Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Hoeffding adaptive trees for multi-label classification on data streams

Aurora Esteban [a], Alberto Cano [b], Amelia Zafra [a,*], Sebastián Ventura [a]

[a] Dept. of Computer Science and Numerical Analysis, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Cordoba, Cordoba, 14071, Spain
[b] Dept. of Computer Science, Virginia Commonwealth University, Richmond, 23284-3068, VA, USA

ARTICLE INFO

Keywords:
Multi-label classification
Data streams
Incremental decision trees

ABSTRACT

Data stream learning is a very relevant paradigm because of the increasing real-world scenarios generating data at high velocities and in unbounded sequences. Stream learning aims at developing models that can process instances as they arrive, so models constantly adapt to new concepts and the temporal evolution in the stream. In multi-label data stream environments where instances have the peculiarity of belonging simultaneously to more than one class, the problem becomes even more complex and poses unique challenges such as different concept drifts impacting different labels at simultaneous or distinct times, higher class imbalance, or new labels emerging in the stream. This paper proposes a novel approach to multi-label data stream classification called Multi-Label Hoeffding Adaptive Tree (MLHAT). MLHAT leverages the Hoeffding adaptive tree to address these challenges by considering possible relations and label co-occurrences in the partitioning process of the decision tree, dynamically adapting the learner in each leaf node of the tree, and implementing a concept drift detector that can quickly detect and replace tree branches that are no longer performing well. The proposed approach is compared with other 18 online multi-label classifiers on 41 datasets. The results, validated with statistical analysis, show that MLHAT outperforms other state-of-the-art approaches in 12 well-known multi-label metrics.

1. Introduction

Nowadays, many real applications in cyber–physical scenarios generate high volumes of data, arriving in unbounded sequences, at high velocities, and with limited storage restrictions [1,2]. In this context, data stream mining emerges as an important paradigm in which the learner does not have access to all data at once, must process each instance rapidly, and must learn in an online fashion. Thus, online learning aims at developing models capable of constantly expanding and adapting, in contrast to the classic batch learning scenario, where a model is trained once all data are available [3]. Since data streams are evolving, the underlying distribution may eventually change, experiencing concept drift, which can impact the decision boundaries and the performance of the classifier. This change can be abrupt, gradual, or incremental, and it also may be recurring when past concepts reappear [4]. Additional challenges include the high dimensionality of the input space [5], the imbalance ratio in the class labels [6], or the presence of missing labels in the training data [7], which increase the complexity and processing time of the learner and can lead to ignore or forget minority classes.

In this context of complex data arriving rapidly and continuously in a system, that may have limited storage or computation capabilities, decision trees have great potential due to their good balance between accuracy and low complexity. For forecasting on sequential data, for example, there are interesting applications such as the work of Li et al. [8], which proposes a method based on XGBoost to predict energy consumption. For classification, IDTs are very popular and effective algorithms for data stream classification [3]. IDTs are designed to handle such scenarios by incrementally updating the decision model as new data arrives, that is, they learn in a truly online manner, processing instance by instance and being able to produce predictions at any time. This allows the model to adapt to the data-changing distribution and detect changes in the underlying patterns concepts, making them suitable for time-sensitive applications [1,2]. Additionally, IDTs are inherently interpretable models, since humans can analyze the path followed in the classification process. This makes them more reliable models in critical applications. Despite these advantages, IDTs have been applied mainly in the classical scenario of multi-class data stream classification, but not in other more complex scenarios such as multi-label data stream classification.

MLC extends the multi-class paradigm by allowing instances to belong to more than one class simultaneously. Traditional MLC methods are not coping well with the increasing needs of large and complex structures [3], as they assume a batch learning environment where all data are available in advance. Thus, solving this problem with

* Corresponding author.
E-mail addresses: aestebant@uco.es (A. Esteban), acano@vcu.edu (A. Cano), azafra@uco.es (A. Zafra), sventura@uco.es (S. Ventura).

https://doi.org/10.1016/j.knosys.2024.112561
Received 1 June 2024; Received in revised form 10 August 2024; Accepted 27 September 2024
Available online 2 October 2024

online learning is a promising approach in these data-intensive environments. However, online MLC presents additional unique challenges, like different concept drifts impacting different labels at simultaneous or distinct times [6,9], or new labels emerging in the stream [7]. These particularities, along with the own challenges related to both MLC and stream mining, make online MLC especially difficult.

One of the most relevant models in IDTs are Hoeffding Trees (HTs), based on the Hoeffding bound [10], which offers mathematical guarantees of convergence with respect to the equivalent batch-learning decision tree. A HT builds the tree incrementally and never revisits decisions already made. In contrast, Hoeffding Adaptive Tree (HAT) [11] is a very popular evolution that incorporates concept drift detection in each split as a mechanism to replace parts of the tree if they become obsolete. In the multi-label scenario, there is only one native adaptation of this paradigm exists, Multi-Label Hoeffding Tree (MLHT) [12], that adapts classic HT by incorporating a multi-label classifier in the leaves. Other proposals [13–15] include this algorithm in different ensemble approaches to increase performance or include concept drift adaptations. However, there is still room for improvement acting directly on the base model, since a single tree versus an ensemble has multiple advantages in terms of computational load, processing speed, or adaptation to concept drift. Our work aims to fill this gap by providing an accurate model for MLC in data streams with additional difficulties such as concept drift and label set imbalance evolution, contributing to a significant advance in the multi-output IDTs research.

This paper presents a novel approach for multi-label data stream classification based on Hoeffding bound named MLHAT. MLHAT evolves over HAT with three main novelties to deal with MLC and its complex decision space, label imbalance, and fast response to concept drift. The main contributions of the work can be summarized as:

- The presentation of the novel MLHAT algorithm, which makes significant advances in multi-label IDT in terms of:
  - Considering relations and labels' co-occurrences in the partitioning process of the decision tree. The Bernoulli distribution is used to model the probabilities.
  - Dynamically adapting to the high imbalance between labels in the multi-label stream. The number of instances arriving at each leaf node and the impurity among them are monitored to choose between different multi-label classifiers to perform the prediction at leaves.
  - Updating the decision model to the change in the data stream distribution or concept drift. A concept drift detector is inserted into each intermediate node of the tree for building a background branch when its performance starts to decrease. This way, at the moment that concept drift is confirmed, the background branch substitutes the main one in the decision tree.
  - Increasing the diversity in the learning process and the accuracy of the results. Two well-known methods in stream learning are adapted to the model. On the one hand, applying bootstrapping in instances to learn and, on the other hand, combining the predictions produced by the main tree and the possible background branches if they are significant enough.
- An extensive experimental study explores the performance of MLHAT compared to the state of the art in tree-based methods for online MLC. Specifically, 18 classifiers, 41 datasets, and 12 metrics are included in the experimental study.
- The source code of MLHAT and experiments are publicly available[1] for the sake of reproducible comparisons in future work. In addition, all the datasets used in the experimentation are public: either they belong to well-known MLC benchmarks, or they are available in the repository in the case of datasets generated synthetically to include explicit information about concept drift in the multi-label context.

[1] https://github.com/aestebant/mlhat.

The rest of this work is organized as follows. The next section provides a comprehensive review of related works in MLC, starting with the fundamentals and going deeper into previous IDT proposals. Section 3 introduces the MLHAT algorithm. Section 4 presents the experimental study and the results, which compare our approach with state-of-the-art methods on benchmark datasets, in addition to studying its potential in an ensemble architecture. Finally, Section 5 presents the conclusions, summarizes our research findings, and discusses future research directions.

2. Related work

2.1. Multi-label data stream classification

The problem of multi-label data stream classification involves predicting multiple labels or categories for incoming data instances in a streaming fashion. Let S represent the data stream as a potential unbounded sequence of instances $(X_1, Y_1), (X_2, Y_2), \dots (X_i, Y_i), \dots$ where X_i is the feature vector of the ith data instance, and Y_i is a binary vector of labels for the ith data instance with n binary indicators $(y_1, y_2, \dots, y_n)$ for the presence or absence of each label of the label space $\mathcal{L}$ in the ith instance. The goal of multi-label data stream classification is to predict the label vectors Y_i for new, unseen instances of X_i in the stream as they arrive, using a predictive model M, which can be a classifier or a combination of them that can handle the multi-label nature of the problem:

$$M(X_i) = Y_i \quad \forall i \in S \tag{1}$$

In the online learning scenario, instances arrive at the model one at a time, so it must be able to learn incrementally, updating its knowledge to the last characteristics seen in the stream, unlike in the traditional batch learning scenario, where models have all the data available from the beginning for training and are built statically. In this context, the so-called concept drift arises, which may affect the decision boundaries. In concept drift, two factors must be considered. On the one hand, when and how the concept drift appears. Concept drift can occur suddenly, incrementally, gradually, or recurrently [4]. In sudden drift, there is an instant change in the data distribution at a particular time, rendering previous models unreliable. Incremental drift is characterized by a steady progression through multiple concepts, with each shift resulting in a new concept closer to the target distribution, and models adapt incrementally to the drift. Gradual drift involves incoming data alternating between two concepts, with a growing bias toward the new distribution over time, and models can adapt gradually. Finally, recurring drift refers to the reappearance of a previously seen concept, and models can be saved and restored when this occurs. On the other hand, it must be studied if drift is contained within one concept [16]. Thus, the real concept drift refers to a change that makes the previous knowledge about a class' decision boundary invalid, i.e., new knowledge is required to adjust to the shift. In contrast, virtual concept drift is a change that only alters the distribution of data within a known concept, but not the decision boundary. Distinguishing between these two types of drift prevents unnecessary modifications to the classifier. Although concept drift is a general problem in stream learning, the additional complexity of the multi-label space increases the potential for label imbalance and label distribution changes.

As in the batch learning scenario, there are two general approaches to deal with MLC in data streams [17]: (i) transforming multi-label data into problems that can be solved using multi-class classifiers, known as Problem Transformation (PT), or (ii) adapting the algorithms from multi-class context to the multi-label paradigm by changing the decision functions, known as Algorithm Adaptation (AA). Attending to

PT, two main approaches can be followed: either Label Powerset (LP), which transforms every combination in the label set into a single-class value to convert the multi-label problem into a multi-class problem; or Binary Relevance (BR), that passes from d-dimensional label vector $\mathcal{L}$ to d binary classification learners that model each one a label to combine the independent results into a multi-label output. Classifier Chain (CC) is a variation of BR that compose a chain where the predictions of the previous learners are fed as extra features to the subsequent classifiers. PT approaches allow an easy and straightforward solution for MLC, but they also have some known issues [18], including over-training, worsening of class-imbalance problem and worst-case computational complexity in the case of LP; loss of label correlation and increase of computational load in the case of BR; or sensitivity to the label order and error propagation for CC.

AA takes the opposite approach with respect to PT, focusing on creating methods that natively support the multi-label output space without transforming it. There are some examples of algorithm adaptations based on k-Nearest Neighbors (kNN) [6,19,20], on rules [21,22], or on neural networks trained following a mini-batch approach [7,9]. In general, models based on AA imply a significant reduction of model complexity and computational load, in addition to being better adapted to the multi-label task, with respect to the PT paradigm. In any case, models and theoretical results obtained so far in online MLC are very limited, and more effort should be put in this direction [17].

A significant challenge in MLC is the inherent class imbalance present in many real-world applications [23]. This imbalance manifests as a non-uniform distribution of samples and their respective labels over the data space, becoming increasingly complex as the number of labels grows. The challenge is further exacerbated in the streaming context, where the imbalance often occurs simultaneously with concept drift. In this dynamic environment, not only do labels definitions change, but the imbalance ratio itself becomes fluid, with labels roles potentially switching over time [5]. This renders static solutions ineffective, as streams may oscillate between varying degrees of imbalance and periods of balance among labels. Moreover, imbalanced data streams can present additional difficulties such as small sample sizes, borderline and rare instances, class overlapping, and noisy labels. These factors compound the complexity of developing effective classification algorithms for multi-label data streams. In this context, current approaches to handling imbalanced data streams typically fall into two big categories: data-level approaches that resample the dataset to make it balanced, or algorithm-level approaches that design methods to make classifiers robust to skewed distributions [5]. However, focusing on the multi-label data stream field, the research focuses on the second approach. Specifically in this context, ensembles are very popular, with BR, previously discussed, as the most straightforward method, although there are other methods such as GOOWE-ML [14] which utilizes spatial modeling to assign optimal weights to a stacked ensemble. In the case of algorithms based on a single model, they need to incorporate specific mechanisms adapted to their nature.

2.2. Incremental decision trees for multi-label data streams

IDTs are a type of decision tree designed to adapt to changes in the data distribution over time. They are based on adapting their structure as new instances arrive, extending the branches, or deleting them if they are no longer accurate. Thus, they differ from classic decision trees such as ID3, C4.5, or CART, in that they do not rebuild the entire tree from scratch to learn from new data. In multi-class classification based on IDTs, the state of the art is based on applying the Hoeffding bound [3]. The Hoeffding bound offers a mechanism for guaranteeing that the incremental tree building at any time would be equivalent to that built in a batch learning scenario with a confidence level given by the user. Thus, Domingos and Hulten proposed the Hoeffding Tree (HT), also known as the Very Fast Decision Tree, [10], which uses the Hoeffding bound to statistically support the decision of the best possible split with the minimum number of instances seen at any moment. Manapragada et al. propose in [24] the Extremely Fast Decision Tree (EFDT), a faster approach in the splitting process that uses the Hoeffding bound to split a node as soon as the split improves the previous node. Another popular approach is Hoeffding Adaptive Tree (HAT), presented by Bifet and Gavaldà [11], that evolves from HT to incorporate a concept drift detector based on Adaptive Windowing (ADWIN). This mechanism monitors the performance of every split to build a parallel tree when drift is detected, then it uses the Hoeffding bound to determine whether to replace the main branch with the alternate sub-tree.

More recently, other IDTs beyond the Hoeffding approach have arisen. Online Stochastic Gradient Tree (SGT) [25] presented by Gouk et al. as an incremental adaptation of the stochastic gradient descent method for building the decision tree. Mourtada et al. presented in [26] an online Mondrian Tree (MT) that uses the recursive properties of the Mondrian process to split the multidimensional space into regions hierarchically. The online growth of the tree is carried out with an adaptation of the context tree weighting algorithm.

The methods discussed so far have been designed for a multi-class classification scenario with a single label as output. Although they could be deployed in MLC by applying any PT technique as BR. For IDTs natively designed for MLC, the main proposal for years has been MLHT, proposed by Read et al. [12] as an adaptation of HT with multi-label classifiers at leaves: the majority labelset in the base case and a tree transformation method in the variation MLHT of Prune Set (MLHTPS). More recently, Osojnik et al. presented Incremental Structured Output Prediction Tree (iSOUPT) [27], a Hoeffding-based decision tree that transforms the MLC problem into a multi-target regression one and places an adaptive perceptron in the leaves to perform the predictions.

Table 1 summarizes these previous proposals on IDT, together with our MLHAT, considering their main limitations regarding achieving high performance in evolving and imbalanced data streams in a multi-label environment. Thus, we analyze if they natively support MLC or if a problem transformation is needed and if the splitting criteria considers the multi-label specific problems like the co-occurrence between labels. We also analyze if the proposals are adaptable to concept drift, i.e., if the algorithm is able to modify previously built branches. For this characteristic, MLHAT stands out as the only proposal that incorporates a concept drift detector adapted to MLC. Finally, we study if the proposals are sensitive to the greater class imbalance that exists in multi-label versus the traditional multi-class scenario. This characteristic is quantified by considering whether the models incorporate some mechanism dependent on the cardinality of the received instances. In this case, MLHAT is the only proposal adaptable to the imbalance between labels, by monitoring metrics associated with this problem that determine which multi-label classifier to use in each leaf of the tree.

3. Multi-label Hoeffding adaptive tree

This section presents the complete specification of the proposed Multi-Label Hoeffding Adaptive Tree (MLHAT), an IDT that attempts to overcome the limitations of previous decision tree-based methods for multi-label data streams. Previous works on adapting HTs to multi-label data streams have three main drawbacks that have caused a lack of popularity, in contrast to the equivalent in traditional data streams. MLHAT evolves from the classical HAT [11] by adding multiple components to address these limitations as follows:

Firstly, previous IDTs for MLC may not consider the relationship between labels when deciding whether to split a leaf node. These IDTs rely on the entropy function to calculate information gain between the original leaf and potential splits, which assumes that labels are mutually exclusive, as in multi-class scenarios. However, in MLC environments, where labels often co-occur, this method creates additional uncertainty and leads to equalized entropies of the original node and its potential splits, impeding tree growth. The proposed MLHAT algorithm uses a

Table 1
IDTs for MLC on data streams.

Algorithm	Acronym	Ref.	Applicable to multi-label	Split criteria	Adaptable to concept drift	Adaptable to class imbalance
Hoeffding Tree	HT	[10]	Transforming the domain	Information gain assuming classes independence	✕	✕
Extremely Fast Decision Tree	EFDT	[24]	Transforming the domain	Information gain assuming classes independence	✕	✕
Hoeffding Adaptive Tree	HAT	[11]	Transforming the domain	Information gain assuming classes independence	✓	✕
Stochastic Gradient Tree	SGT	[25]	Transforming the domain	Loss function minimization between target and prediction	✕	✕
Mondrian Tree	MT	[26]	Transforming the domain	Loss function minimization between target and prediction	✕	✕
Multi-Label Hoeffding Tree	MLHT	[12]	Natively	Information gain considering co-occurrence between labels	✕	✕
Incremental Structured Output Prediction Tree	iSOUPT	[27]	Natively	Reduction of intra-cluster variance	✕	✕
Multi-Label Hoeffding Adaptive Tree	**MLHAT**	**Our proposal**	**Natively**	**Multi-label information gain based on multi-variate Bernoulli process**	✓	✓

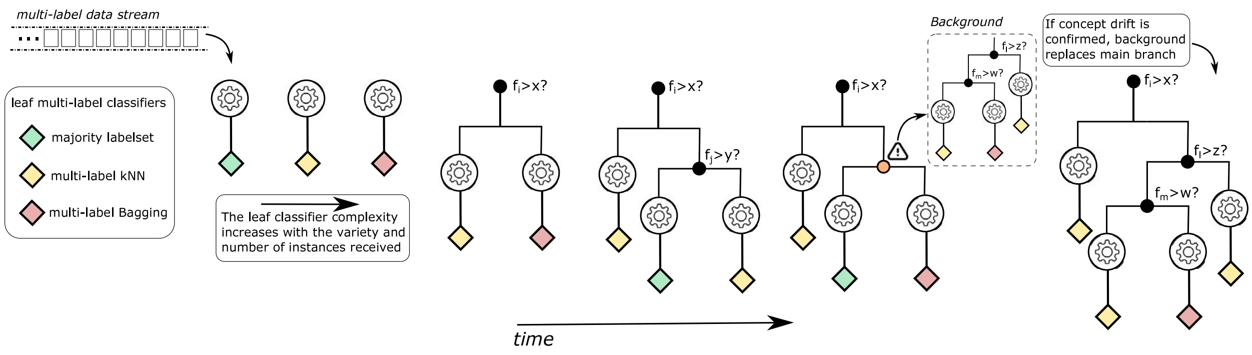

multi-label data stream
leaf multi-label classifiers
majority labelset
multi-label kNN
multi-label Bagging
The leaf classifier complexity increases with the variety and number of instances received
$f_i>x?$
$f_j>y?$
Background
$f_l>z?$
$f_m>w?$
If concept drift is confirmed, background replaces main branch
time

Fig. 1. Flowchart of Multi-Label Hoeffding Adaptive Tree.

multivariate Bernoulli process [28] to calculate the entropy, which considers groups of labels that appear together with higher probability, leading to more accurate approximations of the information gain. This approach also affects the computation of the Hoeffding bound, which determines the significance of the best-split point.

Secondly, previous IDTs for MLC do not account for the imbalance in observed label sets, which may be very severe due to the large number of potential label combinations. This issue affects the final classification at the leaf nodes, as previous models assume either a naive scenario predicting the majority set or a complex multi-label classifier trained on each leaf. The proposed MLHAT deals with imbalanced label sets by incorporating two markers at the leaves, multivariate binary entropy and cardinality, and using four prediction scenarios based on their values and set thresholds. Depending on the difficulty of the scenario, the tree leaf will dynamically alternate between a simpler or a more complex classifier.

Finally, existing proposals for concept drift detection in multi-label IDT [15,29,30] delegate the mechanism to an ensemble approach, leading to not pruning at branch level, which adds unnecessary complexity to the model. In contrast, MLHAT incorporates an ADWIN detector at each node, similar to HAT [11] but with two main differences: it is adapted to monitor multi-label accuracy and it accelerates the reaction to concept drift by triggering an early warning to build a parallel subtree in the background of the current node. The background sub-tree replaces the main one if the concept drift is confirmed.

The pseudocode is presented by Algorithms 1 and 2, which defines the function *leafLearning()* used previously. Fig. 1 shows an example of how the MLHAT tree evolves over time, starting with a single node that splits as more instances from the stream arrive. The general architecture is detailed in Section 3.1. The colored squares represent the different levels of the classifier to be used on the leaves, which will depend on the impurity and cardinality of the instances that reach them (see Section 3.3). When the received instances are sufficient and different enough, the node is split into two sub-branches based on a feature f_i with split point x (see Section 3.4). It may happen that a split point is no longer valid due to concept drift. In this case, it is replaced by another tree built in parallel and updated to the new data distribution (see Section 3.5). In addition to the learning process described above, MLHAT can produce predictions at any time as described in Section 3.6.

3.1. Building the tree

Decision tree algorithms work through recursive partitioning of the training set to obtain subsets that are as pure as possible to a given target class, or set of labels in MLC. Classical decision trees such as ID3, C4.5 and CART, as well as their multi-label adaptation or transformation approximations [31], assume that all training instances are available at training time, building the tree considering all the characteristics simultaneously. However, IDTs learn from data sequences, updating the model as the data distribution evolves and more information is available. This paradigm eliminates the need for retraining the whole model when new data arrives, and for keeping all data in memory. It also allows to perform predictions at any moment, even without having received many instances for training. The general workflow of MLHAT is similar to previous Hoeffding tree works [10–12], but with the particularity of being the first proposal, to the best of our knowledge, of incorporating concept drift adaptation in a native multi-label incremental tree.

Algorithm 1: Multi-Label Hoeffding Adaptive Tree (MLHAT)

```
Input:
$S \leftarrow \{(X_0, Y_0), (X_1, Y_1), \ldots, (X_i, Y_i), \ldots\}$: potentially unbounded multi-label data stream
$\delta_{alt}, \delta_{spl}$: significance levels in the Hoeffding bound for managing alternate trees and splitting nodes respectively
$\kappa_{alt}$: number of instances an alternate tree should see to being considered relevant
$\lambda$: Poisson distribution parameter
Symbols:
$DT(n)$: sub-tree growing from a node $n$
$path$: succession of nodes followed by $X_i$ from an starting point $DT(n_j)$ until a leaf it reached
$\alpha$: multi-label ADWIN concept-drift detector based for each $n$
$W$: instances seen by each $n$, weighted according $Poisson(\lambda)$

MLHAT ← new leaf    ▷ Initialize the tree with a single leaf with empty statistics and fresh classifiers
for $S_i = (X_i, Y_i) \in S$ do
    $w \leftarrow Poisson(\lambda)$ weight for learning
    $path \leftarrow$ traverse $MLHAT$ on $X_i$
    $Z_i \leftarrow MLHAT$ prediction on $X_i$
    for $n \in path$ do
        $\alpha \leftarrow$ ADWIN update based on $Y_i$ and $Z_i$    ▷ Error monitoring and drift update
        if $\alpha$ detects warning and $\nexists DT'(n)$ then    ▷ Start growing an alternate tree in the node's background
            $DT'(n) \leftarrow$ new sub-tree in background
        end if
        if $\exists$ $DT'(n)$ and $W(DT'(n)) > \kappa_{alt}$ then
            $e \leftarrow \alpha$ monitored error in $DT(n)$
            $e' \leftarrow \alpha'$ monitored error in $DT'(n)$
            $\epsilon_{alt} \leftarrow$ Hoeffding bound associated to $e$ (Eq. (16))
            if $(e - e') > \epsilon_{alt}$ then    ▷ Alternate tree is better than main one with confidence $1 - \delta_{alt}$
                $DT(n) \leftarrow DT'(n)$
                $DT'(n) \leftarrow \emptyset$
            end if
            if $(e' - e) > \epsilon_{alt}$ then    ▷ Alternate tree has significantly worsened the main one
                $n \leftarrow$ prune $DT'(n)$
            end if
        end if
        $s \leftarrow$ update statistics of $n$ based on $Y_i$ weighted by $w$
        if $n$ is leaf then    ▷ Incremental learning on leaf classifiers and attempt to split in more branches
            leafLearning($n, S_i, w, \delta_{spl}$)
        end if
        if $\exists$ $DT'(n)$ then    ▷ Alternate tree keeps growing in the background
            $path' \leftarrow$ traverse $DT'(n)$
            for $n \in path'$ do
                $\alpha' \leftarrow$ ADWIN update based on $Y_i$ and $Z_i$
                $s \leftarrow$ update statistics of $n$ based on $Y_i$ weighted by $w$
                if $n$ is leaf then
                    leafLearning($n, S_i, w, \delta_{spl}$)
                end if
            end for
        end if
    end for
end for
```

MLHAT is built on two fundamental components in leaves: (i) the Hoeffding bound that determines when a node should split, and (ii) node statistics about the instances that reach it. Initially, MLHAT starts with a single node, the root, which will be a leaf node receiving the income instances and updating its statistics about label distributions (details in Section 3.2), as well as multi-label classifiers for imbalanced learning (details in Section 3.3). For every κ_{spl} instances received, the model tries to split the node using the feature that minimizes the node entropy. For that purpose, a Hoeffding bound η_{spl} determines with high probability if the estimated entropy minimization is significant enough given the number of instances observed. How to determine possible node split points and associated Hoeffding bound in the multi-label scenario are described in detail in Section 3.4.

Once the Hoeffding bound is passed, the root node is divided into two children nodes at the selected split point in the feature space, having now a branch node with two leaf nodes. Each of these children will generate statistics equivalent to those described above from the new instances it receives according to the new partition. Likewise, at any time a node may encounter a feature whose split point exceeds the Hoeffding bound, which will cause a new splitting of that node, causing the tree to grow to a new level of depth. In this way, the tree would expand incrementally as long as the entropy of the leaf nodes can be minimized with new splits. MLHAT implements a mechanism for pruning branches that are no longer needed due to concept drift. Thus, each node incorporates a concept drift detector α that tracks accuracy during training. If the performance starts to decrease at any node n, a background sub-tree $DT'(n)$ starts to be built from that node. This tree will grow in parallel to the main one, $DT(n)$. For every κ_{alt} instances received, errors of the current and alternate subtrees, e and e', are compared. Again, a Hoeffding bound ϵ_{alt} is used to determine if the difference in performance is significant enough to make changes in the general structure of MLHAT. Section 3.5 describes the complete specification of the concept drift adaptation in MLHAT.

Finally, it should be noted that the entire training process is conditioned by an online bootstrapping following a Poisson(λ) distribution, in order to apply an extra weight in some instances to perform resampling with replacement from the stream. This approach has been used in multiple previous proposals for data stream classification [16,32–35]. This extra weight w affects the node statistics, the count of instances seen W, as well as the multi-label classifiers used in leaves in general. However, if the classifier in question is based on bagging, it will already have its own modification of the weight of the instances following a Poisson(λ) distribution independent of that of the general MLHAT model, since this is the canonical way to simulate bagging in the online paradigm [35]. In this case, applying both modifiers to the instances used to train the classifier would distort the data too much, so only the Poisson(λ) of the classifier is applied to its learning process.

Algorithm 2: Leaf learning in MLHAT

```
Input:
$n$: leaf node belonging to $MLHAT$ to update
$S_i = (X_i, Y_i)$: multi-label instance
$w$: weight of the instance in learning process
$\delta_{spl}$: significance levels in the Hoeffding bound for splitting nodes
$\kappa_{spl}$: number of instances a leaf should observe between split attempts
$\eta$: number of instances a leaf should see to consider high cardinality
Symbols:
$\gamma_{\downarrow C}$: online classifier for low complexity scenario
$\gamma_{\uparrow C}$: online classifier for high complexity scenario
$\mathcal{L}$: labels' space
$\mathcal{F}$: features' space
$\mathcal{N}_f(\mu, \sigma^2)$: Gaussian estimator for conditional probabilities $p(l|x_f)$ $\forall l \in \mathcal{L}$

Function leafLearning($n, S_i, w, \delta_{spl}$):
    $\mathcal{N}_{fl}(\mu, \sigma^2) \leftarrow$ update node stats on $X_i(f)$, $Y_i$, $w$ $\forall f, l \in \mathcal{F}, \mathcal{L}$
    $H_0 \leftarrow$ current entropy in $n$ (Eq. (5))
    if $H_0 > 0$ then
        if $\kappa_{spl}$ instances received since last split try then                ▷ Attempt to split
            for $f \in \mathcal{F}$ do
                $H(f, s) \leftarrow$ entropy at the $s$ that minimizes post-split entropy (Eq. (11)) among possible splits given by $\mathcal{N}_f(\mu, \sigma^2)$
            end for
            $\epsilon_{spl} \leftarrow$ Hoeffding bound associated to $L$ (Eq. (7))
            $G_{f1} \leftarrow$ information gain of the best split candidate $f_i$ at $s_k$ (Eq. (8))
            $G_{f2} \leftarrow$ information gain of the second best $f_j$ at $s_l$
            if $(G_{f1} - G_{f2}) > \epsilon_{spl}$ then                ▷ There is a split outperforming the rest ones with confidence $1 - \delta_{spl}$
                $n \leftarrow$ replace leaf by a split at $(f_1, s_k)$
                $DT(n) \leftarrow$ new branch for $x_f \leq s_k$
                $DT(n) \leftarrow$ new branch for $x_f > s_k$
            end if
        end if
        if $n$ is $leaf$ then                ▷ No split, training of leaf classifiers continues
            if $W(n) < \eta$ then                ▷ Node cardinality is low
                $\gamma_{\downarrow C} \leftarrow$ incremental learning on $\gamma_{\downarrow C}(S_i, w)$
            end if
            $\gamma_{\uparrow C} \leftarrow$ incremental learning on $\gamma_{\uparrow C}(S_i, w)$                ▷ $\gamma_{\uparrow C}$ starts learning before $n$ reaches high cardinality
        else                ▷ Learning on the new leaves grown from $n$
            $leaf \leftarrow$ traverse $DT(n)$
            leafLearning($n, S_i, w, \delta_{spl}$)
        end if
    end if
end function
```

3.2. Modeling label co-occurrences with the multivariate Bernoulli process

In several steps of the MLHAT building process, as in any decision tree, the entropy of the labels observed at each node plays an important role. To obtain a real measure of entropy in MLC, the co-occurrence of labels must be taken into account. In this paper, we use a multivariate Bernoulli distribution [28] to model this behavior, whose purpose is modeling multiple binary random variables simultaneously. Each binary random variable can take on one of two possible values (0 or 1), which represents the presence or absence of a specific event or condition, just like in the MLC. The Bernoulli distribution has interesting properties analogous to the Gaussian distribution that allow us to extend it to high dimensions and construct the so-called multivariate Bernoulli distribution [28].

In our setup, each label $L \in \mathcal{L}$ is defined as a Bernoulli random variable with binary outcomes chosen from $l \in \{0, 1\}$ and with a probability mass function

$$P(L = l) = p^l (1 - p)^{1-l} \tag{2}$$

Since the possible outcomes are mutually exclusive, $P(L = 1) = 1 - P(L = 0)$, the entropy of L is given by

$$H(L) = -p \log_2(p) - (1 - p) \log_2(1 - p) \tag{3}$$

Considering all the labels in the problem $(L_1, L_2, \ldots, L_n)$, we have a n-dimensional random vector of possible correlated Bernoulli random variables that can take values $(0, 0, \ldots, 0)$, $(1, 0, \ldots, 0)$, $\ldots (1, 1, \ldots, 1)$. Extending from (2), the probability mass function in this case is

$$P(L_1 = l_1, L_2 = l_2, \ldots, L_n = l_n) = p_{0,0,\ldots,0}^{\prod_{i=1}^{n}(1-l_i)} p_{1,0,\ldots,0}^{l_1 \prod_{i=2}^{n}(1-l_i)} p_{0,1,\ldots,0}^{(1-l_1)l_2 \prod_{i=3}^{n}(1-l_i)} \ldots p_{1,1,\ldots,1}^{\prod_{l_i}^{n}} \tag{4}$$

And the total entropy is defined as the sum of the entropies of all n Bernoulli random variables:

$$H(L_1, L_2, \ldots, L_n) = \sum_{i=1}^{n} -p_{l_i} \log_2(p_{l_i}) - (1 - p_{l_i}) \log_2(1 - p_{l_i}) \tag{5}$$

Furthermore, as part of the exponential distribution family [36], the multivariate Bernoulli distribution has other properties applicable to MLC, like equivalence of independence and uncorrelatedness, and the fact that both marginal and conditional distributions of a random vector that follows a multivariate Bernoulli distribution are also multivariate Bernoulli. This implies that the conditional probability of any subset of labels $A = (L_i = l_i, ..L_j = l_j)$ given any subset of the rest of them $B = (L_k = l_k, ..L_m = l_m)$ can be computed applying:

$$P(A|B) = \frac{P(A \cup B)}{P(B)} = \frac{p(l_i, \ldots, l_j, \ldots, l_k, \ldots, l_m)}{p(l_k, \ldots, l_m)} \tag{6}$$

3.3. Dynamic multi-label learning at leaves

MLHAT incorporates multi-label classifiers in the leaves to find the last co-dependencies after the discrimination carried out by the rest of the path in the tree. Due to the complexities of the data flows, and especially in MLC, there is a large imbalance between label sets that also affects the number of instances received in each leaf node. Therefore, the learning and prediction strategy should not be uniform for all leaf nodes. There is a previous MLC proposal [37] that employs different classifiers depending on the data partition carried out by a decision tree. However, this proposal is designed for a batch learning scenario, so the inference process, the classifiers employed, and the criteria are not applicable to MLC in the data streams. MLHAT dynamically adapts leaves components as the label distribution of the data stream evolves. For this purpose, each node monitors two markers: (i) the current multi-label entropy at the node H_0 as in previously discussed Eq. (5), and (ii) the cardinality W of the node, which is the number of instances that have reached it at a given time. These markers provide MLHAT with four possible learning/prediction scenarios for the multi-label classifiers used in leaves, affecting both the main tree and alternate ones:

- $H_0 = 0$: entropy cannot be minimized anymore, i.e., the path to the leaf perfectly separates instances of the same label set. Thus, the model does not need to train additional multi-label classifiers because predicting the majority label set is accurate and computationally efficient.
- $H_0 > 0$ & $W \leq \eta$: the leaf is going through an intermediate state with entropy starting to increase but cardinality under a given threshold η. At this point, the leaf incrementally trains a low-cardinality classifier $\gamma_{\downarrow C}$ good at detecting relationships between labels early in data-poor scenarios. In Section 4.2 we study the effect of several multi-label online classifiers to finally select a LP transformation of kNN because its balance between low complexity and high accuracy with few instances.
- $\Delta G(H_0) \leq \epsilon_{spl}$ & $W > \eta$: a high number of instances but a low information gain ΔG between the entropies currently and after the eventual split, imply that it is difficult to separate the label-sets given their feature space. The information gain computation is discussed in Section 3.4. In this scenario, it is necessary to employ a more data demanding and computationally expensive multi-label classifier, but capable of finding deeper relationships, while keeping computational complexity under control so that MLHAT remains competitive in a stream data scenario. Section 4.2 discusses the effect of various multi-label online classifiers based on ensembles and determines BR transformation of the ensemble ADWIN Bagging (ABA)+Logistic Regression (LR) as the most suitable option due to its superiority in learning from larger data streams.
- $\Delta G(H_0) > \epsilon_{spl}$ & $W > \eta$: if after passing the high cardinality threshold it is also observed that entropy increases and there are significant differences between possible splits, the node has seen enough and sufficiently diverse instances, so it is ready to attempt to split following the procedure described in Section 3.4.

3.4. Multi-label splitting into new branches

The Hoeffding bound mathematically supports the split decisions in MLHAT leaves determining the minimum number of instances needed to decide if the split candidate would be equivalent to the one selected in a batch learning scenario where all instances are available. We define the Hoeffding bound for MLC based on [10] but considering the number of known label sets $|L|$ instead of the single labels:

$$\epsilon_{spl} = \sqrt{\frac{\log_2(|L|)^2 \ln(1/\delta_{spl})}{2W}} \tag{7}$$

where W is the number of instances received and δ_{spl} is an error parameter for measuring the confidence in the decision.

The Hoeffding bound is used to determine if the first best-split candidate is significantly better than the second one. To find these two best candidates, each possible candidate in the features space is evaluated using the information gain $G(f,s)$ between the current entropy H_0 at the node that may be partitioned, and the estimated posterior entropy $H(f,s)$ if the data received in that node so far were partitioned at feature f taking value s:

$$G(f,s) = H_0 - H(f,s) \tag{8}$$

The information gain difference ΔG between the attribute f_i with the best split at the value s_k, and the second best split given at the attribute f_j with value s_l, with $i \neq j$, is computed to pass the Hoeffding test:

$$\Delta G = (G(f_i, s_k) - G(f_j, s_l)) > \epsilon_{spl} \tag{9}$$

If this test is satisfied, the best split point $f_i = s_j$, found after observing W instances in the leaf, would be the same as the one that would be selected in a batch learning scenario, with a confidence of $1 - \delta_{spl}$.

To calculate the different entropies implied in the process considering label co-occurrences, we use the multivariate Bernoulli distribution as discussed in Section 3.2. Thus, H_0 is directly obtained from Eq. (5), where label priors $p(l)$ are obtained from the counters maintained by the node:

$$p(l) = \frac{W_l}{W} \;\forall\; l \in \mathcal{L} \tag{10}$$

The entropy after the candidate binary split $H(f,s)$ is also estimated as a succession of binary entropies, aggregating the entropies from the generated branches given by whether the splitting criterion is met, $(x_f \in s)$ or not $(x_f \notin s)$:

$$H(f,s) = p(s)H(x_f \in s) + p(\neg s)H(x_f \notin s) \tag{11}$$

where $p(s)$ and $p(\neg s)$ are the probabilities of occurrence of the splitting criterion, to balance the importance of each partition, and $H(f \in s)$ and $H(f \notin s)$ measure the entropies of each branch by conditioning the labels probabilities to the splitting criterion:

$$\begin{aligned} H(x_f \in s) &= \sum_{l \in \mathcal{L}} -p(l|s)\log_2(p(l|s)) - p(\neg l|s)\log_2(p(\neg l|s)) \\ H(x_f \notin s) &= \sum_{l \in \mathcal{L}} -p(l|\neg s)\log_2(p(l|\neg s)) - p(\neg l|\neg s)\log_2(p(\neg l|\neg s)) \end{aligned} \tag{12}$$

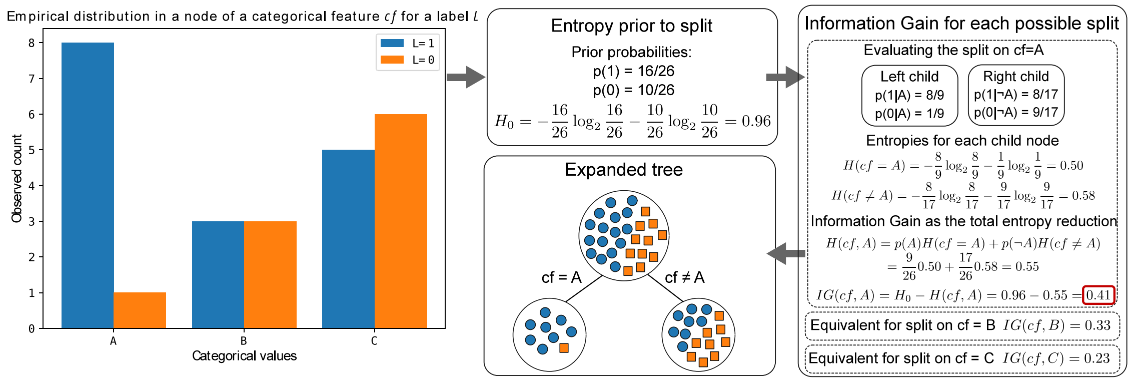
The computation of the conditional probabilities of equation depends on the nature of the feature f. Our MLHAT natively handles categorical and numerical features. In categorical cases, the given feature has already well-defined partitions $f \in \{A, B, \ldots, X\}$. Since both the feature and the label take discrete and mutually exclusive values, conditional probabilities are obtained by counting the occurrences of the given label $W_{l,s}$ among all the observed instances in the node that meet the criterion W_s:

$$\begin{aligned} p(l|x_f = s) &= \frac{W_{l,s}}{W_s}, p(\neg l|x_f = s) = 1 - p(l|x_f = s) \\ p(l|x_f \neq s) &= \frac{W_{l,\neg s}}{W_{\neg s}}, p(\neg l|x_f \neq s) = 1 - p(l|x_f \neq s) \end{aligned} \tag{13}$$

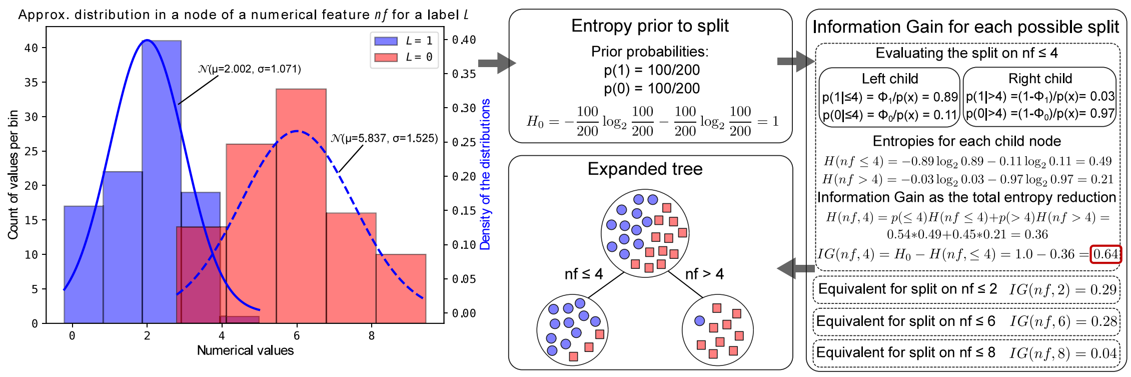
On the other hand, for numerical features that do not have discrete partition points but move in a certain range, $f \in [a,b]$, $a, b \in \mathbb{R}$, we use a common approach in decision trees [38] consisting of modeling the feature space with Gaussian estimators $\mathcal{N}(\mu, \sigma^2)$. This provides an efficient way to calculate conditional probabilities given a splitting point $s \in [a,b]$ to create the branches such that:

$$\begin{aligned} p(l|x_f \leq s) &= \frac{p(x_f \leq s|l)p(l)}{p(x_f \leq s)}, \\ p(\neg l|x_f \leq s) &= \frac{p(x_f \leq s|\neg l)p(\neg l)}{p(x_f \leq s)} \\ p(l|x_f > s) &= \frac{(1 - p(x_f \leq s|l))p(l)}{p(x_f > s)}, \\ p(\neg l|x_f > s) &= \frac{(1 - p(x_f \leq s|\neg l))p(\neg l)}{p(x_f > s)} \end{aligned} \tag{14}$$

where $p(x_f \leq s)$ and $p(x_f > s)$ are normalizing constants for all labels, and $p(l)$ and $p(\neg l)$ are defined in (10). Finally, $p(x_f \leq s|l)$ and $p(x_f \leq s|\neg l)$ are obtained from the cumulative density functions, $\Phi_{f,l}(f \leq s)$ and $\Phi_{f,\neg l}(f \leq s)$, associated with the respective Gaussian estimators maintained in the node for each pair of feature $f \in \mathcal{F}$ and target in


Empirical distribution in a node of a categorical feature cf for a label l
L= 1
L= 0
Observed count
Categorical values
A
B
C
Entropy prior to split
Prior probabilities:
p(1) = 16/26
p(0) = 10/26
$H_0 = -\frac{16}{26}\log_2\frac{16}{26} - \frac{10}{26}\log_2\frac{10}{26} = 0.96$
Expanded tree
cf = A
cf ≠ A
Information Gain for each possible split
Evaluating the split on cf=A
Left child
p(1|A) = 8/9
p(0|A) = 1/9
Right child
p(1|¬A) = 8/17
p(0|¬A) = 9/17
Entropies for each child node
$H(cf = A) = -\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} = 0.50$
$H(cf \neq A) = -\frac{8}{17}\log_2\frac{8}{17} - \frac{9}{17}\log_2\frac{9}{17} = 0.58$
Information Gain as the total entropy reduction
$H(cf, A) = p(A)H(cf = A) + p(\neg A)H(cf \neq A)$
$= \frac{9}{26}0.50 + \frac{17}{26}0.58 = 0.55$
$IG(cf, A) = H_0 - H(cf, A) = 0.96 - 0.55 = \boxed{0.41}$
Equivalent for split on cf = B $IG(cf, B) = 0.33$
Equivalent for split on cf = C $IG(cf, C) = 0.23$

(a) Evaluating a categorical feature


Approx. distribution in a node of a numerical feature nf for a label l
L = 1
L = 0
Count of values per bin
Density of the distributions
Numerical values
𝒩(μ=2.002, σ=1.071)
𝒩(μ=5.837, σ=1.525)
Entropy prior to split
Prior probabilities:
p(1) = 100/200
p(0) = 100/200
$H_0 = -\frac{100}{200}\log_2\frac{100}{200} - \frac{100}{200}\log_2\frac{100}{200} = 1$
Expanded tree
nf ≤ 4
nf > 4
Information Gain for each possible split
Evaluating the split on nf ≤ 4
Left child
p(1|≤4) = Φ₁/p(x) = 0.89
p(0|≤4) = Φ₀/p(x) = 0.11
Right child
p(1|>4) =(1-Φ₁)/p(x)= 0.03
p(0|>4) =(1-Φ₀)/p(x)= 0.97
Entropies for each child node
$H(nf \leq 4) = -0.89\log_2 0.89 - 0.11\log_2 0.11 = 0.49$
$H(nf > 4) = -0.03\log_2 0.03 - 0.97\log_2 0.97 = 0.21$
Information Gain as the total entropy reduction
$H(nf, 4) = p(\leq 4)H(nf \leq 4) + p(> 4)H(nf > 4) =$
$0.54*0.49+0.45*0.21 = 0.36$
$IG(nf, 4) = H_0 - H(nf, \leq 4) = 1.0 - 0.36 = \boxed{0.64}$
Equivalent for split on nf ≤ 2 $IG(nf, 2) = 0.29$
Equivalent for split on nf ≤ 6 $IG(nf, 6) = 0.28$
Equivalent for split on nf ≤ 8 $IG(nf, 8) = 0.04$

(b) Evaluating a numerical feature

Fig. 2. Example of information gain computation in MLHAT in the categorical and the numerical cases.

$l \in \mathcal{L}$. Thus, the inverse conditional probability in each case is obtained as:

$$p(x_f \leq s|l) = \Phi_{f,l}(f \leq s) = 0.5\frac{1 + \operatorname{erf}(s-\mu)}{\sigma\sqrt{2}} \tag{15}$$

For the categorical features, the splitting values to evaluate are given by the support of the given feature, while for the numerical features, a binning process from the observed range of the feature is applied to obtain possible splitting points s. Fig. 2 shows a very simple example for each case, obtaining the information gain of the possible splits given a problem with only one feature and one label. In the first case, the node has so far received 26 instances, which are distributed in 16 positive and 10 negative for the only label considered, which gives it an initial entropy of $H_0 = 0.96$. For these instances, the information gain of a hypothetical categorical variable that takes three possible values is evaluated. After applying the process described above, it is obtained that splitting on $s = A$ provides the highest discriminant power so that split value would be the candidate to apply the Hoeffding bound and, if it passes it, expand the node as shown in the figure to create two new branches. In the second case of Fig. 2, a node has received 200 instances equally distributed between positive and negative for the considered label, so the initial entropy is maximum. These instances have a numerical variable that has been observed to move between in the range $[0, 9]$ approximately. The figure shows the observed distribution of the variable for each target and the associated Gaussian estimate. In the example, four possible split values are considered, obtained from applying the binning process in the observed range. It is determined that $s = 4$ is the one that maximizes the information gain. If this candidate exceeded the Hoeffding limit, it would be used to expand the node into two branches, as shown in the example.

3.5. Concept drift adaptation

ADWIN [30] is an adaptive algorithm that detects changes in data distribution on a set number of instances. It works by comparing the statistical properties of two sub-portions of a window and determining if there is a significant difference in the mean values. Many methods in the field of stream learning implement ADWIN [11,16,29]. MLHAT utilizes ADWIN by incorporating a detector α at each node of the decision tree and potential background sub-trees. This allows ADWIN to detect changes in the data distribution that could only impact specific features determined by different paths in the tree. Specifically, α monitors the error e in the Hamming loss between the baseline Y_i and the prediction Z_i of the training instances. Upon detection of a warning drift, a background sub-tree starts growing from the node affected, with the same components and expanding upon the new data distribution. More discussion of the effect of the monitored metric on the final performance is presented in Section 4.2.

The Hoeffding bound is used to determine the moment when the difference between errors in main e and alternate e' sub-trees are significant enough, with a confidence of $1 - \delta_{alt}$, to perform structural changes. The Hoeffding bound ϵ_{alt} for alternating trees is based on [11] and defined from monitored errors and instances seen by both main W and alternate W' sub-trees as:

$$\epsilon_{alt} = \sqrt{\frac{2e(1-e')(W + W')\ln(2/\delta_{alt})}{W \cdot W'}} \tag{16}$$

Given this threshold, we have three possible scenarios:

- $(e - e') > \delta_{alt}$: the alternate tree is better than the main one so it will be replaced, pruning the previous structure at the node affected by the concept drift and resetting the drift detector in the alternate tree for future concept drifts.

Algorithm 3: Classification in MLHAT

```
Input:
$X_i$: an unknown multi-label instance in the $S$ domain
$\eta$: number of instances a leaf should see to consider high cardinality
$\kappa_{alt}$: number of instances an alternate tree should see to being considered relevant
Symbols:
$MLHAT$ : Initialized model trained with at least one instance
$DT(n)$: subtree growing from a node $n$
$e$: monitored error by $\alpha$ at a node
$W$: instances seen by each $n$, weighted according $Poisson(\lambda)$
$\gamma_{\downarrow C}$: online classifier for low complexity scenario
$\gamma_{\uparrow C}$: online classifier for high complexity scenario

1  $path \leftarrow$ traverse $MLHAT$ on $X_i$
2  $leaf \leftarrow$ leaf node reached by $X_i$ through $path$
3  if node entropy is 0 then          ▷ All the instances discriminated by $path$ belong to same labelset
4      $Z_i \leftarrow$ final prediction as the majority labelset
5  else
6      if $W(leaf) < \eta$ then
7          $p(Z_i) \leftarrow \gamma_{\downarrow C}$ probabilistic multi-label prediction on $X_i$ pondered by $e$
8      else
9          $p(Z_i) \leftarrow \gamma_{\uparrow C}$ probabilistic multi-label prediction on $X_i$ pondered by $e$
10     end if
11     for $n \in path$ do          ▷ $Z_i$ is complemented with the ones provided by significant alternate trees
12         if $\exists\ DT'(n)$ then
13             $leaf' \leftarrow$ leaf node reached traversing $DT'(n)$ on $X_i$
14             if $W(path) > \kappa_{alt}$ then
15                 if $W(leaf') < \eta$ then
16                     $p(Z_i) + \leftarrow \gamma_{\downarrow C}$ probabilistic multi-label prediction on $X_i$ pondered by $e$
17                 else
18                     $p(Z_i) + \leftarrow \gamma_{\uparrow C}$ probabilistic multi-label prediction on $X_i$ pondered by $e$
19                 end if
20             end if
21         end if
22     end for
23     $Z_i \leftarrow$ final prediction after accumulated probabilities normalization
24 end if
```

- $(e' - e) > \delta_{alt}$: the alternate tree is not improving the main one because the concept drift was reversed. In this case, the main tree remains without changes and the alternate tree is pruned.
- $\delta_{alt} \geq |e - e'|$: if no significant differences are found, the two sub-trees continue to receive instances and expand to eventually differentiate their performance.

3.6. Classification at leaves

In the stream learning paradigm, the instances to be classified can arrive at any time, without there being a differentiated phase between building the model and generating the predictions. In MLHAT, the classification process is detailed in Algorithm 3 and consists of two phases: (i) traversing the tree with the incoming instance using the tree branches that partition the feature space and (ii) assigning a label set to the instance depending on the leaf state reached. Three scenarios are considered in the leaf node, related to the entropy and cardinality thresholds used in learning (see Section 3.3):

- $H_0 = 0$: minimal entropy implies that all the instances seen until the moment by the reached node belong to the same label set, so the target instance is also assigned to that label set.
- $H_0 > 0$ & $W \leq \eta$: the cardinality of the node is still low, so the least data demanding classifier $\gamma_{\downarrow C}$ is used to assign a label set to the target instance.
- $H_0 > 0$ & $W > \eta$: the cardinality of the node is high and, because it is still not partitioned, the decision boundaries in the feature space should be difficult to determine. Therefore, the classification of the target instance is performed with the most computationally complex classifier $\gamma_{\uparrow C}$.

As mentioned, if MLHAT is in the midst of a concept change, it will have a main tree and one or more alternate sub-trees that are candidates to replace the branches whose performance is declining. In these cases, the instance to be classified could simultaneously reach a leaf node of the main tree and one or more leaves of the alternate sub-trees. In the context of additional complexity of multi-label stream classification, one of the main challenges is to accelerate the speed of reaction to concept drift. Along these lines, in Section 3.5 we have proposed the alarm-based drift early adaptation mechanism. And following this line, in this section we propose a system for combining predictions between the main tree and the possible alternate sub-trees that an instance to be predicted can reach. The idea is to maintain a balance between the main structure that has the statistical guarantees of the Hoeffding bound, and to provide a fast response to possible concept drifts. Thus, this weighting will be done under two conditions: (i) only leaves from alternate trees that have received at least κ_{alt} instances (the same threshold as for considering the replacement of an alternate tree) will be taken into account, and (ii) the prediction will be weighted based on the error monitored in each of the leaves involved. The Section 4.2 discusses how this measure affects model performance in a practical way.

4. Experimental study

This section presents the experimental study and comparison with the state-of-the-art works. The experiments are designed to answer the following research questions:

- **RQ1:** Can MLHAT demonstrate competitive performance compared to the state of the art incremental decision tree methods for multi-label data streams?
- **RQ2:** Is it more beneficial to use MLHAT's native multi-label split criterion rather than applying Problem Transformation (PT) to use a non-adapted split criterion?
- **RQ3:** Is MLHAT robust in a multi-label data stream scenario where concept drift may appear?

Table 2
Algorithms included in the comparative studies.

Family	Ref.	Year	Acr	Algorithm	Adaptable to concept-drift	Adaptable to class-imbalance
BR+IDT	[10]	2000	HT	Hoeffding Tree	✕	✓
BR+IDT	[11]	2009	HAT	Hoeffding Adaptive Tree	✓	✓
BR+IDT	[24]	2018	EFDT	Extremely Fast Decision Tree	✕	✓
BR+IDT	[25]	2019	SGT	Stochastic Gradient Tree	✕	✓
BR+IDT	[26]	2021	MT	Mondrian Tree	✕	✓
BR+Bayesian	[32]	2009	NB	Gaussian Naïve Bayes	✕	✓
BR+Distance	[39]	2021	kNN	*k*-Nearest Neighbors	✕	✓
BR+Rules	[40]	2016	AMR	Adaptive Model Rules	✓	✓
BR+Ensb	[32]	2009	ABA	ADWIN Bagging + Logistic Regression	✓	✓
BR+Ensb	[32]	2009	ABO	ADWIN Boosting + Logistic Regression	✓	✓
BR+Forest	[29]	2017	ARF	Adaptive Random Forest	✓	✓
BR+Forest	[26]	2021	AMF	Aggregated Mondrian Forest	✕	✓
AA+IDT	[12]	2012	MLHT	Multi-Label Hoeffding Tree	✕	✕
AA+IDT	[12]	2012	MLHTPS	MLHT of Prune Set	✕	✕
AA+IDT	[27]	2017	iSOUPT	Incremental Structured Output Prediction Tree	✕	✕
AA+NN	[7]	2024	MLBELS	Multi-Label Broad Ensemble Learning System	✕	✕
AA+Ensb+DT	[14]	2018	GOCC	Online Stacked Ensemble of Classifier Chain of HT	✕	✓
AA+Ensb+DT	[14]	2018	GORT	Online Stacked Ensemble of iSOUPT	✕	✕
AA+IDT	This	2024	MLHAT	Multi-Label Hoeffding Adaptive Tree	✓	✓

4.1. Experimental setup

4.1.1. Algorithms

Table 2 presents a taxonomy of the 16 multi-label incremental algorithms used in this experimental study, including Binary Relevance (BR) to transform the problem and Algorithm Adaptation (AA) methods, as well as whether they are adaptable to concept drift and adaptable to class-imbalance (yes (✓), no (✕)). As discussed in Section 2, we have considered whether algorithms are adaptive to class imbalance if they include explicit elements such as an ensemble-based construction or our proposal with dynamic leaf classifiers. The same for concept drift, they are considered adaptive if they incorporate explicit elements for this purpose. All algorithms are implemented in Python and are publicly available in the River library [39]. The source code of MLHAT is also publicly available in the repository associated with this work[2] for the sake of reproducibility. In model selection, approaches based on IDT have special prevalence. Thus, we consider 5 multi-class IDTs transformed to MLC using BR; 3 IDTs for MLC, and 2 forest-based methods, i.e., ensembles designed for specific trees as base models. Moreover, the experimentation includes other algorithms from well-known paradigms like neighbors, rules, Bayes, and bagging and boosting, as well as the recent MLBELS, based on neural network and trained in mini-batches [7]. In this last case, in order to make the experimentation conditions as similar as possible to the other proposals, which work purely online, the smallest batch size studied by the authors, 50 instances, is used. In general, the hyperparameter setting has followed three principles: (i) not to make individualized adjustments per dataset, (ii) to start from the configurations suggested by the authors or mostly used in the literature, and (iii) to keep equivalent configurations in models of the same family. Thus, all Hoeffding trees, including our MLHAT, have been configured with Hoeffding significance $\delta_{spl} = 1e{-}5$ and a grace period of $\kappa_{spl} = 200$, and all ensembles have been configured with 10 base models. The complete parameter specification per algorithm can be found in the associated repository.

4.1.2. Datasets

Multi-label datasets used in the experimental study cover a wide range of properties. On the one hand, we evaluate the performance of the algorithms included in the study on 29 real datasets publicly available[3] of up to 269,648 instances, 31,802 features, and 374 labels. These datasets are used to address RQ1 and RQ2. The complete specification of each dataset is shown in Table 3 attending to the number of instances, features, and labels, as well as other statistics like cardinality (average amount of labels per instance), density (cardinality divided by the number of labels), and the mean imbalance ratio per label (average degree of imbalance between labels). Moreover, the last column indicates whether the instances in the dataset are presented in a temporal order: yes (✓), no (✕), or this information cannot be known from the provided description (–). Since in these datasets there is no explicit information about concept drift, this factor can give an idea about the chances of finding concept drift in them.

To the authors' knowledge, there are no public multi-label datasets that expressly contain information about concept-drift [12,13,41].9 Therefore, to address RQ3, 12 additional datasets are synthetically generated to study the impact of various types of concept drift. These datasets have been generated with the MOA framework [42] and are public in the repository associated to this paper. We employ three multi-label generators based on Random Tree, Radial Basis Function (RBF) and Hyper-plane, under two configurations widely used [12]: 30 attributes and 8 labels, and 80 attributes and 25 labels. The drifts are controlled by two parameters: the label cardinality Z and the label dependency u, that are altered in different ways to generate the four main types of concept drift in MLC: sudden, gradual, incremental and recurrent. In all cases, the drifts take place at 3 times depending on the size N of the stream: at times $N/4$, $2N/4$ and $3N/4$, and each drift changes the underlying concept of the stream. Thus, the controlled parameters in the first stretch of the stream are $Z = 1.5$, $u = 0.25$; in the second only the label dependency changes, so we have $Z = 1.5$, $u = 0.15$; in the third stretch only the cardinality changes to have $Z = 3.0$, $u = 0.15$; and finally, in the fourth, they both change to end up with $Z = 1.5$, $u = 0.25$ again. In the recurrent case, only the configurations of the 1st and 3rd stretches are alternated twice. Table 4 shows the main characteristics of these synthetic datasets regarding features, labels, type of concept drift and its width, i.e., for how many instances the drift change extends. For more details, refer to the paper repository.

4.1.3. Evaluation metrics

Due to the incremental nature of the data streams, the canonical way to evaluate performance working with them follows a test-then-train scheme known as prequential evaluation [43]. In this work, we apply this methodology to all experiments setting a forgetting factor of $\alpha = 0.995$ and 50 steps between evaluations.

For comparing the performance between algorithms, we employ 11 different metrics that evaluate the total and partial correctness of multi-label prediction [31], as well as the total computing time in seconds. The metrics are defined as follows on the labels L and the instances n.

[2] https://github.com/aestebant/mlhat.
[3] https://www.uco.es/kdis/mllresources/.

Table 3
Real-world datasets used in the experimental study.

Dataset	Abbreviation	Instan.	Features	Labels	Card.	Dens.	MeanIR	Temporal order
Flags	Flags	194	19	7	3.39	0.48	2.255	×
WaterQuality	WQ	1,060	16	14	5.07	0.36	1.767	–
Emotions	Emo	593	72	6	1.87	0.31	1.478	×
VirusGO	Virus	207	749	6	1.22	0.20	4.041	×
Birds	Birds	645	260	19	1.01	0.05	5.407	–
Yeast	Yeast	2,417	103	14	4.24	0.30	7.197	×
Scene	Scene	2,407	294	6	1.07	0.18	1.254	×
GnegativePseAAC	Gneg	1,392	440	8	1.05	0.13	18.448	–
CAL500	CAL500	502	68	174	26.04	0.15	20.578	×
HumanPseAAC	Human	3,106	440	14	1.19	0.08	15.289	–
Yelp	Yelp	10,806	671	5	1.64	0.33	2.876	✓
Medical	Med	978	1,449	45	1.25	0.03	89.501	✓
EukaryotePseAAC	Eukar	7,766	440	22	1.15	0.05	45.012	–
Slashdot	Slashdot	3,782	1,079	22	1.18	0.05	19.462	✓
Hypercube	HC	100,000	100	10	1.00	0.10	–	–
Hypersphere	HS	100,000	100	10	2.31	0.23	–	–
Langlog	Langlog	1,460	1,004	75	15.94	0.21	39.267	✓
StackexChess	Stackex	1,675	585	227	2.41	0.01	85.790	✓
ReutersK500	Reuters	6,000	500	103	1.462	0.01	54.081	–
Tmc2007500	Tmc	28,596	500	22	2.22	0.10	17.134	✓
Ohsumed	Ohsum	13,929	1,002	23	0.81	0.04	7.869	✓
D20ng	D20ng	19,300	1,006	20	1.42	0.07	1.007	✓
Mediamill	Media	43,907	120	101	4.38	0.04	256.405	–
Corel5k	Corel5k	5,000	499	374	3.52	0.01	189.568	×
Corel16k001	Corel16k	13,766	500	153	2.86	0.02	34.155	×
Bibtex	Bibtex	7,395	1,836	159	2.40	0.02	12.498	×
NusWideCVLADplus	NWC	269,648	129	81	1.87	0.02	95.119	–
NusWideBoW	NWB	269,648	501	81	1.87	0.02	95.119	–
Imdb	Imdb	120,919	1,001	28	1.00	0.04	25.124	×
YahooSociety	YahooS	14,512	31,802	27	1.67	0.06	302.068	–
EurlexSM	Eurlex	19,348	5,000	201	2.21	0.01	536.976	✓

Table 4
Synthetic datasets used in the experimental study.

Dataset	Instances	Features	Labels	Generator	Drift type	Drift width
SynTreeSud	50,000	20 num.+10 cat.	8	Random tree	Sudden	1
SynRBFSud	50,000	80 numeric	25	Random RBF	Sudden	1
SynHPSud	50,000	30 numeric	8	Hyper plane	Sudden	1
SynTreeGrad	50,000	20 num.+10 cat.	8	Random tree	Gradual	500
SynRBFGrad	50,000	80 numeric	25	Random RBF	Gradual	500
SynHPGrad	50,000	30 numeric	8	Hyper plane	Gradual	500
SynTreeInc	50,000	20 num.+10 cat.	8	Random tree	Incremental	275
SynRBFInc	50,000	80 numeric	25	Random RBF	Incremental	275
SynHPInc	50,000	30 numeric	8	Hyper plane	Incremental	275
SynTreeRec	50,000	20 num.+10 cat.	8	Random tree	Recurrent	1
SynRBFRec	50,000	80 numeric	25	Random RBF	Recurrent	1
SynHPRec	50,000	30 numeric	8	Hyper plane	Recurrent	1

Example-based metrics evaluate the difference between the actual and predicted labelsets, averaged over n. Given a true labelset $Y_i = \{y_{i1}...y_{iL}\}$ and a predicted one $Z_i = \{z_{i1}...z_{iL}\}$, the example-based metrics considered are:

$$\begin{aligned}
&\text{Subset accuracy} = \frac{1}{n}\sum_{i=0}^{n} 1|Y_i = Z_i\\
&\text{Hamming loss} = \frac{1}{nL}\sum_{i=0}^{n}\sum_{l=0}^{L} 1|y_{il} \neq z_{il}\\
&\text{Example-based precision} = \frac{1}{n}\sum_{i=0}^{n}\frac{|Y_i \cup Z_i|}{|Z_i|}\\
&\text{Example-based recall} = \frac{1}{n}\sum_{i=0}^{n}\frac{|Y_i \cup Z_i|}{|Y_i|}\\
&\text{Example-based F1} = \frac{1}{n}\sum_{i=0}^{n}\frac{2|Y_i \cup Z_i|}{|Y_i| + |Z_i|}
\end{aligned} \tag{17}$$

Label-based metrics measure the performance across the different labels, that can be micro- or macro-averaged depending on if it is used the joint statistics for all labels or the per-label measures are averaged into a single value. Given for each label l its true positives $tp_l = \sum_{i=0}^{n} 1|y_{il} = z_{il} = 1$, true negatives $tn_l = \sum_{i=0}^{n} 1|y_{il} = z_{il} = 0$, false positives $fp_l = \sum_{i=0}^{n} 1|y_{il} = 0, z_{il} = 1$, and false negatives $fn_l = \sum_{i=0}^{n} 1|y_{il} = 1, z_{il} = 0$, the macro-averaged metrics considered are:

$$\begin{aligned}
&\text{Macro-averaged precision} = \frac{1}{L}\sum_{l=0}^{L}\frac{tp_l}{tp_l + fp_l}\\
&\text{Macro-averaged recall} = \frac{1}{L}\sum_{l=0}^{L}\frac{tp_l}{tp_l + fn_l}
\end{aligned} \tag{18}$$

while the considered micro-averaged metrics are defined as:

$$\begin{aligned}
&\text{Micro-averaged precision} = \frac{\sum_{l=0}^{L} tp_l}{\sum_{l=0}^{L} tp_l + \sum_{l=0}^{L} fp_l}\\
&\text{Micro-averaged recall} = \frac{\sum_{l=0}^{L} tp_l}{\sum_{l=0}^{L} tp_l + \sum_{l=0}^{L} fn_l}
\end{aligned} \tag{19}$$

Additionally, Macro-averaged F1 and Micro-averaged F1 are also considered, defined as the harmonic mean of precision and recall in an equivalent way to the example-based F1 defined above.

Table 5
Hyperparameter study for MLHAT.

Parameter	Studied values	Final value
Significance for split δ_{spl}	Fixed by state-of-the-art	1e−7
Leaf grace period κ_{spl}	Fixed by state of the art	200
Significance for tree replacement δ_{alt}	[0.01, 0.10]	0.05
Alternate tree grace period. κ_{alt}	[100, 500]	200
Cardinality threshold η	[100, 1000]	750
Poisson parameter λ	[1, 6]	1
Drift metric e	Subset Acc., Hamming loss, Micro-F1, Macro-F1	Hamming loss
$\gamma_{\downarrow C}$ base	HT, kNN, NB, LR	kNN
$\gamma_{\downarrow C}$ transformation	BR, LP, CC	LP
$\gamma_{\uparrow C}$ base	NB, LR	LR
$\gamma_{\uparrow C}$ ensemble	BA [35], BO [35], BOLE [33], SRP [46]	BA
$\gamma_{\uparrow C}$ transformation	BR, LP, CC	BR

4.2. Analysis of components in MLHAT

As discussed in the model description in Section 3, MLHAT is highly configurable through a wide variety of hyperparameters ranging from the entropy and cardinality thresholds to the classifiers to be used in the leaves and the metrics to be used to monitor the concept drift. The configuration of these hyperparameters is non-trivial, as it encompasses a multitude of possible combinations that evolve categorical, integer, decimal, and logarithmic parameters. In response to this challenge, we employ a well-known systematic approach for hyperparameter optimization [44] that combines the Tree-structured Parzen Estimator (TPE) algorithm to generate configurations that maximize the objective function, with the Hyperband pruner that accelerates the search by discarding the less promising combinations. This methodology allows us to efficiently traverse the vast hyperparameter space of MLHAT, as well as to understand the importance of each hyperparameter and its influence on the final performance. The study has been implemented in Python through the Optuna framework [45].

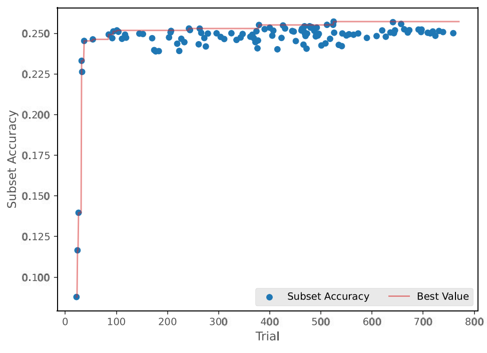
The conditions of the study are primarily determined by the optimization function and the range of each hyperparameter. The optimization function is defined as the Subset accuracy, measured through prequential evaluation (see Section 4.1.3). To make the iterative search feasible, a minimum subset of datasets was selected from the full experimental set. These datasets were chosen based on lower complexity and maximizing differences in terms of cardinality, density, and concept drift. The selected datasets, representing 19% of the total set used in the experiment, are: *Flags*, *Emotions*, *VirusGO*, *Birds*, *Yeast*, *Scene*, *SynHPSud*, and *SynHPGrad*. Details on these datasets can be found in Tables 3 and 4.

The search ranges for each hyperparameter are shown in Table 5. For numerical attributes, typical ranges from Hoeffding tree literature have been used. To monitor concept drift, any MLC metric can be employed. However, it is preferable to use a metric that covers overall performance and is sensitive to class imbalance, as it will operate at each tree node level. In this study, two example metrics and two label metrics are tested. For the leaf nodes, any pair of online multi-label classifiers can be used, provided they meet two criteria. First, they must not incorporate drift detection mechanisms, as this is done at the tree level. Second, they should balance short and long-term performance to cover various scenarios a leaf may encounter, as discussed in Section 3.3. The study examines different classifier complexities. For low-complexity classifiers, it represents four main paradigms in online classification: decision trees, distance-based models, Naïve Bayes, and linear models. For high-complexity classifiers, it compares different ensembles, using low-complexity methods as base models to control overall model complexity. All ensembles use 10 base models, as is standard in online learning paradigms [16]. Both low and high-complexity classifiers explore various transformations from multi-label to single-label space.

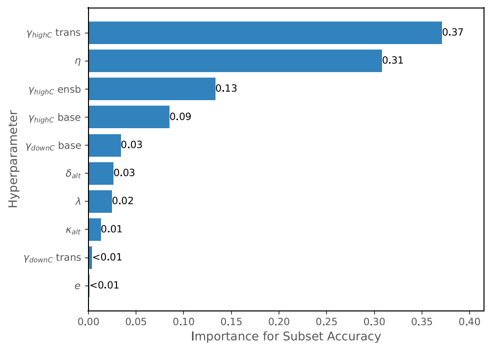
Fig. 3 shows the results of the hyperparameters search, including the optimization history obtained with the framework described above, and the importance of each hyperparameter in the final performance according to the ANOVA functional framework [47], that quantifies the influence of both single hyperparameters and the interactions between them from the marginal predicted performance. Finally, the parallel coordinate plot shows the most promising hyperparameter combination based on the average performance in the subset of datasets selected for this phase. Please note that not all combinations are present because of the guided search method employed. The best hyperparameter configuration found by the optimization method is shown in the last column of Table 5. This configuration will be used in the rest of the experimental study over the whole set of datasets.

The study indicates that the high-complexity classifier is the most relevant parameter for MLHAT performance, specifically the label transformation method, where BR performs much better than LP and CC, probably because it deals more efficiently with label imbalance. In contrast, the transformation used in the low-complexity classifier does not have very important differences in the final result: when working with little data, the multi-label space is small and the three studied methods provide equivalent results. The second most important parameter is the threshold for establishing a high-cardinality scenario. Although the results will vary depending on the pair of classifiers used, it appears that around 600 instances received at a leaf node is the optimal point to move to the most complex classifier. When dealing with the highly complex scenario, it becomes evident that bagging-based ensembles, represented by Oza Bagging (BA) and Streaming Random Patches (SRP), offer more advantages compared to boosting-based ensembles, namely Oza Boosting (BO) and Boosting Online Learning Ensemble (BOLE). This observation arises from the fact that boosting primarily concentrates on enhancing the performance of misclassified instances. However, in the context of MLHAT, where the feature space has already undergone discrimination through the tree structure, this boosting emphasis may result in overfitting. As a base classifier, LR far outperforms Gaussian Naïve Bayes (NB), which, having not completed any runs because they were pruned earlier, does not appear in the graph. In the low complexity case, working with a single model and not an ensemble, it is required a more sophisticated learner to obtain good results. Thus, HT and kNN are the most promising. Finally, attending to the metric for monitoring concept drift, although not as influential as other parameters, there is a clear loser: macro-average is not a good approach, as it does not take into account label imbalance. The rest of the metrics have a similar potential to detect concept drift in general, although probably in studies where only a specific type of concept drift is affected, one metric would respond better than another to accelerate detection. The final configuration for MLHAT, showed in Table 5, follows these lines: for the low complexity leaf classifier a multi-label kNN obtains better performance than the closest alternative of HT. The LP transformation is preferred due to the possibility to capture better the label correlation than BR. The high complexity classifier is conformed as a bagging of LRs, that combines good performance and fast execution in data-intense scenarios. Hamming loss is the metric selected for monitoring the concept drift. This metric that maintains a balance between the proportion of correct and incorrect labels inside the labelset, rather than relying on subset accuracy, which is too strict to be informative. With this configuration, 620 instances are the optimal threshold a leaf should see to pass from low to high cardinality scenarios, and 450 are the minimum instances an alternate tree should see to try to replace the main tree. Finally, the $Poisson(\lambda = 1)$ indicates that is desirable to model the instances' weight as events that occur relatively infrequently but with a consistent average rate.

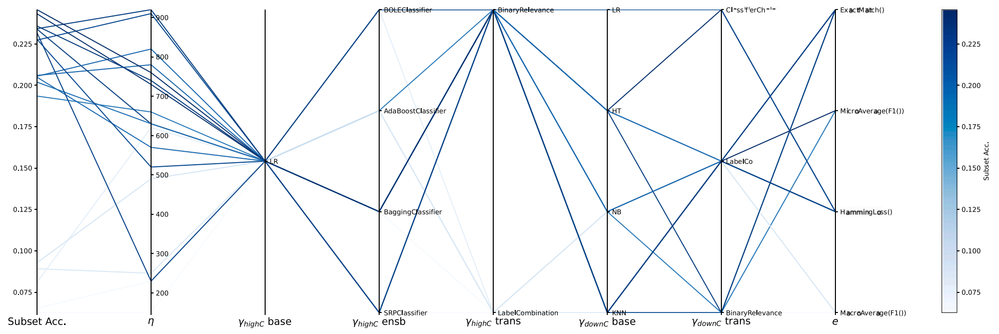
Once the MLHAT hyperparameters have been established, the last study examines the effect of combining the predictions of the alternate



(a) Optimization history



(b) Hyperparameter importance according functional ANOVA



(c) Parallel coordinate plot with the most influential parameters. Some parameters have been omitted for visual clarity as they do not have a sufficiently meaningful range of variability.

Fig. 3. Automatic optimization of MLHAT.

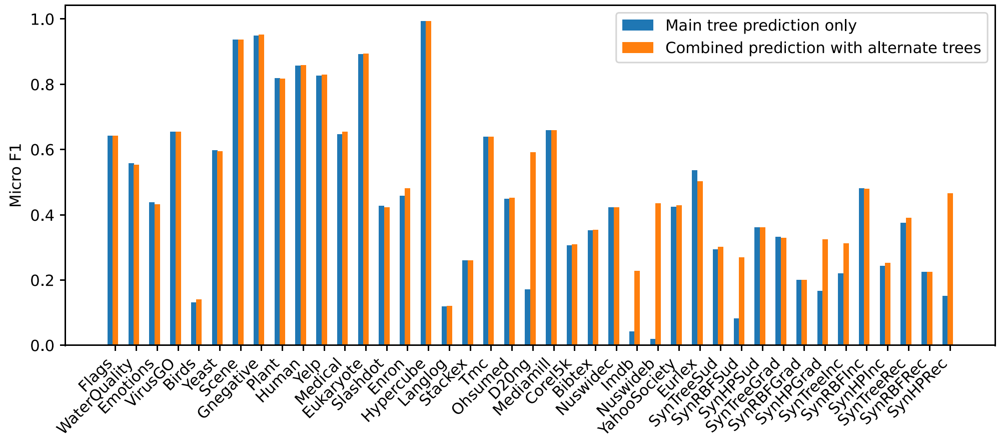


Fig. 4. Effect on the performance of combining predictions from the main and the alternate trees in MLHAT.

trees that are constructed during the occurrence of concept-changes explained in Section 3.6. This study focuses on the micro-average F1 score obtained with the fixed parameters of Table 5 and altering only the way of obtaining the predictions during the prequential evaluation. Fig. 4 shows the differences between results per dataset. Combining the predictions gives an average F1 score of 0.4916, while using the main tree alone gives 0.4475. In other words, the method proposed in Section 3 improves the results on average by 5%. The results per dataset will depend on whether MLHAT detects the concept drift alarm and whether it is finally confirmed or not. In most datasets, this measure has little effect because either the concept drift does not occur or it is quickly confirmed and the alternate structure becomes part of the main tree. In any case, since the grace period κ_{alt} must be exceeded to consider the predictions of the alternate trees, and the predictions are weighted with the branch error in real time, the inclusion of the possible alternate trees is done in a fairly conservative manner that prevents it from having significant adverse effects. Moreover, we see datasets such as *D20ng*, *Nuwideb* or synthetic datasets with explicit concept drift information where combining the predictions has a very positive effect on the average result, with differences of up to 30 times in F1. This is due to the fact that in these cases there are more gradual concept drifts in which it is positive to consider at the same time the main tree and the alternatively emerging branches. In conclusion, this measure is considered positive for the overall performance of MLHAT and will be used in the remainder of the comparative study.

Table 6
Results for example-based F1 on each dataset.

Dataset	MLHAT	KNN	NB	AMR	HT	HAT	EFDT	SGT	MT	ARF	AMF	ABALR	ABOLR	GOCC	GORT	MLHT	MLHTPS	iSOUPT	MLBELS
Flags	0.605	0.619	0.635	0.554	**0.696**	0.677	**0.696**	0.634	0.613	0.694	0.617	0.561	0.557	0.553	0.557	0.615	0.628	0.554	0.450
WQ	**0.533**	0.524	0.525	0.528	0.394	0.406	0.430	0.259	0.330	0.503	0.480	0.524	0.528	0.530	0.517	0.239	0.361	0.522	0.428
Emo	0.446	0.445	**0.601**	0.291	0.544	0.542	0.492	0.318	0.019	0.461	0.338	0.273	0.283	0.294	0.288	0.333	0.477	0.291	0.514
Virus	0.730	0.702	0.095	0.561	0.196	0.439	0.196	0.355	0.428	0.373	**0.841**	0.539	0.561	0.577	0.537	0.337	0.164	0.547	0.520
Birds	**0.482**	0.062	0.000	0.089	0.000	0.000	0.038	0.026	0.023	0.039	0.041	0.072	0.070	0.377	0.380	0.000	0.000	0.089	0.134
Yeast	0.588	0.562	0.533	0.464	0.582	0.575	0.570	0.422	0.454	0.548	0.461	0.463	0.465	0.563	0.503	0.519	0.551	0.476	**0.591**
Scene	**0.948**	0.811	0.611	0.916	0.419	0.533	0.646	0.193	0.538	0.803	0.812	0.913	0.912	0.933	0.933	0.184	0.622	0.876	0.786
Gneg	0.913	0.724	0.682	0.948	0.623	0.632	0.621	0.205	0.678	0.728	0.865	0.947	0.946	**0.953**	**0.953**	0.392	0.628	0.950	0.696
CAL500	**0.351**	0.349	0.348	0.327	0.314	0.305	0.314	0.280	0.305	0.317	0.325	0.330	0.325	0.327	0.322	0.338	0.297	0.327	0.318
Human	0.885	0.588	0.348	0.896	0.358	0.478	0.410	0.095	0.209	0.672	0.467	0.890	0.894	0.904	**0.904**	0.295	0.234	0.862	0.678
Yelp	**0.836**	0.631	0.373	0.705	0.552	0.617	0.650	0.388	0.453	0.758	0.616	0.741	0.743	0.770	0.771	0.438	0.510	0.723	0.666
Med	0.539	0.419	0.002	0.199	0.375	0.395	0.525	0.129	0.196	0.135	0.453	0.130	0.121	0.212	0.166	0.231	0.002	0.217	**0.667**
Eukar	0.912	0.726	0.317	0.903	0.509	0.682	0.617	0.038	0.403	0.786	0.579	0.912	0.911	0.922	**0.922**	0.260	0.207	0.882	0.790
Slashdot	0.361	0.149	0.000	0.018	0.060	0.108	0.159	0.049	0.007	0.085	0.175	0.007	0.006	0.058	0.077	0.150	0.006	0.045	**0.421**
HS	0.855	0.846	0.401	0.839	0.726	0.873	0.820	0.324	0.587	0.867	0.614	0.853	0.853	**0.892**	0.879	0.501	0.562	0.689	–
HC	0.993	0.994	0.966	0.981	0.906	0.963	0.915	0.015	0.994	0.993	**0.995**	0.987	0.987	0.993	0.993	0.786	0.806	0.909	0.987
Langlog	**0.221**	0.047	0.000	0.030	0.000	0.000	0.054	0.013	0.006	0.002	0.006	0.018	0.019	0.198	0.199	0.001	0.029	0.033	0.125
Stackex	0.222	0.098	0.001	0.082	0.009	0.018	0.120	0.030	0.014	0.028	0.033	0.097	0.096	0.094	0.100	0.144	0.000	0.103	**0.309**
Reuters	0.034	0.165	0.000	0.001	0.113	0.129	0.171	0.012	0.047	0.066	0.092	0.000	0.000	0.035	0.027	0.069	0.002	0.000	**0.460**
Tmc	0.628	0.451	0.294	0.590	0.489	0.489	0.425	0.288	0.389	0.482	0.480	0.600	0.590	0.609	0.619	0.215	0.446	0.580	**0.633**
Ohsum	**0.398**	0.042	0.233	0.144	0.286	0.309	0.257	0.142	0.017	0.104	0.004	0.188	0.180	0.283	0.260	0.142	0.294	0.174	0.378
D20ng	**0.600**	0.090	0.000	0.079	0.329	0.307	0.375	0.124	0.067	0.108	0.066	0.208	0.185	0.367	0.327	0.083	0.259	0.162	0.530
Media	**0.633**	0.564	0.170	0.528	0.491	0.498	0.492	0.294	0.450	0.553	0.504	0.532	0.531	0.624	0.623	0.426	0.357	0.500	0.548
Corel5k	0.197	0.111	0.012	0.016	0.059	0.114	0.121	0.012	0.108	0.259	0.163	0.013	0.014	**0.467**	**0.467**	0.043	0.012	0.014	0.204
Corel16k	0.226	0.165	0.010	0.029	0.054	0.132	0.141	0.016	0.059	0.285	0.099	0.012	0.011	0.510	**0.511**	0.059	0.082	0.015	0.225
Bibtex	**0.310**	0.019	0.000	0.159	0.171	0.183	0.190	0.145	0.095	0.120	0.111	0.144	0.145	0.124	0.129	0.067	0.088	0.159	0.289
NWC	**0.404**	0.258	0.219	0.049	0.125	0.172	0.160	–	–	0.162	–	0.041	0.040	0.394	0.403	0.008	0.138	0.050	–
Imdb	0.192	0.141	0.175	0.017	0.029	0.035	0.060	–	–	0.023	–	0.039	0.042	0.198	0.196	0.252	0.006	0.027	**0.347**
NWB	**0.431**	0.247	0.106	0.060	0.085	0.174	0.177	–	–	0.173	–	0.058	0.057	–	0.387	0.006	0.050	0.048	–
YahooS	0.450	0.205	0.367	0.387	0.202	0.257	0.214	–	–	0.216	–	0.443	0.430	0.410	0.403	0.401	0.019	0.413	**0.483**
Eurlex	0.527	0.205	0.091	0.478	0.334	0.348	0.234	–	0.085	0.271	–	**0.588**	0.572	0.535	0.536	0.090	0.018	0.553	–
Average	**0.531**	0.386	0.262	0.383	0.324	0.367	0.364	0.185	0.280	0.375	0.394	0.391	0.390	0.493	0.484	0.246	0.253	0.380	0.488
Ranking	**3.387**	8.196	13.484	9.661	11.097	9.419	8.806	15.823	14.726	9.355	11.306	9.435	9.629	5.806	5.839	13.032	13.984	9.839	7.177

Best results are in bold.
Non finished experiments due to scalability problems marked with –. The physical limitations for all the experiments are set to 200 GB of RAM and 360 h of executions.

4.3. Analysis of MLHAT compared to previous proposals

This section evaluates and compares the overall performance of the 19 online algorithms in the 31 real world datasets for the 11 MLC metrics, plus time of execution, to address RQ1. The analysis is organized as follow. Table 6 analyzes in detail the results broken down by model and dataset, showing the detail of one of the MLC metrics of the study: the example-based F1. We have chosen to show this metric in detail due to its relevance in the MLC field as it adequately summarizes the balance between sensitivity and specificity of the different labels that make up the dataset. Due to space limitations and the large number of methods and datasets included in the experimentation, the details of the rest of the metrics included in the study are available in the repository associated with the work.[4] The last two rows of Table 6 show the summary of the metric under study in two useful ways to get an idea of the overall performance of each method: On the one hand, it is indicated the average that each model has obtained taking into account all the datasets that make up the body of the table. On the other hand, the ranking of each model is shown as the position it obtains with respect to the other methods in the comparison, averaged over all the datasets that make up the body of the table. These two statistics are in turn obtained for the rest of the MLC metrics that make up the study, and are shown in Tables 7 and 8 respectively. Thus, Table 6 serves to make a detailed study of the particularities of each model by type of dataset, while Tables 7 and 8 show the general trends by approach and family of algorithms. In addition, our experiments are supported with statistical tests that validate the significant differences between methods. Specifically, the Friedman test [48] is applied for each metric under study with the rankings in Table 8. Friedman's test shows that there are significant differences with high confidence ($p-$value $\rightarrow 0$) for all metrics. Therefore, the post-hoc Bonferroni–Dunn procedure [48] is applied to a selection of the most representative metrics to find between which groups of algorithms these differences occur. Fig. 5 shows the results at confidence level $\alpha = 99\%$ for subset accuracy, hamming loss, example-based F1, micro-averaged F1, macro-averaged F1, and time.

The results show that MLHAT outperforms on average the other methods in accurate prediction of the labelset of each instance with subset accuracy of 33.23%, as well as the best balance between false positives and false negatives at the instance level (example-based F1 of 53.06%), and at the label level both absolute (macro F1 of 40.64%) and prevalence-weighted (micro F1 of 53.42%). These differences are supported by statistical tests, which indicate that for these metrics MLHAT outperforms the baseline of the Bayesian and rules approaches, NB and AMR, as well as non-Hoeffding IDTs SGT and MT. This indicates that most single-model proposals do not deal well with the additional complexity of MLC transformed with BR in terms of predicting the exact labelset, the average match by instances, nor the average match by labels. In this line, MLBELS and iSOUPT obtain better results than these proposals, showing the potential to adapt specific components of the decision tree to the multi-target paradigm. In the same way, we see that GOCC and GORT, that also incorporate specific MLC mechanisms, are also very competitive. However, these methods obtain worse results than MLHAT with differences per metric. In the case of MLBELS, it is observed that it has problems matching exactly the predicted labelset, although it maintains a good average of hits per label. This makes it significantly worse than MLHAT in subset accuracy, hamming loss, and macro F1, while in Micro F1 and example-based F1, it is at the same level. This may be due to the method used for keeping the co-relation between labels, which is modeled with a neural network. For iSOUPT,

[4] https://github.com/aestebant/mlhat.

Table 7
Average results for each evaluation metric considering all datasets.

Algorithm	Su. Acc	H. Loss	Ex. F1	Ex. Pre	Ex. Rec	Mi. F1	Mi. Pre	Mi. Rec	Ma. F1	Ma. Pre	Ma. Rec	Time (s)
MLHAT	**0.3323**	**0.0745**	**0.5306**	**0.7494**	**0.5397**	**0.5342**	0.6296	**0.4910**	**0.4064**	**0.5899**	**0.3833**	17,355
KNN	0.2575	0.0850	0.3857	0.4238	0.3808	0.4198	0.5405	0.3746	0.2802	0.3953	0.2629	137,827
NB	0.1216	0.1499	0.2618	0.2597	0.3550	0.2725	0.3346	0.3620	0.1925	0.1920	0.2877	10,122
AMR	0.2534	0.0861	0.3828	0.4212	0.3826	0.4076	0.6589	0.3718	0.2911	0.3823	0.2764	39,651
HT	0.1902	0.0891	0.3236	0.3694	0.3205	0.3682	0.5291	0.3142	0.2309	0.3045	0.2190	56,964
HAT	0.2262	0.0842	0.3674	0.4135	0.3650	0.4095	0.5436	0.3558	0.2618	0.3326	0.2480	58,059
EFDT	0.2101	0.0857	0.3642	0.4066	0.3661	0.4095	0.5223	0.3599	0.2689	0.3283	0.2555	49,423
SGT[a]	0.0474	0.2863	0.1848	0.2096	0.3119	0.2059	0.2057	0.3162	0.1510	0.1566	0.3010	66,433
MT[a]	0.1593	0.1010	0.2804	0.3443	0.2652	0.3185	0.5954	0.2601	0.2053	0.3156	0.1925	61,381
ARF	0.2489	0.0750	0.3746	0.4307	0.3596	0.4164	**0.7160**	0.3498	0.2714	0.4287	0.2490	36,633
AMF[a]	0.2458	0.0874	0.3938	0.4584	0.3786	0.4351	0.6870	0.3716	0.3026	0.4280	0.2801	119,653
ABALR	0.2702	0.0826	0.3911	0.4336	0.3856	0.4176	0.6859	0.3734	0.2986	0.3988	0.2774	16,920
ABOLR	0.2669	0.0837	0.3896	0.4305	0.3858	0.4144	0.6817	0.3734	0.2982	0.3969	0.2787	18,540
GOCC[a]	0.2980	0.0853	0.4935	0.6797	0.5103	0.5022	0.5801	0.4689	0.3928	0.5472	0.3687	74,899
GORT	0.2887	0.0844	0.4836	0.6848	0.5018	0.4920	0.5810	0.4565	0.3802	0.5407	0.3560	35,117
MLHT	0.1667	0.1247	0.2459	0.2816	0.2366	0.2476	0.3072	0.2251	0.1155	0.1370	0.1356	**1,934**
MLHTPS	0.1588	0.1012	0.2534	0.3033	0.2465	0.2727	0.4522	0.2438	0.1749	0.2211	0.1772	39,472
iSOUPT	0.2510	0.0880	0.3803	0.4216	0.3779	0.4103	0.6762	0.3677	0.2748	0.3624	0.2596	8,577
MLBELS[a]	0.2779	0.1027	0.4880	0.5595	0.5251	0.4978	0.5234	0.4805	0.2977	0.5280	0.2973	110,060

Best results are in bold.

[a] Results computed without considering all datasets due to scalability limitations. The physical limitations for all the experiments are set to 200GB of RAM and 360 h of executions.

Table 8
Friedman's test and average ranks for each evaluation metric.

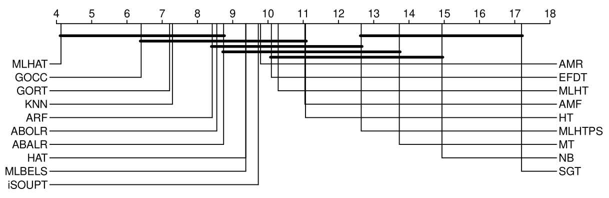
Algorithm	Su.Acc	H.Loss	Ex.F1	Ex.Pre	Ex.Rec	Mi.F1	Mi.Prec	Mi.Rec	Ma.F1	Ma.Prec	Ma.Rec	Time (s)
MLHAT	**4.129**	6.323	**3.387**	**3.548**	**4.290**	**3.774**	7.710	**4.613**	**3.839**	**3.935**	**4.161**	6.419
KNN	7.290	9.355	8.194	9.290	9.032	8.000	10.226	9.065	8.419	7.290	9.129	15.226
NB	14.935	13.468	13.484	15.194	11.710	14.113	15.016	11.387	13.081	14.629	11.677	3.710
AMR	9.790	8.048	9.661	9.855	10.081	9.403	7.242	9.919	9.468	8.823	9.726	8.613
HT	11.065	9.258	11.097	11.129	11.403	10.968	10.645	11.339	11.177	11.855	11.339	7.581
HAT	9.371	9.500	9.419	9.903	9.677	9.065	10.516	9.645	9.194	10.355	9.548	9.387
EFDT	10.097	11.129	8.806	9.226	8.984	8.839	11.129	8.887	8.532	10.435	8.887	11.065
SGT	17.194	18.339	15.823	16.177	11.919	16.145	17.758	11.435	13.113	16.016	10.435	17.726
MT	13.726	11.952	14.726	13.113	14.984	14.177	10.113	15.113	13.758	12.919	14.177	12.371
ARF	8.419	**5.129**	9.355	8.871	9.871	8.968	5.161	9.871	9.516	6.677	10.226	10.742
AMF	11.048	9.016	11.306	10.403	12.081	11.210	7.855	12.145	11.048	9.597	11.565	17.597
ABALR	8.742	6.161	9.435	9.371	10.177	9.145	**5.210**	10.177	10.048	8.371	10.403	6.194
ABOLR	8.548	6.806	9.629	9.468	10.048	9.355	5.581	10.177	9.919	8.661	10.048	6.032
GOCC	6.403	8.500	5.806	4.919	5.581	5.984	9.468	5.774	4.532	5.048	5.129	14.742
GORT	7.210	8.726	5.839	4.339	5.903	5.887	8.952	5.935	4.758	4.919	5.323	11.323
MLHT	10.290	15.226	13.032	13.871	13.194	13.645	15.806	13.355	15.774	16.871	14.677	**1.000**
MLHTPS	12.645	12.323	13.984	13.339	14.113	14.419	12.516	14.242	14.419	15.000	14.210	9.806
iSOUPT	9.726	7.694	9.839	10.161	10.194	9.532	6.629	10.000	10.484	10.419	10.548	3.935
MLBELS	9.371	13.048	7.177	7.823	6.758	7.371	12.468	6.919	8.919	8.177	8.790	16.532
Friedman's χ^2	165.39	201.67	185.01	205.32	143.37	190.97	224.76	138.02	187.20	251.68	145.12	418.51
p-value	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16	2.2e−16

Best results are in bold.

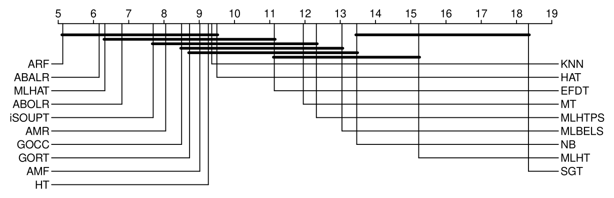
the results are worse because it does not control the concept drift as MLHAT does. This factor also makes MLHAT outperform previous Hoeffding trees, although they will be discussed in detail in Section 4.4. GOCC and GORT are also close to MLHAT in subset accuracy and F1 score, as they are potent ensembles that combine several IDT geometrically averaged. However, combining 10 base trees still performs worse than MLHAT because they do not incorporate their specific tree-level adaptations. kNN works quite well without the need to explicitly consider the concept drift because it already works with a window of the n most recent instances. Finally, ensembles in general also perform better because they combine several classifiers and incorporate concept drift detectors usually. Thus, statistical tests indicate that MLHAT only may not differ from kNN and ensemble-based methods like MLBELS, GOCC, ARF, or ABA in different metrics, which are significantly more complex as the differences in time indicate. Although when this occurs, there is no algorithm that excels in more than one metric or that achieves better results on average, so we can state that there is no clear alternative to MLHAT in terms of overall accuracy.

On the other hand, the results indicate a weakness of MLHAT in terms of the accuracy of the positive labels, that is, it tends to predict an accurate subset of the real labelset, but without including all the active labels. This affects the results in Hamming loss, making MLHAT not statistically superior to the most potent previous methods based on ensembles. Although it is not statistically inferior either. This also affects the micro-averaged precision, where MLHAT does not obtain the best average results either. Thus, it can be seen that obtaining a good accuracy across instances, or in other words, minimizing Hamming loss, is a complicated problem, where the performance of the main proposals is blurred. Thus, in this metric, the ensemble-based methods of ABA and Adaptive Random Forest (ARF) obtain the best results, although MLHAT obtains good enough recall results to be superior overall through the F1 score. Other models like iSOUPT and Adaptive Model Rules (AMR) are also at the same level of performance in this metric according to statistical tests. However, we can see that these algorithms obtain these results at the cost of worse recall rates, while MLHAT remains competitive in this metric in all scenarios, making the overall performance in the end superior.

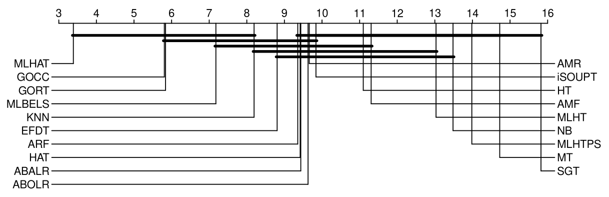
Regarding the real-time factor, the results show that MLHAT is in the middle part of the ranking, close to the BR+HAT transformation. It is slower than the rest of the multi-label IDTs and other less sophisticated proposals, such as NB or LR's ensembles. On the other hand, it is faster than non-Hoeffding IDTs SGT and MT, which have scalability



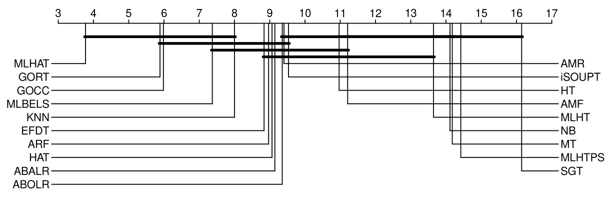
(a) Subset Accuracy



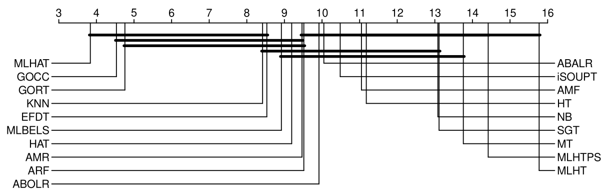
(b) Hamming loss



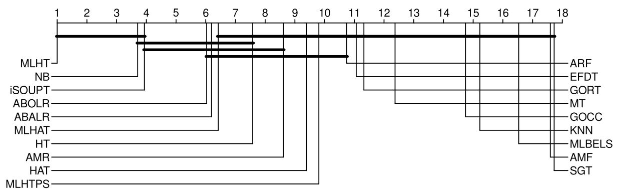
(c) Example-based F1



(d) Micro-averaged F1



(e) Macro-averaged F1



(f) Time (s)

Fig. 5. Critical distance of Bonferroni–Dunn's post-hoc tests (α = 99%).

problems for larger datasets. This is accentuated in the forest approach AMF. MLHAT is also faster than the most similar performing methods: kNN, ARF, and MLBELS. On the one hand, kNN is directly dependent on the number of features and instances. Although MLHAT also maintains kNNs in the leaves, the initial partition of the problem using the tree structure considerably accelerates the process. ARF, GOCC and GORT are slower because they assembles IDTs in a more computationally demanding system with concept drift detectors at tree level, while MLHAT drift detectors operate at branch level, maintaining smaller alternate models. Finally, MLBELS has scalability issues that cause larger datasets to fail. This is due to the combination of an ensemble that maintains as many models as labels, and a mechanism that maintains a pool of discarded models that increases up to 100 per label. Section 4.6 will discuss this factor in more detail in conjunction with the computational complexity and memory consumption evolution of the model.

The results per dataset for a specific metric from Table 6 confirm that MLHAT is not only better on average but consistently outperforms for datasets with varying characteristics, being the best in 12 of 31 data sets and ranking in the first quartile in 23 of 31 data sets. Beyond comparison, these results help identify the problems where MLHAT performs best and where it may be more appropriate to use other algorithms. Thus, we can see that for extremely short flows, less than 1000 instances, MLHAT works well in high-dimensionality cases such as *Medical* and *VirusGO* where there are more features than instances. If there are fewer features, other simpler models perform better. As other authors discussed [12], this is due to Hoeffding trees being conservative models that need a large number of instances to grow. For longer streams, MLHAT generally performs very well, as the tree can grow properly. However, if the number of features is low (less than 500), ensemble-based methods like ABA or ARF can obtain better results, as seen in the cases of *EukaryotePseACC*, and *Corel5k*. MLHAT performs better than the other algorithms in high dimensional spaces, probably because of the partitioning of the truly multi-label feature space. By considering label co-occurrence in the decision tree partitioning process, it better handles the additional complexity of this scenario compared to transforming the problem or using other multi-label decision trees that do not incorporate Bernoulli at this stage. Additionally, keeping two types of classifiers in the leaves allows MLHAT to remain competitive across streams of all lengths: in shorter cases, MLHAT's behavior more closely mimics a kNN, while in longer ones, it approximates an ABA. Finally, we can also highlight the good performance of MLHAT on unbalanced datasets (given by a high MeanIR) like *Mediamill*, *YahooSociety*, *EurlexSM* or *NuwWideCVLAD-plus* among others. This is also a consequence of adapting the leaves classifier to the cardinality of the labels seen so far, which prevents overfitting of the majority labels and, on the other hand, moves to a more advanced classifier when the complexity of the dataset requires it.

4.4. Analysis of MLHAT compared to previous Hoeffding trees

After comparing MLHAT with all other online MLC paradigms, the second study addresses RQ2 by focusing the comparison on previous Hoeffding IDTs and specifically HAT, an algorithm with several common elements with MLHAT but ignoring the additional complexity of multi-label learning. For this analysis, Tables 7, 8, and 6, and Fig. 5 previously discussed are analyzed putting the focus on the Hoeffding trees.

As has been shown in the previous section, MLHAT is 10.61% better at predicting the exact labelset for an instance than the transformation BR+HAT, and around 15% better in the different versions of the F1 score, indicating better accuracy per label. These differences are confirmed by Bonferroni–Dunn, that indicates the superiority of MLHAT in all the metrics except for Hamming loss and the time of execution, in which they are equivalent. This implies that introducing a natively multi-label split criterion that takes into account label co-occurrence through a multivariate Bernoulli distribution is a more effective approach than trying to classify each label separately with BR+HAT. In addition, considering this occurrence allows our model to adapt to the label imbalance that affects the number of labels that each leaf separates, which also contributes to increase the benefits of MLHAT over HAT. Datasets with explicit temporal information generally show that MLHAT is more responsive to possible concept drifts than HAT. This means that, although both use ADWIN-based branch-level detectors, MLHAT's approach of considering all labels together obtains better results than considering the accuracy of each label individually. However, as the RBF generator-based datasets indicate, the HAT drift detector may perform better with data following this distribution. Finally, in terms of execution time, MLHAT is on average 2.7 times faster than HAT. This is because in BR, a separated tree must be maintained for each label, which is slower than maintaining a single tree, even though it has an extra computational load due to the leaf classifiers.

HT and EFDT are Hoeffding trees that have the same problems as HAT due to not adapting the algorithm to the multi-label context. In addition, they slightly worsen the results by not incorporating concept drift detection. In particular, the way of applying the Hoeffding bound ẃith less guarantees" of EFDT versus HT seems to benefit it more, since it compensates for the absence of drift adaptation with a faster tree growth method. MLHAT applies the Hoeffding bound more similarly to HT and HAT, although the other modifications and components specific to the multi-label problem have more effect on the performance of the algorithm, so it obtains better results than EFDT, and also than HT, in all criteria.

Attending to previous Hoeffding trees adaptations to MLC, MLHT and MLHTPS, the results show that MLHAT outperforms them in all performance metrics. In these cases, we observe how the results penalize a partitioning criterion that does not consider the co-occurrence of labels, as well as the absence of concept drift adaptation. Thus, we see that only a multi-label classifier on the leaves is not enough to obtain competitive results. In this line, it can be extracted from the results in Table 6 that, in general, the more complex MLHTPS leaf classifier is more beneficial. On the other hand, the differences in the execution times of MLHT and MLHTPS allow us to quantify the influence of the leaf classifier on the real-time factor in these models. Thus, we can see that, while MLHT is the fastest model in the experiment, the change from the base classifier to an expensive model such as the Prune Set ensemble of HTs increases the execution time by about 20 times on average. This gives us an idea of the importance of the classifiers in MLHAT for the complexity of the model. Thus, the MLHAT time metrics show that in this sense it is more beneficial to maintain two less computationally expensive classifiers such as kNN and ABA+LR for each leaf.

4.5. Analysis of concept drift adaptation

This section addresses RQ3 by making a detailed analysis of the evolution of MLHAT performance along the data stream and how it responds to different concept drift. In this study, we focus on a comprehensive analysis of the evolution of the performance across the data stream. Thus, we include a subset of the real-world datasets with a temporal component that allows us to study complex concept drifts that occur in real scenarios. To conduct a categorized study, and because there are no available multi-label datasets that include explicit concept drift information, synthetic datasets have been generated for this study, as mentioned in Section 4.1.2, in order to evaluate the performance of MLHAT under concept drifts of different types. Finally, to facilitate the presentation of results, this section focuses the comparative study on the most competitive algorithms according to the results of Section 4.3 and tries to include only one representative in the case of families with more than one algorithm. Thus, HAT, kNN and AMR are included as the most competitive BR transformations of single models; ABA and ARF represent the BR transformations of ensembles; and MLBELS, iSOUPT and GORT cover the previous AA.

Fig. 6 shows the evolution of the example-based F1 score, every 50 instances, on the subset of the real-world datasets that have a temporal order, and all the synthetic datasets. These results show great variability and that there is no one algorithm that fits all data distributions without exception. The main conclusion is that there is a relationship between the distribution of the input data and the algorithms that perform better: streams generated with random tree benefit more from tree-based algorithms, while those generated with RBF will be better solved with NB or kNN, although when working with high dimensionality, ensemble-based methods obtain better results. MLHAT in general is more consistent than other proposals, performing acceptably across all distributions. Specifically, MLHAT excels in real-world datasets. However, with more complex datasets, ensemble-based methods such as MLBELS or ARF obtain better results. The reasons for this are detailed below.

It can be extracted that kNN is characterized by having acceptable performance in the early stages of the stream and being resistant to class imbalance and concepts changes. However, the high dimensionality and impossibility of recalling past features make it stagnate compared to other models like HAT or ARF, as can be seen in *D20ng* or *Yelp*. Models that in different ways are based on the cooperation of linear models, AMR, ABA, and iSOUPT, follow similar patterns: they perform better after having seen a large number of instances and gradually concept changes as in *Yelp* or *SynHPGrad*. MLHAT in a way evolves on these two approaches thanks to the synergy of its two base classifiers: at the beginning of the stream it improves rapidly thanks to the kNN it uses as low complexity leaf classifiers. Later, if complexity increases, the LR baggings, improved by the built tree structure, take part in this scenario. This is seen in *D20ng* or *SynHPRec*. On the other hand, ARF and HAT, the BR transformations of Hoeffding trees, may perform better than *MLHAT* on data generated with RBF. However, they are not perfect in these scenarios because they do not perform uniformly for all distributions through which the stream passes. This may be due to the greater complexity of these data and the larger number of labels, making the BR ensemble necessary, especially when it is a forest ensemble, as in the case of ARF. In the case of GORT, also an ensemble of IDTs in this case applying the stacking technique, it is observed that it may have potential in real-world datasets such as *Yelp*. It is also observed that it improves in all cases to individual iSOUPT, the model used as a basis for this proposal. However, in most cases it is significantly worse than MLHAT despite assembling 10 models versus the single MLHAT tree. This indicates the power of MLHAT design elements versus the assembly of multiple IDTs. Finally, MLBELS' performance depends on the type of stream. In general, it responds well if the labels are linearly separable, as with synthetic generators based on Hyperplane and RandomTree. However, with real data and RBF, the nature of its neural networks does not allow it to excel with more complex patterns.

By concept drift type, MLHAT offers uniform performance in the different types. Although some models may achieve better results in specific scenarios, MLHAT handles concept changes better in sudden, gradual and especially recurrent cases, as seen in *SynHPRec* and *SynTreeRec*. This indicates that using a multi-label metric to monitor concept drift is highly beneficial, as it detects changes more quickly than alternatives that monitor each label separately. This flexibility in handling different types of concept drift makes MLHAT particularly effective with real datasets, which often exhibit more mixed concept drift patterns. The incremental concept drift is the most challenging for all algorithms and MLHAT is not an exception. We see how in *SynHPInc* it is outperformed by MLHT and MLBELS, and in *SynRBFInc* by ARF HAT and kNN. In these cases, it might be worth exploring an ensemble option. Hamming loss is the metric used in the experimentation for detecting the concept drift in MLHAT, for serving well in general for all the types of concept drifts, especially those found in real datasets (see Section 4.2). However, to maximize adaptability to abrupt changes, it might be more beneficial to employ another metric, such as subset accuracy.

We can summarize the contributions of MLHAT in terms of concept drift in the following key points:

- MLHAT shows consistent performance across various data distributions, more so than other proposals.

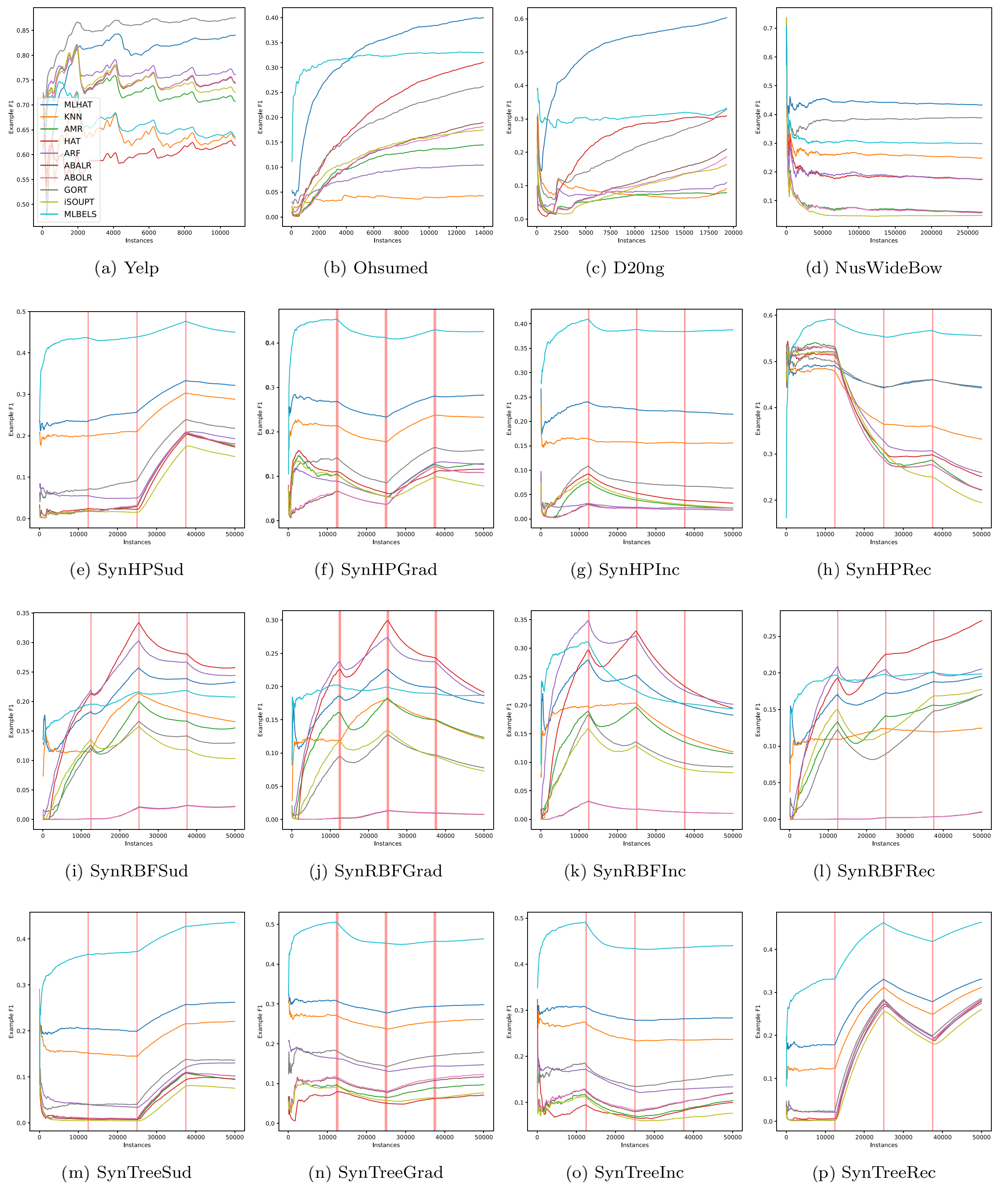


Fig. 6. Prequential evaluation of example-based F1 on a selection of datasets with concept drift (expressed as vertical lines). The type of concept drift is given by the name of the synthetic dataset; more information in Table 4.

- MLHAT combines strengths of different approaches: On the one hand, it used kNN as low-complexity leaf classifiers for rapid improvement at stream start. On the other hand, it employs LR baggings, enhanced by tree structure, for handling increased complexity.
- MLHAT offers uniform performance across different types of concept drift: Handles concept changes better in sudden, gradual, and especially recurrent cases. This flexibility makes MLHAT particularly effective with real datasets, which often exhibit mixed-concept drift patterns.
- MLHAT's use of a multi-label metric (Hamming loss) to monitor concept drift is highly beneficial, as it detects changes more quickly than alternatives that monitor each label separately. However, other metrics might be more beneficial for maximizing adaptability to abrupt changes.
- Attending to the performance in specific scenarios, we can observe that MLHAT excels particularly in real-world datasets; that it may be outperformed by ensemble methods (e.g., MLBELS, ARF) on more complex datasets; and that can struggle with incremental concept drift, like all tested algorithms.

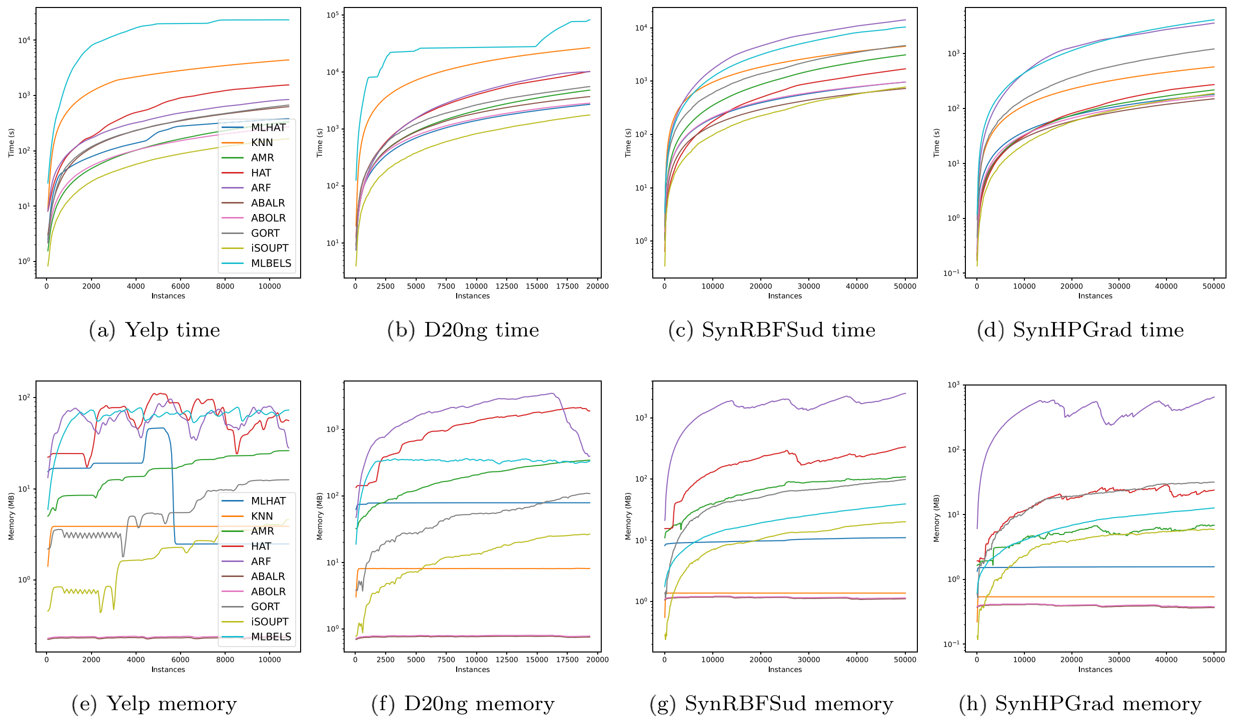


Fig. 7. Resources consumption on a selection of the biggest datasets using logarithmic scale.

- While not always the top performer, MLHAT generally provides more consistent results across various data distributions and drift types compared to other algorithms tested.

4.6. Analysis of efficiency

In the context of data stream classification, the efficiency and speed of processing instances are of great importance, as they are expected to arrive at high speed. Thus, the model must not only be accurate but also scalable, keeping its growth controlled over time. As shown in Section 4.3, MLHAT is a time-competitive algorithm. In this section, we analyze the details of the time and memory consumption of MLHAT.

The space and time complexity of MLHAT is upper-bounded by HT and lower-bounded by the ADWIN method to prune nodes. Following the reasoning of [49], we can divide the operations in the tree into three main components: updating the statistics in the leaf, evaluating possible splits, and learning in the corresponding leaf classifier. Updating the statistics requires traversing the tree to the leaf and updating the label counts at each node along the path. This has a complexity of $O(hl)$ where h is the height of the tree and l is the number of labels in the multi-label problem. To look for possible splits, f splits will be evaluated, where f is the number of features or attributes. For each f split, v computations of information gains are required. Finally, each information gain requires c operations. Thus, split evaluations require $O(fvc)$ operations. Lastly, the complexity of learning in leaves is in the upper bound of the high-complexity classifier, a bagging method that aggregates $k = 10$ logistic regressions, whose complexity depends mainly on the number of features f. Thus, this step has a complexity of $O(kf)$. In conclusion, the total complexity of MLHAT in the worst-case scenario will be $O(hlf^2vck)$. Empirical results are shown below to observe the real resource consumption of MLHAT and how the concept drift affects it.

Fig. 7 shows the evolution in time and memory consumption of the different paradigms for a selection of datasets that combine the largest number of instances and a different number of attributes. To facilitate the visualization of results, only the most representative methods selected in the previous section are plotted here. The results show that MLHAT is faster and more efficient than most models, especially models of similar performance like kNN and MLBELS. This is largely due to the use of a single decision tree and its adaptation to concept drift. Thus, when concept drift is detected and pruning of obsolescent parts of the tree occurs, the memory occupied by the model decreases, which also affects the processing speed of the instances. In contrast, other IDT-based models such as MLHTPS, iSOUPT, SGT or AMF grow without limits, which can be problematic for longer streams. This is accentuated in ensemble versions of these ones like GORT. This is evident in the case for SGT and AMF in *NusWideBow*, for example. Their results have not been able to finish due to requiring hundreds of GB of RAM and/or more than 240 h of execution. Other models based on HT and with concept drift, such as ARF and HAT also scale poorly, especially with datasets that have many labels. This is due to the multi-label transformation of BR requires training one model for each label. In the case of ARF, this implies a memory consumption that can be 100 times higher than that of MLHAT and takes 10 times longer to process the same amount of information. Other proposals such as kNN or ABA imply minimal or no build-up of the model, which makes them very efficient in memory and also in time. MLHAT builds its leaf classifiers on these models, so their presence hardly affects the total computation time and memory used by our proposal, keeping it competitive with other tree-based models like MLHTPS or iSOUPT.

5. Conclusions and future work

This paper introduced Multi-Label Hoeffding Adaptive Tree (MLHAT), an IDT for multi-label data streams. MLHAT incorporates a multi-probabilistic split criterion to natively consider co-occurrences in multi-label data and monitors the labelsets' entropy and cardinality at the leaves to adapt the classification process complexity. We also proposed to monitor the concept drift on the intermediate nodes of the tree, allowing the review of split decisions if performance decreases, and replacing affected branches with new ones adapted to the new data distribution. These combined mechanisms allow MLHAT to overcome various multi-label stream difficulties, such as concept drift and class imbalance. We presented an exhaustive experimental study to assess the competitiveness of MLHAT against 16 different multi-label online classifiers, including the main previous IDTs and representations of other online paradigms. The experimentation covered 41 datasets and 12 multi-label metrics, for which MLHAT achieved the top average result in 10 metrics. We also analyzed the MLHAT evolution across the stream through prequential evaluation, which allowed us to observe that it can perform very well from an early stage of the data stream and that it also adapts quickly to concept drift.

Future work on MLHAT can focus on several aspects. Firstly, while our work has shown that MLHAT is a competitive method for MLC in data streams, it may depend on a non-trivial parameter setting process. Therefore, it would be interesting to study modifications to reduce the number of parameters to be adjusted, for example, with hyperparameterization or dynamic adjustments. Secondly, our experiments have shown that there are experiments with a high label density in which MLHAT performance can be improved by ensembles. Thus, it can be interesting to explore certain techniques, such as bagging, boosting, or random subspaces. Finally, it would be valuable to explore the application of MLHAT to real-world scenarios and domains where multi-label stream learning has great potential, such as predictive maintenance, pose estimation, or autonomous driving.

CRediT authorship contribution statement

Aurora Esteban: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation. **Alberto Cano:** Writing – review & editing, Validation, Supervision, Software, Resources, Formal analysis, Conceptualization. **Amelia Zafra:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Sebastián Ventura:** Writing – review & editing, Supervision, Investigation, Funding acquisition, Formal analysis, Conceptualization.

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

There is a link in the paper for repository with data/code.

Acknowledges

This research was supported in part by grant PID2020-115832GB-I00 funded by MICIN/AEI/10.13039/501100011033, by the ProyExcel-0069 project funded by Departament of University, Research and Innovation of the Andalusian Government, and by a FPU predoctoral grant FPU19/03924 from the Spanish Ministry of Universities. High Performance Computing resources provided by the HPRC core facility at Virginia Commonwealth University (https://hprc.vcu.edu) were used for conducting the research reported in this work.

References

[1] N. Åkerblom, Y. Chen, M. Haghir Chehreghani, Online learning of energy consumption for navigation of electric vehicles, Artificial Intelligence 317 (2023) 1–25.
[2] B. Krawczyk, Active and adaptive ensemble learning for online activity recognition from data streams, Knowl.-Based Syst. 138 (2017) 69–78.
[3] M. Bahri, A. Bifet, J. Gama, H.M. Gomes, S. Maniu, Data stream analysis: Foundations, major tasks and tools, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 11 (3) (2021) 1–17.
[4] G.J. Aguiar, A. Cano, A comprehensive analysis of concept drift locality in data streams, Knowl.-Based Syst. 289 (2024) 1–18.
[5] G. Aguiar, B. Krawczyk, A. Cano, A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, Mach. Learn. (2023) 1–60.
[6] M. Roseberry, B. Krawczyk, Y. Djenouri, A. Cano, Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams, Neurocomputing 442 (2021) 10–25.
[7] S. Bakhshi, F. Can, Balancing efficiency vs. effectiveness and providing missing label robustness in multi-label stream classification, Knowl.-Based Syst. 289 (2024) 1–14.
[8] C. Li, D. Zhu, C. Hu, X. Li, S. Nan, H. Huang, ECDX: Energy consumption prediction model based on distance correlation and XGBoost for edge data center, Inform. Sci. 643 (2023) 1–13.
[9] J. Du, C.M. Vong, Robust online multilabel learning under dynamic changes in data distribution with labels, IEEE Trans. Cybern. 50 (1) (2020) 374–385.
[10] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
[11] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: Proceedings of the 8th International Symposium of Intelligent Data Analysis, 2009, pp. 249–260.
[12] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classification for evolving data streams, Mach. Learn. 88 (1–2) (2012) 243–272.
[13] Z. Shi, Y. Xue, Y. Wen, G. Cai, Efficient class incremental learning for multi-label classification of evolving data streams, in: Proceedings of the 2014 International Joint Conference on Neural Networks, 2014, pp. 2093–2099.
[14] A. Büyükçakir, H. Bonab, F. Can, A novel online stacked ensemble for multi-label stream classification, in: Proceeding of the 27th International Conference on Information and Knowledge Management, 2018, pp. 1063–1072.
[15] S. Liang, W. Pan, D. You, Z. Liu, L. Yin, Incremental deep forest for multi-label data streams learning, Appl. Intell. 52 (12) (2022) 13398–13414.
[16] G. Alberghini, S. Barbon Junior, A. Cano, Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams, Neurocomputing 481 (2022) 228–248.
[17] W. Liu, H. Wang, X. Shen, I. Tsang, The emerging trends of multi-label learning, IEEE Trans. Pattern Anal. Mach. Intell. 44 (11) (2021) 7955–7974.
[18] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: Models, experimental study and prospects, Inf. Fusion 44 (2018) 33–45.
[19] M.L. Zhang, Z.H. Zhou, ML-KNN: A lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048.
[20] M. Roseberry, S. Džeroski, A. Bifet, A. Cano, Aging and rejuvenating strategies for fading windows in multi-label classification on data streams, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, 2023, pp. 390–397.
[21] R. Sousa, J. Gama, Online multi-label classification with adaptive model rules, in: Proceedings of the XVII Conference of the Spanish Association for Artificial Intelligence, 2016, pp. 58–67.
[22] R. Sousa, J. Gama, Multi-label classification from high-speed data streams with adaptive model rules and random rules, Prog. Artif. Intell. 7 (3) (2018) 177–187.
[23] A.N. Tarekegn, M. Giacobini, K. Michalak, A review of methods for imbalanced multi-label classification, Pattern Recognit. 118 (2021) 107965.
[24] C. Manapragada, G.I. Webb, M. Salehi, Extremely fast decision tree, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1953–1962.
[25] H. Gouk, B. Pfahringer, E. Frank, Stochastic gradient trees, in: Proceedings of the 11th Asian Conference on Machine Learning, 2019, pp. 1094–1109.
[26] J. Mourtada, S. Gaïffas, E. Scornet, AMF: Aggregated Mondrian forests for online learning, J. R. Stat. Soc. Ser. B Stat. Methodol. 83 (3) (2021) 505–533.
[27] A. Osojnik, P. Panov, S. Džeroski, Multi-label classification via multi-target regression on data streams, Mach. Learn. 106 (6) (2017) 745–770.
[28] B. Dai, S. Ding, G. Wahba, Multivariate Bernoulli distribution, Bernoulli 19 (4) (2013) 1465–1483.
[29] H.M. Gomes, A. Bifet, J. Read, J.P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Mach. Learn. 106 (9–10) (2017) 1469–1495.
[30] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, 2007, pp. 443–448.
[31] J. Bogatinovski, L. Todorovski, S. Džeroski, D. Kocev, Comprehensive comparative study of multi-label classification methods, Expert Syst. Appl. 203 (February) (2022) 1–18.
[32] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 139–147.
[33] R.S.M. De Barros, S.G.T. De Carvalho Santos, P.M.G. Junior, A boosting-like online learning ensemble, in: Proceedings of the 2016 International Joint Conference on Neural Networks, 2016, pp. 1871–1878.
[34] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, 2010, pp. 135–150.

[35] N.C. Oza, S. Russell, Online bagging and boosting, in: Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, 2001, pp. 229–236.
[36] E.B. Andersen, Sufficiency and exponential families for discrete sample spaces, J. Amer. Statist. Assoc. 65 (331) (1970) 1248–1255.
[37] A. Law, A. Ghosh, Multi-label classification using binary tree of classifiers, IEEE Trans. Emerg. Top. Comput. Intell. 6 (3) (2022) 677–689.
[38] Ł. Korycki, B. Krawczyk, Streaming decision trees for lifelong learning, in: Machine Learning and Knowledge Discovery in Databases. Proceedings of the 2021 ECML PKDD Research Track, 2021, pp. 502–518.
[39] J. Montiel, M. Halford, S.M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H.M. Gomes, J. Read, T. Abdessalem, A. Bifet, River: Machine learning for streaming data in python, J. Mach. Learn. Res. 22 (2021) 1–8.
[40] J. Duarte, J. Gama, A. Bifet, Adaptive model rules from high-speed data streams, ACM Trans. Knowl. Discov. Data 10 (3) (2016) 1–22.
[41] R. Cerri, J.D.C. Junior, E.R. Faria, J. Gama, A new self-organizing map based algorithm for multi-label stream classification, in: Proceedings of the 36th ACM Symposium on Applied Computing, 2021, pp. 418–426.
[42] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[43] J. Gama, P.P. Rodrigues, R. Sebastião, Evaluating algorithms that learn from data streams, in: Proceedings of the 2009 ACM Symposium on Applied Computing, 2009, pp. 1496–1500.
[44] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.L. Boulesteix, D. Deng, M. Lindauer, Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 13 (2) (2023).
[45] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019, pp. 2623–2631.
[46] H.M. Gomes, J. Read, A. Bifet, Streaming random patches for evolving data stream classification, in: Proceedings of the 2019 IEEE International Conference on Data Mining, 2019, pp. 240–249.
[47] F. Hutter, H. Hoos, K. Leyton-Brown, An efficient approach for assessing hyperparameter importance, in: Proceedings of the 31st International Conference on Machine Learning, Vol. 2, 2014, pp. 1130–1144.
[48] T.G. Dietterich, Statistical tests for comparing supervised classication learning algorithms, Neural Comput. 10 (7) (1998) 1–24.
[49] E. Garcia-Martin, A. Bifet, N. Lavesson, R. König, H. Linusson, Green accelerated Hoeffding tree, in: Proceedings of the 2021 TinyML Research Symposium, 2021, pp. 1–8.