Upgraded Very Fast Decision Tree: Energy Conservative Algorithm for Data Stream Classification

Mai Lefa[1], Hatem Abd-Elkader[2], Rashed Salem[2]
Department of Information System, Sadat Academy for Management Science, Cairo, Egypt[1]
Department of Information Systems-Faculty of Computers and Information, Minoufia University, Egypt[2]

***Abstract*—Traditional machine learning (ML) techniques model knowledge using static datasets. With the increased use of the Internet in today's digital world, a massive amount of data is generated at an accelerated rate that must be handled. This data must be handled as soon as it arrives because it is continuous, and cannot be kept for a long period of time. Various methods exist for mining data from streams. When developing methods like these, the machine learning community put accuracy and execution time first. Numerous sorts of studies take energy consumption into consideration while evaluating data mining methods. However, this work concentrates on Very Fast Decision Tree, which is the most often used technique in data flow classification, despite the fact that it wastes a huge amount of energy on trivial calculations. The research presents a proposed mechanism for upgrading the algorithm's energy usage and restricts computational resources, without compromising the algorithm's efficiency. The mechanism has two stages: the first is to eliminate a set of bad features that increase computational complexity and waste energy, and the second is to group the good features into a candidate group that will be used instead of using all of the attributes in the next iteration. Experiments were conducted on real-world benchmark and synthetic datasets to compare the proposed method to state-of-the-art algorithms in previous works. The proposed algorithm works considerably better and faster with less energy while maintaining accuracy.**

Keywords*—*Classification*; *energy consumption*; *Hoeffding bound*; *Information gain*; *massive online analysis*; *stream data*; *very fast decision tree

I. Introduction

In recent years, the amount of generated data is growing significantly. This data is made up of n ($n \rightarrow \infty$) data samples, and it is described as a series of ordered data sequences, with starting and stopping bytes. Data stream (DS) = $\{x_1, x_2, \dots, x_i, \dots, x_n\}$, where $x_n$denotes the most recently visible data object in a stream of data [1]. Because the samples arrive in the form of continuous data flow at a great speed, the traditional classifiers cannot access the data instances in real-time [2]. The instances can only be scanned once or stored for a short period of time, so an ideal data stream classifier must be well prepared to deal with a large number of instances in real-time for better classification performance [1,3]. Data stream mining is a subset of machine learning that involves examining data flows that are continuously expanding as time series and developing a classification algorithm based on them. It is highly different from traditional data mining in terms of processing the mining operation, but it is similar to it in terms of purpose [2].

Machine Learning (ML) techniques consist of three types: unsupervised, semi-supervised, and supervised. Unsupervised algorithms can operate with unlabeled data; the data is clustered into groups with similar characteristics. Semi-supervised algorithms partially work with labeled data [4]. Supervised algorithms require data to be labeled, which they can categorize the data according to a distinct pattern for each class based on the label, such as data stream classification algorithms [5].

Very Fast Decision Tree (VFDT) is an effective classification technique for DS classification, and it has a high level of classification accuracy. It builds the decision tree by learning examples in real-time at a reasonable cost, and its sequential feature can match the data stream's timeliness requirement. To dynamically generate the decision tree, it is based on the enhancement of the Hoefdding tree. The Hoeffding bound (ε) is used to make sure that the data utilized to build each sub tree contains enough information [6,7].

The algorithm determines the information gain (G) for all noticed attributes after reading the minimum number of examples ($nmin$) at that leaf. The two top features are received through using the function $Best\ Split\ Suggestion()$. The difference between these features is compared with the ε, which was calculated previously. As a result, if the difference exceeds the ε, there will be a split on the tree by changing the leaf with a new internal node. This function is considered among the functions with the highest energy in the algorithm [8,9]. This point is considered the main problem that the paper solves. Therefore, the proposed mechanism focuses on working to reduce bad features that take a lot of energy when calculating their G, without impacting on the performance of algorithm. These are the motivations and objectives for our mechanism:

1) Decrease the amount of calculations by reducing undesirable features on each leaf.

2) Maintain the same level of accuracy by deleting just the computations that aren't necessary.

The rest of this paper is structured as follows: Section II clarifies related work. Section III introduces the background. Our algorithm and the corresponding theoretical derivation are explained in Section IV. Experiments and results are covered

in Section V. Finally, the paper is concluded with the conclusions and the future work in Section VI.

II. Related Work

Many works have gone into improving the VFDT in various ways. The Random Forest algorithm was created by Dong Zhenjiang and Li Lingjuan [10], it is a combinational classifier that performs well in terms of classification. It is made up of multiple decision trees, and it can compensate for the lack of a single decision tree. This technique upgrades the Random Forest algorithm by sliding the time window to match the unlimited data streams, and introduces the criterion of constructing decision trees in the random forest classifier. The limitation of this study is that it improves the accuracy levels without trying to improve other evaluation criteria.

R. J. Lyon et al. proposed the GH-VFDT, a novel classification technique for unbalanced data flows. The Hellinger distance was combined with a stream classifier depending on the Hoeffding bound in an empirical study. They can be combined to create a skew-insensitive decision tree split criterion that enhances minority class recall rates significantly. On unbalanced data, the algorithm can successfully enhance minority class recall rates, with same performance levels [11].

In the research [12], Ariyam Das et al. presented a memory-efficient bootstrap simulation heuristic (Mem-ES) that effectively accelerates the learning process. Experiments show that performing resampling techniques efficiently speeds up node splits for online decision tree learning**.**

Victor Guilherme Turrisi da Costa et al**.** proposed two versions of a novel VFDT-based algorithm, SVFDTs were developed to minimize the size of VFDT-induced trees, resulting in a memory-conserving decision tree. Both SVFDTs produce trees that are substantially smaller than those produced by the VFDT, while it doesn't reduce prediction performance statistically [13].

Gayathiri Kathiresan and Krishna Mohanta [14] present the compact method that uses the adaptive reservoir sampling methodology. It helps to reduce memory usage, and handle unbalanced stream data by restricting the information gain deviation based on the improved splitting metric in the information gain measurement. The limitation of this study is that it applied the proposed method to only one dataset, neglecting the other types of datasets with different characteristics, in order to determine the reliability of the findings.

Liang, Chunquan et al. use the Hoeffeding bound theory and the Uncertain Naive Bayes classifier (UNB) to improve the speed of construction and classification performance. They use gradual pruning and an adaptive tie-breaking criterion [15]. In a different study, Eva Garcia et al. proposed the *nmin* adaptation approach to enhance parameter adaptation in Hoeffding trees. They dynamically adjust the total amount of examples needed to make a split, this approach saves energy. The limitation of this study is that the *nmin* adaption method adjusts to the best *nmin* parameter based on the assumption, that newly received data would maintain the same distribution as previously seen data [16].

III. Background

A. Hoeffding Tree

There are several types of supervised classification techniques: k-Nearest Neighbor (kNN), Neural Network, Support Vector Machine (SVM), Naïve Bayes (NB) and Decision Tree (DT). Hoeffding tree algorithm is an extension of the decision tree algorithm for performing DS classification. It reads data in a single pass, and builds model $b = f(a)$ that maps test example a to class b. Every node represents a class feature, and each leaf represents the forecasted class label for that node. Beginning from the root node, the DT grows by exchanging leaf nodes with newly arrived test attributes. It would be the most efficient way of classification [17,18]. VFDT extends the Hoeffding tree algorithm by separating the current best features based on a user-specified threshold value [5].

B. VFDT

VFDT is a tree-based ML technique for DS based on the Hoeffding bound (ε) principles. The nodes represent the features of the dataset. The edges indicate the different possible outcomes for each attribute, and the leaf nodes represent the dataset's class labels. Once the model is complete, test data passes through it, and the decision tree will determine the class label. Each instance is read one by one, it is then sorted into the appropriate leaf, and the statistics are updated [5,19].

At the leaf node, denote $G(X_i)$ as the heuristic measure of attribute X_i. The algorithm is used to determine the heuristic measure (information gain (G)) for all observed features after reading the minimum number of instances ($nmin$) at that leaf. Assume that $X_a$and X_b are the attributes with the best and second-best G after seeing n pieces of data. The difference in G between both the best and second best features (ΔG), $\Delta G = G(X_a) - G(X_b)$, is compared to the ε after calculating it according to Eq. 1. If $\Delta G > \varepsilon$, then the attribute X_a is the best attribute of the current leaf node with a probability of 1− δ. A node will replace that leaf, and there is a split on the best feature X_a.

$$\varepsilon = \sqrt{\frac{R^2 ln\left(\frac{1}{\delta}\right)}{2n}} \quad (1)$$

The feature X_a is removed from the list of all features x when calculating the tested attributes in next iterations X_m as shown in Eq. (2), then the information gain for X_m must be calculated [20].

$$X_m = x - X_a. \quad (2)$$

The statistics required for attribute splitting are stored in each node. When two discrete attributes have similar split gains G or the highest and second-highest G are not significantly different, a tiebreak hyper parameter (τ) is introduced to enable tree growth. This is done by ignoring the Hoeffding bound condition and checking if $G(X_a) - G(X_b) < \varepsilon < \tau$ is true [18]. Because the two top attributes have extremely comparable G values, the algorithm can split into either of them [21].

The algorithm contains a set of fixed parameters that are determined before the work of the algorithm begins. The $nmin$ parameter defines the minimum number of samples that the algorithm must observe before computing. If there are sufficient statistics for a good split; the default value is 200. The parameter τ is utilized to break a tie in the event of a tie. When the difference between the two features is tiny enough, it suggests that both are equally as good. Therefore, waiting a long time for more instances to make a split is pointless. The δ parameter denotes one minus the likelihood of selecting the correct feature to split on. The researchers' default split criterion is information gain or the Gini index [6].

C. Energy Consumption for VFDT Functions

It is necessary to identify the energy consumption at each function level of the VFDT. The specific functions of the VFDT are four key functions: $homogeneous()$, $label()$, $split()$ and $bestSplit()$. If all of the tree's instances can be labeled with a single class, the homogenous() function returns true. The leaf node's label is returned by the $label()$ function. The $split()$ function encompasses all of the functions involved in splitting an internal node into several children. The best attribute to split on is returned by the $Split()$ function. This can be accomplished in a variety of ways, including by utilizing the information gain function [17].

IV. Theoretical Model of Upgraded U-VFDT

In this paper, U-VFDT is proposed to enhance the VFDT and improve its overall performance, which VFDT is one of the most well-known methods for handling data streams**.** The splitting is performed based on the current best attributes, whereas the algorithm determines the G from all noticed attributes after reading the $nmin$ at that leaf. This process is considered one of the most energy-consuming processes in the algorithm. In this case, the $bestSplit()$ function performs unnecessary operations that increase the wasted energy in the algorithm.

The difference in G between the top and the second-top feature (ΔG) is compared with the Hoeffding Bound (ε). If $\Delta G > \varepsilon$, the leaf is replaced by a node, and there is a split on the best feature. That feature is deleted from the list of features available to split on that branch. This method requires a periodic check because the best attributes are prone to change. The proposed method is introduced to limit the number of checked attributes which are noticed at the leaf. Because the G of all features is calculated in each split, and it leads to leveling up the algorithm's energy usage. The proposed method introduces a dynamic mechanism for appropriate features group through two basic steps.

The first step is adaptive appropriate features.

At this step, the performance of all attributes is analyzed to exclude the attributes with small G, through calculating the G for all attributes . If the G for a specific attribute X_i is less than the G of the best attribute X_a by more than a difference of ε $G(X_a)$-$G(X_i) \geq \varepsilon$, this attribute is ignored for that leaf. If G (X_a)-G $(X_i) \leq \varepsilon$, then the attribute is regarded as top feature.

The second step is appropriate features group.

At this step, the appropriate features group (P) is used in each leaf node, which stores the top X features selected at the previous step, and excludes the bad ones. Therefore, the P surely has captured the true split attribute [19]. This method leads to a reduction of the energy usage, because only appropriate features information gain will be evaluated in each split.

To implement this method, we assumed X_a is the attribute with highest G, for any other attribute X_i. X_i is said to be appropriate if $G(X_a) - G(X_i) \leq \varepsilon$ and X_i is shown in the appropriate group P ($X_i \in P$) , other features are removed. After making a split on the best attribute X_a, the algorithm removes X_a from the list of features in P available to split on that branch. In the next iteration, when algorithm calculates G for all tested attributes in next iterations X_m , only the list of attributes in the appropriate group will be recalculated, with the deletion of X_a as shown in Eq. 3.

$$X_m = p\text{-}X_a. \quad (3)$$

In theory, this approach should improve accuracy, reduce the number of calculations, and decrease the energy usage. It will be shown during the practical application. In Algorithm 1, a pseudo code displays the implementation of the U-VFDT with appropriate features group.

Algorithm 1: The U-VFDT with appropriate features group mechanism

Require:
S : the stream of instances
ε : Hoeffding bound
δ: the error probability
HT: Tree with a single leaf (the root)
X: set of attributes
$G(\cdot)$: split evaluation function
P : Appropriate features group
τ: the tiebreak parameter set by the user
Ensure:
Enhanced Very Fast Decision Tree
1. While stream is not empty do
2. Read instance I_i from S
3. Sort I_i to corresponding leaf *l* using ε
4. Update statistics at leaf *l*
5. Incrementn_i: instances seen at leaf *l*
6. If $n_i \geq nmin$ then
7. Compute ε
8. Compute $G(X_i)$ for each attribute X_i
9. If $G(X_a)$-$G(X_i) \leq \varepsilon$ then
10. $X_i \in p$
11. Calculate $\Delta G(.) \leftarrow G(X_a) - G(X_b)$
12. If $\Delta G(.) > \varepsilon$
13. Split on best attribute X_a and Replace *l* with a node.
14. For each branch of the split do
15. Update new leaves,
16. Add New Leaf L_m with empty p_m
17. Let $X_m \leftarrow$ p-X_a
18. End for
19. End if
20. End if
21. Else

22. Do not split
23. Do not update HT
24. End if
25. End while

V. Experiments and Results

In this section, the mechanism is analyzed using reliable datasets that have been utilized in previous researches, with the goal of determining how big of an influence of our algorithm; using the results of its prior efforts as a benchmark, and comparing it with the results of proposed mechanism.

A. Datasets

The experiment was conducted on several different types of datasets to see the impact of the proposed mechanism on each of those data. There are real-world dataset, synthetic dataset and real-world benchmarks datasets. These datasets were chosen as a baseline for the standard behavior of the algorithm. We obtained the real-world datasets from https://archive.ics.uci.edu/theml/datasets.php.

The airlines dataset is the real-world dataset. This classification dataset divides flights into two categories: delayed and not delayed, based on the flight's path and departure and arrival airports. It has eight attributes. The random tree dataset is a synthetic dataset that was created using MOA (Massive Online Analysis).

The synthetic generator generated one million cases. The last two datasets consist of real-world benchmarks, the first one is abalone dataset and this study includes predicting the age of abalone from physical measurements. The second one is adult dataset, which predicts whether income exceeds fifty thousand dollars per year based on census data. Their main characteristics can be seen in Table I.

TABLE I. Datasets Summary

Datasets	Name	Type	Instances	Numeric features	Binary features
1	airlines	Real-world	539,383	3	5
2	random tree	Artificial	1,000,000	5	5
3	abalone	Real-world benchmark	4177	2	6
4	adult	Real-world benchmark	48842	6	8

B. Tools

Massive Online Analysis (MOA) is a well-known framework for developing algorithms and conducting experiments. It has a number of ML methods, such as classification, regression, clustering, concept drift detection, and a recommender system. It also includes a number of evaluation tools.

MOA can be employed with WEKA's many classification and clustering approaches. It is used for online real-time stream data, while WEKA is used for offline data [6,22]. The MOA framework is running in parallel with IPPET (Intel(R) Platform Power Estimation Tool) which can measure how much power different processes are consuming.

A set of specifications for the device utilized in this experiment is also included in the environment for practical application, which impacts energy-related calculations such as the operating system: Windows 7 professional 64-bit (6.1, Build 7601), the processor: Intel(R) core(TM) i3-2350M CPU @ 2.30GHz (4 CPUs), ~ 2.3GHz, and memory: 4096MB RAM.

C. Experimental Design

The primary goal of the model is to increase the efficiency of VFDT. It focuses on understanding and developing the functions of the algorithm that consume the most energy. An efficient method has been proposed for selecting best attributes which are used to perform splitting. It avoids the calculation of the heuristic measure for unnecessary attributes that consumes high energy.

To the best of our knowledge, there are no previous works that have limited the energy consumption of the VFDT except the reference [16]. It has set $\varepsilon = \Delta G$ in the ε equation as shown in Eq. (4) to ensure that $\Delta G \geq \varepsilon$ is satisfied during the next iterations, resulting in a split.

$$nmin = \left\lceil \frac{R^2 . \ln\left(\frac{1}{\delta}\right)}{2.\Delta G^2} \right\rceil \quad (4)$$

The following is a comparative study of the practical experiment between the performance of our method U-VFDT, the standard algorithm VFDT, and this previous modification on algorithm P-FVDT. Run-time, power, memory usage, accuracy and energy are measured using four different datasets.

1) Run-time of VFDT algorithms: There is a convergence of execution time in our algorithm and other two algorithms as shown in Fig. 1. The increase of examples in random tree dataset leads to an increase in times of the heuristic measure for all attributes; thus, increases the running time. The U-VFDT achieves less time to reach the highest efficiency in random tree dataset. The reason of this result is that U-VFDT works on reducing those attributes that waste time. Also, adult dataset contains a large number of features, so the proposed method is effective in working to reduce these features.

U-VFDT does not achieve effective results in the other two datasets due to two reasons. The first reason is the small airlines dataset features, and the other reason is the few examples of abalone dataset. Thus, our method is not efficient, because it depends on reducing the number of features, where time is wasted. Unlike the VFDT, the time increases dramatically as soon as it receives new examples, whereas heuristic measure is counted for each attribute of a new example when it is received.

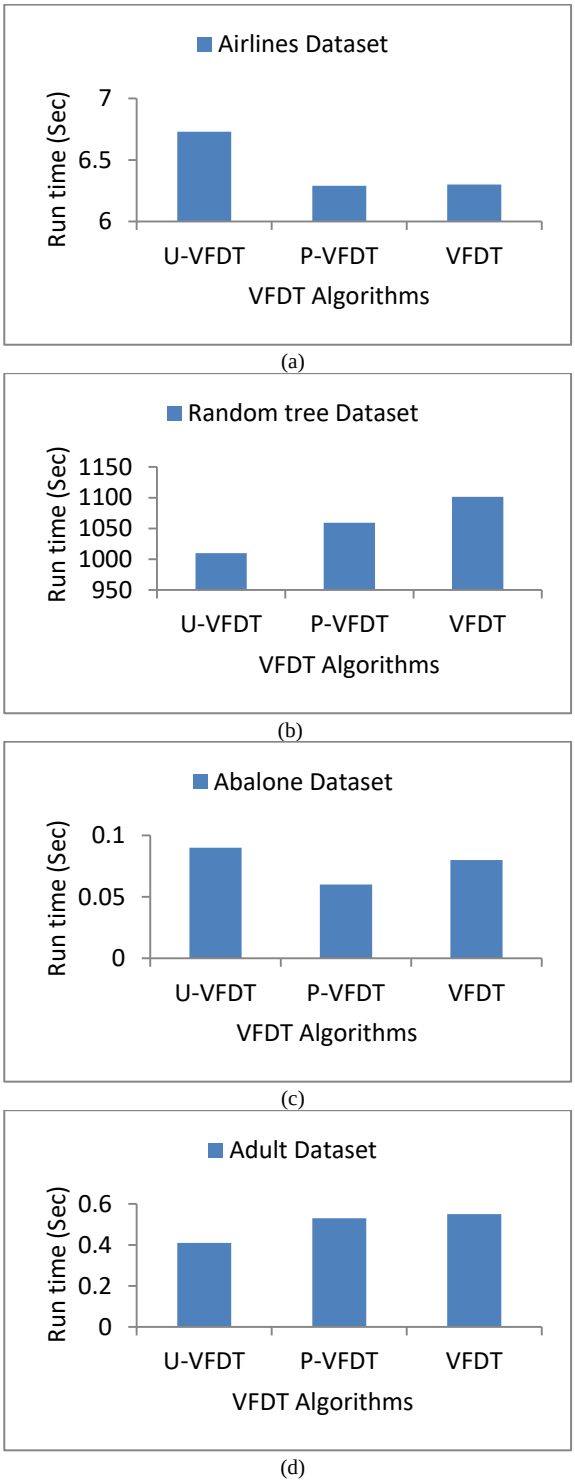


Fig. 1. Run time of VFDT algorithms for all datasets.

2) Power consumption of VFDT algorithms: As shown in Fig. 2, lower power values of U-VFDT in some datasets, and other higher values in other datasets. The power levels differ in each algorithm, due to the difference in the nature of the dataset in terms of its examples and features.

U-VFDT does not achieve an effective result in the abalone dataset, because this dataset contains a small number of examples. U-VFDT proves more efficiency than other two algorithms for other datasets due to two reasons. The first reason is the large number of examples for airlines and random tree datasets, and the other reason is the large number of features for adult dataset. Thus, our method is efficient, because it depends on reducing the number of features.

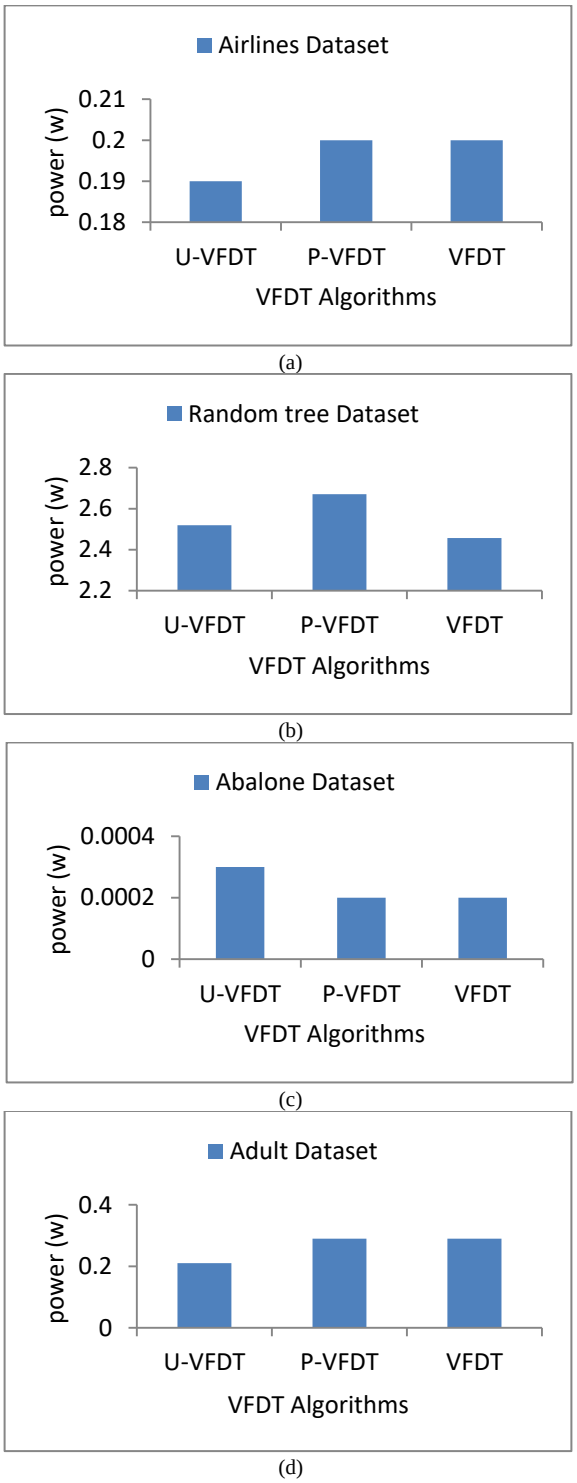
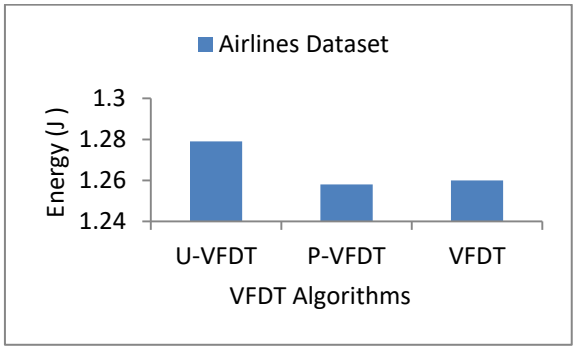


Fig. 2. Power consumption of VFDT algorithms for all datasets.

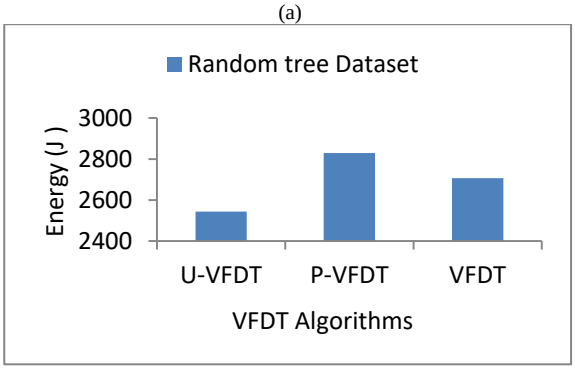
3) Energy consumption of VFDT algorithms for all datasets: U-VFDT succeeded in saving the extra time wasted, which helped in saving energy significantly, based on the Eq. (5). It saves energy only in datasets that contains many examples and attributes, by ignoring the bad features that have no chance of splitting.

$$\text{Energy} = \text{Power} \times \text{Time} \quad (5)$$

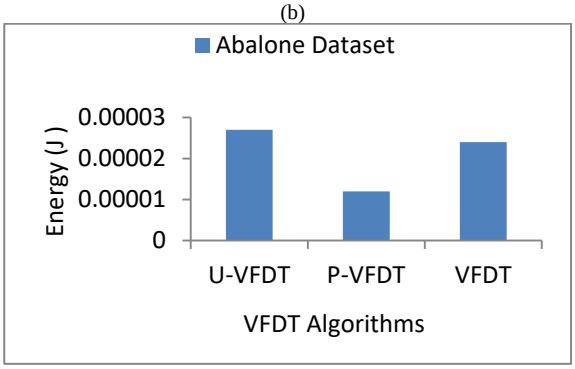
VFDT makes high computations because of the heuristic measure calculations for unnecessary attributes that is required to make a split as shown in Fig. 3.



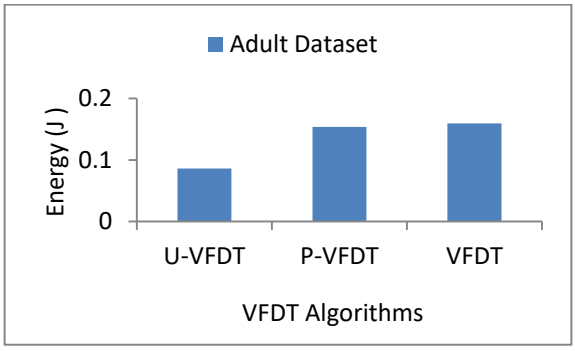
(a)



(b)



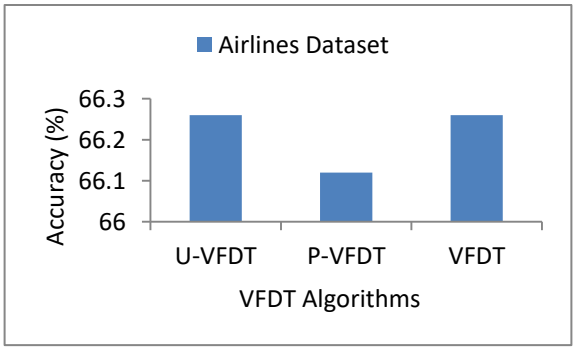
(c)



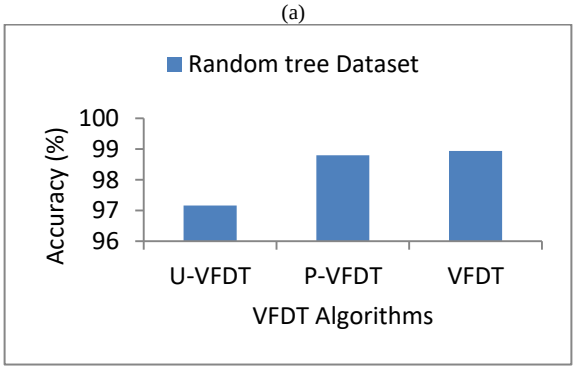
(d)

Fig. 3. Energy consumption of VFDT algorithms for all datasets.

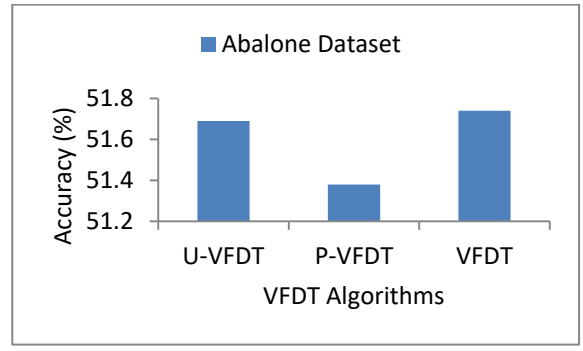
4) Accuracy of VFDT algorithms: As shown in Fig. 4, U-VFDT is maintaining the performance of the algorithm with light impact on accuracy; this is for data with few features and examples such as airlines and abalone datasets. On the contrary, U-VFDT significantly affects the accuracy of the algorithm, but in a greater proportion in the case of data containing a large number of examples and features, such as random tree and adult datasets, which is considered a drawback in our method.



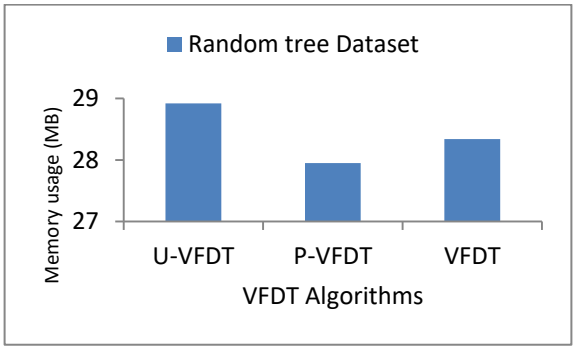
(a)



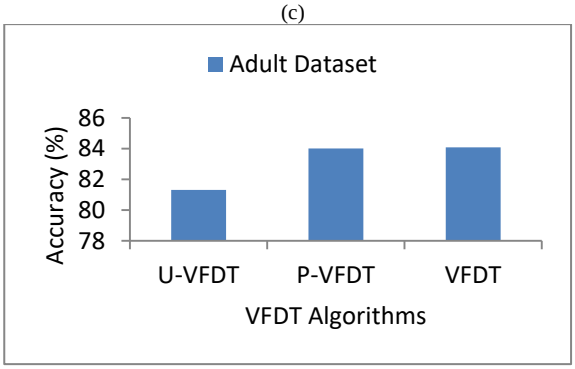
(b)



(c)



(b)

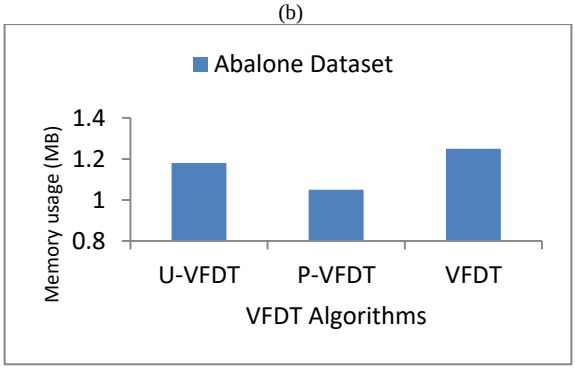


(d)

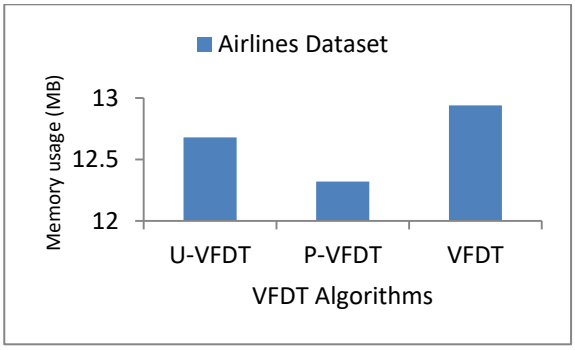
Fig. 4. Accuracy of VFDT algorithms for all datasets Accuracy (%).

5) Memory usage of VFDT algorithms: For memory usage, an adult dataset contains a large number of attributes. U-VFDT proves more efficiency than other two algorithms. It achieves the least required memory for making a split as shown in Fig. 5, which it excludes bad features, thus does not perform any unnecessary operations.

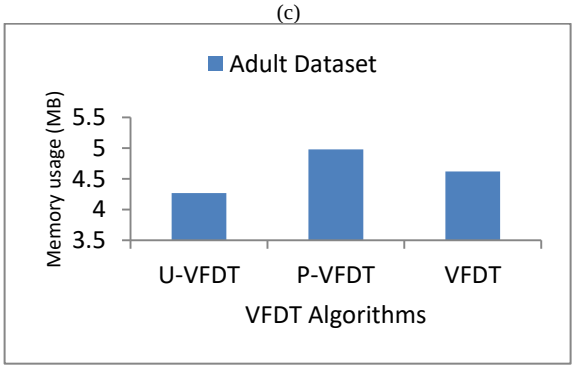
In the random tree dataset, despite the large number of its instances, U-VFDT does not achieve an effective result for it, because this dataset contains noise and concept drift that wastes memory. In the other two small datasets, our method does not achieve an effective result, because it depends on reducing the features.



(c)



(a)



(d)

Fig. 5. Memory usage of VFDT algorithms for all datasets.

The experiments clarify that U-VFDT has a better performance than other two algorithms. The proposed algorithm was limited to the heuristic measure of only the good attributes, through which the split could take place. It was excluded of bad attributes that are consuming massive levels of memory, and energy as well as a large run-time. Accordingly, the energy is decreased and the processes are speed up, because of the reduction in both running time and the memory usage. The limitation of this work is that it consumes less energy only in the dataset with a large number of instances and attributes. Also, it does not achieve an effective result for the datasets with noise and concept drift.

The total computational complexity of the VFDT is $O(n)$. The computational complexity of the U-VFDT does not exceed its value in the original algorithm, and still retains the same value after modification, since it was $O\ (n)$.

VI. CONCLUSIONS

This research developed a new technique to improve the VFDT, which allows for an energy conservative algorithm to construct Hoeffding trees without affecting their predictive performance, resulting in lower energy consumption and minor accuracy loss. In VFDT, after the splitting occurred, and the leaf turned into a node, all of features are recalculated to determine which splitting will occur through.

The proposed mechanism recalculates information gain for only the list of attributes in the appropriate group, with the deletion of the feature used for the previous split. It leads to the reduction of unnecessary calculations of bad attributes. Thus, an evolution occurred in the performance of the algorithm in terms of saving time and memory and reducing wasted energy consumption with maintaining the accuracy of the algorithm.

Finally, the mentioned algorithms are compared in different datasets with standard algorithm, and the previous modification work. The U-VFDT used less energy than the VFDT and the P-VFDT only in the datasets with large number instances and attributes. It does not achieve effective results in data that is small in size and has few features. The main work is based on limiting the useless features, thus reducing the number of unnecessary operations that increase running time, energy and memory usage.

Further methods are offered for future work in order to enable an energy-efficient method to build Hoeffding trees for datasets with noise and concept drift without compromising their predictive effectiveness.

REFERENCES

[1] Zheng, Xiulin, et al. "A survey on multi-label data stream classification." IEEE Access 8 (2019): 1249-1275.

[2] Li, Xiangjun, et al. "A classification and novel class detection algorithm for concept drift data stream based on the cohesiveness and separation index of Mahalanobis distance." Journal of Electrical and Computer Engineering 2020 (2020).

[3] Rad, Radin Hamidi, and Maryam Amir Haeri. "Hybrid forest: A concept drift aware data stream mining algorithm." arXiv preprint arXiv:1902.03609 (2019).

[4] Kok, S., et al. "A comparison of various machine learning algorithms in a distributed denial of service intrusion." Int. J. Eng. Res. Technol 12.1 (2019): 1-7.

[5] Rutuja Jadhav, Neha Sharma “Classification Methods For Data Stream Mining” vol.6. 2018.

[6] Ashish P. Joshi, Biraj V. Patel.” Comparative Study of Different Classification Algorithms for Stream Data Mining Using MOA” International Journal of Computer Sciences and Engineering.vol.6. 2018.

[7] Jia, Shuangying. "A VFDT algorithm optimization and application thereof in data stream classification." Journal of Physics: Conference Series. Vol. 1629. No. 1. IOP Publishing, 2020.

[8] Garcia-Martin, Eva, Niklas Lavesson, and Håkan Grahn. "Identification of energy hotspots: A case study of the very fast decision tree." International Conference on Green, Pervasive, and Cloud Computing. Springer, Cham, 2017.

[9] Garcia-Martin, Eva, Niklas Lavesson, and Håkan Grahn. "Energy efficiency analysis of the very fast decision tree algorithm." Trends in Social Network Analysis (2017): 229-252.

[10] Dong, Z. J., et al. "Random forest based very fast decision tree algorithm for data stream." Res. Paper 12 (2017): 52-57.

[11] Lyon, Robert J., et al. "Hellinger distance trees for imbalanced streams." 2014 22nd International Conference on Pattern Recognition. IEEE, 2014.

[12] Das, Ariyam, et al. "Learn Smart with Less: Building Better Online Decision Trees with Fewer Training Examples." IJCAI. 2019.

[13] da Costa, Victor Guilherme Turrisi, André Carlos Ponce de Leon Ferreira, and Sylvio Barbon Junior. "Strict very fast decision tree: a memory conservative algorithm for data stream mining." Pattern Recognition Letters 116 (2018): 22-28.

[14] Kathiresan, Gayathiri, Krishna Mohanta, and Khanaa VelumailuAsari. "COMPACT: Classifying Stream Data Optimally Using a Modified Pruning and Controlled Tie-threshold." vol.8. 2019.

[15] Liang, Chunquan, et al. "Learning accurate very fast decision trees from uncertain data streams." International Journal of Systems Science 46.16 (2015): 3032-3050.

[16] García-Martín, Eva, et al. "Hoeffding Trees with nmin adaptation." 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2018.

[17] Krawczyk, Bartosz, et al. "Ensemble learning for data stream analysis: A survey." Information Fusion 37 (2017): 132-156.

[18] Masrani, Aastha, Madhu Shukla, and Kishan Makadiya. "Empirical Analysis of Classification Algorithms in Data Stream Mining." International Conference on Innovative Computing and Communications. Springer, Singapore, 2021.

[19] Desai, Sharmishta, et al. "Very fast decision tree (VFDT) algorithm on Hadoop." 2016 International Conference on Computing Communication Control and automation (ICCUBEA). IEEE, 2016.

[20] Sun, Jiang, et al. "Speeding up very fast decision tree with low computational cost." Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. 2021.

[21] Bifet, Albert, et al. "Extremely fast decision tree mining for evolving data streams." Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017.

[22] Srimani, P. K., and Malini M. Patil. "Performance analysis of Hoeffding trees in data streams by using massive online analysis framework." International Journal of Data Mining, Modelling and Management 7.4 (2015): 293-313.