International Journal of Data Warehousing and Mining
Volume 18 • Issue 1

Improvement of Data Stream Decision Trees

Sarah Nait Bahloul, LSSD Laboratory, University of Science and Technology Mohamed Boudiaf, Oran, Algeria
https://orcid.org/0000-0001-9219-8381
Oussama Abderrahim, University of Science and Technology Mohamed Boudiaf, Oran, Algeria
Aya Ichrak Benhadj Amar, University of Science and Technology Mohamed Boudiaf, Oran, Algeria
Mohammed Yacine Bouhedadja, Univesrity Abou Bakr Belkaid, Tlemcen, Algeria
https://orcid.org/0000-0002-4975-1476

ABSTRACT

The classification of data streams has become a significant and active research area. The principal characteristics of data streams are a large amount of arrival data, the high speed and rate of its arrival, and the change of their nature and distribution over time. Hoeffding tree is a method to, incrementally, build decision trees. Since its proposition in the literature, it has become one of the most popular tools of data stream classification. Several improvements have since emerged. Hoeffding anytime tree was recently introduced and is considered one of the most promising algorithms. It offers a higher accuracy compared to the Hoeffding tree in most scenarios, at a small additional computational cost. In this work, the authors contribute by proposing three improvements to the Hoeffding anytime tree. The improvements are tested on known benchmark datasets. The experimental results show that two of the proposed variants make better usage of Hoeffding anytime tree's properties. They learn faster while providing the same desired accuracy.

KEYWORDS

Classification, Data Stream, Decision Trees, Hoeffding Anytime Trees, Hoeffding Trees

INTRODUCTION

Social networks, mobile applications, and IoT produce an avalanche of data on a large scale and in an unpredictable way. Digital data explosion has forced researchers to find new ways to analyze and exploit the world, to manage new scalability issues about the capture, storage, analysis, and representation of data.

One interesting characteristic of these data is the big flows generated, arriving sequentially and at high speed. The analysis has to take place in real-time to handle these flows. A data stream is a real-time, continuous, and orderly sequence of elements. It is impossible to control the order in which data arrives, nor to locally store a stream in its entirety because the instances arrive at a high rate leading to a massive volume of data or even infinite (Rutkowski et al., 2020).

Data Stream Mining, or the exploration of data streams, is currently a very active field of research and is developing rapidly (Amudha et al., 2021, Bahri et al., 2021). Predictive analytics on data streams plays a primary role in modern data analytics. Data is streaming in and needs to be analyzed in real-time to make a future decision. One of the most common data mining methods for prediction is classification. The purpose of classification is to build a function, called a classifier,

DOI: 10.4018/IJDWM.290889

based on the data provided for training, which associates each element of data with a label, and then uses the classifier to classify (or predict) new unlabeled data. A spam filter is a good example where we want to predict if new emails are considered spam or not.

The decision tree is a data mining tool commonly used in data classification tasks. In the data stream field, The Hoeffding tree (HT) (Domingos and Hulten, 2000) is currently considered as the main decision tree. It is an incremental decision tree induction algorithm that is capable of learning from massive data streams. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. The approach was applied to different problems, such as (Choudhury (2020), Deepa et al. (2020), and Soe et al. (2020)).

Several researchers have been interested and are still interested in improving the Hoeffding tree algorithm according to various criteria (accuracy, execution time, memory usage, etc.). In this paper, the authors propose to explore the most recent related works on the Hoeffding tree algorithm. They are particularly interested in the Hoeffding AnyTime Tree algorithm (HATT) (Manapragada et al., 2018), which is considered the latest most promising algorithm in the literature. HATT attempts to select and perform a division as soon as it is confident enough that the division is useful. Then unlike HT, it will revisit that decision after some examples when a potential better division is available.

In this work, the authors intend to further enhance the performances of the HATT algorithm by proposing three improvements, which better exploit the ability to revise decisions. The authors test their variants through several experiments. The results show a precision very close to the original algorithm while considerably reducing the additional processing time.

The remainder of this paper proceeds as follows. It first introduces some basic concepts related to this work, more precisely, data stream characteristics and decision trees. The authors then present the Hoeffding Tree algorithm and its implementation, namely, Very Fast Decision Tree (VFDT). They discuss in the related work section the most relevant improvements of VFDT. The paper also presents some variants of the Hoeffding criteria. In the contribution section, the authors propose three improvements to the Hoeffding Anytime Tree. They then discuss their experimental results. Finally, they conclude and present some potential future works.

BASIC CONCEPTS

Data Stream

Data streams are continuous and ordered sequences of data elements that can potentially grow into infinity (Rutkowski et al., 2020). There are three main characteristics of data streams that make the classic Data Mining algorithms inapplicable to them.

1. **The amount of data that has arrived (and will arrive) is immense**, even potentially infinite. As a result, it is impossible to store it entirely. Even if it is possible to stock everything, it would be unfeasible to go over this data more than once for further processing.
2. **The speed and rate of arrival of this data are high:** This implies that each item must be processed in real-time, for a limited time, then immediately move on to the next example.
3. **Obsolete data:** The nature and distribution of data can change over time, so old data can quickly become useless (sometimes harmful) for a current model. This phenomenon is called Concept Derivation, or Concept Drift.

From there, we can draw four requirements in the handling of data streams: (1) Each example or item must be considered and processed only once (at most). (2) A suitable algorithm must be fast enough to update the model or ignore some instances if the flow rate is too high compared to the computational capacity of the machines. (3) The algorithm must be able to store the information

learned about the processed items in a compact form or a summarized overview, and (4) The algorithm must be available to predict (classify) at any time.

Several researchers in the processing of data streams limit themselves to the two characteristics: the size and the rate of arrival of the data, and design their algorithms for Big Data. However, the most significant difference with static data is in the third characteristic, which is Concept Drift, and must often be taken into account in any non-trivial learning model.

Concept Drift

The majority of machine learning algorithms assume that the data is stationary. In other words, the statistical properties that the model is trying to predict do not change over time. This assumption is unfortunately not respected by the majority of data sources today. Therefore, models trained traditionally drift little by little, and their performance degrades over time. This phenomenon of evolving data is called Concept Drift, or Conceptual Drift (Lu and al., 2018). We can address the concept drift differently: Treat the model as static (ignore the change); re-train, periodically, the model from zero; Update periodically if the algorithm is incremental; Use weighted data where new entries are more important than old ones; Learn the change once detected or process the drift during the data preparation phase.

Decision Trees

A decision tree is a predictive model used to represent classification and regression (Oded and Lior, 2014). A decision tree is built from a dataset, where each record has several attributes (numeric or nominal/categorical type). The model consists of dividing the space of these data into regions through divisions on the data attributes.

The process of building a decision tree is called induction. An inductor is an algorithm that takes a dataset as input and an automatically constructed decision tree as its output. Inducing an optimal tree turns out to be an NP-Difficult problem (Hancock et al., 1996). Therefore, heuristic methods are needed to solve this problem.

There are different approaches to forming a decision tree. The most widely used is the Top-Down approach, based on recursive divisions of the training dataset according to different criteria and algorithms such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and CART (Breiman et al., 1984). In this approach, each iteration selects an optimal division (starting with the root node). Each node subdivides the training set into separate smaller subsets until no splitting is possible or that a stop criterion is satisfied. The splitting criteria are determined based on the decision tree algorithm used and are generally greedy and local. Each induction algorithm uses a well-defined division criterion.

Division criteria: There are several criteria characterized by a measure used or by their origin (from Information Theory, Distance, etc.). Each induction algorithm uses a well-defined division criterion. This section will focus on those based on impurity that is most common in the literature. The purpose of division in a decision tree is to create more homogeneous subgroups, in other words, purer groups.

The profit of a division on an attribute a_i is defined by the reduction of the impurity after the partition of the instances of the training set (S) according to the values of a_i , relatively to the impurity of the set S .

For a discrete attribute a_i , the set S can be partitioned into Q disjoint sets $\left\{S_q^i\right\}_{q=1}^{Q}$ ($Q = 2$ for a binary tree). A general formula for the reduction of impurity is therefore defined as (Rutkowski et al., 2020):

$$\Delta\Phi\left(a_i, S\right) = \phi\left(y, S\right) - \sum\nolimits_{q=1}^{Q} \frac{\left|S_q^i\right|}{\left|S\right|} . \phi\left(y, S_q^i\right)$$

Where:

- $\Delta\Phi\left(a_i, S\right)$: the reduction value of the impurity of S after a division on a_i.
- ϕ : the used measure of impurity.
- y : the attribute (variable) to predict.
- S_q^i: a subset of S which will be a branch of the node to be divided on the attribute a_i

The best known and most widely used criteria based on impurity are the Gini Index and the Information Gain.

Gini index: The Gini index measures the divergence between the distributions of attributes. It is mainly used in the CART algorithm (Breiman et al., 1984).

$$Gini\left(y,S\right) = 1 - \sum\nolimits_{c_i \in dom\left(y\right)} \left(p_{c_i}\right)^2$$

With, $p_{c_i} = \dfrac{n_{c_i}\left(S\right)}{n\left(S\right)}$ where, $n_{c_i}\left(S\right)$ is the number of elements of S with the class c_i.

We can define a criterion for an attribute a_i as follows:

$$GiniGain\left(a_i, S\right) = Gini\left(y,S\right) - \sum\nolimits_{q=1}^{Q} \frac{\left|S_q^i\right|}{\left|S\right|}.Gini\left(y, S_q^i\right)$$

Information Gain: It is a criterion based on impurity that uses the measure of entropy (derived from information theory). It was introduced in the ID3 algorithm (Quinlan, 1986). The more random a variable, the higher the entropy. For example, uniformly distributed variables have the highest entropy.

$$InformationGain\left(a_i, S\right) = Entropy\left(y,S\right) - \sum\nolimits_{q=1}^{Q} \frac{\left|S_q^i\right|}{\left|S\right|}.Entropy\left(y, S_q^i\right)$$

Where,

$$Entropy\left(y,S\right) = \sum\nolimits_{c_i \in dom\left(y\right)} - p_{c_i}.\log_2.p_{c_i}$$

We will therefore seek to maximize the gain of information since a greater gain implies a greater reduction of impurity, and hence a more interesting division.

Misclassification Error: Another measure of impurity rarely mentioned in the literature is the Misclassification Error, a very simple measure given by the following equation:

$$ME\left(y,S\right) = 1 - \max_{c_i \in dom\left(y\right)} \left\{p_{c_i}\left(S\right)\right\}$$

This measure has the same properties as the other impurity measures, one can therefore use it in a test of impurity.

$$MEGain\left(a_i, S\right) = ME\left(y,S\right) - \sum\nolimits_{q=1}^{Q} \frac{\left|S_q^i\right|}{\left|S\right|}.ME\left(y, S_q^i\right)$$

In building a decision tree, the goal is to minimize the error rate of classification in each node. So, it would be intuitive to use this measure. However, this measure is not sensitive enough when the number of elements is too small. The difference between the impurity between parent and child nodes can be 0, and the tree construction will stagnate. That does not apply to entropy and Gini index, thanks to the strict convexity of these functions.

HOEFFDING TREE

The Hoeffding Tree Algorithm (HT) is a decision tree learning method designed for classifying large data streams. The authors have proved in Domingos and Hulten (2000) that the resulting tree of this algorithm will converge to the same tree if the stream was stored and processed by a classical algorithm.

Much like conventional algorithms, ideally, it will need to determine the division of a node based on the entire data set. In streaming data, the data arrives continuously. The division is then determined by the impurity measures estimation (or depending on the used criterion). Of course, this estimate cannot be 100% sure. However, it should be as close as possible. That implies that the number of elements to be considered in the stream must be large enough to determine, with satisfaction, the validity of the division.

The authors proposed a new criterion based on the Hoeffding inequality (Hoeffding Bound) to determine if the number of examples seen is large enough to divide. The Hoeffding inequality ensures that an attribute chosen by applying a division on this sample would be the same as the one chosen from the entire set by a classical algorithm.

Operating principle: The algorithm incrementally builds a tree, gathering in the leaves enough information to be able to designate at a given moment the best attribute to transform these leaves into nodes. So, given a data stream, the former of the data will determine the root test. Then, the following examples will be passed to the corresponding leaves and used to designate the appropriate attributes there, and so on recursively. Each leaf, therefore, keeps a trace of the statistics of the stream examples which reached it. It determines the two best attributes according to the division criterion (Gini index as an example). A data structure called sufficient statistics stores the needed information. Sufficient statistics is a three-dimensional structure containing the number of data elements for each class and each attribute value (or each numeric interval in the case of numeric attributes).

Hoeffding limit: Domingos and Hulten (2000) have proposed a generic strategy for faster machine learning. The approach uses the Hoeffding limit (Hoeffding Bound) to find the minimum number of instances required in a sample to assert, with a degree of trust, an assumption on the whole data.

Let $\Delta g_i\left(S\right)$ be the division measure for the i^{th} attribute calculated for a sample of data S, collected in the considered decision tree leaf and let $n = \left|S\right|$ the number of data elements in S. The idea is to formulate an inequality in the following general form:

$\Delta g_i\left(S\right) - \Delta g_j\left(S\right) > \varepsilon\left(n,\delta\right)$

If i and j denote indices of the current best attribute and the second-best attribute respectively (having the best value of $\Delta g\left(S\right)$), then the inequality should guarantee, if satisfied, that with a probability of at least $1-\delta$ (δ chosen arbitrarily) the inequality is satisfied for the whole set of data.

The value of $\varepsilon\left(n,\delta\right)$ will be denoted ε and called the *confidence interval*. Several forms of the function $\varepsilon\left(n,\delta\right)$ have been proposed in subsequent research. The first one presented by Domingos and Hulten (2000) is the *Hoeffding inequality* and can be used as a division criterion in the generalized Hoeffding Tree algorithm.

Hoeffding inequality: The Hoeffding inequality, proposed by Hoeffding (1963), says that for each random variable Z with an extent R, the real mean of Z (noted $\bar{Z}$) does not deviate from the observed mean $\hat{Z}$ more than a value ε. This assumption is subject to an error rate of δ:

$$\left|\bar{Z} - \hat{Z}\right| < \varepsilon \text{, with } \varepsilon = \sqrt{\frac{R^2.\ln(\frac{1}{\delta})}{2n}}$$

Where, n is the number of observed instances.

This inequality poses two preconditions:

1. The random variable must be identically distributed and most likely bounded (since its range is used in the calculation).
2. The observations of the variable must be independent.

In the majority of algorithms using the Hoeffding limit principle, the variable Z is the value returned by the function that measures the profit of a division. By choosing a value δ, the Hoeffding inequality will state whether the number of instances seen (n) so far is sufficient to choose the best attribute.

VFDT

Very Fast Decision Tree (VFDT) is the implementation of the Hoeffding Tree algorithm, also proposed by Domingos and Hulten (2000). This implementation is a kind of system that introduces several parameters and practical improvements for real cases, especially those related to memory. The authors present in the following the most important parameters:

Initialization: VFDT can be initialized by a decision tree produced by a classical algorithm (such as C4.5). The model is trained on a sample of already available data. The tree can be entered as is or pruned to contain only the nodes that VFDT would have accepted, given the number of examples they contain.

Memory: VFDT can be set with a limit on usable memory, which is the only hardware hurdle of this algorithm. When the maximum available memory of VFDT is reached, the less promising leaves are deactivated to make a place for new ones. The algorithm can reactivate leaves if they become more promising than the currently active leaves.

The δ parameter: Also called split confidence, this is a user-chosen value. 1-δ denotes the desired probability to choose the correct attribute at each point in the tree. This parameter is usually fixed to a small value since a high probability (close to 1) of accuracy is desired. In the literature, the default value for this parameter is 10^{-7}.

The n_{min} parameter: It is computationally expensive to evaluate the information gain of the attributes after each training example. Since a single example will have little influence on the calculation results, it is reasonable to wait for more ones before re-evaluating. The n_{min} parameter also called grace period, dictates how many examples, since the last evaluation, must be seen in a leaf before a division attempt. This parameter allows to speed up the calculation without harming the precision. In the literature, the default value for this parameter is 200.

The A parameter: A situation may arise where one cannot choose the best attribute between two or more competing ones. If they are also equal and superior to some of the other split options, waiting too long to decide between them can do more harm than good to the precision of the tree. This situation can slow down tree growth. As a solution, the parameter τ was introduced by Domingos and Hulten (2000), also called the Tie-Breaking parameter. If the Hoeffding limit (ε) is small enough (less than τ), the node is immediately split on the current best attribute, regardless of how close the second-best attribute is. In the literature, the default value for this parameter is 0.05.

RELATED WORK

VFDT Improvements

The VFDT algorithm has been the subject of several studies that have proposed extensions or variations of the original approach. We focus in this section on the most recent one.

Bifet and Gavaldà (2009) present the Hoeffding Adaptive Tree (HAT). This approach constructs alternative subtrees when a drop-in performance is observed on a branch. If the alternative subtree achieves more satisfactory accuracy, it can replace the original one. The HAT uses an error estimator at each node. It determines whether the prediction error due to recent examples is significantly greater than the prediction error of a longer historical run. That can detect a concept drift.

Several research studies have aimed to optimize others criteria on data stream classification using decision trees. In Da Costa et al. (2018), the authors propose Strict VFDT (SVFDT). SVFDT is a novel algorithm that minimizes unnecessary tree growth, substantially reducing memory usage and keeping competitive predictive performance. The experiment results showed that the proposed algorithm obtained similar predictive performance and significantly reduced processing time and memory use. In García-Martín et al. (2021), the authors propose a new method to reduce the energy consumption of the VFDT algorithm with only minor effects on accuracy. The experiments show a real impact on energy consumption.

Several works, such as Jia (2020) and Ducange et al. (2021), introduce the concept of fuzzy logic to make VFDT more robust to noisy and vague data. Their experimental results show that the proposed approaches can effectively improve the accuracy of stream data classification, especially in the case of concept drift.

The most recent and significant improvement of VFDT is, without a doubt, the Hoeffding Anytime Tree algorithm (HATT). In Manapragada et al. (2018), the authors present the HATT's implementation, the Extremely Fast Decision Tree (EFDT). EFDT learns faster about data streams (compared to VFDT) while ensuring convergence to the same tree built on static data. It attempts to select and perform a division as soon as it has sufficient confidence in its usefulness. Then unlike HT, it will revisit that decision after some examples when a potential better division is available. HATT's strategy is more statistically efficient, learning faster from a stationary distribution while naturally handling the concept of drifting. It can replace the Hoeffding tree algorithm in most scenarios at a low resource cost.

Different works have taken advantage of the speed and efficiency of EFDT to solve classification problems in various fields, such as (Khine et al. (2020), Benllarch et al. (2021), Khairi et al. (2021)).

In this paper, the authors propose to improve the HATT algorithm for more efficiency by varying the criteria and the Hoeffding limit.

Hoeffding Criteria Variants

The correct use of the Hoeffding inequality is critical since a false division (especially for nodes close to the root) can seriously affect the performance of a classification model (Matuszyk et al., 2013). Due to its satisfactory results in practice, HAT is a meaningful step in this field. It is the basis of a large number of research and algorithms that have followed it. However, several researchers have pointed out some inaccuracies in Hoeffding's algorithm. The most important one is that the Hoeffding inequality is only correct for linear sums of random variables. We cannot express, for example, the information gain and the Gini index under this form. Hence, there is no theoretical justification to use the inequality of Hoeffding for these two criteria (Rutkowski et al., 2013). Attempts to replace or improve the Hoeffding limit started very quickly. In this paper, the authors will only focus on the more recent ones and those that have correctly demonstrated the flaws of this algorithm.

In Rutkowski et al. (2013), the authors recommend using the McDiarmid inequality. It is a generalization of Hoeffding's inequality, applicable for non-linear functions like entropy and the Gini index. However, this also leads to less precise estimates and, consequently, a higher number of instances to be observed to reach the desired level of confidence.

In their paper, Matuszyk et al. (2013) propose, among other things, a correction of the Hoeffding limit. The limit ε should be double that given by the Hoeffding inequality to guarantee the desired property. Thus, to guarantee (with an error rate δ) that the best attribute observed in the sample is the same as the one observed from the whole set, the limit of Hoeffding becomes:

$$\Delta g_i\left(S\right) - \Delta g_j\left(S\right) > 2.\varepsilon\left(n,\delta\right)$$

Besides, they propose a new criterion (AccuracyGain) based on a linear measure similar to the Misclassification Error. They express that measure as a sum of independent variables. They solve the problem cited with the Hoeffding limit. The experimental results show that the number of incorrect decisions is significantly smaller by using the double limit. However, the set of examples to see is much greater than the original limit.

Rutkowski et al. (2015) propose a new criterion based on the Misclassification Error measure. The characterization of this measure is that we can write it as a sum of independent variables, and therefore is considered linear. Thus, it solves the incorrect use of the Hoeffding limit cited in Rutkowski et al. (2013). In this work, the Hoeffding limit obtained by the Hoeffding inequality is equivalent to that obtained by the McDiarmid inequality.

The experimental results have shown satisfactory prediction accuracy, particularly in the early stages of tree construction. The algorithm learns faster at the start of training compared to other measures. That is because the Hoeffding limit of this measure is relatively smaller. However, towards the advanced stages of learning, the algorithm begins to slow down or even stagnate (due to its non-convexity). In the same paper, another hybrid criterion was proposed. It combines the misclassification error with the Gini index. The approach benefits from the speed of the misclassification error at the start of learning while avoiding stagnation towards advanced stages. Numerical simulations comparing a Hoeffding tree with the Gini index, another with the Misclassification Error, and a third with this hybrid criterion have shown a satisfactory performance of the hybrid criterion for a small additional computational cost.

CONTRIBUTION: IMPROVEMENT OF EFDT

The proposal in Manapragada et al. (2018) claims to be statistically more efficient than the original Hoeffding tree algorithm, further pushing the potential of Hoeffding Trees in a new direction. It would be interesting to revisit the various previous improvements of the basic algorithm and try to adapt them to HATT.

In this work, the authors propose three improvements to the EFDT algorithm (the HATT implementation). They resume in the following the main of their three propositions:

1. **Misclassification Error:** The application of the Misclassification Error has given satisfactory results for the HT algorithm. Experimenting with its effects on the HATT algorithm would be interesting since both (HATT and Misclassification Error) have an impact on the learning speed of the decision tree.
2. **Hybrid Criterion:** The authors propose to combine the Gini Index and the Misclassification error. That fills the weakness of the latter during the advanced phases of the learning.
3. **Variations of the Hoeffding inequality:** the HATT algorithm uses the same Hoeffding inequality proposed by Domingos and Hulten (2000). It is then interesting to vary the Hoeffding limit on HATT. With the EFDT algorithm, it will not be interesting to test the proposition of Matuszyk et al. (2013) (Double the confidence interval) since the division decisions will be re-explored. Thus, the decision errors with the Hoeffding limit will be caught up in future revisions.

The authors propose, in this work, to use half of the confidence interval. Although this is not justified, and that this inequality can induce more bad decisions, numerical simulations show that this variant is even better than the original Hoeffding tree (Rutkowski et al., 2020). We can also count on HATT's ability to re-explore its decisions to afford divisions that do not guarantee the desired properties and replace them if it found a better one.

To validate their results, the authors reproduce the experimental results of the authors of HATT (comparison between EFDT and VFDT). Then, they test their propositions using the same scenarios.

EXPERIMENTS AND RESULTS

Datasets

For their experiments, the authors chose to use the same datasets used by Manapragada et al. (2018). That would make it possible to directly visualize the impact of their improvements on the original EFDT algorithm. It also ensures that the choice of data sets conforms to those commonly used in the literature. Manapragada et al. (2018) compared the two algorithms on all UCI datasets (Dua and Graff, 2017) with more than 200,000 instances, having a clear prediction target, requiring no word processing, and containing no missing values. With these UCI data, in addition to the Airlines dataset proposed by MOA (Bifet et al., 2010), the authors have a total of 8 quite diverse reference datasets to use, described in table 1.

Table 1. Dataset's characteristics

	Datasets	Dataset characteristics	Attribute characteristics	Attributes number	Instances number
01	KDD intrusion detection	Multivariate	Categorical Numerical	42	4 000 000
02	Poker	Multivariate	Categorical Numerical	11	1 025 010
03	Skin	Univariate	Numerical	4	245 057
04	Forest covertype	Multivariate	Categorical Numerical	54	581 012
05	Gas sensor	Multivariate Time series	Numerical	11	919 438
06	Airlines	Multivariate	Categorical	8	539 383
07	PAMAP Activity Recognition	Multivariate Time series	Numerical	52	3 850 505
08	Hepmass	Synthetic	Numerical	28	10 500 000

Settings and Environment

MOA (Massive Online Analysis) (Bifet et al., 2010) is an open-source framework for dealing with growing and massive data streams. It is linked to the WEKA (Waikato Environment for Knowledge Analysis) project presented in Bouckaert et al. (2010). WEKA provides a workspace containing comparison and visualization tools for different algorithms for data analysis, classification, and other machine learning technologies. MOA provides the same tools as WEKA but adapted for a continuous data stream environment, whose requirements are different from the traditional batch learning environment.

The experiments are run in a Jupyter environment under a UNIX system (Ubuntu 18.04). The authors use Python version 3.7 and MOA version 2020.07. The tests are carried out on a local machine with the following characteristics: Intel Core i7 8th gen CPU @ 2.20GHz and 16 GB of RAM. This machine was sufficient to successfully perform the majority of tests, even for large data sets. The authors use the same default parameters for variants of VFDT and EFDT. Table 2 specifies the evaluation parameters and algorithms.

To compare between the different variants, the authors chose the following metrics:

Table 2. Experiments settings

Parameter	Value	Description
Evaluation method	Test-Then-Train	Each example is used to test the model before it is used for learning.
Evaluation frequency	1000	Performance metrics are calculated every 1000 instances.
Number of mixes	03	Number of mixed streams to test for each algorithm.
δ	10^{-7}	Error rate allowed in the choice of divisions in the Hoeffding algorithm.
n_{min}	200	The algorithm should see at least 200 instances between each split attempt.
n_{min} interne (R)	2000	Specific to EFDT, the interval between re-evaluations of internal nodes.
τ	0.05	Tie-Breaking value.

- **E (Error rate)**: value between 0 and 1, calculated by 1-AverageAccuracy. The authors chose accuracy since it was used in Manapragada et al. (2018).

- **T (Processor time):** execution time in seconds.

- **S (Number of divisions):** The number of divisions performed can be a good indicator of how fast a tree grows.

Experiment 1. EFDT McError

The first experiment consists of testing the performance of the Misclassification Error measure on the EFDT algorithm. According to the results of Rutkowski et al. (2015), measuring Misclassification Error allows the tree to grow faster at the start of training, requiring fewer items to divide. This property will speed up the EFDT algorithm to catch up when a decision is revised and changed. The authors compare in this experiment the original EFDT algorithm (EFDT InfoGain) with the EFDT McError algorithm, in addition to their equivalent on VFDT for reference.

The error rates of the McError are worse than the others (table 3 and figure 1). The reason is due to measurement stagnation (misclassification error property) in the advanced stages of learning. It can be further observed by the small number of divisions (S) in the case of the EFDT McError. The algorithm would block for a very long time in division attempts (costly in computing time) without (rarely) dividing. It is important to mention that the expected property is observed in VFDT McError, where the algorithm performs better in time and precision than VFDT InfoGain. This suggests that poor performance is only due to the possible blocking of the measurement. As previously mentioned, the McError metric typically crashes when the number of instances in leaves is very small or very out of balance, which will happen more often with EFDT because it starts new branches more often when it revisits a decision.

Experiment 2. Hybrid EFDT

The second experiment consists of testing the performance of the hybridization of the criterion based on the Misclassification Error with the Information Gain.

The results are presented in Figure 2 and detailed in Table 4. The **Hybrid EFDT** results are principally better compared to those of McError, in precision and in time. In some cases, the execution time is halved. The McError will often have more potential when used in combination with another metric (preventing it from blocking), be it in VFDT or EFDT. The experimental results show that the

Table 3. Results of the first experiment (Evaluation of the Misclassification Error)

Datasets	VFDT1e InfoGain			VFDT1e McError			EFDT1e InfoGain			EFDT1e McError		
	S	E	T	S	E	T	S	E	T	S	E	T
Pamap2 Activity Recognition	109	18.62%	604	1236	13.38%	705	5653	**7.36%**	**981**	1923	18.47%	1158
Forest covertype	35	32.45%	66	296	26.30%	65	1971	28.18%	162	389	28.18%	**121**
KDD intrusion detection	24	0.32%	183	113	0.07%	164	328	0.10%	**245**	133	**0.08%**	274
Gas Sensor	124	15.75%	41	406	10.87%	43	1742	**6.36%**	63	605	10.68%	**58**
Skin	19	2.04%	5	62	1.26%	5	103	**0.46%**	**6**	76	1.04%	6
Hepmass	761	14.85%	334	1675	15.19%	329	5708	15.66%	523	1700	**15.44%**	**491**
Poker	41	28.72%	31	360	17.75%	30	2619	**16.76%**	54	486	21.62%	**45**
Airlines	74	35.45%	19	28	35.67%	17	385	35.82%	27	56	**35.78%**	**21**

Figure 1. Performance of the Misclassification Error on different datasets

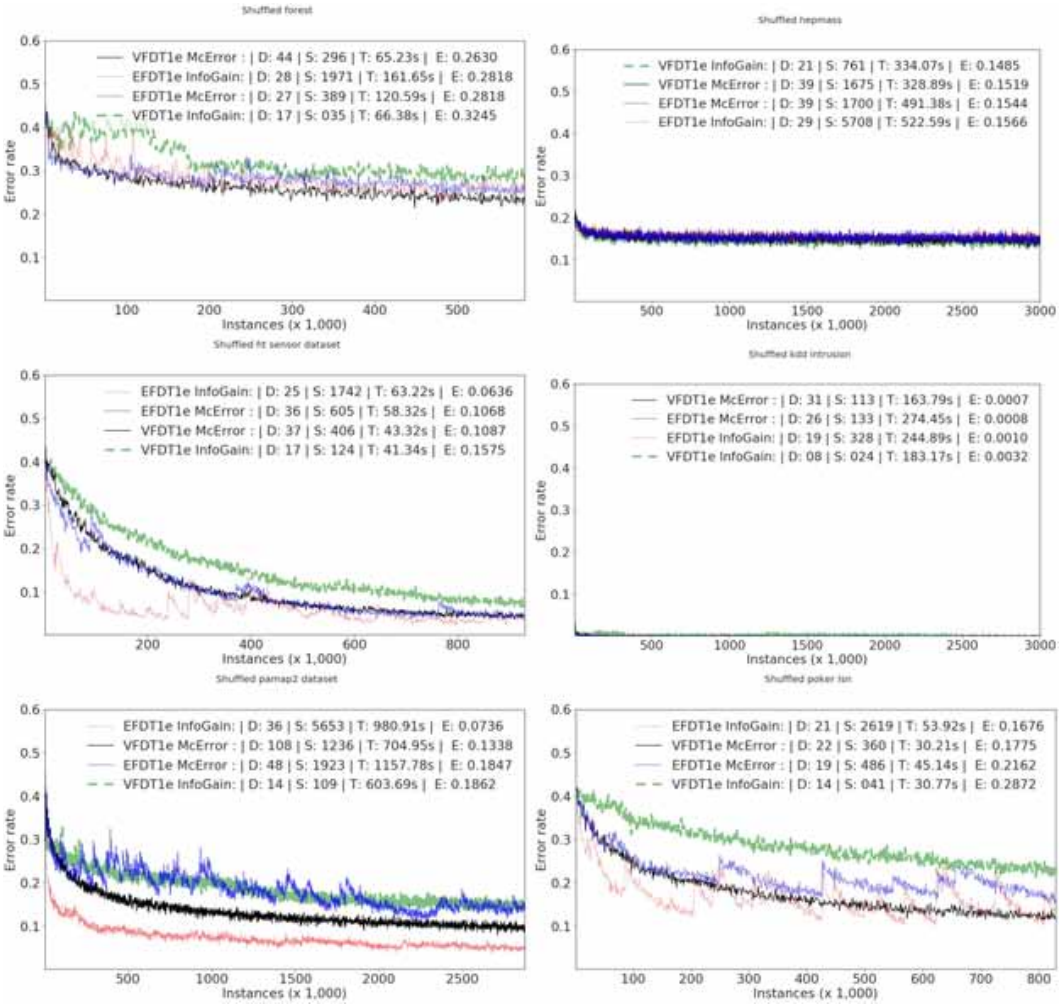

Table 4. Results of the second experiment (Evaluation of the hybrid criterion)

Datasets	VFDT1e McError			Hybrid VFDT			EFDT1e McError			Hybrid EFDT		
	S	E	T	S	E	T	S	E	T	S	E	T
Pamap2 Activity Recognition	1236	13.38%	705	1310	12.98%	397	1923	18.47%	1158	6445	**9.27%**	**552.2700**
Forest covertype	296	26.30%	65	318	25.88%	37	389	28.18%	121	2025	**28.15%**	**88.4800**
KDD intrusion detection	113	0.07%	164	117	0.07%	168	133	**0.08%**	274	464	0.13%	**221.6433**
Gas Sensor	406	10.87%	43	449	10.12%	28	605	10.68%	58	1780	**6.60%**	**37.5267**
Skin	62	1.26%	5	71	1.16%	5	76	1.04%	6	132	**0.51%**	**5.3800**
Hepmass	1675	15.19%	329	1780	15.20%	216	1700	**15.44%**	491	6462	15.71%	**324.3300**
Poker	360	17.75%	30	401	17.89%	21	486	21.62%	45	2519	**16.74%**	**32.8333**
Airlines	28	35.67%	17	109	35.36%	18	56	**35.78%**	21	656	36.02%	30.9600

proposed Hybrid EFDT variant provides very satisfactory results: making it possible to compensate well for the computational cost introduced by the EFDT algorithm of Manapragada et al. (2018) while keeping very high precision similar to the latter.

Experiment 3. Half Hoeffding EFDT

The last experiment consists of testing the performance of a different Hoeffding limit on EFDT. The confidence interval will be divided by 2, denoted Half Hoeffding. As mentioned, the authors believe that EFDT's ability to revise its decisions means that it can afford to choose a potentially less efficient division first and then reconsider it if a better one occurs. That, rather than waiting for the level of confidence guaranteed by the Hoeffding inequality to divide. This variant will be denoted by EFDThe, followed by the name of the measure used: EFDThe InfoGain or EFDThe McError.

The results are presented in Figure 3 and detailed in Table 5. We observe in this experiment results that the precision between EFDT1e and EFDThe never differs by more than 0.005. However, in all cases, the execution time is much lower, sometimes falling to less than half of the first. The EFDTHe variant remains better than VFDTHe in precision in almost all cases. That leads to conclude that the Half Hoeffding criterion is much more efficient to use with EFDT because it makes the most of the ability of EFDT to revise its decisions.

Comparison EFDT With the Three Proposed Variants

Table 6 compares the three proposed variants between them and with the original EFDT. Thus, EFDTHe and Hybrid EFDT results are very similar. They significantly reduce the execution time of the original EFDT while maintaining the same satisfactory precision. In addition to confirming the results of Rutkowski et al. (2015) (Misclassification Error and Hybrid criterion) and Rutkowski et al. (2020) (Half Hoeffding) on the VFDT, the results of the 2nd and 3rd variant showed a considerable improvement in the speed of the EFDT. These improvements manage better compensation for the additional processing cost of the EFDT algorithm against VFDT while keeping a precision similar to that won by the EFDT.

CONCLUSION AND PERSPECTIVES

Many modern data sources generate data as a continuous stream at a very high frequency and in enormous or potentially infinite amounts. Storing or traditionally processing them is no longer feasible. Data Stream Mining is the processing of these data streams through several algorithms and methods to extract information, classify or predict decisions immediately and quickly while performing a

Figure 2. Performance of Misclassification/Gini hybridization on different datasets

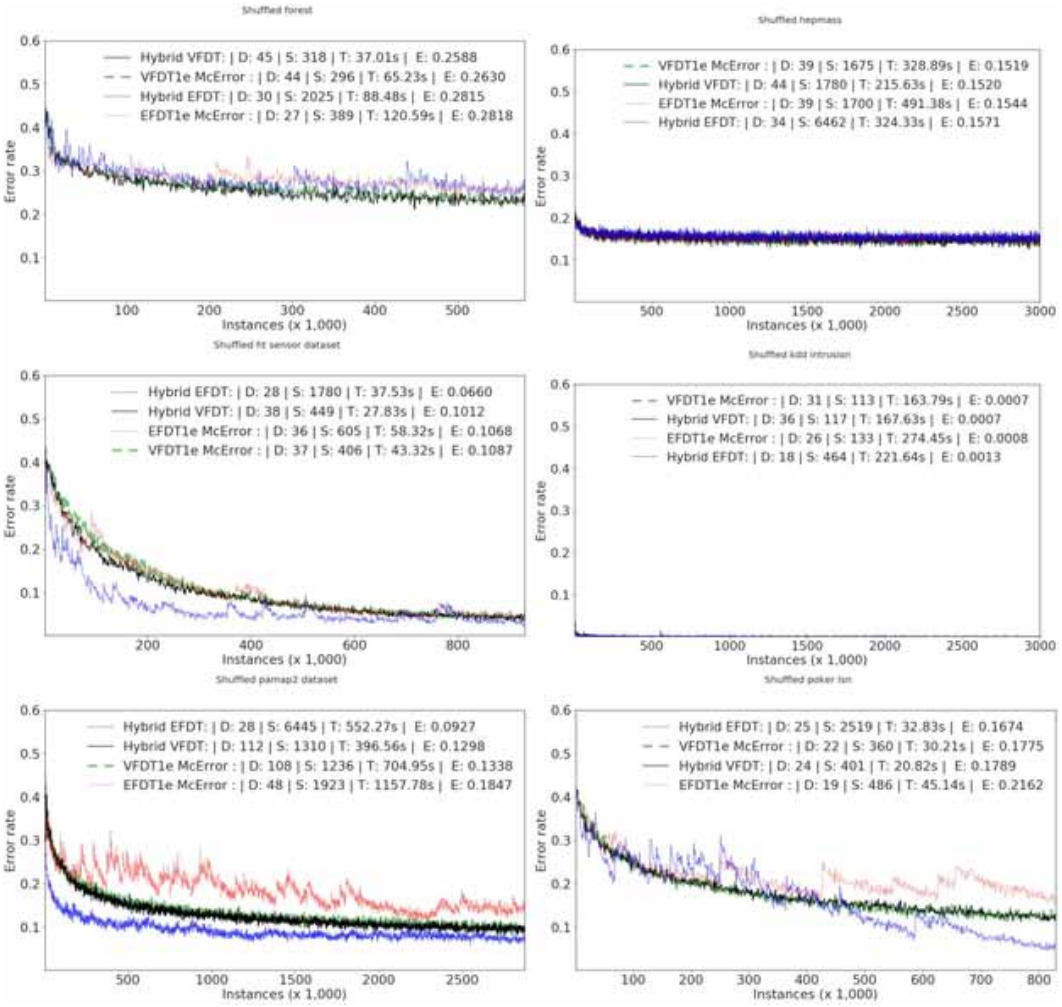

Table 5. Results of the third experiment (Evaluation of Half Hoeffding)

Datasets	VFDT1e InfoGain		VFDThe InfoGain		EFDT1e InfoGain		EFDThe InfoGain	
	E	T	E	T	E	T	E	T
Pamap2 Activity Recognition	18.62%	604	13.39%	315	**7.36%**	981	**7.70%**	**504**
Forest covertype	32.45%	66	28.26%	36	**28.18%**	162	**28.72%**	**91**
KDD intrusion detection	0.32%	183	0.15%	157	**0.10%**	245	**0.11%**	**230**
Gas Sensor	15.75%	41	7.80%	26	**6.36%**	63	7.19%	**36**
Skin	2.04%	5	0.95%	5	**0.46%**	6	**0.48%**	**5**
Hepmass	14.85%	334	14.89%	218	**15.66%**	523	**15.78%**	**296**
Poker	28.72%	31	19.34%	21	**16.76%**	54	**17.00%**	**33**
Airlines	35.45%	19	35.43%	15	**35.82%**	27	**35.93%**	**22**

Figure 3. Performance of the Half Hoeffding criterion on different datasets

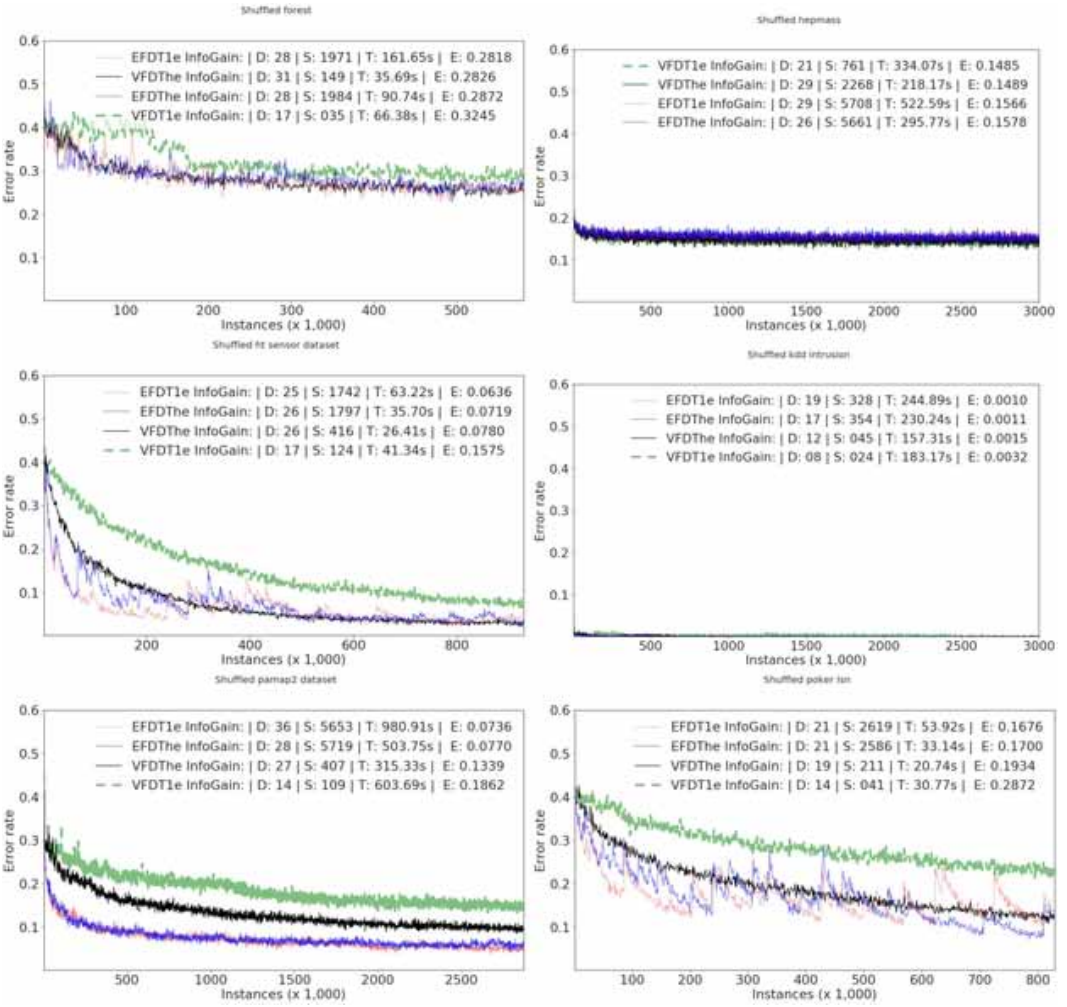

Table 6. Comparison between the 03 variants of the EFDT with the original one

Datasets	EFDT1e InfoGain		EFDT1e McError		Hybrid EFDT		EFDThe InfoGain	
	E	T	E	T	E	T	E	T
Pamap2 Activity Recognition	**7.36%**	980.9133	18.47%	1157.7800	9.27%	552.2700	7.70%	**503.7500**
Forest covertype	28.18%	161.6500	28.18%	120.5900	**28.15%**	**88.4800**	28.72%	90.7400
KDD intrusion detection	0.10%	244.8933	**0.08%**	274.4533	0.13%	**221.6433**	0.11%	230.2400
Gas Sensor	**6.36%**	63.2200	10.68%	58.3167	6.60%	37.5267	7.19%	**35.7033**
Skin	**0.46%**	5.5900	1.04%	5.9467	0.51%	5.3800	0.48%	**5.2233**
Hepmass	15.66%	522.5933	15.44%	491.3767	15.71%	324.3300	**15.78%**	**295.7667**
Poker	16.76%	53.9233	21.62%	45.1433	**16.74%**	**32.8333**	17.00%	33.1367
Airlines	35.82%	26.5967	**35.78%**	**20.7267**	36.02%	30.9600	35.93%	21.6367

single pass on the instances of the stream. The Hoeffding tree algorithm is considered the basis of decision trees in the processing of data streams.

In this work, the authors went through the main extensions and variants of Hoeffding trees. More specifically, on the aspect of the inaccuracy of the Hoeffding criterion, that does not always guarantee the desired properties. One of the relevant variants of this algorithm is the Hoeffding Anytime Tree or HATT, whose idea is to divide as soon as possible then, unlike the original algorithm, to revisit the decisions later and divide if a better division occurs. HATT presents, often, more satisfactory experimental results for an additional cost in processing time. The authors used the HATT algorithm (and its EFDT implementation) by projecting the different variations brought to the Hoeffding trees. They ended up with two different variants, denoted Hybrid EFDT and EFDT Half Hoeffding. These improvements better exploit the ability of the EFDT to revise its decisions, allowing them to provide precision very close to the original EFDT while considerably reducing the additional processing time of the latter. These two variants can, potentially, be added to the options to consider for classification tasks or as valuable elements in an aggregation of models such as Random forests.

The results of the proposed variants show that the idea of the Hoeffding Anytime Tree has even more potential to exploit. For example, we can modify the algorithm to allow it to use a criterion in the re-evaluations of decisions different from that used during the first attempts. In the end, the authors saw in the various corrections proposed to the Hoeffding criterion that a mathematical justification does not necessarily mean a better result. The Hoeffding limits given by experimental results (Hoeffding inequality and Half Hoeffding) are, often, more attainable. That leads to thinking that the search for the optimal limit is still open.

REFERENCES

Amudha, L., & Pushpalakshmi, R. (2021). Applications, Analytics, and Algorithms—3 A's of Stream Data: A Complete Survey. In *Intelligence in Big Data Technologies—Beyond the Hype* (pp. 599–606). Springer. doi:10.1007/978-981-15-5285-4_60

Bahri, M., Bifet, A., Gama, J., Gomes, H. M., & Maniu, S. (2021). Data stream analysis: Foundations, major tasks and tools. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, *11*(3), e1405. doi:10.1002/widm.1405

Benllarch, M., Benhaddi, M., & El Hadaj, S. (2021). Enhanced Hoeffding Anytime Tree: A Real-time Algorithm for Early Prediction of Heart Disease. *International Journal of Artificial Intelligence Tools*, *30*(03), 2150010. doi:10.1142/S021821302150010X

Bifet, A., & Gavaldà, R. (2009). Adaptive Learning from Evolving Data Streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII, IDA '09* (pp. 249–260). Lyon, France: Springer-Verlag.

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive online analysis. *Journal of Machine Learning Research*, *11*, 1601–1604.

Bouckaert, R. R., Frank, E., Hall, M. A., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2010). *WEKA—Experiences with a Java Open-Source Project*. Academic Press.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees. The Wadsworth Statistics/Probability Series. Wadsworth & Brooks/Cole Advanced Books & Software.

Choudhury, A. (2020). Predicting cancer using supervised machine learning: Mesothelioma. *Technology and Health Care*, (Preprint), 1-14.

Da Costa, V. G. T., de Leon Ferreira, A. C. P., & Junior, S. B. (2018). Strict Very Fast Decision Tree: A memory conservative algorithm for data stream mining. *Pattern Recognition Letters*, *116*, 22–28. doi:10.1016/j.patrec.2018.09.004

Deepa, M., & Sumitra, P. (2020). An enhanced classification approach for network intrusion detection using Hoeffiding induction tree algorithm. *European Journal of Molecular & Clinical Medicine*, *7*(09).

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '00* (pp. 71–80). ACM Press. doi:10.1145/347090.347107

Dua, D., & Graff, C. (2017). *UCI machine learning repository*. Academic Press.

Ducange, P., Marcelloni, F., & Pecori, R. (2021). Fuzzy Hoeffding Decision Tree for Data Stream Classification. *International Journal of Computational Intelligence Systems*, *14*(1), 946–964. doi:10.2991/ijcis.d.210212.001

García-Martín, E., Lavesson, N., Grahn, H., Casalicchio, E., & Boeva, V. (2021). Energy-aware very fast decision tree. *International Journal of Data Science and Analytics*, *11*(2), 105–126. doi:10.1007/s41060-021-00246-4

Hancock, T., Jiang, T., Li, M., & Tromp, J. (1996). *Lower Bounds on Learning Decision Lists and Trees*. Academic Press.

Hoeffding, W. (1963). Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, *58*(301), 13–30. doi:10.1080/01621459.1963.10500830

Jia, S. (2020). A VFDT algorithm optimization and application thereof in data stream classification. *Journal of Physics: Conference Series*, *1629*(1), 012027. doi:10.1088/1742-6596/1629/1/012027

Khairi, M. H., Ariffin, S. H., Latiff, N. M. A. A., Yusof, K. M., Hassan, M. K., Al-Dhief, F. T., & Hamzah, M. (2021). Detection and Classification of Conflict Flows in SDN Using Machine Learning Algorithms. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 76024–76037. doi:10.1109/ACCESS.2021.3081629

Khine, A. A., & Khin, H. W. (2020). Credit Card Fraud Detection Using Online Boosting with Extremely Fast Decision Tree. In *2020 IEEE Conference on Computer Applications (ICCA)* (pp. 1-4). IEEE.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

Manapragada, C., Webb, G., & Salehi, M. (2018). *Extremely Fast Decision Tree*. arXiv:1802.08780. 10.1145/3219819.3220005

Matuszyk, P., Krempl, G., & Spiliopoulou, M. (2013). Correcting the Usage of the Hoeffding In-equality in Stream Mining. In A. Tucker, F. Höppner, A. Siebes, & S. Swift (Eds.), *Advances in Intelligent Data Analysis XII* (pp. 298–309). Lecture Notes in Computer Science. Springer. doi:10.1007/978-3-642-41398-8_26

Oded, M., & Lior, R. (2014). *Data Mining with Decision Trees: Theory and Applications* (2nd ed.). World Scientific.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. doi:10.1007/BF00116251

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Rutkowski, L., Jaworski, M., & Duda, P. (2020). *Stream Data Mining: Algorithms and Their Probabilistic Properties. Studies in Big Data*. Springer International Publishing. doi:10.1007/978-3-030-13962-9

Rutkowski, L., Jaworski, M., Pietruczuk, L., & Duda, P. (2015). A New Method for Data Stream Mining Based on the Misclassification Error. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(5), 1048–1059. doi:10.1109/TNNLS.2014.2333557 PMID:25051560

Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision Trees for Mining Data Streams Based on the McDiarmid's Bound. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1272–1279. doi:10.1109/TKDE.2012.66

Soe, Y. N., Feng, Y., Santosa, P. I., Hartanto, R., & Sakurai, K. (2020). Towards a lightweight detection system for cyber-attacks in the IoT environment using corresponding features. *Electronics (Basel)*, *9*(1), 144. doi:10.3390/electronics9010144