Choosing the Right Time to Learn Evolving Data Streams

Alessio Bernardo
DEIB - Politecnico di Milano
Milano, Italy
alessio.bernardo@polimi.it

Emanuele Della Valle
DEIB - Politecnico di Milano
Milano, Italy
emanuele.dellavalle@polimi.it

Albert Bifet
University of Waikato, New Zealand
LTCI, Télécom ParisTech, France
abifet@waikato.ac.nz

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386551

***Abstract*—Continuous data generation over time presents new challenges for Machine Learning systems, which must develop real-time models due to memory and latency limitations. Streaming Machine Learning algorithms analyze data streams one sample at a time, progressively updating their models. However, is it necessary to utilize all the data for model updates? This paper introduces the Online Ensemble SPaced Learning (OE-SPL) strategy, an ensemble meta-strategy that combines online ensemble learning and the Spaced Learning heuristic to rapidly learn underlying concepts without using all samples. We evaluated OE-SPL on synthetic and real data streams containing various concept drifts, providing statistical evidence that OE-SPL achieves comparable performance to state-of-the-art ensemble models while recovering from multiple concept drift occurrences more efficiently, using less time and RAM-Hours.**

***Index Terms*—SML, Spaced Learning, Online Ensemble Learning, Constrained Environment**

I. Introduction

The IT world is continuously in motion, as are the enormous amounts of data produced daily. By 2025, we will produce $175ZB$ of data per year, an order of magnitude bigger than the storage production capability [1]. This exponential inflation will thus require analysing the 30% of them in real-time [1].

These pose new challenges for the ML systems that have to analyze them. ML models are static and incapable of adapting over time, resulting in models no longer being relevant to the newly observed data. This phenomenon is named concept drift [2] and, nowadays, an ML model must deal with it. This would require restarting the training each time new data become available (a.k.a., *stateless retraining*). This practice quickly becomes intractable for data flows (namely data streams) that are available temporarily due to storage constraints (time and memory). Data streams call for novel *stateful systems* that adapt continuously and keep learning over time.

In recent years, *Streaming Machine Learning* (SML) [3] was introduced to address this challenge. SML maintains models online, incorporating one sample at a time and incrementally updating the model, instead of retraining it anew. To cope with the stream's unboundedness, each instance, once used, is discarded. Moreover, an SML model adapts its acquired knowledge to model new concepts.

SML real-world applications include tools for i) monitoring the mobility of people in cities [4], ii) analysing the industrial processes [5] through IoT sensors, iii) capturing the new trends coming from the social media [6], and iv) disaster control and prevention (e.g., COVID-19) [7].

However, the advent of IoT sensors and Edge Computing [8] is increasingly demanding for low-latency and low-resource consumption algorithms to be deployed in resource-constrained environments. For this reason, we tried to understand which is the right time to learn evolving data streams. Intuitively, the right time is when a concept drift detector detects a change. However, to do so, it uses all the data generated. Instead, in this paper, we tried to figure out if an SML model needs to keep learning using the massive quantity of observed data to understand the problem better. We got inspired by how we "train" our brain: we do not always use all the data around us to learn a concept. At a certain point in time, we stop learning. In neuropsychiatry, one heuristic to better learn a concept is Spaced Learning [9]. It involves revising the information over regular time intervals without continuously using all the data.

The main contribution of this paper is Online Ensemble SPaced Learning (OE-SPL), an ensemble meta-strategy based on online ensemble learning and the Spaced Learning heuristic. It is an ensemble meta-strategy that gives more importance to the instances seen in an initial phase aiming at learning the underlying concept as quickly as possible (Oversampling) and then decreases the new instances' importance (Undersampling) since the underlying concept is already learned. As shown in Fig. 1, the OE-SPL final scope is to achieve i) performance similar to other ensemble strategies (upper bounds) using less computational resources (time and RAM-Hours) and ii) better performance than its baselines (lower bounds) using a similar amount of computational resources.

In class balanced scenarios and for binary classification tasks, this paper investigates the following **research question**: *Can a tree model assume that it has learnt the underlying concept and stop training itself with new data?* In particular:

Q1 *Does the* OE-SPL *ensemble meta-strategy, combined with several SML tree models as base learners, improve the performance w.r.t. the SML tree models alone (lower bounds)? What about the time and RAM-Hours consumption?*

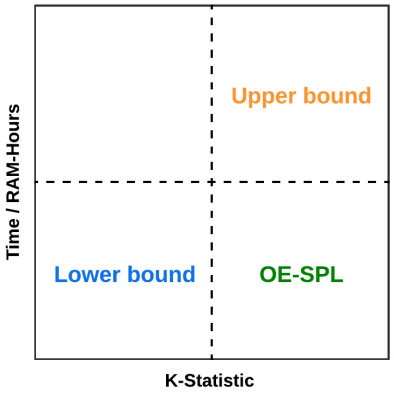


Fig. 1: The OE-SPL's aim is to achieve i) similar K-Statistic performance to its *upper bounds* using less time/RAM-Hours, and ii) better K-Statistic performance to its *lower bounds* using a similar amount of time/RAM-Hours.

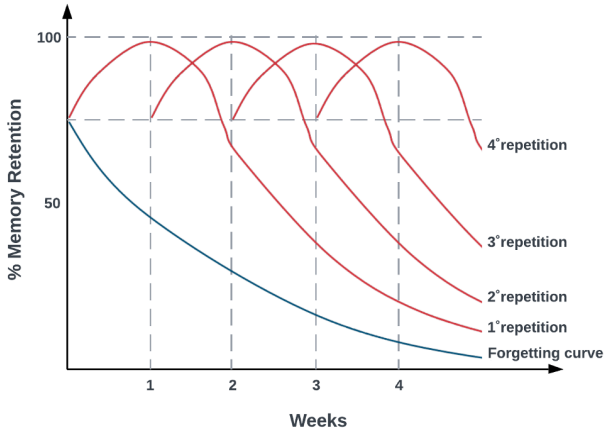


Fig. 2: Spaced Learning Heuristic.

Q2 *How does* OE-SPL*, combined with several SML tree models as base learners, perform w.r.t. state-of-the-art ensemble methods with the same SML tree models as base learners (upper bounds)? What about time and RAM-Hours consumption?*

Q3 *Does the Spaced Learning heuristic make the difference in answering Q1 and Q2? If we ablate the proposed approach by removing it (let us name this ablated version* OE-SPL$_{\text{ABL}}$*), do the Q1 and Q2 answers stay the same?*

Q4 *How does* OE-SPL *combined with several SML tree models as base learners behave with multiple concept drift occurrences w.r.t. the state of the art?*

In more detail, the main contributions of this paper are:

- OE-SPL, a combination of online ensemble learning and the Spaced Learning heuristic to learn as soon as possible the underlying concept;
- Statistical evidence that OE-SPL used with 2 SML tree models outperforms the SML tree models alone (lower bounds) in terms of K-Statistic and takes similar time and RAM-Hours (i.e., positively answering to Q1);
- Statistical evidence that OE-SPL used with 2 SML tree models performs similarly to other state-of-the-art ensemble methods (upper bounds) in terms of K-Statistic and takes less time and RAM-Hours (i.e., answering to Q2);
- Statistical evidence that the use of the Spaced Learning heuristic improves the K-statistic performance (i.e., positively answering to Q3); and
- Statistical evidence that OE-SPL, using less time and RAM-Hours, recovers from concept drift occurrences as other methods (i.e., answering to Q4).

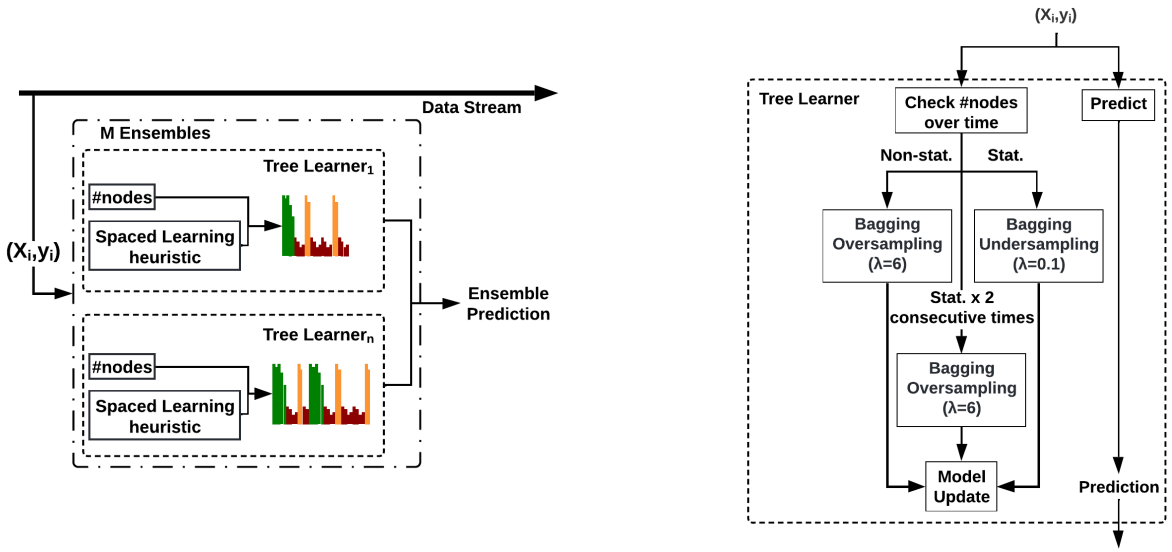
The remainder of this paper is organized as follows. Section II introduces more information about Spaced Learning and some ensemble models presented in the literature. Section III describes OE-SPL. Section IV introduces the data streams and SML models used in the experiments and presents our research hypotheses. Section V discusses the evaluation results, while Section VI discusses the conclusions and some future research.

II. Related Work

Our ability to remember information depends on the number of times we have reviewed it, the temporal distribution of the reviews, and the time elapsed since the last review. The effect of these factors has been investigated in the psychology literature [10], [11], particularly in language acquisition research [12], [13]. These studies motivated the use of flashcards, small pieces of information a learner repeatedly reviews following a schedule, whose goal is to ensure that learners spend more (less) time working on forgotten (recalled) information. This heuristic is called Spaced Learning, and recent studies [9] prove what H. Ebbinghaus[1] already affirmed during his research on human forgetting curve, as Fig. 2 shows: "with any considerable number of repetitions, a suitable distribution of them over a space of time is decidedly more advantageous than the massing of them at a single time". There are only a few Spaced Learning heuristic applications in IT. Examples are online platforms as Duolingo[2], or research to improve the learning of the concepts in real life [14]. None of them refers to the AI field, which should be inspired by how the human brain learns and replicates the most efficient heuristics.

Ensemble Learning is one of the most popular approaches for improving the accuracy of AI models. An ensemble is a composition of multiple weak learners to form one with higher predictive performance, such that a weak learner is loosely defined as a learner that performs slightly better than random guessing [15]. An ensemble example is Bagging. The non-streaming bagging [16] builds a set of M base learners training each model with a bootstrap sample of size N created by drawing random samples with replacements from the original training set. Each base learner's training set contains the original training examples K times where $P(K = k)$ follows

[1]https://en.wikipedia.org/wiki/Hermann_Ebbinghaus
[2]https://www.duolingo.com



(a) Ensemble architecture. *Green* bars represent the learning phase speed up, *red* bars the slow down, and *orange* bars the Spaced Learning heuristic

(b) Base learner architecture. *Non-stat:* non-stationary, *stat.:* stationary.

Fig. 3: Architecture of OE-SPL ensemble.

a binomial distribution. Since the stream is unbounded in the SML scenario, N is supposed to assume large values. So, the binomial distribution tends to a Poisson(1) distribution, where $Poisson(1) = exp(-1)/k!$. Using this fact, Online Bagging (OB) [17], instead of sampling with replacement, gives each example a weight according to Poisson(1).

Leveraging Bagging (LB) [18] is an Online Bagging extension. It increases the instances' weights using a larger λ to compute the value of the Poisson distribution. Using $\lambda = 1$, OB takes out 37% of the examples and repeats 26% of them, similarly to non-streaming bagging. Instead, using a value of $\lambda > 1$, LB increases the weights' diversity and modifies the classifiers' input space inside the ensemble. In LB, $\lambda = 6$ gives the best results for the artificial data streams from the most common data generators with drifts found in the literature and for some real-world benchmark data regardless of the stream nature (stationary, dynamic) or the frequency of concept drifts.

Adaptive Random Forest (ARF) [19] grows binary Hoeffding Trees [20] while i) using a random subset of features for the split at every node, and ii) simulating sampling with replacement via Online Bagging. Moreover, ARF uses ADWIN [21] as a concept drift detector per tree, such that, after a warning detection for one tree, another (background tree) starts growing in parallel and replaces the tree only if the warning escalates to a drift.

Streaming Random Patches (SRP) [22] outperforms ARF allowing globally to select a subset of features for each model once and before constructing it. Randomly resampling both features and instances, i.e. random patches.

The last ensemble learning example is Boosting. It is a sequential method that combines weak learners no longer fitted independently from each other. It fits models iteratively such that the model's training at a given step depends on the models fitted at the previous steps. The idea is to give more importance to samples that the previous models in the sequence poorly handled. Since from the state-of-the-art [17], [23] it seems that Online Bagging performs better than Online Boosting methods, in this paper, we will focus only on the former methodology. Moreover, we tested only the above Online Bagging strategies since they are the most similar ones w.r.t. OE-SPL, sharing the basic concepts. In fact, Online Bagging applies a small level of resampling, while LB, ARF, and SRP apply a high level of it. OE-SPL, varying the resampling level over time, stands exactly among them.

Another paradigm in ML is Active Learning [24]. An active learner asks queries in the form of unlabelled instances to be labelled by an oracle (e.g., a human annotator). In this way, it aims to improve the models' performance by selecting the most informative samples to label from data and using as few labelled instances as possible, thereby minimizing the cost of obtaining labelled data. Even if, for some aspects, this paradigm is similar to the OE-SPL way of working, i.e., not using all available data, it completely differs both from OE-SPL and the more general SML scenario because it selects the most informative samples in input and asks queries for labelling them. Instead, SML models assume to have in input all labelled data. This is why Active Learning is excluded from the comparison in this paper.

III. Online Ensemble SPaced Learning (OE-SPL)

This section describes the proposed meta-strategy OE-SPL. Its acronym stands for Online Ensemble Spaced Learning because the Spaced Learning heuristic is continuously applied to train an ensemble of SML tree models thanks to online

ensemble learning. The source code is available here[3]. Section III-A firstly introduces the concepts underlying OE-SPL, while Section III-B explains the algorithm.

A. Algorithmic Concepts

Fig. 3a shows the meta-strategy architecture. It keeps an ensemble of M SML tree models as base learners and uses the number of tree nodes and the Spaced Learning heuristic to speed up/slow down the tree learners' learning phases. The small bar charts in each tree learner box illustrate the weights associated with each sample over time. Higher weights correspond to oversampling, while the lower ones to undersampling. As a consequence, at the beginning and after a concept drift occurrence, the learning phase is speeded up (green bars), while in other cases, it is slowed down (red bars). Moreover, due to the Spaced Learning heuristic, in certain conditions, the learning phase is awakened in time, aiming at refreshing the model's memory, as in the human brain (orange bars).

Fig. 3b shows the base learner training phase. OE-SPL manipulates the λ parameter of the Online Bagging Poisson distribution for modifying each sample weight based on the number of tree nodes and the Spaced Learning heuristic. Since $\lambda = 1$ corresponds to sampling with probability $\frac{1}{M}$, we can simulate sampling an instance with probability $\frac{C}{M}$ by instead presenting it to the base learner $w \sim Poisson(C)$ times. C represents the sampling rate to achieve and $C \propto w$ means that the larger C is, the higher the sampling rate: a $C < 1$ value corresponds to undersampling. More specifically, this means that, on average, all the samples will have $weight = C$, and if $C < 1$, most of the samples will be discarded. Notably, this method introduces in SML a functionality similar to the learning rate in Neural Networks [25]. OE-SPL monitors the tree's growth over time, checking the no-change hypothesis in the nodes' size through a Page Hinkley statistical test [26]. If the number of nodes is statistically non-stationary, i.e., the tree is still learning the underlying concept, OE-SPL sets $\lambda = 6$, as Leveraging Bagging does, to get a high sample weight (random oversampling) and to learn as quickly as possible. Instead, if the number of nodes is statistically stationary, i.e., the tree is not growing anymore, OE-SPL sets $\lambda = 0.1$ to get a low sample weight (random undersampling) and, so, to stop learning. Moreover, when the Page Hinkley test finds that the node size is stationary for two consecutive stationarity checks, OE-SPL applies the Spaced Learning heuristic. It "awakens" the model for a fixed number of samples setting $\lambda = 6$, aiming to refresh the model's memory to better fix the already learnt concept. Notice that, in the case of a large number of samples, the used trees have a fixed maximum depth and do not continue to grow indefinitely.

B. Algorithm Explanation

Algorithm 1 presents OE-SPL pseudo-code. For each of the $m \in \{1, .., M\}$ tree learners, OE-SPL initializes i) the model e_m (Line 1), ii) the growth detector $growthDet_m$ with the Page Hinkley statistical test [26] (Line 2), iii) the $sample_m$ counter to keep track of the number of samples seen (Line 3), iv) the $patience_m$ variable with P to set every how many samples to check the tree node growth (Line 4), v) the $awakening_m$ variable with 0 to set for how many samples to apply the Spaced Learning heuristic (Line 5), vi) the $growth_m$ variable with $True$ to save the current growth detector result (Line 6), and vii) the $pastGrowth_m$ variable with $False$ to save the last growth detector result (Line 7).

For each sample in the stream and for each tree learner $m \in \{1, .., M\}$, OE-SPL increments the $sample_m$ counter

[3]https://github.com/alessiobernardo/OE-SPL

Algorithm 1 OE-SPL for M models

Input: stream *S*, ensemble size *M*, patience *P*, awakening *A*

1: Initialize ensemble e_m for all $m \in \{1, .., M\}$
2: Initialize growth detector $growthDet_m$ for all $m \in \{1, .., M\}$ with PageHinkley
3: Initialize samples seen $sample_m$ for all $m \in \{1, .., M\}$ with 0
4: Initialize $patience_m$ for all $m \in \{1, .., M\}$ with P
5: Initialize $awakening_m$ for all $m \in \{1, .., M\}$ with 0
6: Initialize current tree node growth $growth_m$ for all $m \in \{1, .., M\}$ with $True$
7: Initialize last tree node growth $pastGrowth_m$ for all $m \in \{1, .., M\}$ with $False$
8: **for each** sample in *S* **do**
9: **for** $m = 1, .., M$ **do**
10: Increment $sample_m$
11: Set $nodes$ = number of e_m tree nodes
12: Update $growthDet_m$ with $nodes$
13: **if** $sample_m$ is the $patience_m{}^{th}$ sample or multiple **then**
14: Set $pastGrowth_m = growth_m$
15: Set $growth_m$ = current node growth from $growthDet_m$
16: Reset $growthDet_m$
17: **if** $growth_m$ AND $pastGrowth_m$ are $False$ **then**
18: Set $awakening_m = A$
19: Set $patience_m = patience_m * 2$
20: **else if** $growth_m = True$ AND $pastGrowth_m = False$ **then**
21: Set $awakening_m = 0$
22: Set $patience_m = P$
23: **if** $growth_m = True$ OR $awakening_m > 0$ **then**
24: Set $\lambda = 6$
25: Decrement $awakening_m$
26: **else**
27: Set $\lambda = 0.1$
28: Set $w = Poisson(\lambda)$
29: Update e_m with the current sample with weight w
30: **if** ADWIN detects a change in the error of one of the base learners **then**
31: Reset base learner and its parameters

(Line 10), gets the number of e_m tree nodes (Line 11) for updating the growth detector $growthDet_m$ (Line 12), and, every $patience_m$ samples, checks the tree node growth (Lines 13-22). OE-SPL saves the last growth detector result into the $pastGrowth_m$ variable (Line 14) before getting the current one into the $growth_m$ variable (Line 15) and resetting the growth detector $growthDet_m$. Then, if both the $growth_m$ and $pastGrowth_m$ variables are $False$ (Line 17), i.e., $growthDet_m$ finds twice in a row that the node size is stationary, OE-SPL applies the Spaced Learning heuristic setting the $awakening_m$ variable to A (Line 18) and doubling the current $patience_m$ value (Line 19), increasing the time among the tree node growth checks. This means that the e_m model will be awakened and learn the next $awakening_m$ samples. Instead, if the current value of $growth_m$ is $True$ (Line 20), i.e., the Page Hinkley test $growthDet_m$ finds that the node size is non-stationary, OE-SPL resets the $awakening_m$ (Line 21) and $patience_m$ (Line 22) variables. Since the node size is non-stationary, the "standard" training phase will be resumed in this case.

Based on the $growth_m$ and $awakening_m$ values, OE-SPL sets the λ value to use and updates the e_m model (Lines 23-30). If the node size is non-stationary ($growth_m = True$) or the Spaced Learning is active ($awakening_m > 0$) (Line 23) then $\lambda = 6$ (Line 24), otherwise $\lambda = 0.1$ (Line 27). Then, OE-SPL uses the λ to set the weight w of the current sample and updates the e_m model (Lines 28-29). Finally, OE-SPL uses the ADWIN [21] drift detector to detect concept drifts, and so, resume the learning phase (Line 30). For each e_m model, it keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window". More precisely, an older fragment of the window is dropped if and only if there is enough evidence that its average value differs from that of the rest of the window. So, ADWIN allows each base learner to speed up, slow down, and resume learning by always using data belonging to the same concept. In fact, in the case of any concept drift, it resets the e_m model and its parameters ($growthDet_m$, $sample_m$, $patience_m$, $awakening_m$, $growth_m$, $pastGrowth_m$), resuming the learning phase (Line 31).

IV. Experimental Setting

This section introduces the hypotheses tested, the data streams used (both synthetic and real), the SML models tested, and the various experimental settings.

A. Research Hypotheses

We formulate our hypotheses as follows:

- *Hp. 1*: OE-SPL coupled with M models improves the K-Statistic performance w.r.t. $\underline{OB}$ coupled with the same M models (lower bound).
- *Hp. 2*: OE-SPL coupled with M models takes a similar time and Ram-Hours to execute w.r.t. $\underline{OB}$ coupled with the same M models (lower bound).
- *Hp. 3*: OE-SPL coupled with M models performs similar K-Statistic w.r.t. $\overline{LB}$ and $\overline{SRP}$ coupled with the same M models (upper bounds).
- *Hp. 4*: OE-SPL coupled with M models, doing less oversampling, takes less time and Ram-Hours to execute w.r.t. $\overline{LB}$ and $\overline{SRP}$ coupled with the same M models (upper bounds).
- *Hp. 5*: OE-SPL outperforms its ablated version $\text{OE-SPL}_{\text{ABL}}$ in the term of K-Statistic performance.

We tested OE-SPL with both $M = 1$ tree learner and $M = 10$ tree learners (ensemble). Depending on the value of M, the lower bound changes, too. In the case of $M = 10$, we used the $\underline{OB}$ strategy with 10 base learners as a lower bound, while, in the case of $M = 1$, $\underline{OB}$ is not necessary, and we used the single model alone ($\underline{HT}$ and $\underline{HAT}$).

B. Data Streams

This study covered 14 synthetic data streams and 5 real ones, as shown in Table I. To study the phenomena in a controllable way, we synthetically generated some streams containing 3 concept drift types [2] and 2 speeds of occurrence [2] using the SINE1 [27] and SEA [28] generators. For each drift type (2 virtual, 1 real), each generator produced 2 data streams of $100,000$ instances with a sudden (instantaneously, the first concept finishes, and the next one starts) and gradual (smooth transition from the first concept to the next one) drift speed. In the former case, the drift happened at time step $50,000$ (SEA_s, SINE1_s); in the latter, it started at the time step $45,000$ and took $10,000$ time steps to complete (SEA_g, SINE1_g). So, we have 2 generators, 3 drift types and 2 speeds for a total of 12 streams. Moreover, we generated 2 streams with a recurrent drift (SEA_r, SINE1_r) happening every $15,000$ instances, for a total of 14.

After that, we tested 5 real streams with a higher number of features [29], [30]: Forest Cover Type (Covtype) [31], Electricity (Elec) [32], Give Me Some Credit (GMSC)[4], KDDCup [31], and PAKDD [33].

C. Algorithms

We used M $\underline{HT}$ [20] and $\underline{HAT}$ [34] models having a fixed maximum depth of 32 as lower bounds and OE-SPL, $\overline{LB}$, and $\overline{SRP}$ base learners. Only in the case of ensembles ($M = 10$), we used the $\underline{OB}$ strategy having 10 HT and HAT base learners as the lower bound. In this way, the lower bound is always represented by models that do not apply resampling ($\underline{HT}$ and $\underline{HAT}$) or apply a small level of it ($\underline{OB}$ with $\lambda = 1$). The SOTA method, instead, as a OE-SPL upper bound, is always represented by $\overline{LB}$ with $\lambda = 6$ with a high level of resampling. As additional terms of comparison, we tested $\overline{SRP}$ and $\overline{ARF}$. Indeed, they induce more diversity among the base learners than OE-SPL and $\overline{LB}$ by globally ($\overline{SRP}$) or locally ($\overline{ARF}$) selecting random subsets of the features. As expected from the literature, $\overline{SRP}$ always outperformed $\overline{ARF}$, thus we only report the comparison of OE-SPL w.r.t. $\overline{SRP}$. Regarding

[4]https://www.kaggle.com/c/GiveMeSomeCredit

TABLE I: The characteristics of the data streams used in the experiments. The possible types are R for Real and S for Synthetic. The dimension column represents the number of *numeric/nominal* features. *Unknown* drift type and speed means that there are concept drifts but we do not know their type and speed.

Data Stream	# Instances	Dimension	Type	Drift Type	Drift Speed
SEA$_s$	100,000	3 / 0	S	$P(y)$	Sudden
SINE1$_s$	100,000	2 / 0	S	$P(y)$	Sudden
SEA$_g$	100,000	3 / 0	S	$P(y)$	Gradual
SINE1$_g$	100,000	2 / 0	S	$P(y)$	Gradual
SEA$_s$	100,000	3 / 0	S	$P(X\|y)$	Sudden
SINE1$_s$	100,000	2 / 0	S	$P(X\|y)$	Sudden
SEA$_g$	100,000	3 / 0	S	$P(X\|y)$	Gradual
SINE1$_g$	100,000	2 / 0	S	$P(X\|y)$	Gradual
SEA$_s$	100,000	3 / 0	S	$P(y\|X)$	Sudden
SINE1$_s$	100,000	2 / 0	S	$P(y\|X)$	Sudden
SEA$_g$	100,000	3 / 0	S	$P(y\|X)$	Gradual
SINE1$_g$	100,000	2 / 0	S	$P(y\|X)$	Gradual
SEA$_r$	100,000	3 / 0	S	Unknown	Recurrent
SINE1$_r$	100,000	2 / 0	S	Unknown	Recurrent
Covtype	581,012	10 / 44	R	Unknown	Unknown
Elec	45,312	7 / 1	R	Unknown	Unknown
GMSC	150,000	10 / 0	R	Unknown	Unknown
KDDCup	494,021	34 / 7	R	Unknown	Unknown
PAKDD	50,000	14 / 13	R	Unknown	Unknown

the ablation test, OE-SPL$_{\text{ABL}}$ removes the Spaced Learning heuristic, meaning that when the Page Hinkley test finds that the node size is stationary for two consecutive stationarity checks, it does not awaken the model. We did not test the ablation of the undersampling part, i.e., using $\lambda = 6$ for the whole time, since it would result in the LB method.

For all the tests, we used the MOA framework [3]. For the lower bounds and the base learners, we used the $\underline{HT}$ and $\underline{HAT}$ models with a maximum depth of 32. We used default hyperparameters for $\underline{OB}$, $\overline{LB}$, and $\overline{SRP}$ models, while we set the $patience$ and $awakening$ OE-SPL and OE-SPL$_{\text{ABL}}$ hyperparameters. In our experimental setting, after several tests, the best $awakening$ value is 1000. In the case of the artificial, Covtype, and PAKDD streams, the best $patience$ value is 3000. With the Elec and KDDCup streams, the best is 1500, while with the GMSC stream, the best is 1000.

D. Settings

We ran all the tests in a machine that used 2 virtual CPUs Intel Skylake P-8175 at 2.5 GHz and 8 GiB of RAM. We evaluated the predictive performance using the prequential evaluation approach [35] and performed 10 runs for each data stream. We used the K-Statistic metric [36], reporting how good a model is w.r.t. a random classifier, and we also took into account the training time and the amount of RAM-Hours. Since we know when the drift occurs with synthetic data, we calculated the metrics over all the time steps after the concept drift ended. In case of *sudden* drifts, we reset the metrics at time step $50,000$. For *gradual* drifts, we reset them at time step $45,000$ (starting drift) and $55,000$ (ending drift), while for *recurrent* drifts, we reset them every $15,000$ steps. Instead, without knowing the true concept drifts in real-world data, we calculated time-decayed metrics by setting the decay factor to 0.995, which means that the old performance was forgotten at the rate of 0.5%.

V. Evaluation Results and Discussion

This section first shows a statistical analysis of K-Statistic, time, and RAM-Hours w.r.t. both the lower and upper bounds. Then, it presents a statistical average computational performance in terms of K-Statistic, time, and RAM-Hours, and a speed recovery analysis from concept drift.

A. Statistical Analysis

Each cell of Fig. 4 and Fig. 5 shows the one-tailed *Welch's* t-test [5], an adaptation of *Student's* t-test for distribution having different variances, results for each data stream, base learner, and metric tested. Using the significance level $\alpha = 0.05$, we checked the *p-value* to determine if there were statistically significant differences among the K-statistic, time, and Ram-Hours results obtained by OE-SPL and its best performant upper ($\overline{LB}$) and lower ($\underline{HT}$, $\underline{HAT}$, or $\underline{OB}$) bounds. In particular, in the case of the K-Statistic analysis shown in Fig. 4, green cells tell that OE-SPL is statistically better than, or at least similar to, $\overline{LB}$ (OE-SPL $\succeq$ $\overline{LB}$), while light green cells tell that $\overline{LB}$ is statistically better than OE-SPL (OE-SPL $\prec$ $\overline{LB}$), but OE-SPL is statistically better than its $\underline{lower\ bound}$ (OE-SPL $\succ$ $\underline{Lower\ bound}$). Finally, red cells tell that the $\underline{lower\ bound}$ is statistically better than or similar to OE-SPL (OE-SPL $\preceq$ $\underline{Lower\ bound}$). Instead, in the case of the time and Ram-Hours analysis shown in Fig. 5, green cells tell that OE-SPL is statistically faster/less memory-intensive than, or at least similar to, its $\underline{lower\ bound}$ (OE-SPL $\succeq$ $\underline{Lower\ bound}$), while light green cells tell that the $\underline{lower\ bound}$ is statistically faster/less memory-intensive than OE-SPL (OE-SPL $\prec$ $\underline{Lower\ bound}$), but OE-SPL is statistically faster/less memory-intensive than

[5] https://en.wikipedia.org/wiki/Welch%27s_t-test

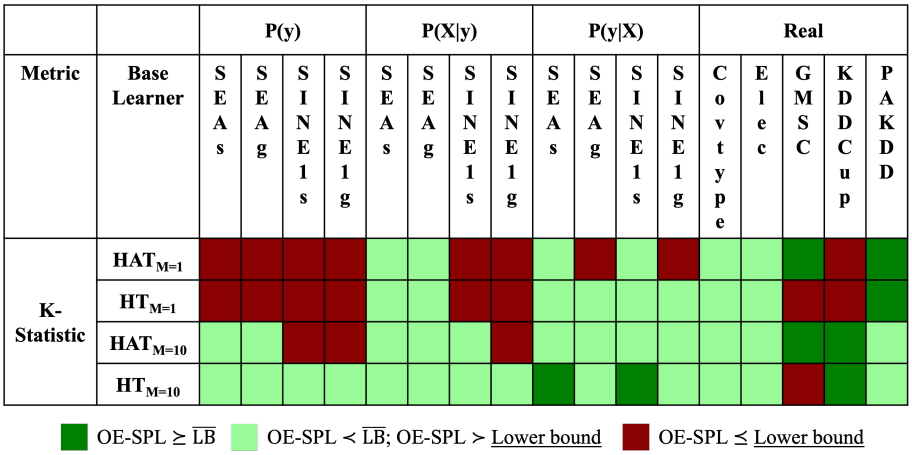


Fig. 4: One-tail Welch's t-test K-Statistic results for each data stream, base learner, and metric tested.

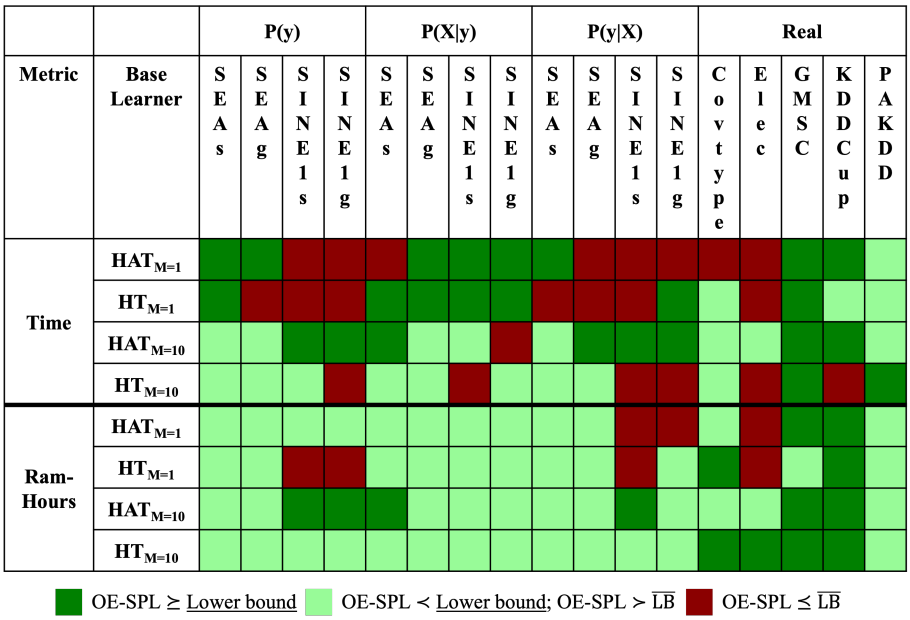


Fig. 5: One-tail Welch's t-test time and RAM-Hours results for each data stream, base learner, and metric tested.

$\overline{LB}$ (OE-SPL $\succ$ $\overline{LB}$). Finally, red cells tell that the $\overline{LB}$ is statistically faster/memory-intensive than or similar to OE-SPL (OE-SPL $\preceq$ $\overline{LB}$).

From both Fig. 4 and Fig. 5, it is easy to notice that OE-SPL, being based on an online ensemble strategy, performs better, is faster and less memory-intensive than upper and lower bounds when its base learners are ensembles. In this particular scenario (2nd and 3rd row), Fig. 4 shows that, in K-statistic results, OE-SPL is statistically better or similar than $\overline{LB}$ in 5 cases, while, in the others, its performance is between $\overline{LB}$ (upper bound) and $\underline{OB}$ (lower bound). Also notice that, in the case of real concept drift ($P(y|X)$) and real data streams, OE-SPL consistently outperforms $\underline{OB}$ and, in some cases, $\overline{LB}$, too. We can conclude that, in this one-to-one comparison, *Hp. 1* is verified in most of the cases, while *Hp. 3* is rejected.

Fig. 5 shows a similar scenario in the ensemble case. In the case of time consumption, in most cases, OE-SPL is faster than or similar to $\underline{OB}$ or, in the worst case, faster than $\overline{LB}$. The reason why OE-SPL is sometimes slower even than $\overline{LB}$ is the need to continuously check the non-stationarity of the number of tree nodes. Instead, in the case of Ram-Hours consumption, we can notice that the combination of the oversampling and undersampling strategies with the Spaced Learning heuristic makes OE-SPL less memory-intensive than the lower bound in most cases, even in the case of single base learners. In conclusion, in this one-to-one comparison, *Hp. 2* is not always verified, while *Hp. 4* is verified in most of the cases. In the next section, we will further discuss these comparisons.

B. Statistical Average Computational Performance

To summarize the results, Tables II and III present the measure values averaged over entire streams (mean perfor-

TABLE II: Avg. results and (avg. ranks) on synthetic and real streams comparing OE-SPL with the lower bounds ($\underline{HT}$, $\underline{HAT}$, $\underline{OB}$), the upper bounds ($\overline{LB}$, $\overline{SRP}$), and OE-SPL$_{\text{ABL}}$. These results are expected from the *Hps.* since OE-SPL performs i) better than the *lower bounds*, using a similar amount of computational resources, ii) similarly to $\overline{LB}$ and $\overline{SRP}$ using less computational resources, and iii) better than OE-SPL$_{\text{ABL}}$ using, of course, more computational resources.

Base learner	Algorithm	K-Statistic[1]	Time[2]	RAM-Hours[3]
HAT$_{\text{M=1}}$	OE-SPL	68.49 (3.16)	2.90 (2.89)	2.01e-07 (2.84)
	OE-SPL$_{\text{ABL}}$	67.85 (3.84)	2.13 (1.16)	1.36e-07 (1.47)
	$\underline{HAT}$	67.19 (3.74)	2.41 (2.37)	1.39e-07 (1.79)
	$\overline{LB}$	70.27 (1.26)	6.90 (3.79)	5.87e-06 (4.21)
	$\overline{SRP}$	60.12 (3.0)	7.08 (4.79)	1.01e-06 (4.68)
HT$_{\text{M=1}}$	OE-SPL	68.48 (3.11)	2.01 (3.26)	1.25e-07 (3.11)
	OE-SPL$_{\text{ABL}}$	68.23 (3.84)	1.50 (1.79)	1.12e-07 (1.89)
	$\underline{HT}$	55.10 (3.74)	0.98 (1.58)	1.41e-06 (1.42)
	$\overline{LB}$	70.79 (1.32)	4.37 (3.58)	2.34e-05 (3.95)
	$\overline{SRP}$	60.43 (3.0)	4.38 (4.79)	1.04e-06 (4.63)
HAT$_{\text{M=10}}$	OE-SPL	71.85 (2.74)	20.67 (2.58)	8.63e-06 (2.63)
	OE-SPL$_{\text{ABL}}$	71.22 (4.11)	17.04 (1.32)	6.53e-06 (1.32)
	$\underline{OB}$	69.13 (4.37)	15.17 (2.26)	1.53e-05 (2.32)
	$\overline{LB}$	72.63 (1.58)	28.62 (3.95)	5.17e-05 (4.0)
	$\overline{SRP}$	71.02 (2.21)	62.99 (4.89)	8.74e-05 (4.74)
HT$_{\text{M=10}}$	OE-SPL	71.06 (2.58)	13.16 (2.95)	4.89e-06 (2.63)
	OE-SPL$_{\text{ABL}}$	70.66 (3.63)	11.14 (2.0)	4.20e-06 (1.11)
	$\underline{OB}$	57.74 (4.68)	6.81 (1.26)	1.94e-04 (2.58)
	$\overline{LB}$	71.33 (1.89)	19.41 (3.95)	4.20e-05 (4.16)
	$\overline{SRP}$	70.33 (2.21)	38.68 (4.84)	7.99e-05 (4.53)

[1] HAT$_{\text{M=1}}$: $\overline{LB} \succ \overline{SRP} \succeq$ OE-SPL $\succ \underline{HAT} \succeq$ OE-SPL$_{\text{ABL}}$
[2] HAT$_{\text{M=1}}$: OE-SPL$_{\text{ABL}} \succ \underline{HAT} \succ$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[3] HAT$_{\text{M=1}}$: OE-SPL$_{\text{ABL}} \succeq \underline{HAT} \succ$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[1] HT$_{\text{M=1}}$: $\overline{LB} \succ \overline{SRP} \succeq$ OE-SPL $\succ \underline{HT} \succeq$ OE-SPL$_{\text{ABL}}$
[2] HT$_{\text{M=1}}$: $\underline{HT} \succeq$ OE-SPL$_{\text{ABL}} \succ$ OE-SPL $\succeq \overline{LB} \succ \overline{SRP}$
[3] HT$_{\text{M=1}}$: $\underline{HT} \succ$ OE-SPL$_{\text{ABL}} \succ$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[1] HAT$_{\text{M=10}}$: $\overline{LB} \succ \overline{SRP} \succ$ OE-SPL $\succ$ OE-SPL$_{\text{ABL}} \succeq \underline{OB}$
[2] HAT$_{\text{M=10}}$: OE-SPL$_{\text{ABL}} \succ \underline{OB} \succeq$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[3] HAT$_{\text{M=10}}$: OE-SPL$_{\text{ABL}} \succ \underline{OB} \succeq$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[1] HT$_{\text{M=10}}$: $\overline{LB} \succeq \overline{SRP} \succeq$ OE-SPL $\succ$ OE-SPL$_{\text{ABL}} \succ \underline{OB}$
[2] HT$_{\text{M=10}}$: $\underline{OB} \succ$ OE-SPL$_{\text{ABL}} \succ$ OE-SPL $\succ \overline{LB} \succ \overline{SRP}$
[3] HT$_{\text{M=10}}$: OE-SPL$_{\text{ABL}} \succ \underline{OB} \succeq$ OE-SPL $\succ \overline{LB} \succeq \overline{SRP}$

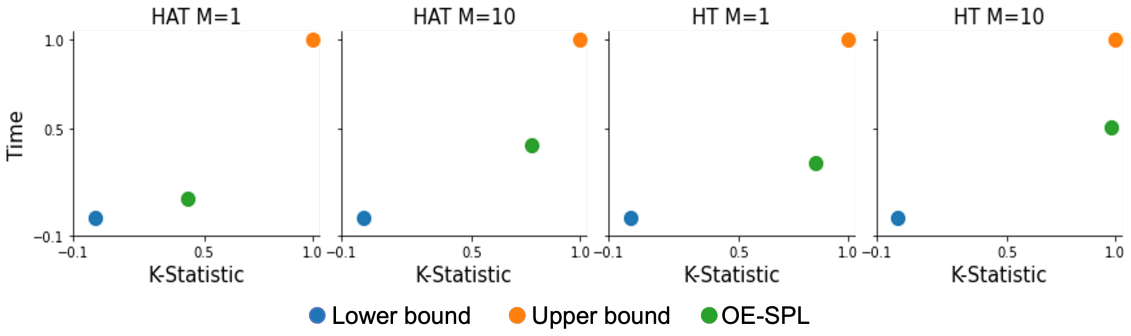
TABLE III: Avg. results and (avg. ranks) on synthetic streams with recurrent drift comparing OE-SPL with the lower bounds ($\underline{HT}$, $\underline{HAT}$, $\underline{OB}$), the upper bounds ($\overline{LB}$, $\overline{SRP}$), and OE-SPL$_{\text{ABL}}$. These results are expected since OE-SPL recovers i) faster than the *lower bounds*, ii) similarly to $\overline{LB}$ and $\overline{SRP}$, and iii) faster than OE-SPL$_{\text{ABL}}$.

Base learner	Algorithm	K-Statistic[1]
HAT$_{\text{M=1}}$	OE-SPL	94.45 (2.5)
	OE-SPL$_{\text{ABL}}$	94.2 (3.5)
	$\underline{HAT}$	92.43 (4.5)
	$\overline{LB}$	95.27 (1.5)
	$\overline{SRP}$	74.92 (3.0)
HT$_{\text{M=1}}$	OE-SPL	94.51 (2.5)
	OE-SPL$_{\text{ABL}}$	94.37 (3.5)
	$\underline{HT}$	44.28 (4.5)
	$\overline{LB}$	95.29 (1.5)
	$\overline{SRP}$	75.53 (3.0)
HAT$_{\text{M=10}}$	OE-SPL	96.76 (2.5)
	OE-SPL$_{\text{ABL}}$	96.62 (3.5)
	$\underline{OB}$	93.28 (4.5)
	$\overline{LB}$	97.33 (1.5)
	$\overline{SRP}$	88.81 (3.0)
HT$_{\text{M=10}}$	OE-SPL	96.34 (2.0)
	OE-SPL$_{\text{ABL}}$	96.18 (3.0)
	$\underline{OB}$	56.59 (4.5)
	$\overline{LB}$	92.87 (2.5)
	$\overline{SRP}$	85.01 (3.0)

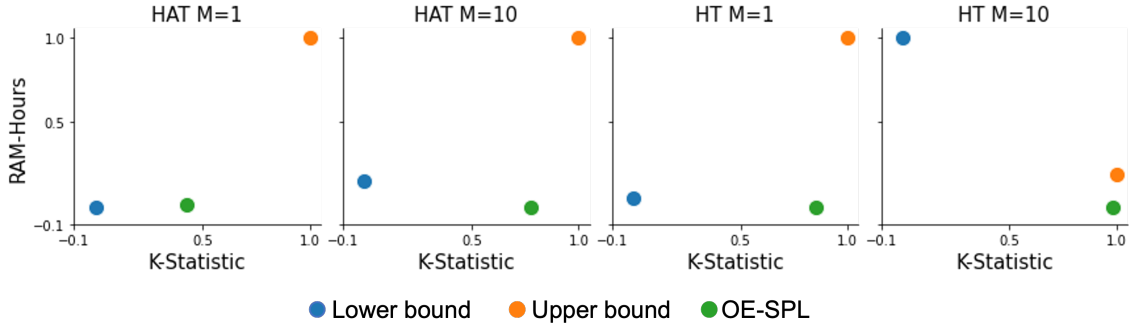
[1] HAT$_{\text{M=1}}$: $\overline{LB} \succeq$ OE-SPL $\succeq \overline{SRP} \succeq$ OE-SPL$_{\text{ABL}} \succeq \underline{HAT}$
[1] HT$_{\text{M=1}}$: $\overline{LB} \succeq$ OE-SPL $\succeq \overline{SRP} \succeq$ OE-SPL$_{\text{ABL}} \succeq \underline{HT}$
[1] HAT$_{\text{M=10}}$: $\overline{LB} \succeq$ OE-SPL $\succeq \overline{SRP} \succeq$ OE-SPL$_{\text{ABL}} \succeq \underline{OB}$
[1] HT$_{\text{M=10}}$: OE-SPL $\succeq \overline{LB} \succeq$ OE-SPL$_{\text{ABL}}$ = $\overline{SRP} \succ \underline{OB}$

mance values), which are used for carrying out a Nemenyi test [37] with significance level $\alpha = 0.05$ to compare classifier performance. In particular, $x \succ y$ indicates that algorithm x is statistically better than y, while $x \succeq y$ suggests that the former is more profitable than the latter, but without any statistical significance, and we will address this case as x is statistically similar to y.

Table II shows the average results and the ranks achieved by OE-SPL, OE-SPL$_{\text{ABL}}$, $\underline{OB}$, $\overline{LB}$, and $\overline{SRP}$ used with different base tree learners: a single HAT (HAT$_{\text{M=1}}$) and HT (HT$_{\text{M=1}}$) model, and 10 HAT (HAT$_{\text{M=10}}$) and HT (HT$_{\text{M=10}}$) models. In terms of K-Statistic, OE-SPL performs i) similar results w.r.t. $\overline{LB}$ and better results than $\overline{SRP}$, even if the last two are statistically better, and ii) better average results w.r.t. OE-SPL$_{\text{ABL}}$ and the single models. This is quite expected since $\overline{LB}$ and $\overline{SRP}$ continuously apply oversampling, while OE-SPL$_{\text{ABL}}$ and the single models do not apply it. Instead, in terms of time and RAM-Hours, OE-SPL i) statistically outperforms $\overline{LB}$ and $\overline{SRP}$, and ii) achieves similar average results w.r.t OE-SPL$_{\text{ABL}}$ and the single models even if the last are statistically better. In particular, it is quite obvious



(a) K-Statistic vs time results.



(b) K-Statistic vs RAM-Hours results.

Fig. 6: Average results achieved by OE-SPL w.r.t. its lower and upper bounds. As the upper bound we used only the best-performant one (LB).

that OE-SPL$_{\text{ABL}}$, doing less oversampling, uses less time and RAM-Hours than OE-SPL. In conclusion, this means that OE-SPL i) achieves similar average K-Statistic results using statistically less time and RAM-Hours than $\overline{LB}$ and $\overline{SRP}$, and ii) statistically outperforms OE-SPL$_{\text{ABL}}$ and the single models used alone (lower bounds) but using a similar amount of time and RAM-Hours. In conclusion, we can say that, in average, *Hp. 1*, *Hp. 2*, *Hp. 3*, *Hp. 4*, and *Hp. 5* are verified.

C. Recovery Speed Analysis

Table III shows the average K-Statistic and ranks achieved using 2 streams with recurrent drifts. For the recovery speed analysis, since drifts occur every $15,000$ samples, we took the K-Statistic values reached $1,000$ samples before every drift occurrence and averaged them. In 3 cases out of 4 (HAT$_{\text{M=1}}$, HT$_{\text{M=1}}$, HAT$_{\text{M=10}}$), OE-SPL performs i) statistically similar to $\overline{LB}$ and $\overline{SRP}$, ii) statistically better than the respective baseline (lower bound), and iii) better than OE-SPL$_{\text{ABL}}$ although with no statistical difference. Moreover, with HT$_{\text{M=10}}$, OE-SPL also outperforms $\overline{LB}$ even if with no statistical difference. In conclusion, OE-SPL recovers from concept drifts i) much faster w.r.t. OE-SPL$_{\text{ABL}}$ and the baselines (lower bounds) but using a similar amount of time and RAM-Hours (*Hp. 1*, *Hp. 2*, *Hp. 5*), and ii) similarly w.r.t. $\overline{LB}$ and $\overline{SRP}$ but using less time and RAM-Hours (*Hp. 3*, *Hp. 4*).

VI. Conclusions

We presented the OE-SPL strategy, inspired by online ensemble learning and Spaced Learning heuristic. Adapting the λ value over time, OE-SPL varies the weight of each sample resulting in an initial phase of oversampling, aiming at learning as soon as possible the underlying concept, and in a subsequent stage of undersampling, aiming at stopping learning. Between the two phases, OE-SPL applies the Spaced Learning heuristic to learn the underlying concept better.

We compared OE-SPL with $\underline{OB}$, $\overline{LB}$, $\overline{SRP}$, and its ablated version OE-SPL$_{\text{ABL}}$ used with a single HAT and HT model, and 10 HAT and HT models, measuring the K-Statistic performance, the time and RAM-Hours used, and the recovery from multiple concepts drifts.

The results statistically demonstrate that OE-SPL i) using a similar amount of time and RAM-Hours, outperforms the lower bounds ($\underline{HT}$, $\underline{HAT}$, and $\underline{OB}$), positively answering Q1, ii) using less time and RAM-Hours, performs similar to $\overline{LB}$ and better than $\overline{SRP}$ (upper bounds), answering Q2, and iii) outperforms its ablated version, showing the importance of the Spaced Learning heuristic and positively answering Q3. These findings are also summarized in Fig. 6a and Fig. 6b, where, comparing both the K-Statistic w.r.t. the time and the K-Statistic w.r.t. the RAM-Hours, OE-SPL always stands in the middle between the lower and upper bounds. Moreover, OE-SPL can recover from multiple concepts drifts faster

than OE-SPL$_{\text{ABL}}$, the $\underline{lower\ bounds}$ and $\overline{SRP}$, and similarly to $\overline{LB}$, using a limited amount of time and RAM-Hours, answering Q4.

In future works, we intend to extend OE-SPL to improve its weaknesses. Firstly, we aim to investigate further the theoretical and statistical properties of Spaced Learning and propose an automatic way to control when to stop and resume learning that does not depend on hyperparameters (patience and awakening). Another fundamental point is to extend OE-SPL to be used with any base learner. This means proposing a model-agnostic way to understand if the model is still growing and learning or not. Moreover, we would like to add a feature selection strategy, as in $\overline{SRP}$, to improve OE-SPL performance. Finally, in the long term, applying the Spaced Learning heuristic to more SML models could be interesting.

REFERENCES

[1] Seagate, "Dataage 2025 - the digitization of the world," https://www.seagate.com/our-story/data-age-2025/, 2020.

[2] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[3] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018.

[4] J. Moreira, "Travel time prediction for the planning of mass transit companies: a machine learning approach," Ph.D. dissertation, ., 12 2008.

[5] K. Kammerer, B. Hoppenstedt, R. Pryss, S. Stökler, J. Allgaier, and M. Reichert, "Anomaly detections for manufacturing systems based on sensor data - insights into two challenging real-world production settings," *Sensors*, vol. 19, no. 24, p. 5370, 2019.

[6] S. Ma, X. Li, Y. Ding, and M. E. Orlowska, "A recommender system with interest-drifting," in *WISE*, ser. LNCS, vol. 4831. Springer, 2007, pp. 633–642.

[7] T. Alberti and D. Faranda, "On the uncertainty of real-time predictions of epidemic growths: A COVID-19 case study for china and italy," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 90, p. 105372, 2020.

[8] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, 2019.

[9] P. Smolen, Y. Zhang, and J. H. Byrne, "The right time to learn: mechanisms and optimization of spaced learning," *Nature Reviews Neuroscience*, vol. 17, no. 2, pp. 77–88, 2016.

[10] A. W. Melton, "The situation with respect to the spacing of repetitions and memory," *Journal of Verbal Learning and Verbal Behavior*, vol. 9, no. 5, pp. 596–606, 1970.

[11] F. N. Dempster, "Spacing effects and their implications for theory and practice," *Educational Psychology Review*, vol. 1, no. 4, pp. 309–330, 1989.

[12] R. C. Atkinson, "Optimizing the learning of a second-language vocabulary." *Journal of experimental psychology*, vol. 96, no. 1, p. 124, 1972.

[13] K. C. Bloom and T. J. Shuell, "Effects of massed and distributed practice on the learning and retention of second-language vocabulary," *The Journal of Educational Research*, vol. 74, no. 4, pp. 245–248, 1981.

[14] A. Zaidi, A. Caines, R. Moore, P. Buttery, and A. Rice, "Adaptive forgetting curves for spaced repetition language learning," in *International Conference on Artificial Intelligence in Education*. Springer, 2020, pp. 358–363.

[15] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[16] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[17] N. C. Oza and S. Russell, "Online bagging and boosting," in *AISTATS*. Society for Artificial Intelligence and Statistics, 2001.

[18] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *ECML/PKDD (1)*, ser. LNCS, vol. 6321. Springer, 2010, pp. 135–150.

[19] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[20] G. Hulten, L. Spencer, and P. M. Domingos, "Mining time-changing data streams," in *KDD*. ACM, 2001, pp. 97–106.

[21] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[22] H. M. Gomes, J. Read, and A. Bifet, "Streaming random patches for evolving data stream classification," in *ICDM*. IEEE, 2019, pp. 240–249.

[23] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *KDD*. ACM, 2009, pp. 139–148.

[24] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of page-hinckley, an approach for fault detection in an agro-alimentary production system," in *2004 5th Asian control conference (IEEE Cat. No. 04EX904)*, vol. 2. IEEE, 2004, pp. 815–818.

[27] J. Gama, P. Medas, G. Castillo, and P. P. Rodrigues, "Learning with drift detection," in *SBIA*, ser. LNCS, vol. 3171. Springer, 2004, pp. 286–295.

[28] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *KDD*. ACM, 2001, pp. 377–382.

[29] K. Bache and M. Lichman, "UCI machine learning repository," 2013.

[30] V. M. A. de Souza, D. M. dos Reis, A. G. Maletzke, and G. E. A. P. A. Batista, "Challenges in benchmarking stream learning algorithms with real-world data," *Data Min. Knowl. Discov.*, vol. 34, no. 6, pp. 1805–1858, 2020.

[31] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[32] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[33] C. Linhart, G. Harari, S. Abramovich, and A. Buchris, "PAKDD data mining competition 2009: New ways of using known methods," in *PAKDD Workshops*, ser. Lecture Notes in Computer Science, vol. 5669. Springer, 2009, pp. 99–105.

[34] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *IDA*, ser. LNCS, vol. 5772. Springer, 2009, pp. 249–260.

[35] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, 2013.

[36] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the 21th ACM SIGKDD*, 2015, pp. 59–68.

[37] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.