Pattern Recognition 138 (2023) 109359

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Meta-learning for dynamic tuning of active learning on stream classification

Vinicius Eiji Martins[a], Alberto Cano[b], Sylvio Barbon Junior[c,*]

[a] Dept. of Computer Science, State University of Londrina, Londrina, Paraná, Brazil
[b] Dept. of Computer Science, Virginia Commonwealth University, Richmond, Virginia, USA
[c] Dept. of Engineering and Architecture, University of Trieste, Trieste, Italy

ARTICLE INFO

Article history:
Received 21 August 2021
Revised 28 October 2022
Accepted 21 January 2023

Keywords:
Meta-learning
Active learning
Data stream
Concept drift

ABSTRACT

Supervised data stream learning depends on the incoming sample's true label to update a classifier's model. In real life, obtaining the ground truth for each instance is a challenging process; it is highly costly and time consuming. Active Learning has already bridged this gap by finding a reduced set of instances to support the creation of a reliable stream classifier. However, identifying a reduced number of informative instances to support a suitable classifier update and drift adaptation is very tricky. To better adapt to concept drifts using a reduced number of samples, we propose an online tuning of the Uncertainty Sampling threshold using a meta-learning approach. Our approach exploits statistical meta-features from adaptive windows to meta-recommend a suitable threshold to address the trade-off between the number of labelling queries and high accuracy. Experiments exposed that the proposed approach provides the best trade-off between accuracy and query reduction by dynamic tuning the uncertainty threshold using lightweight meta-features.

1. Introduction

Present-day supervised machine learning tasks rely on extracting hidden patterns from vast amounts of data. A significant part of them is fashioned as a data stream of continuous data arriving at a very high speed. This demands algorithms that are suitable to deal with fast processing time and limited memory space efficiently [1–3].

Unlike traditional tasks accomplished by static and batch methods on previously collected data, many challenges and particular constraints arise when dealing with the data-stream paradigm. Among the challenges, updating the model could be very tricky, since new data can arrive at any time, often changing their behaviour and leading to a concept drift issue [4,5], which invalidates models built in the past.

Active Learning is an area of machine learning grounded on the idea that if the learning algorithm can choose which data samples are more useful for its learning process, it can be trained better, faster, and with fewer data. There are many Active Learning methods for vastly different scenarios, but Stream-based and Pool-based selective sampling have achieved high-quality results in stream mining [6].

Stream-based and Pool-based methods are a subset of the Selective Sampling strategy grounded on a heuristic to query whether a new data sample is worth taking part in the training of the model. The most commonly used query strategy is Uncertainty Sampling [7], which uses an uncertainty measure. This measure is most commonly the Entropy (H) of the data sample based on the class probabilities obtained from the classifier. If $H(S_i) > Z$, where Z is a threshold value and S_i is a mew sample used in training. However, choosing correctly Z poses an additional challenge because its value is strictly related to each particular piece of a data stream. Furthermore, changes in the stream patterns (concept drift) demand updating Z. Thus, using a static Z is not feasible for real-life data streams.

An alternative approach is first to learn an optimal Z default setting and then redefine Z after a concept drift instead of leaving it at that default value. However, this approach is very demanding, as it relies on the use of an accurate concept drift detector and tuning Z. Detectors are methods to estimate the positions of concept drifts in data streams. This allows for a base learner update to be triggered after changes in the data distribution and thus improve the classification task [8]. Therefore, we have the possible

* Corresponding author.
E-mail address: sylvio.barbonjunior@units.it (S. Barbon Junior).

https://doi.org/10.1016/j.patcog.2023.109359

bias caused by the detector and a challenging tuning task regarding Z due to stream constraints [9].

We believe that using the performance of related previous tasks can help make a suitable recommendation of Z. Thus, we propose using Meta-learning (MtL) to adjust the Z value of Uncertainty Sampling based on statistical meta-features extracted from an adaptive windowing procedure handling the trade-off between the number of labelling queries and high accuracy. We explored the Very Fast Decision Tree as a classification algorithm for different streams and concept drift conditions. The main contributions of this paper are as follows:

- A methodology to improve stream classification performance under a reduced number of class labels, i.e., a real-life scenario;
- The usage of Meta-learning for Active Learning tuning as a regression problem;
- A discussion around the importance of different meta-feature categories in the context of entropy-based hyperparameter selection;
- A discussion on how Stream-based classifiers are impacted by concept drifts over different scenarios and how the classification performance may be improved when some data samples are ignored during training.

The manuscript is organised as follows. Section 2 presents an overview of related work on active learning and meta-learning for data streams. Section 3 introduces the proposed approach. Section 4 presents the experimental evaluation and analysis. Finally, Section 5 makes the concluding remarks.

2. Background

We can define a data stream as a potentially unbounded sequence of instances $< S_1, S_2, \ldots, S_n, \ldots >$, where S_i is a new sample. Addressing as a supervised machine learning task, we can define each sample S_i $p_i(x^i, \ldots, x^d, y) = P_i(X, y)$, where $P_i(X, y)$ is a joint distribution of i-th sample, described by a d-dimensional feature vector related to a class y. Each sample in the data stream is independent and randomly grounded on a stationary probability distribution $D_i(X, y)$. Since X is continuously arriving, one important solution is to get useful information on the fly without storing, incrementally incorporating, and discarding samples. Regarding y, its value is required to access the class labels to train the model following a supervised learning process. However, this requirement does not match real-life scenarios since obtaining a label (ground truth) for each sample is a high-cost process due to the time and availability of the labelling. Formally we can define concept drift as $P_i(X, y) \neq P_{t+\Delta}(X, y)$ at time t.

2.1. Active learning for data stream mining

Active Learning has a multitude of different techniques, but in the context of streaming, it is generally prioritised the use of simple techniques to not add significant overhead. Among these, we have Selective Sampling using Uncertainty Sampling for sample querying selection. Selective Sampling is comprised of two different approaches: *pool-based sampling* and *stream-based sampling*. The difference between them is that *pool-based sampling* collects pools of samples, ranks these samples, and queries only the n first samples, while *stream-based sampling* decides if a sample is worth labelling when it arrives [6]. Since data is both unbounded and fast arriving in a stream setting, stream-based sampling approaches are usually preferred. Among the query strategies used in selective sampling techniques, Uncertainty Sampling tends to be commonly used thanks to its simplicity and speed while also providing better results when compared to Random Sampling in general[6].

Although uncertainty sampling has some less commonly used uncertainty measures such as *least confidence* and *margin sampling*, the most general and commonly used is *entropy* [6], which is the one we refer to when mentioning Uncertainty Sampling in this paper. Eq. 1 defines this measure, where x is the sample, y_i corresponds to a label possibility and $P(y_i)$ is the probability that y_i is the correct class according to the classifier. The logarithmic base is usually set to 2 to represent "bits", but it can be changed, provided that it is kept consistent and the uncertainty sampling variables are scaled properly. Entropy determines the amount of confusion or uncertainty present in a system, which in this case corresponds to the confidence that the classifier knows the correct label. If the entropy is high, the classifier is uncertain since the probabilities for each class are similar.

$$H(x) = -\sum P(y_i)\log P(y_i) \tag{1}$$

One of the biggest problems with the use of Active Learning in streaming environments is dealing with concept drift [10] and unbalanced classes [11]. Since the structure of the data may change, the Active Learning configuration may become outdated and may even negatively impact the learning process.

A few solutions to this are found in the literature, such as training a new classifier in the background when the old classifier begins to lose accuracy and finally replacing it with a new one when drift is detected by a drift detection algorithm, which is also responsible for controlling the labelling rate of instances to accommodate new concepts [12]. This allows the model to avoid losing accuracy because the model is trying to adapt to the changes without altering anything else.

Shan et al. [13] proposed a framework that allows for the adaptation of concept drift during stream classification. It uses an ensemble composed of a stable classifier that always learns the latest data and an array of dynamic classifiers that learn on late sliding windows. With the use of Active Learning, using both Uncertainty Sampling and Random Sampling, the framework is capable of maintaining high accuracy levels and gradually stabilising in the advent of concept drift. Liu et al. [14] proposed a framework that relies on two metrics to decide whether an instance should be labelled. The first, called local density, is a metric based on Ebbinghaus law of human memory cognition [15] to create a sliding window based on the forgetting curve from which the local density is calculated. The second one is the classical uncertainty sampling grounded on the representativeness of the instance and the confidence that the classifier can make the right decision. This method achieves good stability and performance even in the presence of gradual concept drift while also keeping only one classifier unlike the previously seen methods, but it also suffers from a large number of hyperparameters to tune, and under abrupt concept drift scenarios its superiority to other methods is not obvious. In the context of sampling, Bouguelia et al. [16] proposed a method that provided an adaptive threshold value that is adjusted according to the error rate of the model. Although this method was created with the proposed "sufficient weight" sampling method, it is also usable in regular Uncertainty Sampling contexts. The disadvantage of this method is that it does not consider concept drifts. Castellani et al. [17] also propose a method to adapt to concept drifts by using a drift detector to estimate when a change occurs. Once a drift is detected, the querying budget is temporarily increased, after a while it is then set at a low value to comply with the global budget target. Once some time has passed, the budget value is then returned to its regular value. This method was only tested under the occurrence of a single abrupt drift, as such this method is not proven to perform well under complex scenarios.

It can be seen that many propositions that tackle Active Learning in the context of Stream Mining either aim for performance and general adaptability or concept drift handling at the cost of

complexity and resource usage. We aim to propose an approach that allows for both good general and concept drift adaptability while also keeping the task relatively simple with a few new tuning parameters introduced.

2.2. Meta-learning for data stream mining

Meta-learning has some recent applications to data stream mining in literature [18]. One of the preliminary proposals was made by Rossi et al. [19], they proposed a framework for algorithm selection on stream regression tasks using meta-learning. The proposed approach is based on extracting meta-features from the data stream bounded by sliding windows, creating meta-instances that fed the meta-model. The meta-model chooses one or more algorithms to be trained on the current sliding window of data. Different from traditional meta-learning frameworks, the models are always retrained instead of keeping static until recalled by the meta-learner; it also allows for ensembles to be formed when the meta-learner chooses more than one algorithm to build models over the current data.

A similar approach was proposed by van Rijn et al. [20] where a heterogeneous ensemble (different types of classifiers) was used to classify a data stream by choosing the main classifier with a weight of 1 and the others with a weight of 0. The main classifier is selected through the use of a meta-learning model combined with the proposed Online Performance Estimation. This showed a significant improvement over the regular use of meta-learning techniques for data stream classifier selection, although it is still generally worse than the proposed *BLAST* method. Anderson et al. [21] proposed a framework to deal with concept drifts during stream classification. The authors' proposal is based on maintaining a repository of classifiers alongside the current classifier being used; when drift is detected, a meta-learner will then choose a classifier from this repository that it judges as the fittest for the new structure of the data. The framework keeps training the current and maintained classifiers using new instances from right before the drift (when the detector signals a warning state); once a new drift is detected, one of the classifiers that were being used is stored in the repository, and the low accurate one is discarded before the meta-learner selects a new classifier.

It is worth mentioning the concern related to drift adaptation by all meta-learning over data stream approaches presented. The concept drift phenomenon requires attention, which demands a considerable change in the trained model and even an algorithm shift after a drift. In other words, a drift adaptation is beyond model updating, requiring hyperparameter tuning or the use of other algorithm biases. This observation supports our assumption that by using meta-learning to dynamically tune Active Learning algorithms, we can provide boosted results, even under different drift scenarios.

2.3. Meta-learning for active learning

To the best of our knowledge, the use of meta-learning techniques paired with Active Learning is not very well explored in the literature. Ravi et al. [22] propose the use of meta-learning to select samples to be labelled instead of using a query strategy. Through the computation of quality and diversity over the samples from a pool of unlabelled data, a meta-model is capable of selecting promising samples to train a classifier. It is worth noting though that this framework is to be used exclusively under batch scenarios. Yu et al. [23] propose a framework that aims to detect concept drifts and classify them into their different categories. Another contribution is related to the capacity of reducing the cold start effects often present in traditional drift detection methods. This framework can be split into two main phases: a training phase where the meta-model is trained and the detection phase, where the meta-model is deployed. During the training phase, meta-features from examples of various types of concept drift are extracted and used to train the meta-model. At the detection phase, the meta-model is set to detect concept drift while also adapting itself dynamically by training on relevant examples selected by Stream Based Active Learning.

While the previous work does indeed combine both Active Learning and Meta-learning, it tackles the problem of drift detection and not adaptation. With these works, we can see that meta-learning and active learning were not concepts fully explored under stream mining scenarios together.

3. Proposed approach

Meta-learning, or learning to learn, regards learning a model from a previous similar task and predicting unseen related tasks. In particular, after having properly chosen hyperparameters for active learning tasks on stream data it is possible to induce a model to recommend suitable values. Meta-learning for hyperparameter tuning has been successfully applied in different problem domains such as image processing [24], supervised machine learning [25] and coupled with algorithm selection [26]. Current literature does not present a solution for Active Learning based on Streams, exploring meta-characteristics related to the behaviour of streams or sampling uncertainties. Through the use of Stream-based Active Learning alongside Uncertainty Sampling for label querying, our stream classifier can be trained with a lower labelling cost. However, this introduces a hyperparameter that requires tuning for each task: the Z value or *uncertainty threshold*, which determines the amount of uncertainty or entropy necessary for the Active Learning model to deem a sample worthy of labelling.

Our approach aims to dynamically tune Z by adding a meta-model on top of the Active Learning model. This meta-model, grounded in Meta-learning theory, is responsible for selecting an appropriate Z for each stream chunk. Since a stream may change its behaviour through concept drift, the meta-model decides on a new Z routinely, through a set trigger, e.g., a change detector [1,27]. We propose to employ a trigger based on a change detector that detects possible changing behaviour, indicating a concept drift. It is worth mentioning that several algorithms such as ADWIN [27], Page-Hinkley [28] and the Early Drift Detection Method (EDDM)[29] can play this role in the proposed framework.

When a new possible drift point is detected, the data samples bounded by the change detector are used to extract meta-features, which feed our meta-model towards outputting a new Z for the Uncertainty Sampling as Fig. 1 shows. It is worth mentioning that different change detectors could perform as the trigger.

The workflow implementing our meta-learning approach is composed of these five steps:

1. *Meta-Feature Extraction* is a step devoted to describing the characteristics of the event stream based on the lightweight temporal time series features [30].
2. *Meta-Target Definition* identifies the best Z value that provides a suitable trade-off between accuracy a low label querying (Algorithm 1). Since this is a data-driven step, it is necessary to discover suitable Z values able to cover the search space of possibilities under the cited trade-off assumptions.
3. *Meta-Database*, through this step, the meta-features and meta-targets are combined, forming meta-instances used to train the meta-model as described in Algorithm 2.
4. *Meta-Learner* is a step grounded in machine learning to inducing the meta-model using the obtained meta-instances. *Meta-Model*, is the final model able to output the recommendation of a proper Z.

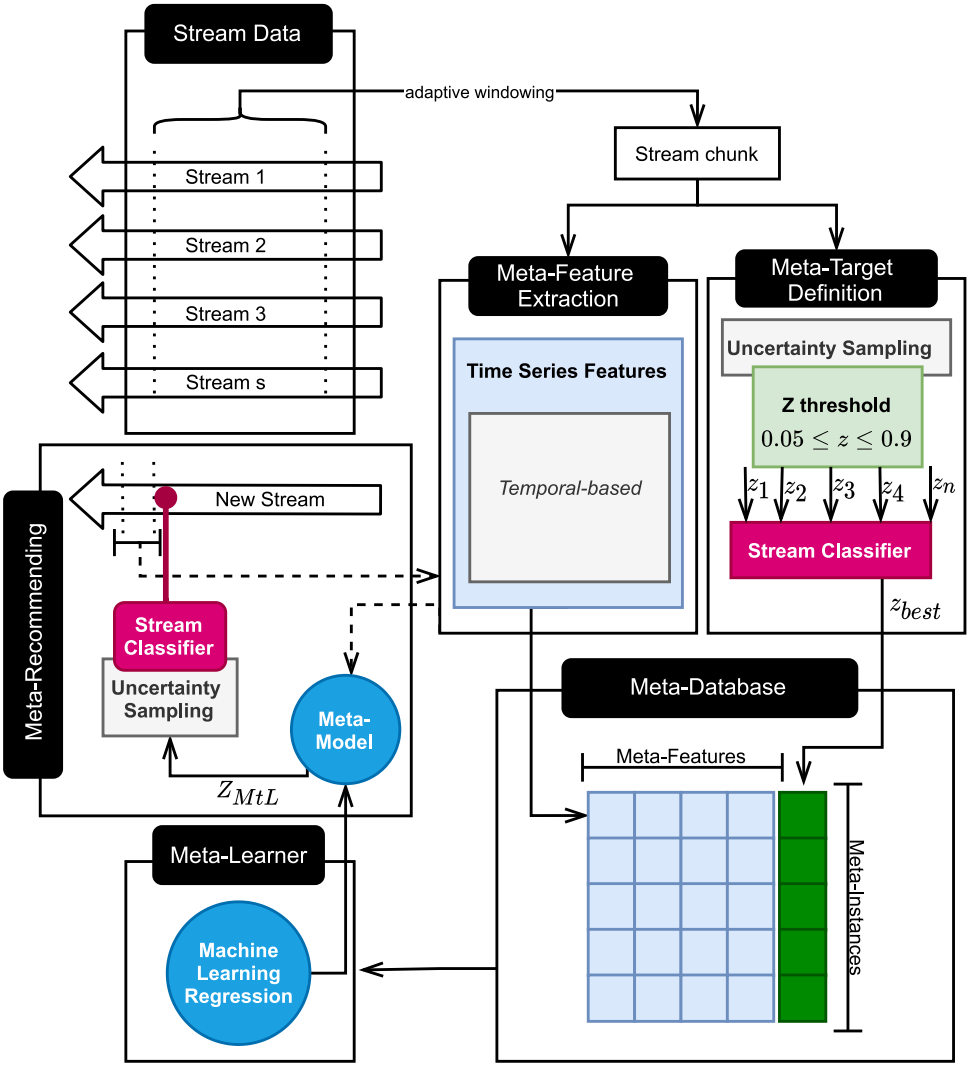

Fig. 1. Proposed approach overview.

Algorithm 1: Best Z Selection.

Input: R: Classification result of Z candidate containing accuracy and query quantity, s: Selection margin for top values
Symbols: $ACC(x)$: Accuracy obtained from the Z value x, $QRY(x)$: Query rate obtained from the Z value x
Output: R_{top}: The Z value with the lowest query rate within the top value selection interval

$R_{acc} \leftarrow Z$ with highest accuracy in R;
$T \leftarrow \{R_i \in R \mid ACC(R_{acc}) - s \leq ACC(R_i) \leq ACC(R_{acc})\}$;
$R_{top} \leftarrow \text{argmin}_T(QRY(T))$;
return R_{top};

5. *Meta-Recommending*, when a change is detected, its samples (stream chunk) have their features extracted towards predicting a new Z by meta-model. The meta-model outcome is inputted into the Uncertainty Sampling to work labelling instances to the stream classifier, as possible to follow in Algorithm 3.

4. Experimental study

The experimental study was designed to answer the following research questions (RQ): **RQ1:** Is our Uncertainty Sampling tuning method (Z_MtL) competitive with standard fixed values? **RQ2:** Does Z_MtL allow Uncertainty Sampling to support high classification accuracy when faced with concept drifts? **RQ3:** Does Z_MtL adapt well to concept drifts by limiting queries when necessary? **RQ4:** How well does our method compare to the regular supervised learning method (with all labels)?

To answer our research questions and prove the contributions provided by our approach, we conduct experiments on different scenarios: synthetic and real-life streams affected or not by concept drifts.

We explored different drift detectors (ADWIN, EDDM and Page-Hinkley) and two classifiers in our experiments. We select the Naive Bayes (NB) and Very Fast Decision Tree (VFDT) [1] with default hyperparameters as our stream classifiers, using the *scikit-multiflow* [31] implementation. There exist different classifiers, such as SVFDT [2] and ensembles. However, the selection was made considering simple and well-known algorithms to provide straightforward insights and possible limitations of our proposal.

Our experimental setup can be split into five main areas: i) Meta-Learning for Uncertainty Sampling threshold tuning, ii)

Algorithm 2: Meta-Database Construction.

Input: G: List of Data Stream Generator possibilities, α: drift detector choice, β: drift classifier choice,p_β: the amount of samples used to pre-train the drift classifier,γ: window classifier choice, ζ: Z value choices,p_γ: the amount of samples used to pre-train the window classifier,t: target number of meta-samples in the Meta-database,q: amount of samples in each stream created
Symbols: *ACT*: Stream-based Selective Sampling Active Learning
Output: M: Meta-database

$M \leftarrow$ empty dataset;
while $|M| < t$ **do**
  $g \leftarrow$ new generator from G;
  $X \leftarrow$ new data stream from g;
  $\alpha_x \leftarrow$ new instance of α;
  $\beta_x \leftarrow$ new instance of β;
  Pre-train β_x on first p_β samples of X;
  for $X_i \in \{X_{p\beta}+1, \ldots, X_q\}$ **do**
    Update α_x with error from prediction $\beta_x(X_i)$;
    if α_x *change detected* **then**
      for $\zeta_j \in \{\zeta_1, \ldots, \zeta_n\}$ **do** `// Z choosing mechanism`
        $W \leftarrow \{X_{last\_drift}, \ldots, X_i\}$;
        $\gamma_x \leftarrow$ new instance of γ;
        Pre-train γ_x on first p_γ samples of W;
        $R_j \leftarrow$ Results from $ACT^{\gamma_x}_{Z=\zeta_j}$ on $\{W_{p\gamma}+1, \ldots, W_q\}$;
      end
      $Z_{best} \leftarrow$ best Z from Algorithm 1 with input R;
      $M_k \leftarrow$ { features generated from W, Z_{best}};
    end
  end
end
return M;

Algorithm 3: Meta-Recommending.

Input: S: data stream, α_x: drift detector, β_x: classifier,θ_x: meta-learner model, p_β: the amount of samples used to pre-train the classifier
Symbols: *ACT*: Stream-based Selective Sampling Active Learning

$Z \leftarrow 0.5$;
Pre-train β_x on first p_β samples of S;
for $S_i \in \{S_{p_\beta+1}, \ldots, S_n\}$ **do**
  $E_i \leftarrow$ run ACT^β_Z on S_i and return error from prediction;
  Update α_x with E_i;
  if α *change detected* **then** `// Z requires change`
    $F \leftarrow$ features generated from $\{S_{last\_drift}, \ldots, S_i\}$;
    $Z \leftarrow$ prediction from θ_x with input F;
  end
end

Uncertainty Sampling stream benchmarks, iii) Stream classification with dynamic thresholding, iv) Querying rate using dynamic threshold tuning, and v) Uncertainty Sampling for drift adaptation.

4.1. Meta-learning for uncertainty sampling threshold tuning

The meta-learning approach proposed in this paper is grounded on gathering knowledge from previous suitable Z values from diverse streams. We took advantage of several stream benchmarks and their variations to create a robust and rich source of stream data. We created a meta-database exploring a large collection of stream datasets generated on-the-fly by 7 benchmark generators (HyperplaneGenerator, LEDGeneratorDrift, MIXEDGenerator, RandomRBFGeneratorDrift, RandomTreeGenerator, SineGenerator, and STAGGERGenerator). HyperplaneGenerator, is creates datasets regarding a classification problem around a rotating hyperplane, introducing drifts by modifying the rotation probability of each example. LEDGeneratorDrift constructs datasets around the problem of predicting the number displayed by a seven-segment LED display, introducing drifts randomly. MIXEDGenerator creates datasets characterised by the occurrence of abrupt concept drifts. Drift is introduced by reversing the classification. RandomRBFGeneratorDrift produces a dataset based on a radial basis function, adding drifts by moving the centroids at a certain speed as samples are generated. RandomTreeGenerator creates data samples with randomised features that are then inputted into a random tree that is built in the background; the label of each instance is defined by this tree. SineGenerator generates datasets containing abrupt concept drift. Drift is introduced by reversing the classification of points. Based on the artificial SINE1 dataset. Finally, the STAGGERGenerator creates specific abrupt concept drifts.

Each generator creates streams with unique behaviours that can be adjusted through their hyperparameters. To ensure that we have the maximum amount of variation in our datasets, each stream was created by randomly selecting a generator and randomly setting the values of their hyper-parameters.

Each stream has approximately 300,000 samples, randomly affected by different drifts (*Gradual, Recurring* and *Abrupt*) covering binary and multi-class classification problems.

To segment each stream into chunks from which features may be extracted, we leveraged the use of a drift detector. Our choice of a drift detector for the generation of our meta-database was ADWIN [27] with a δ value of $10e^{-4}$ due to the detection performance observed in the preliminary tests. With these configurations, we extracted 2026 windows in total, which translates to 2026 meta-instances in the meta-database. The same configuration of ADWIN is used alongside other detectors in the further phases of our experiments.

Since the meta-data from the drift detector is not persisted in the meta-database, we reasoned that it would be unnecessary to create a new meta-database for each drift detector and classifier. This would also allow us to evaluate the ability of our trained meta-model to generalise under different classifiers and drift detectors.

Since our drift detectors are all univariate, the value fed to the algorithm was the prediction result from a classifier (in the case of our meta-database, a VFDT) making predictions on the top of the generated stream and outputting 0 if it predicted wrongly or 1 if predicted correctly. The detector then detects changes when the classifier changes its behaviour by suddenly making too many mistakes or too many hits. This same principle is applied in the *Meta-Recommending* phase, where the prediction result of the stream active learning model is fed to the detector.

With the drift points detected by our drift detector, we can segment the entire stream into windows delimited by the aforementioned points. Each window represents the behaviour of the stream at a certain period of time before (and after) going through a drift. With these windows in hand, we are able to aggregate and extract meta-features that map their behaviours.

Table 1
Meta-features used in our experiments generated by TSFEL [30]. s is the signal vector, t is the time vector represented by $(i/fs)_{i=0}^{N}$ where fs is the sampling rate and N is the size of s, and Δs is the discrete derivative of s defined by $\Delta s = s_{i+1} - s_i$.

Name	Descriptor	Complexity	Function
Absolute energy	Energy	$O(\log n)$	$\sum_{i=0}^{N} s_i^2$
Total energy	Energy	$O(1)$	$\frac{\sum_{i=0}^{N} s_i^2}{t_N - t_0}$
Autocorrelation	Correlation	$O(1)$	$\sum_{n \in Z} s(n)\overline{s(n-l)}$
Centroid	Neighbourhood	$O(1)$	$\frac{\sum_{i=0}^{N} t_i \times s_i^2}{\sum_{i=0}^{N} s_i^2}$
Neighbourhood peaks	Neighbourhood	$O(1)$	Count the peaks found using continuous wavelet transform with n (default = 0) neighbours
Peak to peak distance	Format	$O(1)$	$\lvert max(s) - \min(s)\rvert$
Negative turning points	Format	$O(1)$	$\sum_{i=0}^{N-1} turn(\Delta s_{i+1}, \Delta s_i)$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Positive turning points	Format	$O(1)$	$\sum_{i=0}^{N-1} turn(\Delta s_i, \Delta s_{i+1})$ where $turn(x,y) = \begin{cases} 1, \text{if } x < 0 \text{ and } y > 0 \\ 0, else \end{cases}$
Slope	Format	$O(\log n)$	m coefficient from fitted equation $s = mt + b$
Zero crossing rate	Format	$O(1)$	$\sum_{i=1}^{N} 1_{\mathbf{R}<0}(s_{i-1}si)$ where $1_{\mathbf{R}<0}$ is an indicator function[32]
Mean absolute diff	Distance	$O(1)$	$mean(\lvert\Delta s\rvert)$
Mean diff	Distance	$O(1)$	$mean(\Delta s)$
Median absolute diff	Distance	$O(1)$	$median(\lvert\Delta s\rvert)$
Median diff	Distance	$O(1)$	$median(\Delta s)$
Signal distance	Distance	$O(1)$	$\sum_{i=0}^{N-1} \sqrt{1 + \Delta s_i^2}$
Sum absolute diff	Distance	$O(1)$	$\sum_{i=0}^{N-1} \lvert\Delta s_i\rvert$
Area under the curve	Statistical	$O(\log n)$	$\sum_{i=0}^{N} (t_i - t_{i-1}) \times \frac{s_i + s_{i-1}}{2}$
Entropy	Entropy	$O(\log n)$	$-\sum_{x \in s} P(x) log_2 P(x)$

Table 2
Performance for each meta-model candidate across 10 repetitions of 10 cross-validation runs.

Name	RMSE	
	Mean	STD
CART	0.284	0.020
MLP (2 Layers: 25x25)	0.780	0.122
Random Forest	0.241	0.006
SVM (Linear)	0.305	0.017
SVM (Polynomial)	0.594	0.085

Meta-features need to tackle the challenge of representing the stream behaviour related to a suitable Z, the threshold of Uncertainty Sampling. As such, besides the meta-feature extraction procedure, we also need to define a particular value of Z for each window. Moreover, it is worth noting that the meta-feature extraction step should have a low computational cost. Considering each stream chunk as a time series, we employed the descriptors of Energy, Correlation, Neighbourhood, Format, Distance, Statistics and Entropy. The complete list of descriptors and features can be found in Table 1. From the descriptors, each extracted meta-feature has calculated its mean, minimum, maximum, and standard deviation values.

In our experiments, the meta-feature extraction was performed using the *TSFEL* [30] library with a fixed sampling frequency (fs) of 20% of window size. This value determines the number of samples that should be sampled for each unit of time, e.g. when set to 20%, for every 5 units of time 1 sample will be used, since $20\% = \frac{1}{5}$. Since only a few meta-features require the setting of this value (in our case, only 3), our fs was chosen empirically.

The *TSFEL* library essentially aggregates every sample in our window and calculates the meta-features for each feature present in the original stream. Since this would generate a discrepancy in the number of meta-features across different streams, we further aggregate these "meta-features per feature" into summarised meta-features. They represent the minimum value found in all features for that meta-feature, the maximum value, the average, and the variance between the values. This reduces our meta-database from *Nmeta-features*∗*Mfeatures* to *Nmeta-features*∗4.

Most of our meta-features are calculated in $O(1)$ time with a few taking $O(\log n)$ time. We also need to add the cost of summarising our meta-features, which is comprised of $O(n)$. Even with our summarised meta-features, our costs remain relatively low, ensuring that our framework is capable of readily responding to drifts.

By performing all these steps, we are then able to convert a stream window W into a single meta-database sample M_k. All generated samples are then combined into a single meta-database M that is then used to train our meta-model.

Analysing the feature importance provided by Random Forest, we observed the Energy features as the most important with a score of 0.055, followed by Correlation (0.016) and Neighbourhood (0.010). Entropy meta-features have the lowest importance with a score of 0.001.

The meta-model needs to predict, as a regression task, the Z value that provides the maximum accuracy with the lowest possible number of queries. The values of Z that match this requirement were discovered using the procedure found in Algorithm 1. In our experiments, we set the selection margin s to 0.02. The higher this value, the more the algorithm will favour query rate over accuracy.

We evaluated different machine learning regressors such as Support Vector Machine (linear and polynomial kernel), CART Decision Trees, Multi-Layer Perceptron (with several architectures), and Random Forest. The last one presented the smaller predictive error, the results are summarised in Table 2.

Thus, using a Random Forest as a regressor, we obtained a *RMSE* of 0.241 ($\pm$0.006) after 10 repetitions of 10-fold cross-validation strategy. Our meta-model θ_x was built without a tuning procedure since after testing there are no relevant improvements over the default hyperparameters.

4.2. Uncertainty sampling stream benchmarks

For our experiments, we used 34 benchmark datasets from multiple sources. They are listed in Table 3, the number column is also used with reference to them in the other tables. They are essentially split into 8 groups: CTU-13 [33], Electricity [34], RandomRBF [35], LED24 [36], Hyperplane [37], Poker Hand [38], SEA [39] and Insects [40]. The selected datasets represent different problem complexities, number of instances, features, classes, and drift scenarios. Each of these categories, except out-of-control, makes up two datasets, one with a balanced class distribution and another imbalanced [40].

Table 3
Benchmark stream datasets used in our experiments.

Number	Name	Instances	Features	Classes	Drifts	Type
1	CTU-13 - Scenario 1 [33]	2,824,636	11	2	38	Real
2	CTU-13 - Scenario 2 [33]	1,808,122	11	2	35	Real
3	CTU-13 - Scenario 3 [33]	4,710,638	11	2	5	Real
4	CTU-13 - Scenario 4 [33]	1,121,076	11	2	3	Real
5	CTU-13 - Scenario 5 [33]	129,832	11	2	1	Real
6	CTU-13 - Scenario 6 [33]	558,919	11	2	9	Real
7	CTU-13 - Scenario 7 [33]	114,077	11	2	3	Real
8	CTU-13 - Scenario 8 [33]	2,954,230	11	2	3	Real
9	CTU-13 - Scenario 9 [33]	2,087,508	11	2	43	Real
10	CTU-13 - Scenario 10 [33]	1,309,791	11	2	15	Real
11	CTU-13 - Scenario 11 [33]	107,251	11	2	19	Real
12	CTU-13 - Scenario 12 [33]	325,471	11	2	29	Real
13	CTU-13 - Scenario 13 [33]	1,925,149	11	2	39	Real
14	Electricity [34]	45,312	7	2	27	Real
15	RandomRBF 250k samples, 50 features [35]	250,000	50	2	9	Synthetic
16	RandomRBF 500k samples, 10 features [35]	500,000	10	2	20	Synthetic
17	RandomRBF 1M samples, 10 features [35]	1,000,000	10	2	25	Synthetic
18	LED24 1M samples, 0% noise [35]	1,000,000	24	10	10	Synthetic
19	LED24 1M samples, 10% noise [35]	1,000,000	24	10	1	Synthetic
20	LED24 1M samples, 20% noise [35]	1,000,000	24	10	0	Synthetic
21	Hyperplane [37]	250,000	10	2	4	Synthetic
22	Poker Hand [38]	829,201	10	10	153	Synthetic
23	SEA [39]	60,000	3	2	4	Synthetic
24	Insects - Abrupt Imbalanced [40]	452,044	33	6	73	Real
25	Insects - Incremental Imbalanced [40]	143,323	33	6	43	Real
26	Insects - Incremental-gradual Imbalanced [40]	355,275	33	6	61	Real
27	Insects - Incremental-reoccurring Imbalanced [40]	452,044	33	6	91	Real
28	Insects - Incremental-abrupt Imbalanced [40]	452,044	33	6	84	Real
29	Insects - Abrupt Balanced [40]	57,018	33	6	12	Real
30	Insects - Incremental Balanced [40]	24,150	33	6	14	Real
31	Insects - Incremental-gradual Balanced [40]	52,848	33	6	33	Real
32	Insects - Incremental-reoccurring Balanced [40]	79,986	33	6	47	Real
33	Insects - Incremental-abrupt Balanced [40]	79,986	33	6	56	Real
34	Insects - Out-of-control [40]	905,145	33	24	21	Real

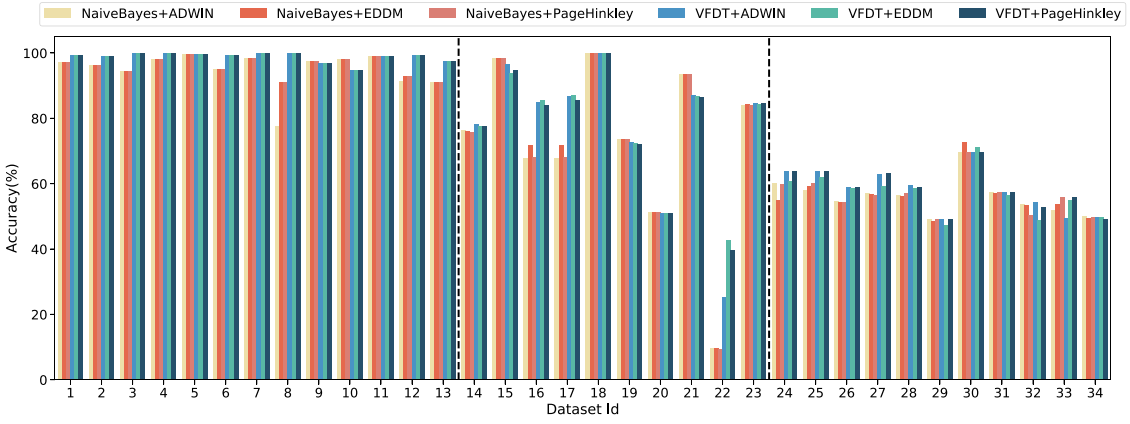

Fig. 2. Accuracy between different combinations of algorithms (Naive Bayes and VFDT) and detectors (ADWIN, Page Hinkley and EDDM). Each bar graph represents a subset of stream datasets.

4.3. Performance with different detectors and classifiers

The first experiment was conducted to support the investigation of possible bias or effects for the use of a particular change detector or classifier. We explored six different combinations of NB and VFDT classifiers to three change detectors (ADWIN, EDDM and Page Hinkley) observing their average accuracy and percentage of queries requested during the classification of the whole stream datasets.

In Fig. 2 is possible to see the accuracy obtained from all datasets. This figure has three regions highlighted (scenarios 1–13, scenarios 14–23, and scenarios 24–34) to provide different perspectives of analysis.

Regarding the first region, CTU datasets (scenarios from 1 to 13), VFDT-based combinations were able to provide superior results for scenarios 1, 2, 3, 4, 6, 7, 8, 12 and 13. In Scenarios 5 and 11 a very similar performance of VDFT and NB was observed. However, in scenarios 9 and 10 it was observed superior results from the NB classifier. It is important to note that the CTU-based datasets present a high predictive performance superior to 98%.

The remaining scenarios (from 14 to 34) are dataset streams with tricky predictions, ranging from 53% to 100% when using all labels to classify new samples. Again, VFDT presented superior accuracy in the major part of the scenarios, except dataset 15 (RandomRBF 250k sample) and 21 (Hyperplane). In particular, when observing different change detectors regardless of the clas-

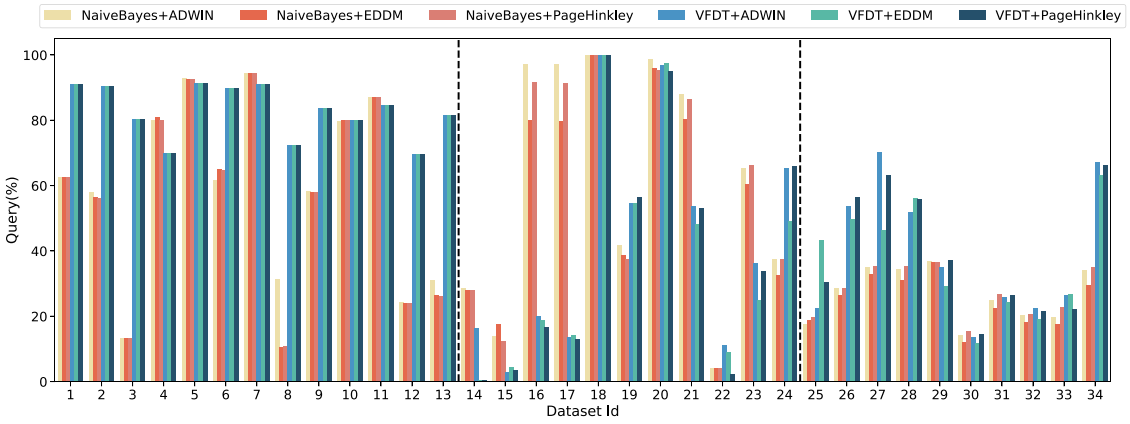

Fig. 3. Percentage of queries between different combinations of algorithms (Naive Bayes and VFDT) and detectors (ADWIN, Page Hinkley and EDDM). Each bar graph represents a subset of stream datasets. 100% of queries regards the usage of requesting labels for all stream sample.

Table 4
Accuracy percentage of all Z values across all 34 datasets and their standard deviation. The best performance per dataset is marked in bold and the standard deviation is shown in parenthesis (also in percentages). The underlined values in the All Labels column indicate that it lost accuracy to Active Learning techniques.

Datasets	Fixed Z					Z_MtL	All Labels
	0.05	0.1	0.2	0.5	0.7		
1	**99.30 (0.62)**	**99.30 (0.62)**	**99.30 (0.62)**	**99.30 (0.62)**	**99.30 (0.62)**	**99.30 (0.62)**	99.80 (0.81)
2	**98.90 (0.60)**	**98.90 (0.60)**	**98.90 (0.60)**	**98.90 (0.60)**	**98.90 (0.60)**	**98.90 (0.60)**	99.60 (0.71)
3	**99.80 (0.19)**	**99.80 (0.19)**	**99.80 (0.19)**	**99.80 (0.19)**	**99.80 (0.19)**	**99.80 (0.19)**	100.00 (0.04)
4	**99.90 (0.07)**	**99.90 (0.07)**	**99.90 (0.07)**	**99.90 (0.07)**	**99.90 (0.07)**	**99.90 (0.07)**	100.00 (0.07)
5	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	**99.70 (0.28)**	99.90 (0.27)
6	**99.30 (0.22)**	**99.30 (0.22)**	**99.30 (0.21)**	**99.30 (0.21)**	**99.30 (0.21)**	**99.30 (0.21)**	99.90 (0.27)
7	**99.90 (0.04)**	**99.90 (0.04)**	**99.90 (0.04)**	**99.90 (0.04)**	**99.90 (0.04)**	**99.90 (0.04)**	99.90 (0.25)
8	**99.80 (0.03)**	**99.80 (0.03)**	**99.80 (0.03)**	**99.80 (0.03)**	**99.80 (0.03)**	**99.80 (0.03)**	100.00 (0.08)
9	**96.90 (3.40)**	**96.90 (3.40)**	**96.90 (3.40)**	**96.90 (3.40)**	**96.90 (3.40)**	**96.90 (3.40)**	98.50 (1.74)
10	**94.80 (4.40)**	**94.80 (4.40)**	**94.80 (4.40)**	**94.80 (4.40)**	**94.80 (4.40)**	**94.80 (4.40)**	100.00 (0.18)
11	**98.90 (2.52)**	**98.90 (2.52)**	**98.90 (2.52)**	**98.90 (2.52)**	**98.90 (2.52)**	**98.90 (2.52)**	99.40 (2.97)
12	**99.20 (0.39)**	**99.20 (0.38)**	**99.20 (0.38)**	**99.20 (0.38)**	**99.20 (0.38)**	**99.20 (0.38)**	99.40 (1.52)
13	**97.60 (0.45)**	**97.60 (0.45)**	**97.60 (0.45)**	**97.60 (0.45)**	**97.60 (0.45)**	**97.60 (0.45)**	99.50 (0.85)
14	**79.90 (2.10)**	79.40 (2.84)	78.80 (2.84)	77.70 (3.37)	77.70 (3.37)	78.00 (3.09)	78.70 (7.61)
15	97.40 (1.89)	95.30 (2.19)	**97.70 (1.69)**	95.70 (1.95)	74.30 (8.66)	97.10 (1.25)	98.40 (1.60)
16	**87.10 (4.35)**	86.60 (4.31)	86.90 (4.11)	86.30 (4.09)	84.30 (3.02)	85.70 (4.41)	89.90 (3.61)
17	**89.10 (3.66)**	88.60 (3.66)	88.80 (3.46)	87.90 (3.31)	85.70 (2.57)	87.40 (3.56)	91.10 (2.96)
18	**99.80 (0.80)**	**99.80 (0.80)**	**99.80 (0.80)**	**99.80 (0.80)**	**99.80 (0.80)**	**99.80 (0.80)**	100.00 (0.34)
19	72.90 (0.62)	**73.10 (0.62)**	72.90 (0.65)	72.50 (0.70)	72.80 (0.63)	72.60 (0.68)	73.70 (2.03)
20	**51.10 (0.36)**	**51.10 (0.36)**	**51.10 (0.35)**	**51.10 (0.36)**	51.00 (0.36)	51.00 (0.36)	51.10 (2.33)
21	**87.70 (0.81)**	87.50 (0.77)	86.50 (0.71)	86.90 (0.80)	87.20 (0.80)	86.80 (0.71)	89.20 (1.69)
22	42.00 (3.98)	43.20 (6.12)	**50.50 (2.04)**	45.80 (7.32)	33.70 (6.12)	42.80 (6.87)	74.10 (11.62)
23	**84.80 (1.26)**	**84.80 (1.26)**	**84.80 (1.36)**	84.60 (1.50)	83.20 (1.69)	84.60 (1.51)	84.70 (2.65)
24	63.00 (3.56)	**63.30 (3.52)**	62.30 (3.54)	63.20 (2.65)	**63.30 (4.39)**	62.10 (3.01)	67.10 (8.89)
25	64.90 (2.58)	**65.90 (2.45)**	64.90 (2.70)	62.80 (3.31)	65.20 (1.90)	64.70 (3.74)	57.70 (12.18)
26	**61.30 (2.48)**	60.70 (2.05)	60.90 (1.62)	57.50 (1.46)	57.70 (1.97)	60.60 (2.39)	66.20 (11.81)
27	62.80 (2.71)	61.80 (2.21)	**64.00 (2.48)**	60.20 (2.57)	58.80 (3.89)	61.60 (2.54)	63.00 (10.28)
28	60.10 (1.29)	**62.30 (2.12)**	61.40 (1.92)	58.90 (1.87)	58.10 (2.55)	60.20 (2.96)	64.60 (11.42)
29	48.40 (2.86)	**49.30 (3.41)**	49.20 (3.16)	49.00 (3.02)	48.20 (2.76)	49.00 (2.89)	52.20 (6.59)
30	69.60 (3.54)	69.50 (3.80)	71.50 (4.16)	69.60 (4.49)	72.10 (5.29)	**75.10 (5.51)**	60.90 (15.46)
31	57.40 (3.12)	57.90 (2.62)	57.80 (2.93)	57.50 (2.81)	57.70 (2.81)	**58.10 (2.89)**	53.80 (15.68)
32	55.00 (1.80)	53.90 (1.79)	55.00 (1.98)	53.40 (2.46)	53.10 (1.80)	**56.30 (2.45)**	53.30 (20.07)
33	57.80 (2.45)	58.70 (1.64)	57.30 (2.41)	58.10 (3.28)	50.60 (5.47)	**59.60 (2.13)**	57.70 (15.86)
34	**50.40 (2.75)**	**50.40 (2.45)**	49.70 (2.95)	49.40 (2.05)	49.00 (2.32)	49.50 (2.98)	56.30 (4.76)
Total Avg.	89.65 (0.17)	89.67 (0.17)	**89.88 (0.17)**	89.52 (0.17)	88.84 (0.19)	89.51 (0.17)	90.12 (0.16)

sifier, there are particular streams where EDDM outperformed ADWIN and Page Hinkley, (i.e., datasets 16, 17, 22, and 30). ADWIN and Page Hinkley presented a very similar performance, except for dataset 22 (Poker Hand). It is important to note that during the experiments using CTU datasets, the impact in terms of accuracy was observed in classifier selection. On the contrary, in the INSECTS datasets, one noted the influence of both classifier and change detector effect on the accuracy.

Fig. 3 reveals a slight modification of query percentage for different detectors, most of them presented in the INSECTS datasets. Conversely, in the CTU streams, just scenarios 8 and 13 presented variations, particularly when classifying the samples using Naive Bayes. Regarding classifiers, it is not possible to observe a classifier as more economic than others in terms of querying. On one hand, VFDT queried less for RandomRBF-based streams. On the other hand Naive Bayes did the same for the Insects streams.

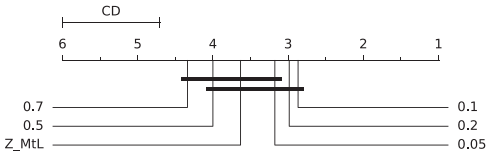

Fig. 4. Nemenyi post hoc test (significance of $\alpha = 0.05$ and critical distance (CD) of 1.29) considering the accuracy obtained using static thresholds Z (0.05, 0.1, 0.2, 0.5 and 0.7) and our proposal (*Z_MtL*).

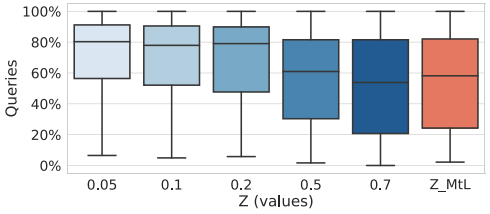

Fig. 5. Query demand all datasets for static thresholds (0.05, 0.1, 0.2, 0.5, 0.7) and our proposal (*Z_MtL*).

Table 5
Relative query number (in percentage) of all Z values across all 34 datasets. The most reduced number of queries was marked in bold.

Datasets	Fixed Z					*Z_MtL*
	0.05	0.1	0.2	0.5	0.7	
1	**91.20**	**91.20**	**91.20**	**91.20**	**91.20**	**91.20**
2	**90.50**	**90.50**	**90.50**	**90.50**	**90.50**	**90.50**
3	**80.30**	**80.30**	**80.30**	**80.30**	**80.30**	**80.30**
4	**70.00**	**70.00**	**70.00**	**70.00**	**70.00**	**70.00**
5	**91.30**	**91.30**	**91.30**	**91.30**	**91.30**	**91.30**
6	92.60	90.40	**89.90**	**89.90**	**89.90**	**89.90**
7	**91.00**	**91.00**	**91.00**	**91.00**	**91.00**	**91.00**
8	**72.50**	**72.50**	**72.50**	**72.50**	**72.50**	**72.50**
9	**83.60**	**83.60**	**83.60**	**83.60**	**83.60**	**83.60**
10	**80.00**	**80.00**	**80.00**	**80.00**	**80.00**	**80.00**
11	**84.60**	**84.60**	**84.60**	**84.60**	**84.60**	**84.60**
12	70.00	**69.70**	**69.70**	**69.70**	**69.70**	**69.70**
13	**81.50**	**81.50**	**81.50**	**81.50**	**81.50**	**81.50**
14	75.80	49.00	34.60	1.70	**0.50**	21.30
15	7.00	4.90	5.80	3.90	**3.30**	4.50
16	57.00	52.20	41.20	24.40	**16.80**	20.40
17	50.90	47.50	35.10	18.20	**12.90**	15.30
18	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
19	91.50	94.40	89.80	60.90	59.10	**58.20**
20	99.90	99.90	99.90	98.70	**98.40**	99.40
21	91.60	88.50	77.80	56.50	**46.10**	54.00
22	6.50	5.80	14.00	3.70	5.50	**2.20**
23	95.40	93.20	84.80	38.40	**28.20**	36.00
24	80.10	78.80	79.30	65.40	53.80	**45.90**
25	56.10	52.10	47.70	30.30	31.20	**19.90**
26	70.90	66.10	67.80	41.40	**28.80**	38.40
27	82.00	70.50	76.10	65.40	**47.70**	56.60
28	80.30	77.50	79.40	51.20	49.80	**48.90**
29	56.40	51.40	49.10	38.10	**30.90**	36.60
30	29.90	25.00	21.70	15.50	**11.30**	11.90
31	44.20	42.50	35.40	27.40	**20.80**	25.30
32	48.70	44.10	34.10	24.10	**17.70**	20.30
33	51.70	45.20	36.50	27.00	17.00	**16.60**
34	85.60	77.90	79.00	70.30	**49.10**	67.60
Total Avg.	78.68	77.80	77.22	73.06	**71.24**	72.08

In general, we can affirm that classifiers influence more the predictive performance and the number of queries than the change detectors. VFDT was superior in terms of accuracy, but it is not possible to affirm the best classifier in terms of querying reduction. There are particular improvements provided by different classifiers for specific datastream scenarios. Regarding change detectors, ADWIN and Page-Hinkley behave similarly and EDDM provided more accurate results in some specific cases.

4.4. Stream classification with dynamic thresholding

This experiment is focused on comparing the overall performance of five static Z values (0.05, 0.1, 0.2, 0.5, and 0.7) on 34 stream datasets to address RQ1 using ADWIN and VFDT as change detector and classifier. This combination was selected because it balances the trade-off between predictive performance and query reduction, providing good insights for future discussions (Section 4.4, Section 4.5, and Section 4.6). Table 4 collects the results of the average accuracy throughout the stream for all datasets with the most accurate values of Z marked in bold. It is possible to observe several similar high accuracy results across all static configurations and the dynamic one (*Z_MtL*), particularly on 13 datasets (all from the CTU-13 group). This shows that the different uncertainty sampling setups could not lead to improvements in these datasets, since all experimented Z deliver similar predictive performance.

On the one hand, static Z values of 0.05, 0.1, and 0.2 led to highly accurate results. On the other hand, 0.7 and 0.5 delivered lower accuracy.

It is important to note that the stream datasets affected by concept drift (datasets from 24 to 33) revealed different rankings of the best accurate Z value. Instead of the most restrictive ones taking over the top rankings, we observed the presence of medium and high Z, e.g., 0.2 and 0.7, achieving better performance than 0.05 and 0.1. Furthermore, there are particular real and synthetic datasets in which *Z_MtL* obtained the best performance (datasets 30, 31, 32, and 33), even increasing the accuracy of using all labels. We marked with underline in Table 4 those streams that the uncertainty sampling strategy improves the accuracy when compared to all available labels. A great part of them underwent the concept drift effect.

Table 4 also provides the average accuracy across each Z, and here we can observe that, while all values are very similar, less restrictive values of Z, such as 0.05, 0.1, and 0.2 obtained the highest accuracy values with (89.65%), (89.67%) and (89.88%) respectively, losing only to the non usage of Active Learning. As discussed before, the more restrictive Z, the more reduced the accuracy is, with a Z of 0.7 leading to the lowest average performance (88.84%), also interesting to note that this Z achieved the highest standard deviation, showing that it was the least stable among the other values. We note that *Z_MtL* achieved a similar median value (88.42%) to the usage of all labels when using a VFDT. This achievement brought some insights related to RQ2, since the presence of concept drift can deteriorate the incremental tree performance when fed by all labels. This topic is discussed in Section 4.6.

Focusing on finding a superior method to setup Z, we evaluated the results based on statistical analysis based on the non-parametric Friedman test to determine any significant differences among the fixed Z values and the dynamic recommended ones (*Z_MtL*) using 34 datasets. We used the post hoc Nemenyi test to infer which differences are statistically significant. As Fig. 4 shows, differences between populations are significant. Particularly, we assume that there are no significant differences within Z of 0.1, 0.05, 0.2, 0.5 and *Z_MtL*; Z of 0.2, 0.5 and *Z_MtL*. All other differences are significant. In other words, *Z_MtL* achieved results statistically equal to the usage of the most reduced Z values, those that query more frequently for labels and provide high-accuracy values.

It is worth mentioning that low Z values tend to obtain high predictive performance since the demand for more labels. Thus, the number of queries required for each Z value is a complementary analysis to address the trade-off between performance and querying expected by using Active Learning.

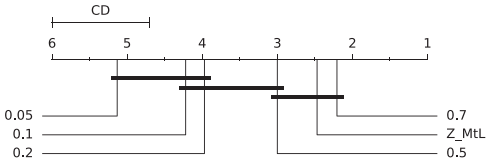

Fig. 6. Nemenyi post hoc test (significance of $\alpha = 0.05$ and critical distance (CD) of 1.29) considering the number of queries requested using static thresholds (0.05, 0.1, 0.2, 0.5 and 0.7) and our proposal (*Z_MtL*).

4.5. Querying rate using dynamic threshold tuning

Fig. 5 shows the percentage of queries from the evaluated datasets across the static thresholds and *Z_MtL*. Low *Z* values (0.05, 0.1, and 0.2) required an average of 70.15% of instance labels. On the other hand, the more restrictive *Z* (0.5 and 0.7) and the dynamic one (*Z_MtL*) restricted the demand for labels to 54.78% of the total samples. Similarly to the accuracy evaluation, it was possible to observe several similar querying numbers when classifying CTU-13 streams (datasets 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13), as in Table 5 is shown. Moreover, the accuracy was similar to the usage of all labels, but training the VFDT with fewer labels, e.g., UCT13 - Scenario 12, achieved 99.20% of accuracy querying 69.70% of labels.

The datasets 15 (RandomRBF 250k) and 22 (Poker Hand) had a number of queries inferior to 10% of the total available instances, without compromising the accuracy when compared to the usage of all labels. In particular, all labels were required when processing Dataset 18 (LED24 1M samples with 0% noise). *Z_MtL* recommended values were able to reach the smaller number of queries in 6 datasets (19, 22, 24, 25, 28, and 33), most parts of them were affected by concept drift.

We evaluate the performance of the reduction in terms of queries requested based on a statistical analysis grounded in the non-parametric Friedman test to determine any significant differences between the fixed *Z* values and the dynamic recommended ones (*Z_MtL*) using 34 datasets. Again, we took advantage of the post hoc Nemenyi test to infer which differences are statistically significant. Fig. 6 shows the Nemenyi post hoc test on the number of queries, where we can observe that differences between populations are significant. Particularly, we assume that there are no significant differences within *Z* of 0.7 and *Z_MtL*; *Z* of 0.7 and 0.5; *Z* of 0.5 and 0.2; *Z* of 0.2, 0.1 and 0.05. All other differences are significant. In other words, *Z_MtL* achieved results statistically equal to the usage of the highest *Z* value, the most restricted when querying labels.

It is important to emphasise that the dynamic recommended *Z* (*Z_MtL*) obtained is statistically similar to the less restrictive *Zs* (0.5, 0.1 and 0.2) in terms of accuracy. Moreover, *Z_MtL* values were statistically similar to the most restrictive *Zs* (0.7 and 0.5) when reducing the query demand. In other words, our proposed

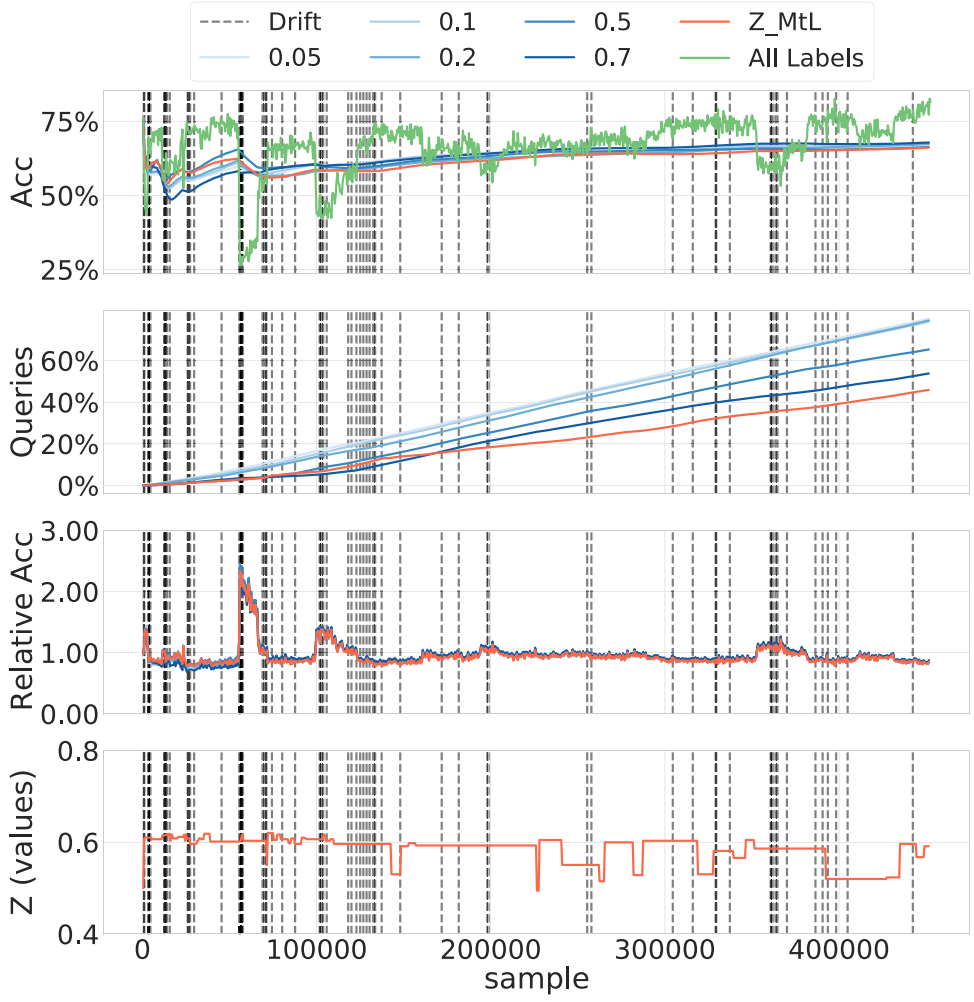

Fig. 7. Accuracy, Queries, Relative Accuracy and dynamically adjusted *Z* values from experiments using Insects - Incremental Imbalanced stream (Dataset 25).

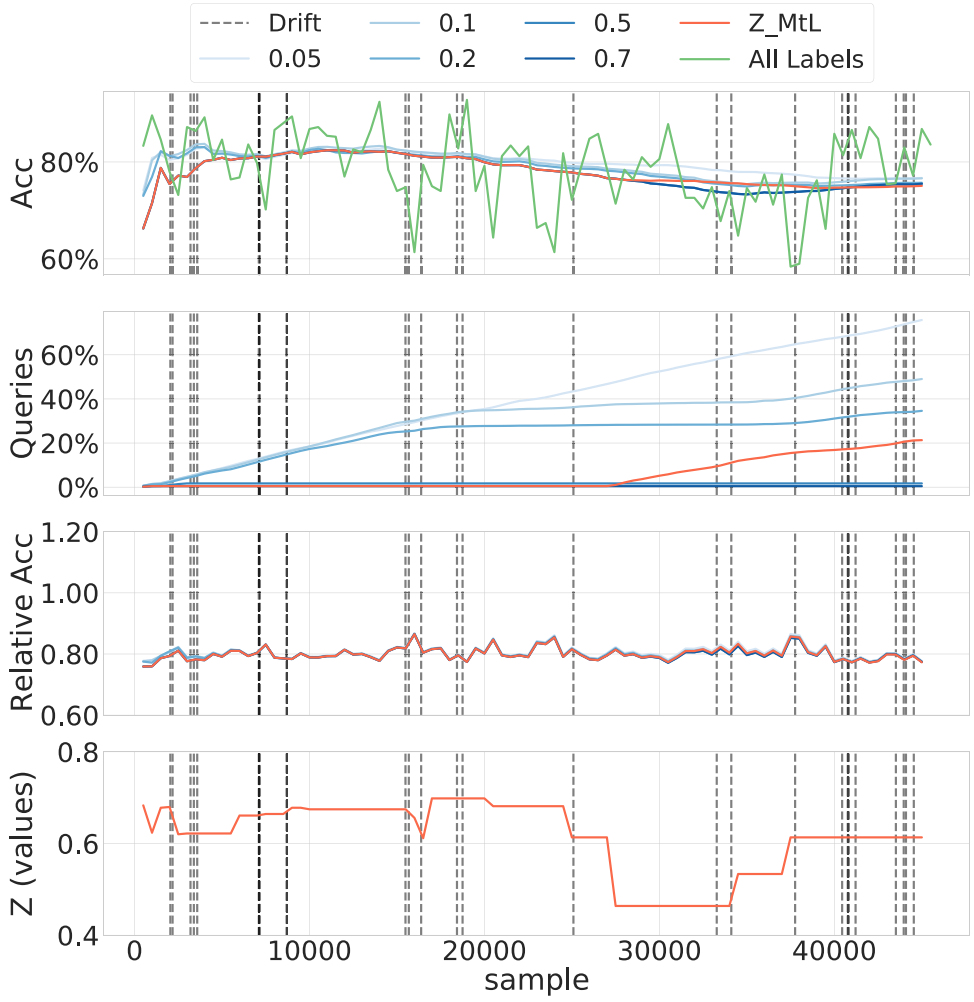

Fig. 8. Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Electricity stream (Dataset 14).

approach provides the best trade-off between accuracy and query reduction by dynamic tuning Z values using lightweight meta-features.

Additionally, we observed the accuracies of datasets 30, 31, 32, and 33 (Table 4) were boosted by the use of uncertainty sampling compared to the use of all available labels to incrementally induce VFDT. Precisely, *Z_MtL* improved from an accuracy from 60.90%(±15%) to 75.10%(±6%) in Dataset 30 (Insects - Incremental Balanced); from 53.80%(±15%) to 58.10%(±6%) in Dataset 30 (Insects - Incremental-gradual Balanced); from 53.30%(±15%) to 56.30%(±6%) in Dataset 30 (Insects - Incremental-reoccurring Balanced); from 57.70%(±15%) to 59.60%(±6%) in Dataset 30 (Insects - Incremental-abrupt Balanced). All of these data sets shared the presence of concept drift in a balanced classification problem. Putting some explanation to this finding and answering RQ3, we observed the effect of concept drift over the accuracy and number of queries, as discussed in Section 4.6

4.6. Uncertainty sampling and drift adaptation

Our experiments brought some insights into the concept drift adaptation using Active Learning on streams. Remarkably, the Uncertainty Sampling reveals the potential of selecting the most informative features when adapting the classifier to a drift. Regarding RQ3, Fig. 7 shows the results for Dataset 25 (Insects - Incremental Imbalanced), where we can see how Z tends to change in accordance with concept drifts when running on *Z_MtL* usually by decreasing it and allowing the learner to query more instances and adapt to changes faster while increasing it in stagnated periods to avoid unnecessary querying, this helps us answer RQ3 since we can see that *Z_MtL* will control the querying rate in drift situations.

Using the Electricity stream from different perspectives as Fig. 8 shows, we can observe the drift starting points and the particular changes in terms of accuracy, the number of queries, and uncertainty sampling Z values. Observing the accuracy, it is evident that for all labels this metric changes considerably after a drift point. On the other hand, using uncertainty sampling, a smooth and regular accuracy was obtained. Regarding the number of queries, a strict relation was not observed between drift points and a rough modification. We can affirm that Z equals 0.05 that grows linearly throughout the stream. This behaviour is similar to 0.1 and 0.2, but with a small gradient. *Z_MtL* started to present this behaviour at 28,000, but was not devoted to a drift point. Relative accuracy was barely affected. Regarding the dynamically adjusted Z (bottom chart in Fig. 8), we can observe variations from 0.43 to 0.71, emphasising the demand for tuning the threshold of uncertainty sampling.

Observing *Z_MtL* results, which overcame the usage of all labels, we selected Dataset 31 (Insects - Incremental and Gradual) to explore some particular behaviours that are shown in

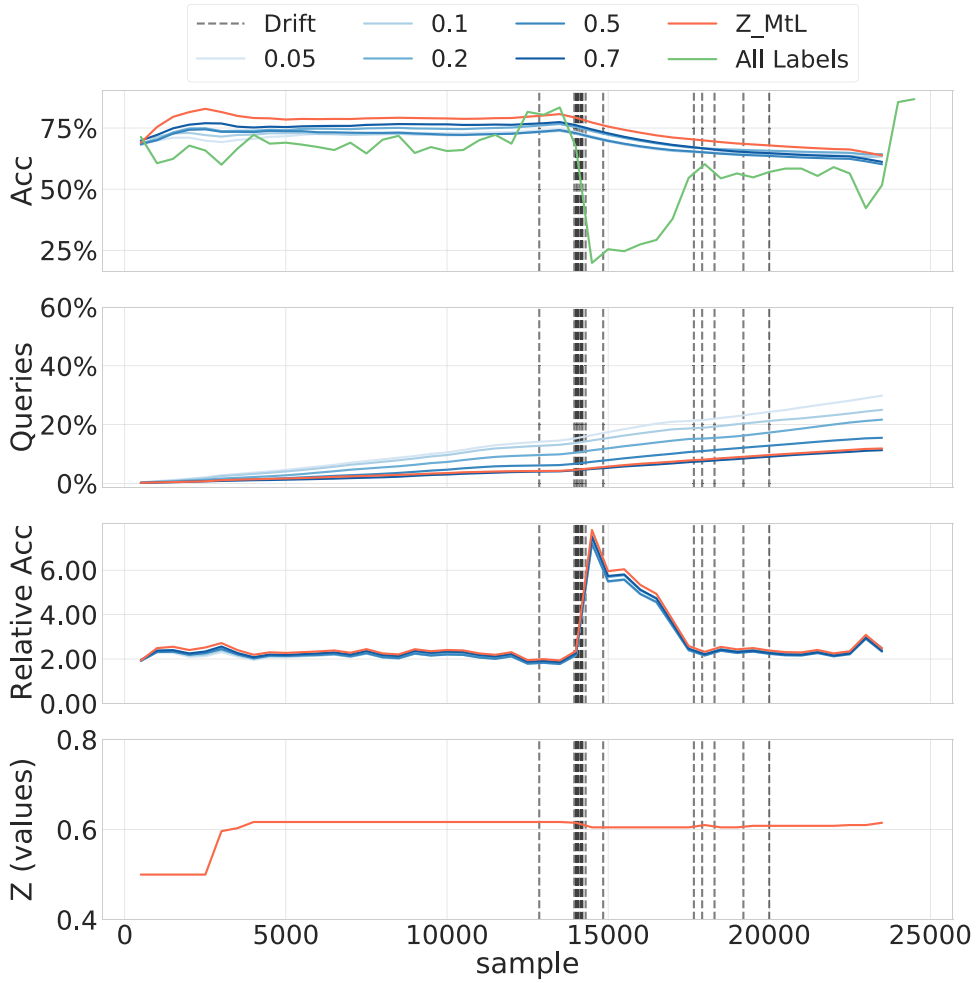

Fig. 9. Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Insects - Incremental and Gradual Balanced stream (Dataset 31).

Fig. 9. Queries and Z values followed similar results. Slightly linear queries have their slope related to Z, achieving higher inclination with less restrictive values. This data stream has a gradual concept drift starting at around 14,000 which reduced the accuracy of VFDT when using all labels. When using Active Learning for training, the VFDT was hardly affected by the gradual drift, presenting a smooth reduction in accuracy. It is important to mention that after the concept drift, the relative accuracy exceeds 6.00, with *Z_MtL* delivering the most accurate result. VFDT was able to recover the performance, i.e., adjust to the drift, approximately at the point 18,000. This window from 14,000 to 18,000 was appropriately handled by uncertainty sampling, allowing the VFDT to achieve superior results compared to the usage of all labels.

The Dataset 26 (Insects - Incremental and Gradual Imbalanced) shown in Fig. 10 further reinforces our observations, in this dataset, notably, the VFDT accuracy suffers dramatically across different points of drift when using all labels while *Z_MtL* remains relatively stable even if its accuracy suffers marginal losses, while also maintaining a very low query rate compared to other Active Learning techniques, never surpassing the 20% mark. These achievements help to answer RQ4 since scenarios over concept drift can take advantage of frequent and accurate updating of Z values. We can confirm that by selecting important features and avoiding irrelevant ones, it is possible to improve the predictive performance.

This robust capacity to adjust to the concept drift by outperforming the usage of all available labels is highly suitable for various concept drift scenarios, leading to increasing VFDT drift adaptability. This improvement was observed in the great part of the experiments with datasets affected by concept drift (datasets 25, 26, 30, 31, 32, 33, and 34), half of them with dynamic tuning (*Z_MtL*) as the most predictive approach.

The main advantages of the proposed method are: i) Online selection of suitable Z values based on lightweight features. ii) Competitive results using *Z_MtL*, which can overcome the usage of all labels with fewer samples. iii) Capacity to provide concept drift adaptation for simple algorithms such as VDFT. Finally, we consider that all research questions were addressed positively, which indicates that our proposal is a promising Stream-based Active Learning taking advantage of Uncertainty Sampling for label querying. Our proposed approach has some limitations when dealing with signals without drifts, since we introduce additional active learning effort with no meaning. Furthermore, by using classifiers that address concept drift, the advantages of our proposal could be reduced, as the active learning approach might not be effective.

5. Conclusion

The proposed approach aims to reduce the overhead and costs of choosing the best Uncertainty threshold (Z), automatically, through Meta-Learning. Also, allowing changes and automatic tuning through the process of processing the stream. Our tests revealed a significant improvement in the use of Uncertainty Sam-

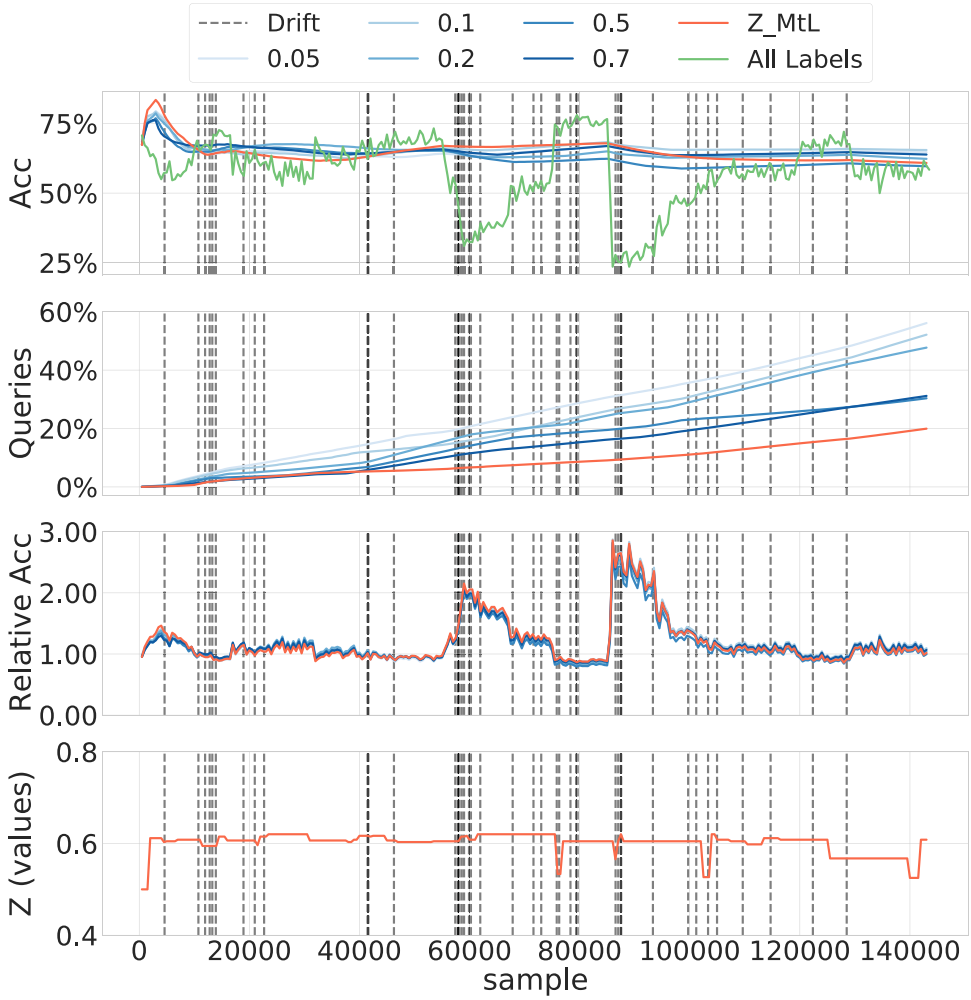

Fig. 10. Accuracy, Queries, Relative Accuracy and dynamically adjusted Z values from experiments using Insects - Incremental and Gradual Imbalanced stream (Dataset 26).

pling when selecting samples for stream classification. By using our solution to dynamically select Z, it was possible to improve the accuracy by considerably reducing the demand for labels, while maintaining a steady adaptation to the introduced concept drifts. In particular, due to our framework adapting well to the newly introduced concept drifts, we managed to obtain superior performance when compared to regular supervised classification by using all labels. The results obtained encourage us to pursue further work in investigating the dynamic tuning of alternative Active Learning approaches, Ensemble and Adaptive Classifiers and the use of budgeting for further querying reduction.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] P. Domingos, G. Hulten, Mining high-speed data streams, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
[2] V.G.T. da Costa, A.C.P. de Leon Ferreira, S.B. Junior, et al., Strict very fast decision tree: a memory conservative algorithm for data stream mining, Pattern Recognit Lett 116 (2018) 22–28.
[3] H. Yu, W. Liu, J. Lu, Y. Wen, X. Luo, G. Zhang, Detecting group concept drift from multiple data streams, Pattern Recognit (2022) 109113.
[4] B. Krawczyk, A. Cano, Adaptive ensemble active learning for drifting data stream mining, in: International Joint Conference on Artificial Intelligence, 2019, pp. 2763–2771.
[5] G. Alberghini, S.B. Junior, A. Cano, Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams, Neurocomputing 481 (2022) 228–248.
[6] B. Settles, Active Learning Literature Survey, Computer Sciences Technical Report, University of Wisconsin–Madison, 2009.
[7] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, in: International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1994, pp. 3–12.
[8] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: a review, IEEE Trans Knowl Data Eng 31 (12) (2018) 2346–2363.
[9] W. Chu, M. Zinkevich, L. Li, A. Thomas, B. Tseng, Unbiased online active learning in data streams, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 195–203.
[10] I. Žliobaitė, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Trans Neural Netw Learn Syst 25 (1) (2013) 27–39.
[11] Ł. Korycki, A. Cano, B. Krawczyk, Active learning with abstaining classifiers for imbalanced drifting data streams, in: IEEE International Conference on Big Data (Big Data), 2019, pp. 2334–2343.
[12] B. Krawczyk, B. Pfahringer, M. Woźniak, Combining active learning with concept drift detection for data stream mining, in: IEEE International Conference on Big Data (Big Data), 2018, pp. 2239–2244.
[13] J. Shan, H. Zhang, W. Liu, Q. Liu, Online active learning ensemble framework for drifted data streams, IEEE Trans Neural Netw Learn Syst 30 (2) (2018) 486–498.
[14] S. Liu, S. Xue, J. Wu, C. Zhou, J. Yang, Z. Li, J. Cao, Online active learning for

drifting data streams, IEEE Trans Neural Netw Learn Syst (2021) 1–15, doi:10.1109/TNNLS.2021.3091681.
[15] H. Ebbinghaus, Memory: a contribution to experimental psychology, Ann Neurosci 20 (4) (2013) 155.
[16] M.-R. Bouguelia, Y. Belaïd, A. Belaïd, An adaptive streaming active learning strategy based on instance weighting, Pattern Recognit Lett 70 (2016) 38–44.
[17] A. Castellani, S. Schmitt, B. Hammer, Stream-based active learning withverification latency innon-stationary environments, in: Artificial Neural Networks and Machine Learning – ICANN 2022, Springer Nature Switzerland, Cham, 2022, pp. 260–272.
[18] R.S. Oyamada, L.C. Shimomura, S. Barbon, D.S. Kaster, A meta-learning configuration framework for graph-based similarity search indexes, Inf Syst (2022) 102123.
[19] A.L.D. Rossi, A.C.P. de Leon Ferreira de Carvalho, C. Soares, B.F. de Souza, et al., MetaStream: a meta-learning based method for periodic algorithm selection in time-changing data, Neurocomputing 127 (2014) 52–64.
[20] J.N. van Rijn, G. Holmes, B. Pfahringer, J. Vanschoren, Having a blast: Meta-learning and heterogeneous ensembles for data streams, in: International Conference on Data Mining, 2015, pp. 1003–1008.
[21] R. Anderson, Y.S. Koh, G. Dobbie, A. Bifet, Recurring concept meta-learning for evolving data streams, Expert Syst Appl 138 (2019) 112832.
[22] S. Ravi, H. Larochelle, Meta-learning for batch mode active learning, 2018.
[23] H. Yu, T. Liu, J. Lu, G. Zhang, Automatic learning to detect concept drift, arXiv preprint arXiv:2105.01419 (2021).
[24] G.F.C. Campos, S.M. Mastelini, G.J. Aguiar, R.G. Mantovani, L.F.d. Melo, S. Barbon, Machine learning hyperparameter selection for contrast limited adaptive histogram equalization, EURASIP J Image Video Process 2019 (1) (2019) 1–18.
[25] R.G. Mantovani, A.L. Rossi, E. Alcobaça, J. Vanschoren, A.C. de Carvalho, A meta-learning recommender system for hyperparameter tuning: predicting when tuning improves SVM classifiers, Inf Sci (Ny) 501 (2019) 193–221.
[26] T. Mu, H. Wang, C. Wang, Z. Liang, X. Shao, Auto-cash: a meta-learning embedding approach for autonomous classification algorithm selection, Inf Sci (Ny) 591 (2022) 344–364.
[27] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: SIAM International Conference on Data Mining, 2007, pp. 443–448.
[28] R. Sebastião, J.M. Fernandes, Supporting the page-hinkley test with empirical mode decomposition for change detection, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2017, pp. 492–498.
[29] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, volume 6, 2006, pp. 77–86.
[30] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, H. Gamboa, TSFEL: Time Series Feature Extraction Library, SoftwareX 11 (2020) 100456.
[31] J. Montiel, J. Read, A. Bifet, T. Abdessalem, Scikit-multiflow: a multi-output streaming framework, Journal of Machine Learning Research 19 (72) (2018) 1–5.
[32] C.-h. Chen, Signal Processing Handbook, volume 51, CRC Press, 1988.
[33] S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, computers & security 45 (2014) 100–123.
[34] M. Harries, N.S. Wales, Splice-2 comparative evaluation: electricity pricing, Citeseer (1999).
[35] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explorations Newsletter 11 (1) (2009) 10–18.
[36] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression trees, Routledge, 2017.
[37] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 97–106.
[38] D. Dua, C. Graff, UCI machine learning repository, 2017.
[39] W.N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 377–382.
[40] V.M. Souza, D.M. dos Reis, A.G. Maletzke, G.E. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Min Knowl Discov 34 (6) (2020) 1805–1858.

Vinicius Eiji Martins, received his B.Sc. from the State University of Londrina (UEL), Brazil, in 2018. He is currently a Data Engineer at Oktagon Games and enrolled in an M.Sc. program at the State University of Londrina. His current research interests include Active Learning, Meta-Learning and Stream Mining.

Alberto Cano is an Associate Professor with the Department of Computer Science, Virginia Commonwealth University, USA, where he heads the High-Performance Data Mining Lab. He obtained his BSc degrees in Computer Engineering and in Computer Science from the University of Cordoba, Spain, in 2008 and 2010, respectively, and his M.Sc. and Ph.D. degrees in Intelligent Systems and Computer Science from the University of Granada, Spain, in 2011 and 2014 respectively. His research is focused on machine learning, big data, data streams, concept drift, continual learning, GPUs and distributed computing. He has published over 50 articles in high-impact factor journals, 50 contributions to international conferences, two book chapters, and two books in the areas of machine learning, data mining, data streams, and GPU computing. His research is supported by an Amazon AWS Machine Learning award (2018) and the VCU Presidential Research Quest Fund (2018). Dr. Cano is IEEE Senior Member and Associate Editor of IEEE Access, Applied Intelligence, and PeerJ Computer Science.

Sylvio Barbon Junior is an Associate Professor at the Department of Engineering and Architecture of the University of Trieste, Italy, where is co-head of the Machine Learning Lab. Previously was the leader of the research group that studies machine learning in the Computer Science Department at the State University of Londrina (UEL), Brazil. He received his B.Sc. degree in Computer Science in 2005, M.Sc. degree in Computational Physics from the University of So Paulo (2007), a degree in Computational Engineering in 2008, and a Ph.D. degree (2011) from IFSC/USP such as the M.Sc. degree. In 2017, he was a visiting researcher at the University of Modena and Reggio Emilia (Italy) working on multispectral analysis and machine learning. In 2021, as visiting professor at Universita Degli Studi Di Milano (Italy) focused on Data Stream and Process Mining. His research interests include Pattern Recognition, Machine Learning, and Process Mining.