Energy Efficiency in Data Stream Mining

Eva García Martín*, Niklas Lavesson†, Håkan Grahn‡
Dept. of Computer Science and Engineering
Blekinge Institute Technology
Karlskrona, Sweden
Email: *eva.garcia.martin@bth.se, †niklas.lavesson@bth.se, ‡hakan.grahn@bth.se

***Abstract*—Data mining algorithms are usually designed to optimize a trade-off between predictive accuracy and computational efficiency. This paper introduces energy consumption and energy efficiency as important factors to consider during data mining algorithm analysis and evaluation. We extended the CRISP (Cross Industry Standard Process for Data Mining) framework to include energy consumption analysis. Based on this framework, we conducted an experiment to illustrate how energy consumption and accuracy are affected when varying the parameters of the Very Fast Decision Tree (VFDT) algorithm. The results indicate that energy consumption can be reduced by up to 92.5% (557 J) while maintaining accuracy.**

I. Introduction

Data stream mining is gaining importance with the evolution of hardware, sensor systems and technology. The rate at which data is generated is increasing day by day, challenging storage and computational efficiency [1]. Digital Universe Study[1] has predicted that by 2020, 40,000 exabytes of data will be processed, most of them originating from devices that automatically generate data. Many algorithms in data stream mining are designed to process fast and potentially infinite streams [2], [3].

Traditionally, the machine learning community has considered accuracy as the main factor when building algorithms. With the appearance of data stream mining, scalability has also been a key factor to consider. In this context, scalability stands for how fast an algorithm can process the incoming data. The problem that we address in this study is the fact that few researchers in the data mining community consider energy consumption as an important measure.

It has been shown that energy consumption can be reduced in every layer of the Open Systems Interconnection (OSI) model [4], [5]. Hardware solutions to reduce energy consumption have been focused on, e.g. using Dynamic Voltage Frequency Scaling (DVFS) and on parallel computing [6], [7]. Software solutions have also been developed in order to reduce the energy consumption of applications, although not in such depth as the hardware solutions. However, no solutions related to data stream mining have addressed energy consumption as a key factor, instead they have been centered towards specific applications.

This paper introduces energy consumption and energy efficiency as important factors to consider during data mining algorithm analysis and evaluation, and to demonstrate the use of these factors in a data stream mining context. The consideration of energy efficiency can help companies and researchers move towards green computing [8] while improving business profits.

We conducted an experiment to illustrate a possible scenario where energy consumption is relevant to study. More specifically, we studied how energy is affected by changing the parameters of the VFDT (Very Fast Decision Tree) algorithm [2]. The results indicate that it is possible to reduce the energy consumption of the VFDT algorithm by up to 92.5% while maintaining similar levels of accuracy. The main contribution of this paper is the introduction of energy consumption as a key factor to consider in data mining algorithms. This is supported by an experiment that illustrates an example of sustainable and efficient algorithms.

II. Background

In this section we first explain the importance of energy consumption in data mining. Then, we briefly explain data stream mining and why it is different from standard data mining, and finally we introduce some terminology related to power, energy, energy efficiency, and computational efficiency.

A. Energy-awareness

The demand for energy is increasing day by day [4]. World leaders and scientists focus on finding a solution towards this problem, centering on two key factors: developing new sources of clean energy and decreasing energy usage [9], [10], which would lead to a reduction in CO_2 emissions. The main reason why researchers and every citizen should be aware of energy consumption is because energy pollutes. Every device that we use on a daily bases that consumes energy produces CO_2. Nowadays, based on a study conducted by the World Health Organization, air pollution kills more people than malaria and aids combined [11].

There was a study conducted by Alex Wissner-Gross measuring the CO_2 emissions of a search query on a search engine. Considering that there are approximately 66k Google queries per second[2], reducing the CO_2 emissions of search queries will significantly impact the environment. If we translate this example to data stream mining, we can picture the execution of data stream mining algorithms in servers running 24 hours a day, for a complete year. In this case, building energy-aware algorithms has the following consequences:

- Reduction of CO_2 emissions to the atmosphere.

[1] https://www.emc.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf

[2] http://www.statisticbrain.com/google-searches/

- Reduction of air pollution, therefore reduction on the number of deaths per year due to this matter.
- Reduction of the money spent on energy.
- Increase of the battery life of mobile devices and sensor networks, if the algorithm is implemented in such scenarios.

B. Data Stream Mining

Data stream mining is the process of building models by exploring and extracting patterns from a stream of data.

The core assumption of data stream mining, in comparison to data mining, is that the examples are inspected only once, so we have to assume that if they are not processed immediately they are lost forever [12], [13]. Moreover, it is considered that the data arrives online, with no predefined order, at a high-speed and with time-changing characteristics. Data stream mining algorithms should be able to process potentially infinite streams while updating the model incrementally [3], [14].

C. Terminology

In this section we clarify several concepts related to energy, power and efficiency. Energy is a measurement of the amount of *fuel* used for a specific application. It is measured in Joules (J) or kWh. Power is a measurement of the rate at which energy is consumed. It is measured in Joules/second, which is equal to Watts (W). The following is an example that illustrates the relationship between power and energy: A process is running for 3.94 seconds consuming an estimate power of 18.1 mW. The total energy consumed is: $3.94 \times 0.0181 = 0.07J = Ws = 19.9 \times 10^{-}6$ Wh.

Energy efficiency has a specific definition at Green500[3], being, "*The amount of operations per watt a computer can perform*". This definition is related to hardware. In this study, whenever we mention energy efficiency we refer to reducing the energy consumption of some process or algorithm.

In theoretical machine learning, researchers introduced the computational learning theory [15], where they analyze the computational complexity of algorithms. They approach computational efficiency as a way of designing less computationally complex algorithms that can run in polynomial time.

III. Related Work

In this section is we first review literature related to energy awareness in software and hardware. Then, we examine relevant works in the data stream mining field, focusing on the VFDT algorithm. Finally, we review papers that are related to both energy consumption and data stream mining.

Research in energy awareness at the software level started many years ago, when researchers began to realize the importance of the energy consumed by a software application. In 1994, the first systematic attempt to model the power consumption of the software components of a system was presented [16]. After that, in 1999, PowerScope was presented [17], a software tool for profiling the energy usage of applications. The novelty of this approach is that energy consumption can be mapped to program structure to analyze which procedures consume more energy. Companies such as Microsoft[4] and Intel[5] have invested in developing software tools to help developers reduce the energy consumption of their applications.

In relation to energy efficiency at the hardware level, one of the most important techniques, implemented in most contemporary processors, is Dynamic Voltage Frequency Scaling (DVFS). DVFS is a power saving technique used and improved by many researchers. One improvement is Real Time DVFS, an implementation of DVFS for real time systems [6]. Based on Koomey's law, the computation per kWh doubles every 1.57 years. Therefore, computers are able to execute applications faster with less energy consumption [18]. Another area that is gaining importance nowadays is parallel computing, where there are relevant energy savings by employing more cores on a processor [7]. Several energy-saving approaches, such as "Cost optimization for power-aware computing" have been developed in the past years [4].

In relation to data stream mining, researchers have developed efficient approaches to mine data streams, as outlined below. There have been several reviews conducted in data stream mining since 2005. Two general reviews [1], [19], portray techniques and concepts such as data-based techniques, task-based techniques, data stream classification and frequent pattern mining. More specific reviews center on topics such as sensor networks [20] and knowledge discovery [14].

From the reviews explained above, we have extracted six main techniques and approaches in data stream mining: Data stream clustering [21], Data stream classification [2], Frequent Pattern Mining [22], Change Detection in data streams [23], [24], Sliding window techniques [25] and Stream mining in sensor networks [20], [26]. We have decided to focus in Data Stream classification and change detection in data streams.

Concept drift refers to a change between the input data and the target variable on an online supervised learning scenario. The first framework that dealt with concept drift was proposed to also address efficiency and robustness [27]. Nowadays, researchers consider concept-drift an important aspect when building algorithms for other specific purposes. A survey on different methods that address concept drift has been conducted in 2014 [28].

Classification is considered a challenging problem in data stream mining [19]. The main reason is that many of the traditional classification techniques and algorithms were designed to build models from static data.

One of the key breakthroughs in supervised online learning was made with the development of the Hoeffding Tree algorithm and the Very Fast Decision Tree (VDFT) learner [2]. In contrast to previous algorithms, such as SPRINT [29] and ID5R [30], this new approach was able to deal with potential infinite streams, arriving at a fast pace and with low computational cost. The VFDT learner is able to process examples at a high rate in constant time. One year later, the same authors created a new version of the VDFT algorithm, CVFDT, that was able to adapt to concept-drift [3]. Another

[3] www.green500.org

[4] http://research.microsoft.com/apps/pubs/default.aspx?id=166288

[5] https://software.intel.com/en-us/energy-efficient-software

extension on the VFDT algorithm appeared two years later, with a new decision tree learner that could efficiently process numerical attributes [31]. In the same line, a decision tree algorithm was created for spatial data streams [32].

We would like to mention relevant methods that address different classification problems, namely: On-Demand classification [23], [33], Online Information Network (OLIN) [34], LWClass [35], ANNCAD [36] and SCALLOP [37].

In relation to energy awareness in data stream mining, several researches have conducted studies where they emphasize the importance of energy consumption [1], [38], [39]. While the first two are concerned on energy saving for sensor networks, the second one centers on examine the energy consumption of different data analysis techniques. To the best of our knowledge, the last work is the one most related to ours.

We can observe how there is no specific research on making energy consumption a key factor on data stream mining, since the research has been centered towards specific applications or hardware modifications. We would like to change this approach by proposing energy consumption as the new factor to consider when building, optimizing or creating new algorithms in data stream mining. We believe that this the next natural step to take, since other researchers in similar fields, hardware and software, have already taken that step.

IV. Proposing Energy Efficiency as Key Factor in Data Stream Mining

Data stream mining is gaining a lot of interest in the research community with the appearance of big data. Research conducted in this area has been mainly focused on improving two factors: the accuracy of the model and the maximum speed at which the model is able to process the data. Nowadays, online learners are able to process fast streams of data in constant time, consume low memory, deal with concept drift and have a high accuracy. However, most of the studies do not consider energy consumption.

We argue that energy consumption is a key factor when dealing with data stream mining problems. Therefore, it should be considered as a top priority when building data stream mining models, together with accuracy and scalability.

We propose a generic framework based on the CRISP (Cross Industry Standard Process for Data Mining) model [40]. CRISP is a data mining process model that clearly describes how to approach data mining tasks. The framework, portrayed in Figure 1, has seven phases: business understanding, data understanding, data preparation, energy measurement, modeling, evaluation, and deployment. They are outlined below:

1) *Problem Definition*: The first step is to clearly define the problem, by getting an understanding of the objectives and requirements from a business perspective.
2) *Data Understanding*: This phase covers data collection, understanding the data and discovering interesting features of the data. We need to identify how to obtain the dataset, or find a way to synthetically generate it.
3) *Data Preparation*: In this step we select and clean the data that we consider relevant for the problem.

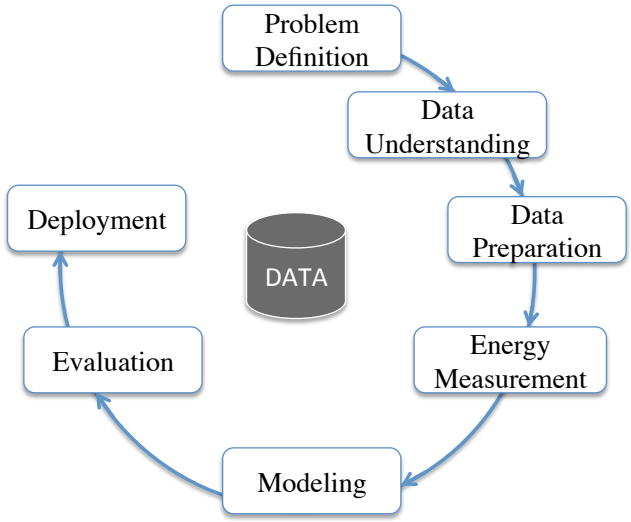

Fig. 1. Generic Framework. Based on CRISP-DM [40]

4) *Energy Measurement*: This step is not part of the CRISP model but directly relates to our goal of measuring energy consumption of data mining algorithms. We need to select a tool to measure energy that, first, estimates the power or energy at the process level and second, is able to run during the exact execution time of a process.
5) *Modeling*: In this step we create the model for the task defined in step 1. We need to choose and motivate the choice of a specific algorithm and parameters. While the model is running, we need to run in parallel the tool that estimates power or energy, to know exactly the amount of energy consumed by the algorithm. We have to find a way to input the execution of the algorithm to the power estimate tool.
6) *Evaluation*: This step centers on evaluating the results based on accuracy and the requirements defined on step one. The researcher needs to also review the process in case there is some missing point.
7) *Deployment*: In this last step we need to organize the results obtained from the model and transform it to knowledge. The way we propose to do this is by creating a script that automates the complete process. Therefore, it can input the data mining modeling algorithm into the energy estimate tool, and then output the results in a file or report showing key values and comparisons.

The goal with this generic framework is to present a simple procedure that any researcher could follow when building data mining or data stream mining models. CRISP is currently the state-of-the-art and standard framework for researchers and companies [41].

V. Experimental Design

The goal of this section is to map the general steps from the framework portrayed in Figure 1, to a concrete experiment that illustrates how different parameter configurations affect energy consumption.

A. Problem Definition

The goal of this experiment is to measure the energy consumption of different parameter configurations on the VFDT algorithm [2]. We compare how energy varies against accuracy when changing the parameters of the VFDT algorithm. Ideally, we want to find a situation where accuracy is maintained while energy consumption significantly decreases.

B. Data Understanding

In this experiment we use synthetic data generated with a stream generator function in MOA [42]. We have generated three different synthetic data streams of 1 million instances each and mapped them to three scenarios. In all scenarios, we compute the measurements for the different configurations of the VFDT algorithm. In *Scenario 1*, the data is generated with the Random Tree Generator. This generator creates a decision tree by randomly splitting the nodes with the different attributes, as explained in the VFDT implementation [2]. In *Scenario 2*, the data is generated with the Agrawal function, creating a binary classification dataset with nine attributes [43]. Finally, in *Scenario 3*, the dataset is generated with the SEA stream generator function [44].

C. Data Preparation

Since the data has been synthetically generated, there is no need to clean the data for this experiment. However, we propose it as a relevant step in a process with real-world data.

D. Energy Measurement

We have chosen PowerTop[6] to measure the power estimate of the experiment. PowerTop is an open source tool released by Intel in 2007 to measure and minimize the power consumption of a computer running on battery power. PowerTop fulfills the requirements defined in Section IV, since it shows in detail the power estimate for different processes during a specific workload. A workload is defined as the execution of a program or script. In this experiment PowerTop is used to measure the power estimate of a task in MOA.

E. Modeling

This subsection describes the specific configuration choices for the experiment, the tool used to build the model and a detailed explanation and motivation of the VFDT algorithm.

1) Parameter choice: Table I summarizes the different parameter configuration of the VFDT algorithm. The meaning of the different parameters is explained in the next subsection. For each parameter configuration, we measure the following values: Elapsed time, CPU usage, power estimate (W), accuracy (percentage of correctly classified instances) and energy (J). Each combination of streamer, parameter configuration and measure (CPU, Power...) is executed 10 times and then the median, mean, maximum, minimum and standard deviation are calculated.

In terms of hardware, on each scenario the battery was charged to at least 95%, Bluetooth and WiFi were turned off. No other program apart from the operating system was running on the laptop and the screen was always on. The experiment was conducted on a Linux machine running Ubuntu 14.04 with a 2.2 GHz Intel Core i7 processor and 16GB of RAM.

2) VFDT: The VFDT algorithm is based on Hoeffding Trees [2]. The Hoeffding tree method proposes to build a decision tree recursively, by picking the best attribute at each node and leaf. In order to pick the best attribute, all attributes from n examples are evaluated based on some heuristic G. The key and novel idea is that the authors use the Hoeffding Bound [45] to compute the value of n, to ensure that the attribute chosen with n examples is the same that would be chosen using infinite examples.

The VFDT is a decision tree learning system based on the Hoeffding Tree algorithm with a number of refinements: ties, G computation, memory, poor attributes, initialization and rescans. We focus in two parameters, ties and the G computation, since we vary them in the experiment. Ties occur when several attributes have similar G. The user can therefore specify a threshold value, τ. This means that if ΔG is lower than τ, hence there is a low difference between G values, there will be a split on the current best attribute. Moreover, computing a G value for every new example is inefficient. Therefore, users can specify an $nmin$ value (parameter -g in MOA). Being $nmin$ the *"minimum number of new examples that a leaf must accumulate before recomputing G"*.

3) MOA: Massive Online Analysis (MOA)[7] is a state-of-the-art framework for data stream mining. It is written in Java and developed at the University of Waikato [42]. We have chosen MOA to perform the complete experiment, from building the model to evaluating the results.

MOA is a feature rich and powerful tool that gives the user the opportunity to build machine learning models on data streams. It has a wide collection of machine learning algorithms, from Bayesian classifiers to concept-drift classifiers. It also includes stream generators to generate synthetic data, a property that is useful to test different algorithms for scalability and performance on big data sets.

4) Motivation for choosing the VFDT learner: There are several reasons why we chose the VFDT algorithm for our experiment. It is an algorithm widely known by many researchers that made a breakthrough in data stream mining when it was presented. Even though many researchers made relevant changes to this algorithm to increase its performance, it has not been tested against energy consumption before. We want to illustrate the reader how we can obtain interesting results by varying certain algorithm parameters.

F. Evaluation

The novel aspect of this experiment is how the classifier was evaluated. Traditionally, classifiers were evaluated in terms of performance, being, the number of instances the model correctly classified. However, our aim is to evaluate not only the accuracy, but also the energy consumption of the classifier. All the measurements about the energy consumption, CPU usage, accuracy and elapsed time are presented in Table II.

[6] https://01.org/powertop

[7] http://moa.cms.waikato.ac.nz/

G. Deployment

In order to compute the energy measurements, we coded a Bash Shell script. This script extracts the measured results from PowerTop of each MOA task, for a number of N iterations. As an output, it calculates the average of the results.

There are four main actions executed in the script:

1) Compute the power estimate with PowerTop of the execution of algorithm *i* in MOA.
2) Repeat *step 1* for *N-1* times.
3) Compute the average, mean, standard deviation, maximum, minimum and median of such N executions.
4) Repeat *steps 1-3* for *I* times, *I* being the number of setups or scenarios we want to test.

VI. Results

In this section we present the numerical results obtained from the experiment explained in Section V. Table I introduces the way we configured the parameters for each execution in the experiment. An execution corresponds to running the VFDT algorithm with a specific parameter configuration. These executions are mapped to a certain index, that will later be used in Table II and in the figures in Section VII. For instance, index E represents the execution of VFDT algorithm setting $nmin$ to 1400, while setting the other parameters to default. These executions are repeated for each scenario. We have designed a total of 19 configurations per scenario, six varying $nmin$, four varying $tie\ threshold$, and eight varying the numerical estimator. As can be observed, when varying one parameter the others are set to their default values.

The results of the experiment and all configurations across all scenarios are presented in Table II. For each index, we show the average and standard deviation (in parentheses) of the accuracy, power, time and energy. The average and standard deviation are computed from running each setup 10 times. The best result for each measure on each scenario is highlighted in ***bold***. Although the best parameter in terms of energy, power and time is the VFML set to 10 bins, it also has a 32.5% lower accuracy on average than the default setup. The best overall parameter considering accuracy and energy is $nmin = 1100$. Section VII explains more in depth the results and the behavior of the data.

VII. Analysis

In this section we detail several analyses extracted from the results presented in Section VI. We start with a general analysis between all parameters and end with specific analyses for each group of parameters.

A. General Analysis

A comparison between energy consumption and accuracy for each scenario, is presented in Figures 2, 3 and 4. The main conclusion that these illustrations suggest is that varying the parameters correctly, energy decreases significantly while maintaining similar levels of accuracy. The biggest gain in energy without decreasing accuracy is of 557 J (92.51%), obtained in *Scenario 3*. The results also indicate that, on average, increasing the value of $nmin$ is the top energy saving measure. The worst configuration across all scenarios is setting the numerical estimator Greenwald-Khanna to 1,000 tuples.

TABLE I. Parameter Configuration Index

INDEX	NMIN	TIE THRESHOLD	NUM. ESTIMATOR
A	Default (200)	Default (0.05)	Default (Gauss. 10 split points)
B	500	Default	Default
C	800	Default	Default
D	1,100	Default	Default
E	1,400	Default	Default
F	1,700	Default	Default
G	2,000	Default	Default
H	Default	0.01	Default
I	Default	0.03	Default
J	Default	0.07	Default
K	Default	0.09	Default
L	Default	Default	VFML 10 bins
M	Default	Default	VFML 100 bins
N	Default	Default	VFML 1,000 bins
O	Default	Default	Gaussian 100 split points
P	Default	Default	Gaussian 1,000 split points
Q	Default	Default	Greenwald-Khanna 10 tuples
R	Default	Default	Greenwald-Khanna 100 tuples
S	Default	Default	Greenwald-Khanna 1,000 tuples

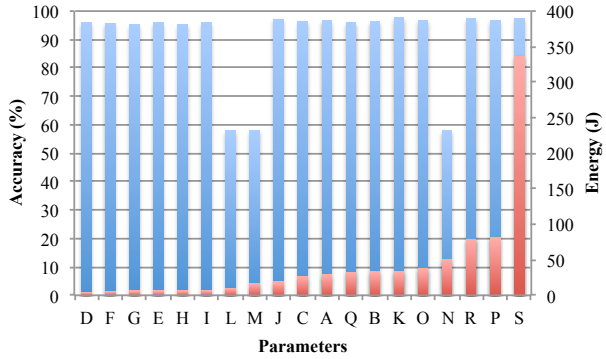

Fig. 2. Comparison between energy (red) and accuracy (blue) for different parameters configuration. Sorted from the lowest to the highest energy consumption parameter in *Scenario 1*.

B. Parameter $nmin$

In this subsection the behavior of parameter $nmin$ is analyzed. In the initial implementation of the Hoeffding Tree algorithm, the heuristic G would be recomputed on the arrival of new instances. However, it is unlikely that one sample affects the value of G. For that reason, the authors suggest to wait $nmin$ examples to recompute G to save computational effort. The default value of $nmin$ is 200. We have tested the following values of $nmin$ to observe how accuracy and energy were affected: 200, 500, 800, 1100, 1400, 1700, and 2000.

In *Scenario 1* and *Scenario 3* we observe how energy decreases when $nmin$ increases. Figure 5 suggests a significant decrease of 22.03 J (82.67 %) when $nmin$ is bigger than 800, while slightly decreasing accuracy by less than 1%. This tendency is expected, since as the authors suggest [2], it is computationally costly to recompute G. As shown in Table II, the decrease in energy is due to a significant decrease on power.

TABLE II. EXPERIMENTAL RESULTS

	SCENARIO 1				SCENARIO 2				SCENARIO 3			
PR[1]	ACC [2]	P(W)[3]	T(s)[4]	E(J)[5]	ACC	P(W)	T(s)	E(J)	ACC	P(W)	T(s)	E(J)
A	96.63 (0.02)	6.86 (1.36)	4.20 (0.53)	28.84 (7.14)	95.07 (0.02)	27.27 (1.72)	3.45 (0.47)	93.70 (11.40)	89.66 (0.02)	29.69 (1.61)	2.39 (0.27)	70.68 (5.49)
B	96.37 (0.02)	8.63 (0.80)	3.85 (0.41)	33.10 (3.45)	95.08 (0.02)	25.88 (1.11)	3.16 (0.35)	81.92 (10.35)	**89.78 (0.02)**	29.23 (2.47)	2.23 (0.26)	65.81 (14.33)
C	96.36 (0.01)	6.90 (2.57)	3.97 (0.61)	26.65 (9.94)	95.08 (0.02)	30.37 (2.73)	3.10 (0.27)	93.77 (9.32)	89.74 (0.03)	28.54 (1.35)	2.11 (0.03)	60.23 (3.72)
D	96.06 (0.02)	**1.19 (0.73)**	3.81 (0.44)	**4.62 (2.91)**	95.08 (0.01)	29.86 (3.48)	3.07 (0.32)	91.02 (7.20)	89.76 (0.02)	28.23 (1.01)	2.06 (0.03)	58.25 (2.63)
E	95.92 (0.02)	1.80 (0.21)	3.65 (0.23)	6.53 (0.61)	95.08 (0.03)	26.57 (1.67)	3.13 (0.15)	83.28 (9.52)	89.72 (0.04)	28.52 (1.07)	2.10 (0.04)	59.89 (3.05)
F	95.68 (0.02)	1.60 (0.20)	3.58 (0.33)	5.75 (1.04)	95.08 (0.02)	26.56 (2.03)	3.18 (0.36)	84.42 (9.85)	89.67 (0.04)	29.49 (1.74)	2.20 (0.24)	65.12 (11.36)
G	95.29 (0.02)	1.70 (0.16)	3.76 (0.59)	6.35 (0.97)	95.08 (0.02)	32.04 (1.94)	2.95 (0.06)	94.31 (4.63)	89.68 (0.04)	30.04 (1.84)	2.15 (0.24)	65.07 (12.12)
H	95.32 (0.02)	1.64 (0.07)	4.13 (0.07)	6.78 (0.41)	94.76 (0.02)	27.98 (2.36)	3.58 (0.38)	100.26 (13.89)	89.19 (0.02)	27.99 (1.33)	1.57 (0.05)	43.97 (3.43)
I	95.92 (0.02)	1.62 (0.64)	4.40 (0.71)	6.94 (2.61)	94.87 (0.03)	27.66 (1.66)	3.52 (0.37)	97.24 (11.15)	89.49 (0.03)	31.63 (5.08)	2.09 (0.22)	65.77 (10.93)
J	97.09 (0.01)	4.52 (1.89)	4.16 (0.06)	18.81 (7.80)	**95.09 (0.01)**	27.28 (1.49)	3.12 (0.06)	85.13 (5.80)	89.49 (0.04)	32.41 (1.48)	2.65 (0.33)	85.76 (8.56)
K	**97.60 (0.01)**	7.45 (0.39)	4.43 (0.51)	33.13 (5.20)	95.04 (0.03)	27.40 (1.47)	3.35 (0.38)	91.92 (13.09)	89.77 (0.02)	36.11 (3.27)	2.92 (0.27)	104.96 (6.65)
L	57.83 (0.04)	5.74 (0.89)	**1.79 (0.33)**	10.22 (2.16)	67.22 (0.04)	**20.48 (1.58)**	**2.07 (0.15)**	**42.54 (6.87)**	64.41 (0.03)	26.13 (1.78)	**0.74 (0.07)**	**19.54 (3.28)**
M	57.81 (0.05)	6.16 (0.61)	2.65 (0.27)	16.31 (2.30)	67.20 (0.05)	27.20 (0.94)	2.97 (0.23)	80.76 (4.21)	64.42 (0.05)	**22.02 (1.85)**	1.35 (0.14)	29.85 (5.80)
N	57.82 (0.05)	4.88 (0.65)	10.40 (0.50)	50.90 (8.56)	67.22 (0.02)	25.49 (0.56)	9.10 (0.59)	231.72 (11.48)	64.39 (0.05)	27.00 (0.75)	6.03 (0.36)	162.62 (8.73)
O	96.82 (0.02)	7.58 (0.73)	4.91 (0.71)	37.04 (4.84)	95.08 (0.02)	29.42 (2.14)	3.98 (0.11)	117.03 (6.12)	89.76 (0.04)	30.24 (0.66)	3.35 (0.02)	101.16 (2.49)
P	96.74 (0.01)	6.52 (0.78)	12.53 (1.09)	81.40 (9.96)	95.08 (0.01)	26.86 (0.90)	10.44 (0.86)	280.30 (23.42)	89.65 (0.03)	27.02 (0.35)	13.87 (1.00)	374.82 (27.67)
Q	96.06 (0.02)	7.48 (0.70)	4.35 (0.47)	32.48 (3.81)	94.52 (0.03)	27.33 (1.75)	3.47 (0.59)	95.17 (19.25)	89.10 (0.02)	28.69 (0.85)	2.43 (0.30)	69.53 (7.07)
R	97.34 (0.02)	8.47 (0.55)	9.35 (1.06)	79.19 (10.34)	95.08 (0.02)	27.15 (0.42)	6.46 (0.75)	175.61 (22.82)	89.47 (0.03)	28.74 (0.69)	5.54 (0.07)	159.26 (5.54)
S	97.36 (0.02)	10.20 (0.12)	33.00 (0.91)	336.60 (10.29)	95.08 (0.03)	25.87 (0.43)	22.70 (1.48)	587.24 (39.17)	89.48 (0.03)	27.12 (0.23)	21.67 (0.35)	587.61 (12.14)

[1] PR = Parameter configuration. [2] ACC = Accuracy. Correctly classified instances. [3] P = Power (Watts). [4] T = Execution time (seconds). [5] J = Energy consumption (Joules).

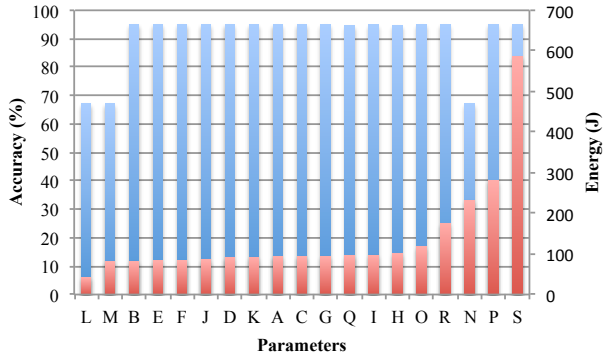

Fig. 3. Comparison between energy (red) and accuracy (blue) for different parameters configuration. Sorted from the lowest to the highest energy consumption parameter in *Scenario 2*.

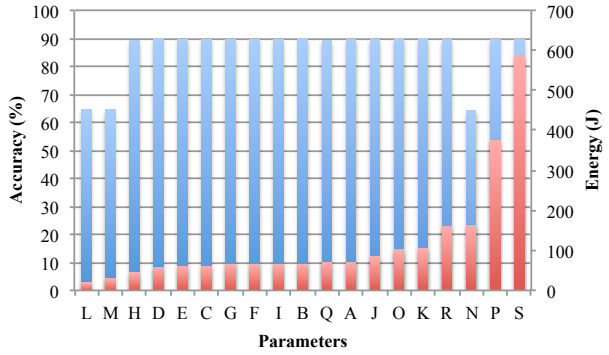

Fig. 4. Comparison between energy (red) and accuracy (blue) for different parameters configuration. Sorted from the lowest to the highest energy consumption parameter in *Scenario 3*.

C. Parameter *tie threshold*

When several attributes have a similar value of G, many instances are needed in order to make a split on the best attribute. In this situation, while there is no difference in the attribute that is chosen, there is indeed a waste of computational power. For that reason, in the VFDT implementation the authors have created the parameter τ. If the difference between the G values of two attributes is lower than τ, the algorithm splits in the current best attribute. The default value of τ is 0.05, and we have tested the following values: 0.01, 0.03, 0.05, 0.07, 0.09.

In *Scenario 1* and *Scenario 3* there is a tendency for energy to increase when τ increases. This phenomenon is shown in Figure 7, extracted from *Scenario 3*. We can observe how there is an increase of 61 J (138.73%) from the 0.01 value to the 0.09 and that accuracy does not vary. As shown in Table II, the increase in energy is due to an increase in execution time.

D. Parameter *NumericalEstimator*

This parameter was not in the original implementation of the VFDT algorithm, since their approach was designed for discrete attributes. However, they later developed a toolkit [46], VFML, that handles numeric continuous attributes. Other ways to handle numeric attributes by applying discretization techniques are Gaussian approximation [47] and Greenwald-Khanna [48] technique.

In the VFML implementation, the numeric attribute values are summarized by a set of ordered bins fixed at creation time. If, for instance, the number of bins is set to 1,000, the summary will be based on the first 1,000 examples that arrive from the stream. In the experiment, the stream has 1 million instances and the summary is based only on the first 1,000. Therefore, since the estimator is sensitive to data order, this is a possible reason why this parameter has a 30% lower accuracy that the rest. The reason why VFML $= 10$ bins is the best parameter in terms of energy consumption is the low execution time that

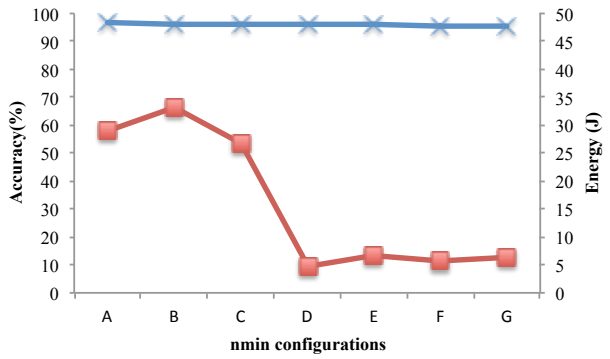

Fig. 5. Comparison between energy (red squares) and accuracy (blue stars) for the $nmin$ parameter on *Scenario 1*.

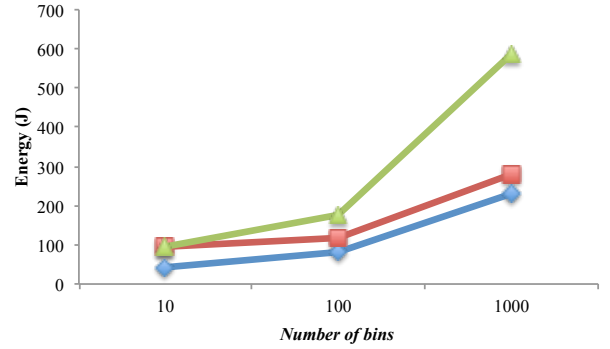

Fig. 6. Comparison on energy between the following numerical estimator parameters from *Scenario 2*: VFML (blue) = Very Fast Machine Learning [46]; Gaussian Approximation (red) [47] and Greenwald-Khanna (green) [48].

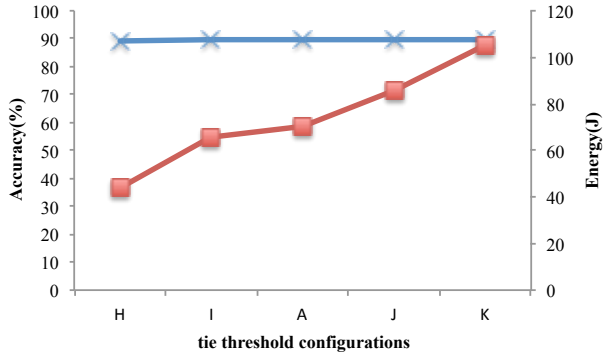

Fig. 7. Comparison between energy (red squares) and accuracy (blue stars) for the $tie\ threshold$ parameter on *Scenario 3*.

it takes to evaluate only the first 10 examples for the summary.

The Gaussian approximation is insensitive to data order and designed to be efficient in terms of computation and memory. The numerical data is summarized using a Gaussian distribution with a fixed number of split points. In Figure 6, we observe that when the number of split points is increased, so is the energy consumption. The reason is that it takes longer time to evaluate 1,000 split points than 10 split points.

The Greenwald-Khanna estimator is a quantile summary method that maintains an ordered set of tuples. It has a high accuracy in comparison to other quantile methods. The main drawback is that the method needs to maintain the set of tuples ordered when new instances arrive. As observed from the results, the worst parameter configuration is when the Greenwald-Khanna estimator is set to 1,000 tuples. The reason is that whenever new instances arrive, we need to sort 1,000 tuples every time, which results in higher execution time.

The aim was to introduce the idea that energy consumption is an important factor seldom considered in the data mining community. The figures in Section VII clearly indicate that there is a difference in energy consumption when varying the parameters of an algorithm correctly. We are able to reduce energy consumption while maintaining predictive accuracy.

American data centers produce 61 billion kWh of electricity every year [4], so even reducing a 1% of the electricity would save a large amount of energy and money.

VIII. Conclusions and Future Work

The aim was to introduce energy consumption as an important factor during data mining algorithm evaluation and analysis. While performance and computational effort are factors usually considered in data mining, energy consumption is seldom evaluated. Energy awareness leads to reducing CO_2 emissions, increasing battery life of mobile devices and reducing air pollution.

In order to measure energy consumption, we proposed a generic framework. This framework was designed by adding energy measurement to the CRISP-DM model. CRISP-DM is considered the state-of-the-art process in data mining. Thus, we believe that adding energy measurement as an extra step to CRISP-DM, is more straightforward than proposing a completely new framework.

We then designed an experiment where we varied the parameters of the VFDT algorithm and measured the energy consumption and accuracy. The results indicate that it is possible to reduce the energy consumption of an algorithm without reducing accuracy by correctly varying the parameters of the algorithm.

Future work is to investigate why certain parameter choices consume more energy than others. For this purpose, we aim to break down data stream mining algorithms into generic sub tasks to allow a more fine-grained comparison of energy consumption across various algorithms and algorithm configurations. Another potential study is to investigate how the VFDT algorithm with different configurations affect energy consumption on a real world scenario with real data.

Acknowledgment

This work is part of the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden.

References

[1] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[2] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[3] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.

[4] C. Reams, "Modelling energy efficiency for computation," Ph.D. dissertation, University of Cambridge, 2012.

[5] H. Zimmermann, "OSI reference model–the iso model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[6] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 89–102.

[7] J. Li and J. F. Martínez, "Power-performance considerations of parallel computing on chip multiprocessors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 2, no. 4, pp. 397–422, 2005.

[8] A. Hooper, "Green computing," *Communication of the ACM*, vol. 51, no. 10, pp. 11–13, 2008.

[9] S. Chu, "The energy problem and Lawrence Berkeley National Laboratory," *Talk given to the California Air Resources Board*, 2008.

[10] "Address of the president, Lord Rees Of Ludlow om kt prs, given at the anniversary meeting on 1 december 2008," *Notes and Records of the Royal Society of London*, vol. 63, no. 2, pp. pp. 183–190, 2009.

[11] M. Naghavi, H. Wang, R. Lozano, A. Davis, X. Liang, M. Zhou, S. E. V. Vollset, A. Abbasoglu Ozgoren, R. E. Norman, T. Vos *et al.*, "Global, regional, and national agesex specific all-cause and cause-specific mortality for 240 causes of death: a systematic analysis for the global burden of disease study 2013," *The Lancet*, vol. 385, no. 9963, pp. 117–171, 2015.

[12] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[13] A. Bifet and R. Kirkby, "Data stream mining a practical approach," 2009.

[14] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[15] M. J. Kearns and U. V. Vazirani, *An introduction to computational learning theory*. MIT press, 1994.

[16] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[17] J. Flinn and M. Satyanarayanan, "PowerScope: A tool for profiling the energy usage of mobile applications," in *WMCSA '99 Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. IEEE, 1999, pp. 2–10.

[18] J. G. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, 2011.

[19] C. C. Aggarwal, *Data streams: models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

[20] M. M. Gaber, "Data stream processing in sensor networks," in *Learning from Data Streams*. Springer, 2007, pp. 41–48.

[21] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering data streams," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 359–366.

[22] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD Record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[23] C. C. Aggarwal, "A framework for diagnosing changes in evolving data streams," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003, pp. 575–586.

[24] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004, pp. 180–191.

[25] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1794–1813, 2002.

[26] M. Garofalakis, J. Gehrke, and R. Rastogi, "Querying and mining data streams: you only get one look a tutorial," in *SIGMOD Conference*, 2002, p. 635.

[27] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 226–235.

[28] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[29] J. Shafer, R. Agrawal, and M. Mehta, "Sprint: A scalable parallel classifier for data mining," in *Proc. 1996 Int. Conf. Very Large Data Bases*. Citeseer, 1996, pp. 544–555.

[30] P. E. Utgoff, "Incremental induction of decision trees," *Machine learning*, vol. 4, no. 2, pp. 161–186, 1989.

[31] R. Jin and G. Agrawal, "Efficient decision tree construction on streaming data," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 571–576.

[32] Q. Ding, Q. Ding, and W. Perrizo, "Decision tree classification of spatial data streams using peano count trees," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, ser. SAC '02. New York, NY, USA: ACM, 2002, pp. 413–417.

[33] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On demand classification of data streams," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 503–508.

[34] M. Last, "Online classification of nonstationary data streams," *Intelligent Data Analysis*, vol. 6, no. 2, pp. 129–147, 2002.

[35] M. M. Gaber, S. Krishnaswamy, and A. Zaslavsky, "On-board mining of data streams in sensor networks," in *Advanced methods for knowledge discovery from complex data*. Springer, 2005, pp. 307–335.

[36] Y.-N. Law and C. Zaniolo, "An adaptive nearest neighbor classification algorithm for data streams," in *Knowledge Discovery in Databases: PKDD 2005*. Springer, 2005, pp. 108–120.

[37] F. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. Riquelme, "Discovering decision rules from numerical data streams," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 649–653.

[38] J. Gama and M. M. Gaber, *Learning from data streams*. Springer, 2007.

[39] R. Bhargava, H. Kargupta, and M. Powers, "Energy consumption in data analysis for on-board and distributed applications," in *Proceedings of the ICML*, vol. 3, 2003, p. 47.

[40] C. Shearer, "The crisp-dm model: the new blueprint for data mining," *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[41] Ó. Marbán, G. Mariscal, and J. Segovia, "A data mining & knowledge discovery process model," *Data Mining and Knowledge Discovery in Real Life Applications*, pp. 1–17, 2009.

[42] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[43] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 5, no. 6, pp. 914–925, 1993.

[44] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 377–382.

[45] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[46] G. Hulten and P. Domingos, "VFML – a toolkit for mining high-speed time-changing data streams," 2003. [Online]. Available: http://www.cs.washington.edu/dm/vfml/

[47] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[48] M. Greenwald and S. Khanna, "Space-efficient online computation of quantile summaries," in *ACM SIGMOD Record*, vol. 30, no. 2. ACM, 2001, pp. 58–66.