y.-C., Yeh J.-S., and Chang C.-C., "Isolated items discarding Strategy for discovering high utility itemsets," Data &Knowledge Engineering, Vol. 64, No. 1, 2008, pp. 198–217.

[18] Sarode, Nutan, and Devendra Gadekar, " A review on efficient algorithms for mining high utility itemsets, "International Journal of Science and Research, Vol. 3, No. 12, 2014, pp.708 –710.

[19] Shankar S., Purusothoman T.P, Jayanthi S., Babu N., "A fast algorithm for mining high utility itemsets" ,In Proceedings of IEEE International Advance Computing Conference (IACC), Patiala, India, 2009, pp.1459-1464.

[20] Tan P.N., Kumar V., and Srivastava J., "Selecting the right objective measure for association analysis," Information Systems,Vol. 29, No. 4, 2004, pp. 293–313.

[21] Tseng V. S., Shie B.-E., Wu C.-W., and Yu P. S., "Efficient algorithms for mining high utility itemsets from transactional databases," IEEE Transactions on Knowledge and Data Engineering, Vol. 25, No. 8, 2013, pp. 1772–1786.

[22] Yao H., Hamilton H. J., Butz C.J., "A foundational approach to mining itemset utilities from databases," ICDM 2004, pp. 482-486.
[23] Yao H., Hamilton H. J., Geng L., "A unified framework for utility-based measures for mining itemsets," in Utility-Based Data Mining," In Special Interest Group on Knowledge Discovery in Data. Association for Computing Machinery, 2006, pp. 28–37.
[24] Zaki M.J., "Scalable algorithms for association mining," IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No.3, 2000, pp. 372–390.
[25] Morteza Zihayat, Yan Chen, Aijun An, "Memory-Adaptive High Utility Sequential Pattern Mining Over Data Stream", Published in Springer Journal on Machine Learning. Vol. 106.Issue 6, 2017, pp.799–836.
[26] Sharayu H. Fukey, Prof. P. M. Chawan ," Survey of High Utility Item sets Mining Algorithms", International Journal of Engineering Development and Research, Vol. 5, Issue 2, 2017, pp. 2321-9939.
[27] Dr. S. Meenakshi1, P. Sharmila, "Review of Mining High Utility Patterns", International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, Issue 8, pp.14055-14061.