Applied Computer Science, vol. 17, no. 3, pp. 82–91
doi: 10.23743/acs-2021-23

Submitted: 2021-06-01 | Revised: 2021-08-16 | Accepted: 2021-09-21

frequent itemset, nodeset, FIN and IPOC

Pannangi NARESH[0000-0003-2932-6699]*, *R. SUGUNA*[0000-0002-8930-9092]*

IMPLEMENTATION OF DYNAMIC AND FAST MINING ALGORITHMS ON INCREMENTAL DATASETS TO DISCOVER QUALITATIVE RULES

Abstract

Association Rule Mining is an important field in knowledge mining that allows the rules of association needed for decision making. Frequent mining of objects presents a difficulty to huge datasets. As the dataset gets bigger and more time and burden to uncover the rules. In this paper, overhead and time-consuming overhead reduction techniques with an IPOC (Incremental Pre-ordered code) tree structure were examined. For the frequent usage of database mining items, those techniques require highly qualified data structures. FIN(Frequent itemset-Nodeset) employs a node-set, a unique and new data structure to extract frequently used Items and an IPOC tree to store frequent data progressively. Different methods have been modified to analyze and assess time and memory use in different data sets. The strategies suggested and executed shows increased performance when producing rules, using time and efficiency.

1. INTRODUCTION

The extraction procedure for information and new data formats is called knowledge mining (Agrawal, Imieliński & Swami, 1993). We may construct association rules and relationships from this derived information and itemsets that reveal some relationship and the link between items in transactions. In numerous fields like marketing, analysis and company improvement forecasts, this may be beneficial. Two qualitative metrics are needed by Association Rule (Naresh & Suguna, 2019) in general for the production of rules. Firstly, support and secondly, trust. This can enhance the procedure of mining and achieve the necessary regulations.

The FP (frequent pattern) growth method for mining was devised by Han et.al (Han et al., 2004). The first way is to develop FP-growth in depth. The tree structure has a left node and a right node; each node is associated with a transaction. About all repeated elements of a data collection, each node has a node count. This data was tree compressed, and two dataset scans were required for the mining of this technique. There was no requirement for a key candidate generation. It is substantially quicker for FP-Growth (UCI machine learning repository: Data sets, n.d.) than for other algorithms. Techniques of optimisation

* Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, pannanginaresh@gmail.com

are more time consuming to minimise. Zaki has suggested a first-hand depth approach such as FP to mining.

Infrequent production of items, the Apriori algorithm (Jain & Sharma, 2016) plays a vital role. This is done on each item-level by producing the candidate key. From these things created, two qualitative measures and trust will be considered to support these regulations. There are two levels in Apriori, one is joined, and the other is prune. All things are produced on a single level with the aid of the candidate key, and those things are cut to support at the pruning level.

Tab.1. Sample Transaction Databse

TID	Transactional Items
T001	{A,B,C,E}
T002	{B,C}
T003	{A,B,E}
T004	{A,C}
T005	{B,D}
T006	{A,C}
T007	{A,B}
T008	{A,C,E}
T009	{A,B,E}
T010	{A,B,C}

Minimum support level: 50% – {A},{B},{C},{A,B},{A,C}

Another type of common itemset production is the FP-growth algorithm. In the key generation, this is different from apriori. We have no candidate key in fp-growth. This works by creating a framework for an fp-tree in which each tree node is connected to an item. The graphic below explains the development of the Fp-tree from the data set (Naresh & Suguna, 2019). The data set is scanned by transaction, from which the fp-tree will be generated. The tree node link number will rise depending on the frequent order of the item if the same item is found again.

The chronological technique is applied to all essential association rule mining algorithms until the dataset amount is minimal. Their efficacy begins to decline with the number of data sets. Similar methods have therefore been devised to handle big datasets. Increasingly complex is the mining of the frequent items in a vast dataset (Chen, Zhao & Liu, 2020). Such problems have now been enhanced a day as enormous data sets and enormous volume databases are dealt with (Naresh & Suguna, 2019). Moreover, it was also a major problem, where datasets contain various attribute values, lack of and noisy data.

The PPC (pre-post code) tree (Lv et al., 2017) is made up of a null-value root node and child-value for that root. Without the root node, all nodes include pre-order (Deng & Lv, 2015), post-order, name of the item, count and list of children. The item name, which is only for which the individual items belong, is the pre-order, which indicates an integer value of the children number of the specific node, and the children list keeps all children of the tree.

The technological improvement in terms of power, communication networks and storage capacity has led to unprecedented data generation in the contemporary digital era. The mining of data from such vast databases helps to draw meaningful insights. These huge datasets nevertheless offer many difficulties. These include data volume, data speed, diversity and truthfulness. It is a problem for computer scientists to represent enormous data sets into a knowledge-based system and extract relevant information. In the field of knowledge engineering, data uncertainty is also a developing issue (Naresh & Suguna, 2019).

For example, data acquired from a location-based service such as GPS or sensor network data from several monitoring systems are generally probabilistic. The relevant information cannot be accurately retrieved for an educated conclusion using such data sets. In the finding of information from these probabilistic data sets, there is always uncertainty. Few new data mining methods (Qu et al., 2020) to overcome this problem have been introduced. The existing databases are nonetheless huge; so, new strategies must be deployed further to organise the data so that cost may be minimised.

2. BACKGROUND SURVEY

Transactional databases are distinct. The additional information offered on each transaction is different. A simple or conventional transactional database is merely a list of transactions with no further details in each transaction. In the database, temporary databases contained supplementary lifespan information. In this instance, transactions may be invalid beyond the end of their lifespan while new transactions are being put in incremental chunks (Abdelhamid et al., 2017) in the database with a fresh lifespan. Many techniques have been developed in general or total time databases to cope with the time association rules (TARs) mining process.

FP-Growt (Maw, 2020) and the so-called frequency pattern tree have been suggested with a scalable data structure (Hong et al., 2008). The advantages of employing them include not requiring candidate items and scanning the database twice so that FD-Growth is much more accurate than Apriori. The negative is that a wide range of body trees have to be constructed and take a great deal of RAM and engine time. The Apriori and FP-Growth approach for maintaining (Han et al., 2004) and controlling persistent databases are also useful. All this can not extend to a dynamic repository and may involve combining operational details over a period and the influence of pre-association rules and entirely new regulations.

To recreate the previous rules and identify new rules, the complete changed database has to be rescanned. However, this technique is time and money wasteful. The rules found in a database represent only the actual database status. However, Association rules found in the old database may no longer be valid and interesting rules for the upgraded databank in a dynamic database (Dhanaseelan & Sutha, 2016; Abdelhamid et al., 2017), where new transactions are often entered. Thus, new business information cannot be found, such as changing client preferences or new seasonal patterns.

In order to establish an intelligent environment that can detect new business data in a dynamic database (Dhanaseelan & Sutha, 2016), the algorithms of the association rules should be able to extract a dynamic database (Pavitra Bai & Ravi Kumar, 2016) gradually.

In order to resolve the problem, the larger database (Agrawal, Imieliński & Swami, 1993) may also be split into several smaller databases. Then, additional knowledge may be obtained in the incremental database without having to review the full repository. This methodology reduces the utilisation of devices and consumes less work to decode association rules from a small, incremental database than a huge dataset.

Several incremental updating strategies for the mining association rules have been developed for dynamic datasets (Hong et al., 2008). Cheung et al. described the FUP method as one of the earlier works for incremental association rule mining. FUP algorithm is the first incremental maintenance association update technology to insert new data into the database. FUP calculates frequent itemsets utilising big itemsets identified in the previous iteration based on the framework of the Apriori method. The main notion of FUP is the reuse with the incremental database (Song & Rong, 2018) of frequently produced items from prior mining. FUP's primary achievement is to cut the number of candidates set in update process.

The FUP extension algorithm (Chen, Zhao & Liu, 2020) is FUP2 which will be erased from a database to handle all the updates when a database has been added. In recent years, three types of data structure have been subsequently presented: Node-list (Deng, Wang & Jiang, 2012; Deng & Lv, 2015), N-list and Nodeset, for improving the efficiency of frequent mining elements. All of them are based on node sets from an encoded node tree. Pre-order number and post-order number are used to encode each node in the pre-order tree used by the node list and the N-list. The main distinction between nodes and N-lists is that nodes employ descendant nodes to represent a group of objects, whereas N-lists (Deng, Wang & Jiang, 2012) are an ancestral item. Two methods, dubbed PPV and PrePost, are presented for mining common items respectively based on the node-list (Deng, Wang & Jiang, 2012) and the N-list and prove to be highly successful and generally outperforming prior methods.

It is memory intensive, though, because node lists and N-lists (Deng, Wang & Jiang, 2012) must encode a node with pre-order and post-order. In addition, the Node-list and N-list encoding models cannot be used for joining N-lists (or Node-lists) of 2 short arrays to construct N-lists of a large array . We provide a data structure for mining frequently used itemsets named Nodeset. Unlike N-list, Nodeset requires the pre-order (or post-order) number of a node to encode the node without pre-order and post-order number requirements. We propose FIN to find frequent things based on the Nodeset structure.

3. PROPOSED APPROACH

New transactions can be inserted via a dynamic database. Not only may current association rules be invalidated but also new association rules be enabled. It is an important problem to maintain association rules for a dynamic database. This research thus offers a novel technique for dealing with these updates. We assume that the statistics of new transactions are progressively changing from the old transactions for the new method. According to the presumption, the old transaction data obtained through prior mining may estimate the new transaction data. Thus, after inserting new transactions into an original database that contains old transactions, supporting counts for objects derived from prior mining may deviate somewhat from support counts.

The new technique employs a maximum of 1 item set support from prior mining to estimate rare items in an original database, which can be frequent items when new transactions in the original database are added. The support count of unusual items that are eligible for frequent itemset, with maximum support number and size of new transactions that enable inserting to an original database

Tab. 2. Transaction database with ordered items of min_sup 2

Transaction ID	Items	Ordered Items
T200	**I1, I6, I7**	**I1, I6**
T201	**I1, I2, I3, I5**	**I2, I3, I5, I1**
T202	**I2, I3, I5, I9**	**I2, I3, I5**
T203	**I2, I3, I5, I8**	**I2, I3, I5,**
T204	**I2, I3, I4, I5, I6**	**I2, I3, I5, I6**

This table contains transaction ids, transaction items and arranged items by min support (2). I4, I7, I8 and I9 are deleted since their support is lower than min support. A POC tree will be generated with the rest of the items. Then create a POC tree for data D in the next step.

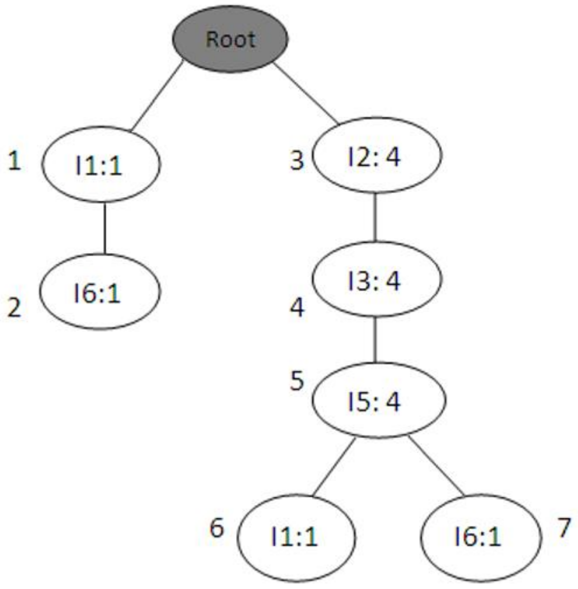

Fig. 1. POC Tree

FIN is a mining method used in transaction databases to locate frequent items. It depends on a new data structure called Nodeset that maintains frequent item information. FIN has three levels. Three levels. A POC-Tree has been established at the first level; frequency 1 itemsets are being mined, 2-item scanned in a second-level POC tree and 3 or more often used items at the final level.

Tab. 3. Ordered items of original (D) and new transaction database (d1)

	Transaction ID	Items	Ordered Items
D	T200	I1, I6, I7	I1, I6, I7
	T201	I2, I3, I5	I2, I3, I5
	T202	I2, I3, I5, I9	I2, I3, I5
	T203	I2, I3, I5, I8	I2, I3, I5, I8
	T204	I2, I3, I4, I5, I6	I2, I3, I5, I6
Incremental data – d1	T205	I1, I6, I7, I8	I1, I6, I8, I7
	T206	I1, I2, I3, I8	I2, I3, I1, I8
	T207	I1	I1

Some items support the min_sup criterion after adding new data set d1 to D. I7 and I8 are added to the current tree, and new frequent items comprising new data are also produced. The command of items changes in the ordered item list depending on the frequency count of each item. It is hence not necessary to me from scratch (existing data). Rather than updating the new item count in the tree structure, it is simple to mine the frequent things by looking at the count. In this tree, the IPOC structure is described.

3.1. FIN with POC and PPC

Although the node list and n list are composed of new mining structures, all PPC or POC tree nodes that demand time and space for often generated items must be encoded in these lists. A nodeset structure was developed to enhance time-space compromise in order to overcome these disadvantages. The nodeset shows that speed and performance are improved.

All frequent itemsets in the supplied dataset are generated by the FIN algorithm. Finish algorithm scans and produces the data set first, then visits each tree node and then calculates its support once it moves to the next node and follows the same way up to all nodes in the tree. It shows all the created FIs together with their support. We can produce association rules for analysis from these common objects.

POC Tree:

POC Tree algorithm works as follows:

- Build a POC tree.
- Identify 1-frequency item sets.
- Build the frequency sets according to the POC tree.
- Scan a POC tree for 2-items often.
- Mine all common dataset k-itemsets.

Each node N in the POC tree has a pair of values: one is pre-order, and the other is counting and N-info. The frequent item I nodeset is the series with every N-info about the POC tree nodes.

IPOC-tree

There are three major fields in the FP trees for each node: the name of the entity, the count and the connection to the node. The number of path components that strike the node reflects the number of operations is one integer in every node. Unlike the FP tree, each IPOC tree node contains four primary fields: object-id, initial number, scaled number, and node linkage. The first number tracks an object's counts in the database. A scaling number is supported in the scaling database by an object count. The increment count is started at 0 per node and increased when the associated entity is created from the main repository in the scaled depository during an IPOC-tree. . This feature is useful because an incremental database is shown in which several actions are added to the node item.

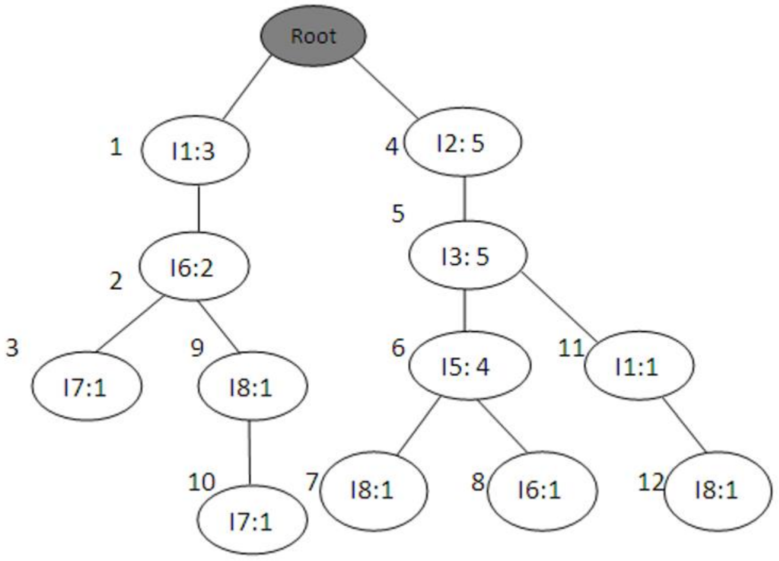

Fig. 2. IPOC tree for old data (D) and new data (d1)

4. RESULTS ANALYSIS

Datasets used: To implement the frequent items some standard datasets were needed. In this implementation Mushroom, Connect, Chess and Online Retail datasets were used. Mushroom, Connect and Chess datasets were downloaded from FIMI Repository and Online Retail Dataset was taken from UCI Machine Learning Repository (UCI machine learning repository: Data sets, n.d.).

Tab. 4. Datasets description

Dataset	No of Instances	No of Attributes	Size
Chess	3196	36	349 KB
Mushroom	8124	22	365 KB
Connect	67557	42	5829 KB
Online Retail	541909	8	23160 KB

The datasets were selected for implementing and analyzing the time and space trade offs for IPOC tree for mining qualitative association rules. The following table describes various datasets performances. Here the dataset sizes was increased dynamically to check efficiency , memory and time.

Tab. 5. Comparison analysis of time and memory for various datasets

Dataset Size/Name	IPOC-Tree							
	Chess		Mushroom		Connect		Online Retail	
	Time	Memory	Time	Memory	Time	Memory	Time	Memory
20%	0.92	12.36	2.51	25.16	7.58	78.91	17.51	180.75
40%	1.14	14.85	2.73	26.25	8.95	81.65	24.65	197.63
60%	1.42	17.87	3.01	26.38	10.87	93.85	31.86	204.36
80%	1.97	18.96	4.23	26.42	11.32	101.25	42.88	221.45
100%	2.12	21.61	4.42	26.64	13.86	109.24	50.81	239.61

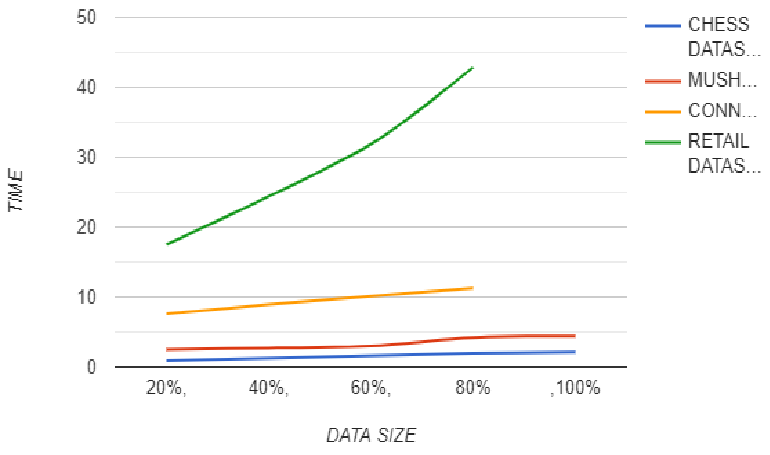

Fig. 3. Time comparisions of increasing datasets

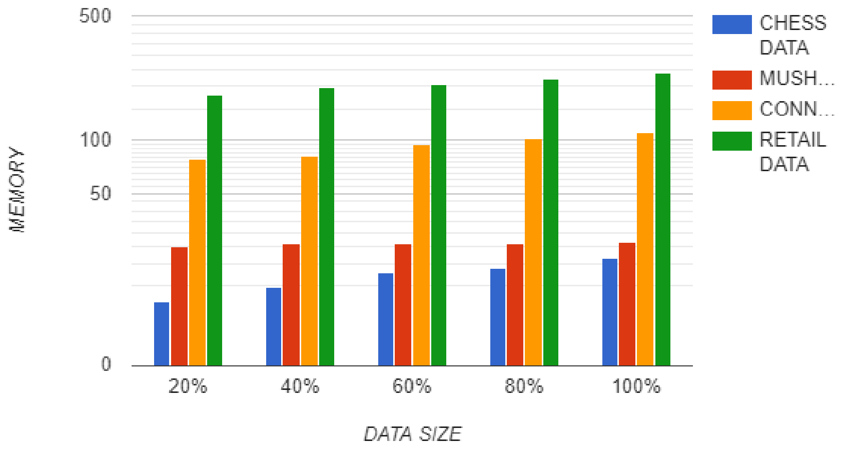

Fig. 4. Memory comparisions of increasing datasets

5. CONCLUSION

This article discusses an implementation employing a concept of FP algorithms for improved progressive association rules discovery for dynamic datasets. The fundamental notion behind the technology is to get and dynamically employ all recurring things in the initial repositories to detect all the common objects. The redesigned IPOC tree will ensure that starting pathways are avoided and that the installed substrates are minimised. The issue of incremental mining under the rule of temporary association was addressed in this work. The difficulty is to locate frequent time items in updated databases that add new transactions depending on a specified timetable. Results also demonstrated that the methodology presented uses less storage space. On the other hand, the method demonstrated linear squealing speed even at lower minsupp threshold values while mining huge datasets. The experimental results show that the proposed methodology has minimised the number of subtrees formed and the time required.

REFERENCES

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data – SIGMOD '93* (pp. 207–216). ACM Digital Library. https://doi.org/10.1145/170035.170072

Deng, Z., Wang, Z., & Jiang, J. (2012). A new algorithm for fast mining frequent itemsets using N-lists. *Science China Information Sciences, 55*(9), 2008–2030. https://doi.org/10.1007/s11432-012-4638-z

Deng, Z., & Lv, S. (2015). PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets via children–parent equivalence pruning. *Expert Systems with Applications, 42*(13), 5424–5432. https://doi.org/ 10.1016/ j.eswa.2015.03.004

Hong, T.-P., Chen, H.-Y., Lin, Ch.-W., & Li, S.-T. (2008). Incrementally fast updated sequential pattern trees. *2008 International Conference on Machine Learning and Cybernetics* (pp. 3991–3996). IEEE. https://doi.org/10.1109/icmlc.2008.4621100

Lv, D., Fu, B., Sun, X., Qiu, H., Liu, X., & Zhang, Y. (2017). Efficient fast updated frequent pattern tree algorithm and its parallel implementation. *2017 2nd International Conference on Image, Vision and Computing (ICIVC)* (pp. 970-974). IEEE. https://doi.org/10.1109/icivc.2017.7984699

Naresh, P., & Suguna, R. (2019). Association rule mining algorithms on large and small datasets: A comparative study. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)* (pp. 587–592). IEEE. https://doi.org/10.1109/iccs45141.2019.9065836

Pavitra Bai, S., & Ravi Kumar, G. K. (2016). Efficient incremental Itemset tree for approximate frequent Itemset mining on data stream. *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)* (pp. 239–242). IEEE. https://doi.org/10.1109/icatcct.2016.7912000

Qu, J., Hang, B., Wu, Z., Wu, Z., Gu, Q., & Tang, B. (2020). Efficient mining of frequent Itemsets using only one dynamic prefix tree. *IEEE Access, 8*, 183722-183735. https://doi.org/10.1109/access.2020.3029302

Maw, S. S. (2020). An improvement of FP-growth mining algorithm using linked list. *2020 IEEE Conference on Computer Applications (ICCA)* (pp. 1–4). IEEE. https://doi.org/10.1109/icca49400.2020.9022857

Chen, R., Zhao, S., & Liu, M. (2020). A fast approach for up-scaling frequent Itemsets. *IEEE Access*, *8*, 97141–97151. https://doi.org/10.1109/ACCESS.2020.2995719

Jain, T., & Sharma, D. V. (2016). Quantitative analysis of Apriori and eclat algorithm for association rule mining. *International Journal Of Engineering And Computer Science*, *4*(10). https://doi.org/10.18535/ijecs/v4i10.18

Dhanaseelan, F. R., & Sutha, M. J. (2016). An effective hashtable-based approach for incrementally mining closed frequent itemsets using sliding Windows. I*nternational Journal of Data Mining, Modelling and Management, 8*(4), 382. https://doi.org/10.1504/ijdmmm.2016.10002313

Abdelhamid, E., Canim, M., Sadoghi, M., Bhattacharjee, B., Chang, Y., & Kalnis, P. (2017). Incremental frequent Subgraph mining on large evolving graphs. *IEEE Transactions on Knowledge and Data Engineering, 29*(12), 2710–2723. https://doi.org/10.1109/tkde.2017.2743075

Song, W., & Rong, K. (2018). Mining high utility sequential patterns using maximal remaining utility. In Y. Tan, Y. Shi & Q. Tang (Eds.), *Data Mining and Big Data. DMBD 2018. Lecture Notes in Computer Science* (Vol. 10943, pp. 466–477). Springer. https://doi.org/10.1007/978-3-319-93803-5_44

Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, *8*(1), 53–87. https://doi.org/10.1023/b:dami.0000005258.31418.83

UCI machine learning repository: Data sets. (n.d.). Retrieved April 8, 2021 from https://archive.ics.uci.edu/ml/datasets