Journal of Applied Research and Technology

www.jart.icat.unam.mx

ICAT
Instituto de Ciencias Aplicadas y Tecnología

Journal of Applied Research and Technology 24 (2026) 610-618

Original

An Approach for Association Rule Mining: Hash-Based Reverse a priori using the Masking Technique

Sharayu Bonde* • Dnyaneshwar Kirange

SSBT's COET Bambhori, KBCNMU, Jalgaon, Maharashtra, India

Received: 26 05 2025; Accepted: 12 08 2025
Available: 31 08 2026

Abstract: Consumer buying patterns are a type of purchase made by consumers, whether by an individual or many individuals, to get the desired thing by making a purchase transaction. Apriori algorithm helps to find patterns among purchased items. But for larger transaction datasets, Apriori has high time complexity, as the database is scanned multiple times with the help of expensive resources. This, in turn, impacts the algorithm when the computer memory is inadequate, and there are a voluminous number of frequent transactions. This study aims to create an optimization of Apriori that is used in determining consumer purchasing patterns. The principal objective of this exploration is to make an advanced proposal framework utilizing hash-based Apriori that can assist with investigating the purchasing behavior of clients and, in view of that, suggest the most appropriate items to them as per their needs.

Keywords: **Reverse Apriori, Association Rule mining, Hash-based Apriori, Pattern Matching**

*Corresponding author.
E-mail address: sharayubonde29@gmail.com (S. Bonde).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

1. Introduction

The technology era in which we currently live has enabled business groups to amass vast amounts of information. However, while database technology has been satisfactory in keeping these data stacks robust, it will ultimately be necessary to request information to increase the association's value.

Businesses in the current customer-centric markets must implement sufficient and affordable upgrading procedures that can respond to changes in client perceptions and information needs. Additionally, it might help businesses identify a completely remote market that they might efficiently target. Altogether, for making key decisions on the market strategy, stable, as much as possible, and secure evidence-based information is required. Shaukat et al. (2015) introduced an elaborate review of ARM applications and examined the various parts of the association rules that are utilized to separate the interesting patterns and connections among the set of items in repositories.

One of the most well-known, significant, and scalable methods for mining frequent Itemsets and mining association rules is called Apriori. In 1993, Agrawal and Srikant introduced apriori. The Apriori technique is used to locate all frequently occurring objects included in a specific database. The Apriori algorithm repeatedly examines databases. To search through entries in the database, it employs a breadth-first approach. Numerous sectors employ the Apriori algorithm for transactional operations, and it may also be used in real-time applications in places like shopping malls, corner stores, and grocery stores by compiling customer purchases over time to create frequent items.

2. Literature Survey

Agarwal et al. (2013) studied the application of the Apriori algorithm in a grocery shop. Additionally, researchers have examined market basket analysis data mining methods (Phani & Murlidher, 2013; Dhanabhakyam & Punithavalli, 2011). Bartik (2009) showed the use of association-based classification for relational data in an online environment.

Sumithra and Paul (2010) presented a distributed Apriori association rule mining and traditional Apriori mining algorithms for grid-based knowledge discovery. Qiang and others, Niu et al. (2009). In their work, they present a classification-based strategy based on rule compactness; however, it has the drawback of overfitting because the classification rules with the least support and confidence are returned to the classifier as strong association rules. The study's authors showed how association rule mining may be used to leverage market basket analysis for business intelligence Abdulsalam (2014).

3. Related Work

Apriori: There are two steps that make up the overall algorithm:
Step 1: Apply minimum support to search a database for all frequent sets with k items.
Step 2: With the use of frequent N-Itemsets, apply the self-join rule to identify the frequent sets with N+1 items.

Continue in this manner from N=1 until the self-join rule cannot be applied. The "bottom up" strategy entails extending a frequent itemset one at a time.

The orthodox working of the Apriori Algorithm as studied is a simple rule-based model that works as follows:

I. Check the Database.
II. Collect each row's data in a variable.
III. For each next row, check if the entries in the previous and current rows match; add such items to a new set.
IV. The Threshold is maintained by support and confidence
V. Now such candidate sets are generated and are formed into rules.

Limitations: Apriori's limitations are, for instance, the assumption that a frequent-1 itemset has 10^4 items from the set. More than 10^7 candidates of length 2 must be produced by the Apriori algorithm code, which will then test and accumulate the results. The technique generates 2^100 potential itemsets or candidates in order to find a size-frequent pattern of size 100 (with v1, v2,..., v100).

Apriori algorithm has higher time complexity for the following reasons.

- generating combinations of items at each candidate set generation step. Therefore, we have scope for improvisation of this complexity by generating the combinations with much more efficient complexity
- The database is scanned repeatedly, utilizing expensive resources. When the system memory is low and there are a lot of frequent transactions, this in turn affects the algorithm. Because of this, the technique is slow and ineffective when dealing with huge databases.

Reverse Apriori Algorithm: Reverse Apriori starts by taking into account the maximum combination of all the

values in pairs and then builds big frequent item sets. It is effective in cases where it is obvious to identify these combinations by quickly scanning the dataset and establishing a minimum support value. Then, under the constraint that it should satisfy the user-defined support, it generates a sizable frequently mined set of items. If it does, the number of items in the item sets continuously decreases until it obtains the greatest collection of frequently occurring items. The frequently mined item sets are then provided by the reverse Apriori algorithm in this way (Kharat & Gupta, 2014; Chawla & Dhindsa, 2014).

The following is the reverse Apriori pseudo code:

1. n = Select the maximum number of items
2. i=0
3. For each combination of (n-i) number of items Do
4. Generate Candidate itemsets of n-i
5. Generate frequent itemsets from candidate itemsets of n-i
6. Where support in generated itemsets>=minsup
7. If true, then go to step 9
8. Else increment i and goto step 3
9. Return set of large itemsets
10. End

Limitations of Reverse Apriori: For a large no of dataset transactions, reverse Apriori needs large database scans. So time complexity increases with the increase in the number of transactions. Here we need a more optimized approach to avoid larger database scans.

4. Proposed Work

The proposed optimization using data encoding considers the data frame in the form of 1 or 0 rather than the string form. Each row is converted into a single binary string rather than a group of strings. This takes less space to store each transaction than the traditional approach of storing all items that a user has bought. With this optimization, we can save space in representing each row. Also, because of converting each row to decimal values, the calculations are done with more ease, and Manipulation is done with ease.

Data Encoding: We have a list of all the items in the dataset. Now we will be converting them based on their presence in the transaction. Suppose any item is present in the transaction, then it will be encoded as 1; else it will be encoded as 0. Suppose we have "m" attributes in the dataset; then our binary string for each transaction will be of length "m". Each bit will indicate the presence of each item in the transaction. Now we will take the whole binary string of the single transaction and convert that to the decimal value corresponding to the binary string. If we have an "m" length binary string, then we will convert it to a decimal value within the range $0 - 2^m$.

After converting each transaction to the respective decimal value, we have now accomplished the first step of optimization. Now the manipulations on the transactions will be easy to do. The conversion of each transaction will now provide an efficient way to deal with the Algorithm and hence serve for the optimization.

Use of Binary String for Transaction details:

E.g., suppose we have 4 items (Milk, Bread, Eggs, Butter) in the shop, so we know that each transaction will either have,

For transaction I, if the customer buys Milk and Bread, then the binary string for the transaction is (1 1 0 0), and the decimal equivalent for the transaction is 12

For transaction II, if the customer buys Bread and Butter, then the binary string for the transaction is (0 1 0 1), and the decimal equivalent for the transaction is 5

Converting each transaction binary string into a decimal equivalent value makes further computation easy. The space needed to store the data is also utilized and efficiently managed. Now, from the decimal value, we can understand and trace back the items purchased by the user, and so there is no need to mention the name of each and every item that the user has purchased in the transaction. This way, we are formatting and converting the string to an int value; the Data encoding is taking place.

Optimization: In the above optimization technique, we represent the transaction as a decimal value rather than a string, so the storage size required for an integer is smaller, and the information gained by the encoding remains the same. The size of transaction representation when using the traditional approach for storage is the sum of the lengths of all item name strings, whereas the new optimized size for each transaction will be the size of an integer, which is 2 bytes. Thus, converting transaction strings to decimal representation makes efficient use of the encoding and is therefore proposed for optimizing the Algorithm.

The hash map: The hash table is created to store values in the form of key-value pairs. The hash map is used to store transactions and the number of times each transaction has occurred in the database. The key value

indicates the transaction no that is used to find the total no of items that are present in the transaction, and the names of those items can be found using the decimal representation of that transaction. The key stores the decimal representation of the transaction, and its value represents the total number of times the specific transaction has occurred, which will help us analyze the frequency of each transaction.

We need to iterate over the transactions once, and based on each transaction's value, we will need to fill the hash map. Whenever we iterate over all transactions, for each transaction we need to do either of the following steps:

I. If the transaction already has an entry in the hash map, then we need to increase the count of that particular hash map entry by incrementing its value by 1.
II. If the transaction has no entry in the hash map, then we need to create a new entry for that transaction value and save its value as 1.

Hash map format:

{ Hash Key : Value}

⇧ ⇧

Decimal value of transaction | No. of times the transaction occurs

Table 1 below shows the transaction.

Table 1. Transaction Dataset Example.

Transaction	Binary representation (Milk, Bread, Butter, Eggs)	Decimal Value
Milk, Bread	1 1 0 0	12
Bread, Butter	0 1 1 0	6
Milk, Bread	1 1 0 0	12
Milk, Bread, Butter	1 1 1 0	14
Milk, Bread, Eggs, Butter	1 1 1 1	15
Bread, Butter, Eggs	0 1 1 1	7
Milk, Bread, Eggs, Butter	1 1 1 1	15

Hash map for Table 1 is given below:

{12:2, 6:1, 14:1, 15:2, 7:1}

Searching for a particular transaction in the hash map tells us whether it is present in O(1) time and hence optimizes the traditional approach, which used to take O(n) time, where n is the total number of transactions in the dataset.

Now, for the calculation of support, we need to find the no. of transactions with a particular value, and hence it will be efficiently found using the hash map. The support calculation helps us to find the next candidate set, and hence we are able to generate the association rules. The support makes use of a hash map to find the total no of transaction values present in the hash map that satisfy the given constraint. Hence, the use of a hash map helps us to make the traditional searching approach much more efficient. Asymptotically, the time complexity of traditional searching is O(N) each time in each iteration, but with the use of a hash map, it only takes O(1) complexity in searching. Hence, hash maps are used to optimize the Apriori algorithm and to overcome the problems in the traditional Apriori algorithm.

Final Optimization (Support Calculation): In this final optimization, we will be finding the support more efficiently. To find the support, we need to find the total no of transactions with the following specification. We will find the transactions using the hashmap, and the total transactions desired need to be calculated.

In this method, we will generate all possible transactions from the given transaction, which will help us determine whether all the generated transactions are present or not.

ex- Suppose we have the following transaction:

Step 1: One-hot encoding

1. Milk Bread

Now converting it to the binary one-hot encoding format

2. Milk =1 Bread=1 Eggs=0 Butter=0

Step 2: Masking Technique(used to access only bits referring to only specific items from the transaction)

3. In this step, we will set the bit 1 only where the required items are to be searched in the transaction. E.g., to get the transaction for milk and bread from any transaction, we will create a mask for all other items except milk and bread; hence, we get the mask as given below.

1 1 _ _ (The _ indicates that the values can be filled with possible 0/1 value)

Step 3: AND Operation (here, to find out the support value of the generated itemset, we make the Bitwise AND operation of the masked value from the above step and each transaction binary stream.)

4. Generate the possible binary string by filling appropriate values in the _

Possible values: 1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

5. Decimal values of the transactions:
12, 13, 14, 15
6. Now we will
7. the no. of transactions with the find if the above transactions are present in the hashmap or not.
8. The above values help us to understand the total no of transactions that have Milk, Bread in it.
9. This helps us to calculate a particular specification.

Hence, we can efficiently calculate the Support.

5. Testing Set-I

For the implementation of the proposed algorithm, we have used the Python platform. Using the above-mentioned various Apriori Algorithms, we try to find some associations between products in orders of a bakery in Edinburgh, Scotland. The dataset has 20507 entries, over 8000+ transactions, downloaded from Kaggle. Here, the top 10 best-selling items are considered for rule Mining. After applying preprocessing techniques, the database is arranged in a suitable format for mining. We compare the performance of traditional Apriori, reverse Apriori, and the proposed hash-based reverse Apriori algorithm for the preprocessed dataset.

The first parameter selected for comparison is the number of database scans. Algorithms are applied on 10 different items over an 8000+ transactions dataset.

Table 2 shows the result of database scan comparison on the above breadbasket dataset

Parameters:

A: No. of database scans for Candidate set Generation
B: No. of database scans for Frequent Item Calculation
C: No. of database scans for Rule pruning
D: Overall required database scans

Table 2. Database Scan for Breadbasket Dataset.

Parameter/ Algorithm	A	B	C	D
Apriori	8339	16327762	79973223	96309324
Reverse Apriori	1050714	1467664	16110948	18629326
Hash Reverse Apriori	257024	4096	3864	264984

The following chart in Figure 1 shows the comparison of database scan required for Traditional Apriori, Reverse Apriori, and the proposed Hash-Based Reverse Apriori algorithm.

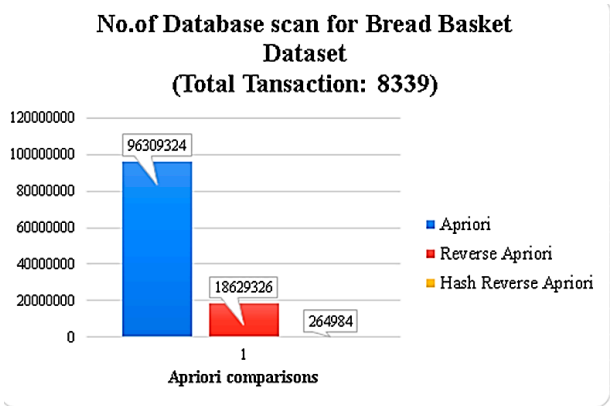

Figure 1. Database Scan chart for Bread Basket dataset.

The second parameter used for comparison is the time required to produce the results. The following.

Table 3 shows the result of the time requirement comparison on the above Bread Basket dataset.

Parameters:

A: Preprocessing time
B: Candidate Set Generation time
C: Frequent Itemset Calculation Time
D: Rule generation time
E: Rule Pruning Time
F: Total Required Timings

Table 3. Timing Requirement for Breadbasket Dataset.

Parameter/ Algorithm	A	B	C
Apriori	1.08562	1.02168	10.223
Reverse Apriori	1.09181	1.61438	1.8467
Hash Reverse Apriori	1.10103	2.65584	1.0119

Parameter/ Algorithm	D	E	F
Apriori	1.099	87.371	100.8
Reverse Apriori	1.011	20.3643	25.99
Hash Reverse Apriori	1.095	1.04145	6.901

The time required to calculate each step of the algorithm is shown in following Figure 2 for BreadBasket, comparing time complexity across algorithms.

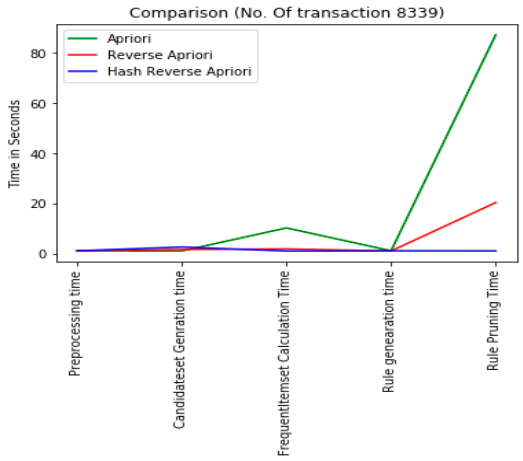

Figure 2. Timing Requirement chart for Bread Basket dataset.

6. Testing Set-II

The dataset contains 11 clinical features for predicting heart disease events. This project helps to predict whether the patient has Heart Disease or not. This prediction is made using patients' clinical data. The dataset was downloaded from Kaggle. This dataset is used to find association rules that cause heart disease. To find out the associations, we have used traditional Apriori, reverse Apriori, and the proposed Hash-Based reverse Apriori algorithm.

To compare the performance of the above algorithms, the first parameter chosen is the number of database scans. The algorithms are applied to 11 different items over a 918-transaction dataset.

Table 4 shows the result of database scan comparison on the heart dataset.

Parameters:

A: No. of database scans for Candidate set Generation
B: No. of database scans for Frequent Item Calculation
C: No. of database scans for Rule pruning
D: Overall required database scans

Table 4. Database Scan for Heart Dataset.

Parameter/ Algorithm	A	B	C	D
Apriori	918	324054	1270438	1595410
Reverse Apriori	27540	1364148	165240	1556928
Hash Reverse Apriori	579584	6144	186600	772328

The above chart in Figure 3 shows the comparison of database scan required for Traditional Apriori, Reverse Apriori, and the proposed Hash-Based Reverse Apriori algorithm.

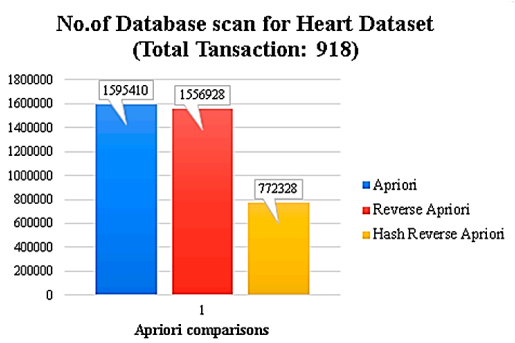

Figure 3. Database Scan chart for Heart dataset.

The second parameter used for comparison is the time required to produce the results. The time required to calculate each step of the algorithm for the heart dataset is presented to compare time complexity in each algorithm.

Table 5 shows the result of the time requirement comparison on the above heart dataset.

Parameters:

A: Preprocessing time
B: Candidate Set Generation time
C: Frequent Itemset Calculation Time
D: Rule generation time
E: Rule Pruning Time
F: Total Required Timings

Table 5. Timing Requirement for Heart Dataset.

Parameter/Algorithm	A	B	C
Apriori	1.02201	1.00317	1.29419
Reverse Apriori	1.02151	1.05049	1.96168
Parameter/Algorithm	D	E	F
Apriori	1.01304	2.27879	6.61120
Reverse Apriori	1.00307	1.32410	6.36077
Hash Reverse Apriori	1.04197	1.01701	9.03900

The graph as shown in Figure 4 reflects the time requirement for heart dataset.

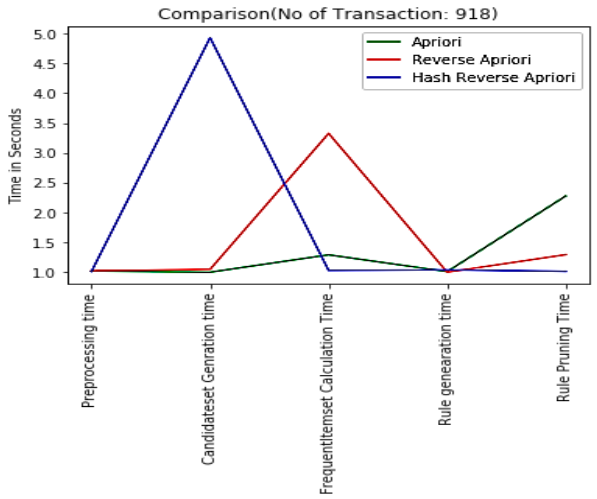

Figure 4. Timing Requirement chart for Heart dataset.

6. Results and Discussion

In this approach, we calculate support using hashmap searching and generate all possible transactions. This saves a lot of time because we search only those transactions that are possible.

Suppose we have m items in the dataset; then the total no. of unique possible transactions is 2^m.

In the traditional approach for finding whether the given transaction is present or not, we need to find all the transactions present in the dataset, but this is not very efficient, as it will take O(N). This is the time complexity for checking whether the transaction is present.

But in an optimized approach, we only need to find the total no of possible binary strings from the given transaction, which will be 2(m-p) where p is the total no of transactions present in the given iteration of transactions.

In large datasets, O(N) will be larger than all possible generated transactions, which means that this approach gives us a more systematic way of finding the support by searching through a hashmap, and finding candidate sets will be easy.

The above approach for finding can also be optimized with masking for searching. The mask will be created for each of the transactions, and hence, rather than finding, we will be performing the AND operation, which is less complex and more productive than other operations. The logical operation has less complexity and hence, in turn, gives us better results for searching and finding the support till the last candidate set generation. This way, the Apriori algorithm can be optimized by using methods like hash map, masking, and one-hot encoding.

Efficiency of Hash-Based Algorithm:

Comparing the results of two different datasets, when there is a larger number of transactions, the hash-based Apriori algorithm provides better efficiency for calculating frequent itemsets, rule generation, and rule pruning. In traditional Apriori and reverse Apriori methods, we need more database scans for a larger number of transactions in the dataset. But in Hash-Based Apriori, we are creating an array of hash values of each transaction, and the count of repeated transactions is calculated in the form of support(key) of those decimal values. Hence, instead of scanning the whole document for support calculation, we are scanning only the hash map table, resulting in high speed of rule generation. So, the overall time requirement is reduced by the proposed hash-based reverse Apriori algorithm by a reduction in the number of database scans.

Conclusions

The Apriori algorithm is useful in cases where we are required to find associative rules; the main issue with the traditional Apriori was recursive access to the database, and it does not use calculation of support and confidence in some cases. To tackle the problem, the use of one-hot encoding of data and the use of hashing and masking proved significant.

The current optimizations that follow reverse Apriori using hashes and masking are things that are useful only when the amount of data is populated in the database; for less populated databases, this method may prove slower.

Conflict of interest

The authors have no conflict of interest to declare.

Funding

The authors received no specific funding for this work.

Acknowledgements

The corresponding author would like to thank co-author Dr. D. K. Kirange for his valuable guidance. His exceptional support made it possible for this paper and the research. The Author is also thankful to SSBT COET, Bambori, for

providing a technical platform for research. This endeavor would not have been possible without the academic support of KBCNMU, Jalgaon

References

Abdulsalam, S. O. (2014). Data mining in market basket transaction: An association rule mining approach. https://d1wqtxts1xzle7.cloudfront.net/35091128/Data_Mining_in_Market_Basket_Transaction_An_Association_Rule_Mining_Approach-libre.pdf?1413082166=&response-content-disposition=inline%3B+filename%3DData_Mining_in_Market_Basket_Transaction.pdf&Expires=1786545548&Signature=OKHND3ZbKf9dM8ZXed2KecnspcKL4W96Tbcp1UK~LsusAnPLxxztqPijxDDLAGYE-aG8vCyQXjEfV25I2uV8nYSbpXa-STNAFHGAcagPG3VMhpjQBW4-vFnQme~fsh4JHMBleNSa03c~RnIVCPsxad9CE8aGRCGpLAy3Qdyh6qNZ9O4qeg1Bem2qH0FsoFsA5bEEZfgCiw~C49PMN3NCS7pxkiSwDwDnFp6wZ2qadRnsM94IcVjzBUfoqs16ANI5rn4bP2Oqy1240RHZAKL3uUjGnoQR7w3xF2YiIfznaH~IvmNIBnWDwAYzYCNnMrf0LVNlUWCRvUNIsl81-YnsBw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Agarwal, P., Yadav, M. L., & Anand, N. (2013). Study on Apriori Algorithm and its Application in Grocery Store. *International Journal of Computer Applications*, *74*(14), 1-8. https://d1wqtxts1xzle7.cloudfront.net/70625281/pxc3889882-libre.pdf?1636115442=&response-content-disposition=inline%3B+filename%3DStudy_on_Apriori_Algorithm_and_its_Appli.pdf&Expires=1786544328&Signature=F0lHgnO-elew~Ua-NASQ9pxkocUrQS~xqum-5YU6wh~Jpm0bTFXFNkrCN4MLRUVCpSqmzY2CL4CgDMoub1G~gf0c04rSgHZoQkh8yanxKjX4z0aX9TZocj0H2HxWYqhNVpyfHCQ0gUfTDPX1IwCKcDtCU1gckIDAK4twrq2buoPK6C6eaC7EAL0xt4bnY1uZ9gXO3nRjUz1oKLiL4QdvUl-HhYtlta3~6WvaY-v6~xtykUGYYWqm-6RFoWPxZGOt79MRyeH7opU4CfACQsID7dcCpIi8lBLIQ4pQzV2yzqpV11s8AtXv6JWQ13o0YGeBuaYFZPSjJXNLoy2MTwXjiw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Bartik, V. (2009). Association based classification for relational data and its use in web mining. In *2009 IEEE Symposium on Computational Intelligence and Data Mining* (pp. 252-258). IEEE. https://doi.org/10.1109/CIDM.2009.4938657

Chawla, A., & Dhindsa, K. S. (2014). Reverse apriori approach—an effective association rule mining algorithm. *Int. J. Comput. Complex Res*, *4*(3). https://d1wqtxts1xzle7.cloudfront.net/76111903/REVERSE_APRIORI_APPROACH__AN_EFFECTIVE_A20211210-20781-wwpbfv.pdf?1738455190=&response-content-disposition=inline%3B+filename%3DReverse_Apriori_Approach_an_Effective_As.pdf&Expires=1786545841&Signature=EfdLnqFob1UPezwVqz2ufJISaYrdfNzec6YF1DIVKaO8MyMpi5LFYFY29uZRZvAMx99vh-eqHJrmUIj-eMC9fPXD5fbfUY524DqcxYeEc8OfY2cnbQGkmpfFYbZ9piovRHdsnfoD4hDpnF5MCseVOmUDlEkQhyeEI2~0rmn20c6aMSITxrzKPYuIrh~HedcGN7CvDTTrDy~AlP0IcElB1uz0b28~ONL5Du1g1t6Z-SaCyehDpUXqzp1fhr3iKOGoldhKCLhWJEmjuUjrkhhPBr1V8z8wAyCmQ08EDkyNEl1282zIVKmdP7pw2JKocqJgAOKYKXnQOZQhB3uN8vUvfw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Dhanabhakyam, M., & Punithavalli, M. (2011). A survey on data mining algorithm for market basket analysis. *Global Journal of Computer Science and Technology*, *11*(11), 23-28. https://d1wqtxts1xzle7.cloudfront.net/34118919/848-833-1-PB-libre.pdf?1404502594=&response-content-disposition=inline%3B+filename%3DA_Survey_on_Data_Mining_Algorithm_for_Ma.pdf&Expires=1786544782&Signature=a~zN8exTEnbG-6lSyZzqBVhhSOBaFelaoOPpUt5Cfq7GXM2cBHds3lTs~99hHahdfDOvw8jukysIEYb4qhSOcUm2n5LFFBncPI7GLSyaEkktUtliU2~zUuk0IfO0RcLjrpaxRmWP~vS0XFebcAFEjxASRvxQAjX9iZahYVx9Jc7O13fAnf8LN7w1jzXujm86pj1-hqq2u9ttMZ3BziXbLbjKF5eEwxJTqqV2wbl~AYfXLfEq8DzK41vp8de~HXk0qjXWhJ7-3DokGV~wpGqn5JK7uaYYbZMlooaB~mfU~99RLmmZwBp2myxI5dsI-Actu2YWQjWmPX8QE3ye1IqtJg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Kharat, S., & Gupta, N. (2014). Result Analysis of Mining Fast Frequent Itemset Using Compact Data. *International Journal of Information Science and Techniques*, *4*(1/2). https://d1wqtxts1xzle7.cloudfront.net/85900739/4214ijist02-libre.pdf?1652422135=&response-content-disposition=inline%3B+filename%3DResult_Analysis_of_Mining_Fast_Frequent.pdf&Expires=1786545706&Signature=e-FVaZqWD32emvrN5ywjXfCmJiqjscVvaV9k5-rYYS4Up0XTG2cQ0NeRgD3oLgtrU4kpI6xAZS9UsihbppbYTS98vRXWL3eaOhTZDMphCHnQK2-DoHMuPTLFpt2TA7TrWxN4UFhODxyIwHMuvEdbEAYIK3P2z1LC~9QKJwvNy1tZOiOD9Q0PbyZfndCw2~YniUCu3g1Pj9oABWGoEv1PRhLHz4t4EwW40HBtlcjjsQ-tIjtRTtP~YpzDKIKzLaIUuZW96n7Y0fTQhE1lDs7UH70tbXBhZz5O6ncAfDPA4500MWEAwqvMQB8ZUf19rfni64XLhtAkg1y2dp0FA3PYqQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Niu, Q., Xia, S. X., & Zhang, L. (2009). Association classification based on compactness of rules. In *2009 Second International Workshop on Knowledge Discovery and Data Mining* (pp. 245-247). IEEE. https://doi.org/10.1109/WKDD.2009.160

Phani, P. J., & Murlidher, M. (2013). A Study on Market Basket Analysis Using a Data Mining Algorithm. *International Journal of Emerging Technology and Advanced Engineering*, *3*(6), 361-363.

Shaukat, K., Zaheer, S., & Nawaz, I. (2015). Association rule mining: an application perspective. *International Journal of Computer Science and Innovation*, *2015*(1), 29-38. https://www.researchgate.net/profile/Kamran-Shaukat/publication/284721728_Association_Rule_Mining_An_Application_Perspective/links/59564727aca272fbb37d21d4/Association-Rule-Mining-An-Application-Perspective.pdf

Sumithra, R., & Paul, S. (2010). Using distributed apriori association rule and classical apriori mining algorithms for grid based knowledge discovery. In *2010 Second International conference on Computing, Communication and Networking Technologies* (pp. 1-5). IEEE. https://doi.org/10.1109/ICCCNT.2010.5591577

Sharayu Bonde, Dr. D.K. Kirange, (2018), "Survey on Evaluation of Student's Performance in Educational Data Mining", IEEE 2nd International Conference on Inventive Communication and Computational Technologies, pp. 209-213. https://doi.org/10.1109/ICICCT.2018.8473228

Bonde, S.N., Kirange, D.K. (2020). Educational Data Mining Survey for Predicting Student's Academic Performance. In: Pandian, A.P., Senjyu, T., Islam, S.M.S., Wang, H. (eds) Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018). ICCBI 2018. Lecture Notes on Data Engineering and Communications Technologies, vol 31. Springer, Cham. https://doi.org/10.1007/978-3-030-24643-3_35