Adaptive Learning on Hierarchical Data Streams using Window-weighted Gaussian Probabilities

Eduardo Tieppo[a,b,*], Júlio Cesar Nievola[a], Jean Paul Barddal[a]

[a]*Programa de Pós-Graduação em Informática (PPGIa), Pontifícia Universidade Católica do Paraná, Curitiba, Brazil*
[b]*Instituto Federal do Paraná (IFPR), Pinhais, Brazil*

Abstract

The hierarchical data stream classification task addresses challenges in both hierarchical and data stream classification primary areas. In these scenarios, machine learning models must simultaneously deal with class hierarchies and adapt to respond to nonstationary data. Given such a challenging set of traits, existing techniques are deficient, as they perform incremental learning and are slow to adapt to newer data, thus not capturing their dynamics in a timely fashion. In this study, we propose two novel adaptive Gaussian Naive Bayes classifiers tailored to classify hierarchical data streams. The models use window-weighted Gaussian probabilities to consider current and historical data and improve the adaptability of the classifiers, especially for nonstationary data streams. As a result of our research, we introduce a unified protocol for evaluating and comparing hierarchical data stream classifiers and establish a benchmark for the hierarchical data stream classification task encompassing the proposed methods and state-of-the-art classifiers. The results demonstrate that our proposed algorithms achieve better prediction

*Corresponding author: eduardo.tieppo@pucpr.edu.br

correctness than their state-of-the-art counterparts while responding more swiftly to changes in data distribution.
Keywords: Hierarchical Classification, Data Stream Classification, Gaussian Naive Bayes

1. Introduction

Learning algorithms are usually tailored for problems with specific characteristics. For instance, learning models tailored to the hierarchical classification task work with labels that are appropriately hierarchical [1]. Similarly, learning models for streaming scenarios utilize continuous incoming data provided by a stream rather than batch data or a full view of the dataset [2]. However, progress in data acquisition and storage technologies is resulting in challenges characterized by features that extend beyond a single classification task. A recent example is the work of [3], in which the authors bring forward a dataset related to public health that exhibits both streaming and hierarchical traits. Therefore, it is necessary to formulate solutions that simultaneously address these characteristics [4] instead of handling each individually.

This paper targets hierarchical data stream classification, which inherits challenges from its foundation classification tasks, i.e., hierarchical classification and data stream classification [4]. On data stream classification, learning models often assume the class structure as flat and without relationships between the classes, thus losing potentially valuable information within the class taxonomy. Meanwhile, on hierarchical classification, learning models assume finite and stationary data, with models not updating themselves, disregarding time or incoming data [2, 1]. In addition, challenges at the in-

tersection of streaming and hierarchical classification may also exhibit data distribution changes, a phenomenon called concept drift [5].

Concept drifts result in dynamic environments where learning models must decide when and how to adapt to non-stationary incoming data. At the same time, models must adapt swiftly to data drifts (plasticity) while retaining older patterns observed in previous data (stability). This trade-off is pursued when developing new algorithms and is referred to as the plasticity-stability dilemma [6].

In this paper, we propose two adaptive learning method variants fitted to work with hierarchical data streams using window-weighted Gaussian probabilities to obtain a balance on the stability-plasticity dilemma trade-off.

The first method variant weights the historical Gaussian probabilities using updated data considering incoming samples from the data stream. Meanwhile, the second method uses only the incoming data to calculate an adaptive Gaussian probability. Our proposal uses window-weighted probabilities along with Bayes' Theorem to perform adaptive learning using historical and recent data to calculate probabilities. Experimentation-wise, we append these window-weighted probabilities to a Gaussian Naive Bayes classifier. The results demonstrate improvements in its application on the predictive performance of the learning model, especially on unstable data streams with well-described concept drifts.

As a byproduct of this research, we propose a unified assessment protocol to compare hierarchical data stream classifiers using different evaluation metrics as criteria in a Multi-Criteria Decision-Making analysis, resulting in a reproducible testbed for further studies with other classifiers in the hierar-

chical data stream classification area.

The remaining sections of this paper are structured as follows. Section 2 provides the theoretical background of this study, including both hierarchical classification and data stream classification areas. Section 3 depicts existing works in hierarchical data stream classification, comprehending their background and research gaps. Next, Section 4 describes our proposed adaptive learning methods. Section 5 outlines the unified assessment protocol planned as a reproducible testbed. Section 6 benchmarks our proposal against existing techniques using real-data hierarchical data streams. Finally, Section 7 concludes this study and states envisioned future works.

2. Theoretical Background

This section describes hierarchical classification, data stream classification, and the result at the intersection of such areas, which is the problem our proposal tackles.

2.1. Hierarchical classification

In hierarchical classification, instances are assigned to a class that is part of a set of related classes, with specific classes in the set representing hierarchical relationships to general classes [7, 8, 1].

A class hierarchy can be defined as regular concept hierarchy [9] structured over a partially ordered set $(Y, \succ)$, where Y stands for a finite set comprising all objective classes from a problem and the relation $\succ$ is defined as a subsumption relation, i.e., a data sample assigned to a specific class (child class) in the hierarchy is consequently also assigned to all linked general nodes (parent classes) [1, 10]. Due to this data trait, hierarchical

classifiers differ from traditional ones in some key aspects. The authors in [1] specified these aspects in a four-tuple with the format $(\Omega, \Delta, \Xi, \Theta)$, where Ω refers to the data structure used in the representation of the hierarchy of classes, Δ and Ξ to the cardinality and label depth applied by the hierarchical classifiers, and Θ to the approach used by the classification algorithm to treat the hierarchy [1, 4].

We focus on hierarchical classifiers built with trees as data structure, assign leaf node classes (mandatory leaf-node prediction - MLNP) as the last class of at most one predicted label path (single path prediction - SPP), and in which their learning models consist of one classifier per node/class to predict between the child nodes of the class (Local classifier per parent node - LCPN) or one single classifier to handle all classes using the hierarchy information (Global classifier - GC).

As hierarchical classifiers assign label paths to instances, it is imperative to consider the entire label path when assessing the predictive performance of the learning model. In this sense, the authors in [11] proposed hierarchical versions of the traditional evaluation metrics Precision, Recall, and F-Measure. Related to the Precision metric, the hierarchical Precision (hP) computes the number of classes that are components of a predicted hierarchical set of classes that also appear in the ground-truth hierarchical set of classes for a given instance. Related to the Recall metric, the hierarchical Recall (hR) counts the number of ground-truth classes comprised by the predicted hierarchical set of classes for a given instance. As in the traditional classification metrics, the hierarchical F-Measure (hF) is the harmonic mean between their hierarchical Precision and hierarchical Recall components [12].

2.2. Data stream classification

Data stream classification places specific demands on classifiers compared to traditional classification scenarios and methods. Traditional classifiers assume finite and stationary data used in a well-defined training step. This premise is not applicable in streaming scenarios, wherein a data stream continuously supplies data to a learning model [2].

Note that this single unique trait in how the stream provides data to the learning model requires some characteristics for data stream classifiers. As data streams are potentially unbounded, they may surpass the capacity of any computational resources available, and their underlying distribution may change over time, which may also lead to changes in the discovered patterns (concept drifts). Data stream classifiers should be able to adapt to concept drift to avoid or reduce the impact of these drifts on the learned concepts [13, 2, 14].

Moreover, a classifier tailored to the streaming scenario must operate with limited memory, where each instance is processed once and discarded to avoid memory overflow. In this sense, models typically store only a summary of the stream, and approximate results are allowed. In addition, classifiers typically follow time-window models that consider only certain parts of the data (usually the most recent). For example, a classifier interested in the most recent data may consider only data within a window that slides along with the data stream and discards data outside the window or that weights incoming data samples differently depending on their recentness [15, 14].

Due to the memory and time constraints imposed by the streaming scenario, stream classifiers must at least use batch mode adaptations (mini-

batches) to process all data. However, it is desirable for methods to be incremental or adaptive [16]. Here, we focus on adaptive stream classifiers that update themselves by using strategies to forget previously learned information and, thus, can adapt to data changes over time [16].

Data stream learning models are assessed utilizing the same metrics employed in traditional classification but on a per-instance basis. In this protocol, known as the prequential evaluation method or interleaved test-then-train method, each instance is initially used to assess the model and calculate the evaluation metrics and then to train and update the model [17].

2.3. Hierarchical data stream classification

The hierarchical data stream classification merges the foundation areas described above. Consequently, it inherits their properties and constraints.

First, a hierarchical data stream classifier must use a data stream as input to the learning process not as a data source only but by effectively processing portions of the data over time (preferably instance by instance) without a complete view of the data set at any point in time.

Hereafter, let $hDS = [(\vec{x}^t, \vec{y}^t)]_{t=0}^{\infty}$ be a hierarchical data stream that provides instances $(\vec{x}^t, \vec{y}^t)$ over time t, where $\vec{x}^t$ stands for the set of features and $\vec{y}^t$ for the corresponding ground-truth set of classes for the t-th instance. Thus, a hierarchical data stream classifier is a learning model function $f^t : \vec{x}^t \mapsto \vec{y}^t$, where the training data consists of features represented by $\vec{x}$ for each t [4]. At this point we highlight that, as in the hierarchical classification, $\vec{y}^t$ includes class labels organized under a regular concept hierarchy structured on a partially ordered set $(Y, \succ)$, where Y is a prior known finite set containing all classes and the relation $\succ$ is defined as an asymmetric,

anti-reflexive and transitive subsumption relation.

As for the data stream classification, a hierarchical data stream classifier must also consider non-stationary data distributions and consequently be able to adapt to possible concept drifts. Given a set C of prior probabilities of classes and a class-conditional probability density function given by $C = \bigcup_{y \in Y} \{(P[y], P[\vec{x}|y])\}$ [18, 19], the hierarchical data stream classifier must be able to update its learning model function f^t if between two distinct timestamps t_i and t_j a concept drift occurs, i.e., C^{t_i} differs from C^{t_j}.

To swiftly adapt f^t, a hierarchical data stream classifier typically uses a windowing strategy to focus on specific data portions (usually the recent). As mentioned above, the stability-plasticity dilemma must be accounted for since small windows lead to a faster adaptation of f^t to concept drifts (plasticity). In contrast, large windows preserve historical data and produce more stable f^t (stability).

Finally, a hierarchical data stream learning model must operate within limited computational resources and analyze each instance $(\vec{x}^t, \vec{y}^t)$ only once in the order in which it arrives and not reuse it. Next, the model must be able to provide a prediction $\vec{x}^t$ on demand given the updated f^t based on the training information provided by $(\vec{x}^t, \vec{y}^t)$.

3. Related Works

This section discusses existing works and efforts regarding the hierarchical data stream classification area, comprehending their background, research gaps, and current methods. First, in the following subsection, we briefly describe the background of the state of the art and highlight the need for

novel methods based on an existing hierarchical data stream set. Next, we list and detail the current state-of-the-art techniques tailored for the hierarchical data stream classification task.

3.1. Background

Despite both fundamental areas being well-established research topics, their properties are not usually addressed simultaneously in the literature. Moreover, current techniques can not solve hierarchical data stream classification problems straightforwardly. For instance, hierarchical classifiers cannot handle changing and unbounded data, while classifiers tailored for streaming data often overlook the hierarchical relationships between classes of data samples [4].

Comprehensive reviews of hierarchical classification have been presented in [20, 7], where the basic concepts and terminologies of the field have been formally defined. Similarly, comprehensive reviews of data stream classification have been presented in [21, 22, 23], which show that substantial attention has been paid to scenarios in which data is provided as a stream and how to overcome the challenges associated with it.

Data stream classification has been revisiting the classification subtasks, resulting in new research connections, initially with the binary classification of data streams and then handling hierarchically structured problems. This evolution seems reasonable considering the improvement in obtaining, collecting, and storing large-scale data.

For example, the authors in [3] proposed a new version of one of the datasets initially introduced in [24], comprehending more instances and with a formal definition of concept drifts within the data stream. This dataset is

composed of instances representing flying insects captured by electronic traps built with optical sensors, obtaining the wing-beat frequency of the insects. The dataset aims to identify disease vectors or agricultural damage-causing species via classifiers.

The dataset generated by the authors contains all instances collected in natural order (randomly) along the data stream. The authors also proposed different filters and arrangements of the dataset based on ambient features, such as the temperature or the luminosity at a given time in which an instance was collected. Hence, the authors controlled the occurrence and representation of concept drift in the datasets. As a result, the authors proposed 11 datasets, including the one containing the complete data, plus another ten sets representing balanced and unbalanced versions of five distinct patterns simulating concept drifts.

Despite the dataset being initially introduced by the authors as a flat dataset, the data stream retains its hierarchical characteristic as proposed in [24]. In other words, each instance component of the data stream represents species of disease vector mosquitoes, which are naturally organized in an entomological taxonomy.

From the perspective of the classification task, it is possible to handle this dataset in a few different ways. Note that despite being a hierarchical data stream, textcolorblueone can apply learning models from other classification subtasks if particular data traits are unaccounted for. For instance, one can apply a hierarchical classifier to a hierarchical data stream, handling it as a batch dataset or a flat data stream classifier by bypassing data hierarchy. However, both strategies have some drawbacks, as discussed below.

On the one hand, regarding hierarchical classification, the main disadvantages of learning models are the premise of finite data available for a well-defined training step and the building of a stationary model. Data streams are potentially unbounded, and data are continuously made available to the model over time. Thus, several strategies used by hierarchical classifiers would fail in a data stream environment, such as storing all data or reprocessing a given instance.

In addition, due to the ephemeral trait of the data, the stationary model could result in model degradation over time since it would use old information as a reference that may no longer describe the current data.

Note that the forced use of exclusively hierarchical classifiers in a hierarchical data stream is possible, for example, if considered in hypothetical circumstances where one can store all data in memory. However, it may render a noticeable drawback to the classification results since the learning model would not use any update strategy to respond to possible changes in data distribution that affect predictions.

On the other hand, concerning data stream classification, learning models have the main drawback of not considering intrinsic class hierarchies present in hierarchical data streams, losing information that could be useful to the classification process.

Note that the forced use of flat data stream classifiers in a hierarchical data stream is even feasible by ignoring the hierarchy of classes, predicting a class in a flat and isolated way, and retrieving a hierarchical label path through further analysis of an attached ontology. However, all the information on the relationship between the classes is wasted mainly in two aspects.

First, the model would not consider any similarity between classes, which may affect the prediction and the classification costs (since the classification costs are distinct at different spots in the hierarchy). Second, the trade-off between prediction reliability and usefulness is not an option since the classifiers do not have enough information to walk through the hierarchy and decide between discriminating generic classes (where there are more examples and thus more confidence but less usefulness) or specifying the prediction to deeper hierarchy classes (where there are fewer examples and thus less confidence but more usefulness) [8].

3.2. State-of-the-art techniques

The authors in [4] presented a comprehensive review of the state of the art of the hierarchical data stream classification field. The survey listed the algorithms used to deal with problems involving hierarchical data streams and tagged the methods with the area constraints to check their adherence to the hierarchical data stream classification task. As a finding, the first method fully adherent to the task was proposed in [24], where the authors proposed a k-Nearest Neighbors (kNN) classifier fitted to work with hierarchically structured data streams.

In addition to this classifier, we retrieved another seven methods published subsequently in the literature that also fulfill hierarchical data stream constraints. To our knowledge, these methods represent all the available classifiers tailored for the hierarchical data stream classification task. In the following subsections, we provide explanations of these techniques and highlight their differences and relevance to our work.

3.2.1. Local kNN-hDS

Local kNN-hDS [24] is a pioneering method designed to work with hierarchically structured data streams. It employs a sliding window strategy to manage data instances, ensuring that older data is forgotten using a first-in/first-out approach. This method also utilizes local classifiers per parent node (LCPN) and computes distances between instances to perform predictions. Its data representation involves maintaining buffers of instances associated with each class node in the hierarchy.

3.2.2. Global kNN-hDS

Built upon Local kNN-hDS, Global kNN-hDS [25] introduces a global classification (GC) variant. It reduces the reliance on distance computations by analyzing the label path of nearest neighbors instead of employing the LCPN approach. This analysis involves splitting the label path into levels and selecting label paths containing the most frequent label for each hierarchy level. This adaptation results in reduced computational resource usage while maintaining prediction correctness.

3.2.3. Local kNC-hDS and Local Dribble-hDS

Local kNC-hDS and Local Dribble-hDS [26] are methods tailored for hierarchical data streams. They employ summarization techniques, specifically incremental centroids and cluster feature vectors (CFs), to represent the data instances. Local kNC-hDS uses incremental centroids at each hierarchy node and performs predictions by comparing incoming instances to

k-nearest centroids. On the other hand, Local Dribble-hDS stores CFs representing instances and utilizes the nearest neighbor as a prediction strategy. Both methods apply adaptive learning through a sliding window, ensuring efficient memory usage and timely updates.

3.2.4. Global kNC-hDS and Global Dribble-hDS

Global kNC-hDS and Global Dribble-hDS [27] merge the strategies of Global kNN-hDS and Local kNC-hDS/Local Dribble-hDS. These classifiers share a global classification approach and employ summarization techniques. They have demonstrated superior prediction correctness and processing speed compared to Local kNN-hDS and their local counterparts.

3.2.5. GNB-hDS

GNB-hDS [28] is an incremental Gaussian Naive Bayes classifier designed for classifying hierarchical data streams. It employs statistical summaries of the data stream instead of storing raw instances and relies on Bayes' Theorem for predictions. The method uses an LCPN approach and a landmark window, summarizing the entire data stream into incremental statistical descriptors. GNB-hDS offers improved processing speed compared to Local kNN-hDS while achieving similar prediction correctness.

3.2.6. GNB-hDS-iYJ

GNB-hDS-iYJ [29] is a variant of GNB-hDS that incorporates an incremental adaptation of the Yeo-Johnson Power Transformation. This trans-

formation reduces the data stream variables' skewness and can be applied to any data stream mining task. GNB-hDS-iYJ utilizes statistical models to determine the appropriate power parameter for data transformation. In the same paper, authors showed that this adaptation improves prediction correctness significantly.

3.3. Overview and comparison

Table 1 summarizes these state-of-the-art techniques, including their data representation and prediction strategies.

Table 1: Overview of the state-of-the-art techniques.

Method	**Data representation**	**Prediction strategy**
Local kNN-hDS [24]	Buffers of instances	Nearest neighbors
Global kNN-hDS [25]	Buffers of instances	Nearest neighbors and label path analysis
Local kNC-hDS [26]	Centroids	Nearest neighbors
Local Dribble-hDS [26]	Cluster Feature Vectors	Nearest neighbors
Global kNC-hDS [27]	Centroids	Nearest neighbors and label path analysis
Global Dribble-hDS [27]	Cluster Feature Vectors	Nearest neighbors and label path analysis
GNB-hDS [28]	Statistical descriptors	Bayesian probabilities
GNB-hDS-iYJ [29]	Statistical descriptors	Bayesian probabilities

Table 2 also specifies the methods concerning the hierarchical data stream classification properties (using the same categorization previously discussed). Each method depicted in Tables 1 and 2 is detailed as follows.

Table 2: Specification of the state-of-the-art techniques concerning the hierarchical data stream classification properties.

Method	Data Structure (Ω)	Label cardinality (Δ)	Label Depth (Ξ)	Hierarchy handling (Θ)	Single-pass	Readiness	Bounded Memory	Concept drift	Time Window	Data stream handling
Local kNN-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	Sliding	Adaptive
Global kNN-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	Sliding	Adaptive
Local kNC-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	Sliding	Adaptive
Global kNC-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	Sliding	Adaptive
Local Dribble-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	Sliding	Adaptive
Global Dribble-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	Sliding	Adaptive
GNB-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	Landmark	Incremental
GNB-hDS-iYJ	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	Landmark	Incremental

Note that the GNB-hDS method was proposed as an incremental model since it does not have a built-in mechanism to forget data. Thus, there is room to improve it by making it adaptive. As mentioned above, adaptive methods are more suited to respond quickly to concept drifts since the classifier's mapping function can emphasize recent data, which is more representative concerning changes, and capture the data dynamics accordingly. Therefore, in this study, we propose a sliding window with a Gaussian Naive Bayes classifier to handle more unstable hierarchical data streams and improve its adaptation to concept drifts.

4. The Window-Weighted Gaussian Probabilities

In this section, we propose window-weighted Gaussian probabilities to enhance the adaptiveness of the Gaussian Naive Bayes classifier tailored for hierarchical data stream classification (GNB-hDS). We present two variations of the classifier, each employing a distinct weighting strategy for integrating window-weighted Gaussian probabilities.

We applied the window-weighted Gaussian probabilities within the incremental Gaussian Naive Bayes GNB-hDS previously introduced in [28], making it adaptive and resulting in two variations of the original classifier using two different weighting strategies.

First, the traditional GNB-hDS must adhere to two fundamental properties: the ability to handle potentially unbounded data streams and hierarchical class structures. This classifier maintains incremental statistical descriptors to capture information about observed instances and efficiently forget older data. It also considers the entire set of classes within the hierarchy when updating these statistical descriptors, traversing from specific nodes to the root.

The incremental statistical descriptors encompass a triplet in the format $(n, \bar{x}_n, \sigma_n)$, where n represents the number of observed instances, and $\bar{x}_n$ and σ_n correspond to the incremental mean and incremental standard deviation of these instances The incremental standard deviation σ_{n_i} is computed using Equation 1 for each feature i [30, 31].

$$\sigma_{n_i} = \sqrt{\frac{(n-2)\,\sigma^2_{n-1_i} + (n-1)\,(\bar{x}_{n-1_i} - \bar{x}_{n_i})^2 + (x_{n_i} - \bar{x}_{n_i})^2}{n-1}} \tag{1}$$

To introduce adaptability into GNB-hDS, we also store an additional triplet $(c, \bar{x}_c, \sigma_c)$, representing the current statistical description of the last c instances.

Let us explore how we use these statistical descriptors to calculate weighted Gaussian probabilities. The prediction of the class for an incoming instance from the data stream involves the calculation of *a priori* probabilities based

on n or c, likelihood probabilities using Bayes' theorem, and the determination of the maximum *a posteriori* probability among all classes.

The likelihood probability for feature i and class j is calculated as shown in Equation 2:

$$p\left(x_i|\ C_j\right) = \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}}\ \exp\left\{-\frac{1}{2}\left(\frac{x_i - \bar{x}_{i,j}}{\sigma_{i,j}}\right)^2\right\} \tag{2}$$

The *a posteriori* probability, depicted in Equation 3, is obtained from the product of the independent feature probabilities for each class C [32].

$$p\left(C_j|x\right) \propto \left\{\prod_i p\left(x_i|\ C_j\right)\right\} p(C_j) \tag{3}$$

The prediction for the incoming instance is determined by selecting the class with the maximum value of the weighted Gaussian probabilities, considering both the sets $(n, \bar{x}_n, \sigma_n)$ and $(c, \bar{x}_c, \sigma_c)$ of statistical descriptors.

Thus, the method follows one of the following weighting approaches:

i. GNB-hDS-Hw (Historical window): In this approach, the historical window, i.e., the historical statistical descriptors $(n, \bar{x}_n, \sigma_n)$, are weighted by the statistical descriptors of the current window $(c, \bar{x}_c, \sigma_c)$. The arguments of the maxima for $p\left(C_j|x\right)$ (Equation 3) are computed as $pw = p_n\left(C_j|x\right) \times p_c\left(C_j|x\right)$, where p_n and p_c represent the *a posteriori* probabilities calculated using their respective sets of statistical descriptors.

ii. GNB-hDS-Cw (Current window): In this approach, only the current window, i.e., the current statistical descriptors $(c, \bar{x}_c, \sigma_c)$, are consid-

ered. The arguments of the maxima for $p\left(C_j|x\right)$ are calculated using only $p_c\left(C_j|x\right)$.

After making a prediction, the statistical descriptors are updated adaptively. It is important to note that discarding the oldest instance in the window is impossible, as it has already been removed right after processing. Instead, the methods employ a forgetting strategy by subtracting one pseudo-instance (a mean instance) from the current statistical descriptors $(c, \bar{x}_c, \sigma_c)$, representing the oldest instance within the current sliding window.

Note that within hierarchical data streams, the sliding window strategy plays a pivotal role in addressing the challenges posed by evolving class distributions and the need for adaptive learning. The sliding window allows the model to focus on recent data while discarding older instances. It ensures that the classifier remains responsive to changes in the data distribution, a critical consideration in hierarchical data streams where class distributions may evolve independently in various branches.

Figure 1 shows an illustrative example of the sliding window and the resulting adaptive learning regarding the current window aforementioned.

At each node in the hierarchy, we maintain both sets $(n, \bar{x}_n, \sigma_n)$ and $(c, \bar{x}_c, \sigma_c)$ of statistical descriptors of the last n (total, or historical) and c (current) observed instances. In the figure, $\bar{x}_n$ and $\bar{x}_c$ represent incremental mean across instances x_n. Note that the incremental mean $\bar{x}_n$ (solid underline) is based on historical window (instances encompassed by the solid border), while the incremental mean $\bar{x}_c$ (dashed underline) relies on current window (instances encompassed by the dashed border). On $\bar{x}_c$, the incremental mean reflects the last c instances by subtracting one pseudo-instance (a

mean instance) from $\bar{x}_c$ before performing the computation. Note that this strategy speeds up the convergence of the incremental mean to the ground-truth mean of the last c observed instances. Also, removing the observed oldest instance from the window is unfeasible since the model maintains only statistical descriptors and current x_n (in the figure, in black).

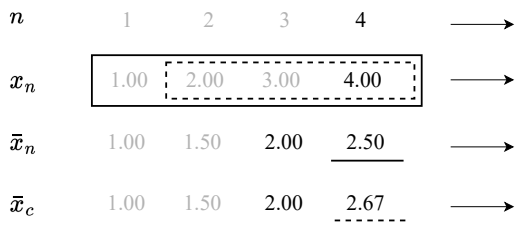

Figure 1: Illustrative example of the sliding window applied in the incremental mean computation.

Additionally, it is worth observing that the described sliding window strategy occurs at each hierarchy node. Thus, the model can heed recent data and adapt to concept drifts at different levels in the hierarchy. This hierarchical adaptation is crucial for maintaining classification accuracy, as it allows the classifier to prioritize recent data that may be more indicative of changes in specific branches of the hierarchical structure.

Algorithm 1 shows the pseudocode for the proposed Adaptive Gaussian Naive Bayes for Hierarchical Data Streams with Historical Window method (GNB-hDS-Hw). Likewise, Algorithm 2 depicts the pseudocode for the Adaptive Gaussian Naive Bayes for Hierarchical Data Streams with Current Window method (GNB-hDS-Cw). Both algorithms are similar and differ in the statistical descriptor sets to calculate the Gaussian probabilities.

Algorithm 1: GNB-hDS-Hw - Adaptive Gaussian Naive Bayes for Hierarchical Data Streams with Historical Window

input : hDS: a hierarchical data stream providing instances $(\vec{x}^t, \vec{y}^t)$ over time t
w: maximum number of instances (window size) on the current statistical descriptors

output: $\widehat{\vec{y}}^t$: a predicted label path for the input instance

```
1  Tree ← classTaxonomy(hDS);
2  foreach (x⃗^t ∈ hDS) do
3      predictedNode ← Tree.root;
4      while ¬(predictedNode.isLeaf) do
5          foreach childNode ∈ predictedNode.children do
6              priors_n ← priorProbability(childNode.Class_n);
7              priors_c ← priorProbability(childNode.Class_c);
8          end
9          likelihood_n ← likelihoodProbability(x⃗^t, priors_n);
10         posterior_n ← posteriorProbability(likelihood_n, priors_n);
11         likelihood_c ← likelihoodProbability(x⃗^t, priors_c);
12         posterior_c ← posteriorProbability(likelihood_c, priors_c);
13         weightedPosterior ← posterior_n × posterior_c;
14         predictedNode ← arg max(weightedPosterior);
15         ŷ⃗^t ← ŷ⃗^t ∪ {predictedNode.label};
16     end
17     UpdateStatisticalDescriptors(x⃗^t, y⃗^t);
18     correctNode ← Tree.y⃗^t;
19     if correctNode.data_c.count > w then
20         UpdateStatisticalDescriptors(correctNode.data_c);
21     end
22 end
```

Algorithm 2: GNB-hDS-Cw - Adaptive Gaussian Naive Bayes for Hierarchical Data Streams with Current Window

input : hDS: a hierarchical data stream providing instances $(\vec{x}^t, \vec{y}^t)$ over time t
w: maximum number of instances (window size) on the current statistical descriptors

output: $\widehat{\vec{y}}^t$: a predicted label path for the input instance

```
1  Tree ← classTaxonomy(hDS);
2  foreach (x⃗^t ∈ hDS) do
3      predictedNode ← Tree.root;
4      while ¬(predictedNode.isLeaf) do
5          foreach childNode ∈ predictedNode.children do
6              priors_c ← priorProbability(childNode.Class);
7          end
8          likelihood_c ← likelihoodProbability(x⃗^t, priors_c);
9          posterior_c ← posteriorProbability(likelihood_c, priors_c);
10         predictedNode ← arg max(posterior_c);
11         ŷ⃗^t ← ŷ⃗^t ∪ {predictedNode.label};
12     end
13     UpdateStatisticalDescriptors(x⃗^t, y⃗^t);
14     correctNode ← Tree.y⃗^t;
15     if correctNode.data_c.count > w then
16         UpdateStatisticalDescriptors(correctNode.data_c);
17     end
18 end
```

Both algorithms receive a hierarchical data stream hDS providing instances $(\vec{x}^t, \vec{y}^t)$ over time t, and the w (window size) parameter representing the maximum number of instances to be considered on the current statistical descriptors $(c, \bar{x}_c, \sigma_c)$.

At any time, if required, the algorithms can output a set of predicted class

labels (a hierarchical label path) $\widehat{\vec{y}}_t$ for each given instance $(\vec{x}^t, \vec{y}^t)$, where $\vec{x}^t$ represents a d-dimensional feature set and its values, and $\vec{y}_t$ represents the corresponding ground-truth set of hierarchical classes of that instance.

The algorithms start by representing the class hierarchy from the hierarchical data stream hDS (line 1). On both algorithms, the first loop (line 2 onwards) receives an arriving instance from the hierarchical data stream, and the following loop (line 4 onwards) handles the hierarchy using an LCPN approach by predicting one of the children labels possible for a current parent node.

The *a priori* probabilities are calculated using the counting of class instances, and the likelihood and *a posteriori* probabilities are calculated by the application of Equations 2 and 3, respectively. Note that each algorithm uses different statistical descriptor sets to calculate the probabilities. On GNB-hDS-Hw, historical and current stats are used to calculate prior (lines 6 and 7) and likelihood and posterior probabilities (lines 9-12). On GNB-hDS-Cw, only current stats are used (lines 6, 8, and 9).

On GNB-hDS-Hw, the historical posterior probabilities are weighted by current ones (line 13). Note that this step is unneeded on GNB-hDS-Cw since the model only uses the current posterior probabilities. The predicted node for the evaluated parent node is then obtained via the arguments of the maxima for $p\left(C_j|x\right)$ (line 14 on GNB-hDS-Hw, and line 10 on GNB-hDS-Cw), and the respective single label is appended to a partial label path $\widehat{\vec{y}}_t$. This process is repeated until a leaf node is reached and the label path $\widehat{\vec{y}}_t$ is complete and ready to be output by the algorithm.

Finally, the algorithms update the statistical descriptors from the leaf to

the root class. After that, both methods test whether the number of instances represented in the current statistical descriptors $(c, \bar{x}_c\sigma_c)$ $(data_c)$ exceeds the stipulated maximum number w allowed. If so, the sliding window strategy is applied by forgetting a representation of the oldest instance of $(c, \bar{x}_c\sigma_c)$.

Finally, in the context of our proposed adaptation, which employs window-weighted Gaussian probabilities, it is essential to address the assumption of Gaussian data distribution. While our approach utilizes Gaussian probabilities as weights in a Gaussian Naive Bayes model, it may encounter limitations when dealing with data that deviates significantly from Gaussian distribution, such as data with annular shapes or one-dimensional signals.

To address this concern, we experimented with the previously described Incremental Yeo-Johnson transformation as part of our testing process. This transformation aims to mitigate the non-Gaussian nature of the data by applying incremental adjustments and can be attached to the proposed GNB-hDS-Hw and GNB-hDS-Cw. While it may not entirely resolve the non-Gaussian characteristics, the Incremental Yeo-Johnson transformation enhances the adaptability of our approach to a broader range of diverse data distributions. In subsequent sections, we describe and discuss our experiments and the impact of the Incremental Yeo-Johnson transformation on the data conformity to Gaussian assumptions.

5. Evaluation Protocol

In this section, we propose a unified assessment protocol to compare hierarchical data stream classifiers using different evaluation metrics as criteria in a Multi-Criteria Decision-Making analysis, resulting in a reproducible testbed.

Besides, we instantiate the protocol and describe the experimental setup to assess the proposed methods against the previously described state-of-the-art techniques.

5.1. Unified protocol

The literature review presented in [4] revealed no baseline for a testbed creation in the hierarchical data stream classification area regarding hierarchical data stream sets or evaluation protocols.

Overall, all the retrieved studies in the literature review use the previously described Hierarchical Precision (hP), Recall (hR), and F-Measure (hF) evaluation metrics. Note that, except for studies in specific areas, these metrics are widely applied in the evaluation of hierarchical classification learning models [1]. Furthermore, the standard protocol for evaluating data stream learning models, regardless of the metric used, is the prequential evaluation method [16]. Besides, the review also pointed out the need for metrics for measuring the computational performance of the learning methods.

In this sense, the authors in [25, 28, 26, 27] use the number of instances per second ($inst/s$) that a learning model can process and classify per second under the prequential protocol, and at the same experimental setup when performing comparisons with other learning models.

In this section, we propose a unified protocol to fill this gap by merging the fundamental aspects of the evaluation strategies used in recent hierarchical data stream studies. Figure 2 illustrates the proposed unified assessment protocol. The main steps of the protocol are represented in the boxes, where dashed ellipses describe user-defined instantiations related to that step, and dashed arrows and notes depict possible outputs.

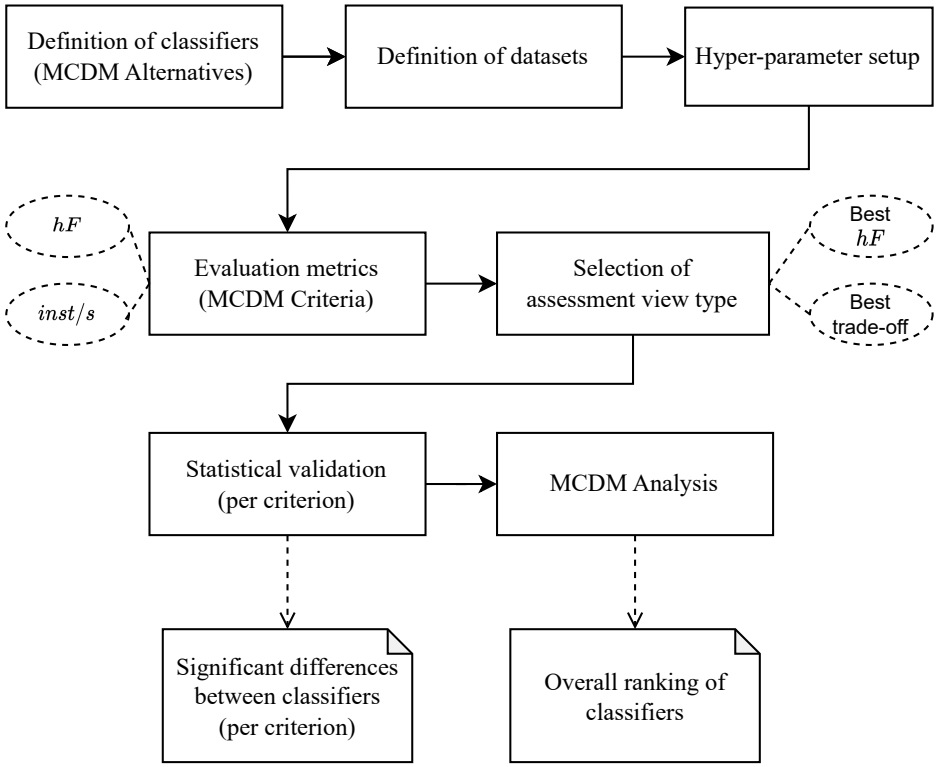

Figure 2: Steps performed on the proposed unified assessment protocol.

The first three steps of the protocol are commonly applied in evaluation protocols in classification tasks and involve the definition of tested classifiers (including state-of-the-art techniques), the definition of datasets with different features, instances, and domains, thus allowing a reasonable assessment of the behavior of the classifiers in different scenarios, and the configuration of classifiers' parameters comprehending at least a moderate range of values.

The next step regards the definition of the evaluation metrics obtained by the classifiers to be measured during the experiments. Here, we suggest the previously reported hF and $inst/s$ metrics.

Furthermore, we highlight two key points to be considered in this step.

First, the evaluation metrics should comprise prediction correctness and computational performance in a balanced way, preferably equally. Note that the chosen metrics are used as Multi-Criteria Decision-Making (MCDM) criteria afterward and will be discussed in detail below. Thus, having more metrics related to one type of evaluation (prediction correctness or computational performance) can unbalance the analysis for one measure at the expense of the other one(s). Second, it is important to observe that each evaluation metric has intrinsic usage requirements. For instance, when considering imbalanced datasets, the hierarchical F-Measure metric is more representative than the hierarchical Precision. Similarly, the $inst/s$ metric must be evaluated only internally to a set of experiments since it depends on the environment in which the experiment is performed.

Next, the chosen evaluation metrics are observed in one or more assessment view types, depending on the problem characteristics and requirements. Here, we suggest two assessment view types, i.e., a best average performance regarding prediction correctness and a trade-off performance concerned with computational performance. The first view represents the common target of the classification task and intends to obtain the most accurate model possible regarding the input data. The second view adds feasibility to the first by considering the computational resources usage and environment limitations, such as hardware limitations in real-world applications, which may be prohibitive in streaming scenarios as discussed in previous work [4, 33].

At that point, the assessment view types represent value sets obtained on each evaluation metric by each classifier on all datasets. It is not uncommon for studies with proposals of novel classifiers to stop the evaluation process at

this stage and draw conclusions based only on local differences in an isolated view of the result matrices. However, it is critical to highlight the need to verify statistical differences between classifiers [34].

Therefore, the statistical validation step understands the results obtained by all methods as sample sets used as a basis for statistical tests. These sample sets comprehend ordinal, non-parametric data, representing the evaluation metrics obtained by the classifiers in different hierarchical data stream sets, assuming the null hypothesis that there is no significant difference between the results of the classifiers. Observe that statistical validation, claims, or conclusions occur for each criterion. This step outputs significant differences between classifiers for each selected criterion isolated [34].

Finally, in the last step of the protocol, a separate analysis may be conducted to compare all classifiers together. In addition to the above-mentioned statistical validation tests, this analysis compares the methods using the evaluation metrics (i.e., hF and $inst/s$ rates) understood as multiple attributes in a problem of Multi-Criteria Decision-Making. Multi-Criteria Decision-Making (MCDM) is a problem-solving technique fitted to deal with multiple conflicting objectives. It was first introduced in [35] and used to develop multiple criteria metrics for the evaluation of data mining algorithms in [36].

The MCDM analyzes distinct alternatives representing possible solutions to a problem. These alternatives are assumed to be finite and eventually ranked. The problem is associated with multiple decision criteria representing the dimensions from which the alternatives can be evaluated. Also, the criteria are usually conflicting and depicted via incommensurable units [37].

Formally, one can express an MCDM problem as a finite set $A = a_i, i =$

$\{1,\ 2,\ 3, \ldots, n\}$ of alternatives and a finite set $G = g_j, j = \{1,\ 2,\ 3, \ldots, m\}$ of goals or criteria, and each goal has an associated weight W (or desirability). The solution determines the optimal alternative a that maximizes the degree of desirability concerning all goals g_j [37].

The analysis performed in our protocol understands the classifiers as the alternatives in the MCDM problem and the goals, or multiple decision criteria, as the evaluation metrics (e.g., hF and $inst/s$).

Specifically, the MCDM is applied through a weighted product model (WPM) where each alternative (classifier) is compared against other alternatives by multiplying ratios for each weighted criterion (hF and $inst/s$ rates). When used to compare or rank multiple alternatives, a variant approach of the WPM that uses only products without ratios can be applied. Equation 4 shows the MCDM-WPM calculation, where n is the number of criteria, a_{ij} is the metric value obtained by the $i\text{-}th$ classifier concerning the $j\text{-}th$ criterion, and w_j is the weight of $j\text{-}th$ criterion [37].

$$\mathrm{WPM}\,(A_i) = \prod_{j=1}^{n} \left(a_{ij}\right)^{w_j} \tag{4}$$

Note that the evaluation metrics should be normalized to avoid impacts of different value scales. Here, we suggest the linear scale transformation (Max-Min method), considering both maximum and minimum performances of evaluation metrics. The normalized value r_{ij} obtained by the linear scale transformation is calculated via Equation 5, where a_{ij} is the metric value obtained by the $i\text{-}th$ classifier concerning the $j\text{-}th$ criterion, and $a_j^{\max}$ and $a_j^{\min}$ are the maximum and the minimum performance rate obtained by the classifiers concerning the $j\text{-}th$ criterion [38].

$$r_{ij} = \frac{a_{ij} - a_j^{\min}}{a_j^{\max} - a_j^{\min}} \tag{5}$$

Besides, a constant equal to 10^{-n}, with $n \in \mathbb{N}$ can be added to r_{ij} to avoid zero multiplication (which would result in ignoring the *j-th* criterion) when $a_{ij} = a_j^{\min}$.

Finally, the criteria weights can assume complementary values in distinct scenarios representing different importance given to prediction correctness and speed performance. As a final output of the protocol, this step results in an overall ranking of the tested classifiers regarding all evaluation metrics together.

5.2. Experimental Setup

We now instantiate the above-described protocol and detail the experimental setup to assess the proposed GNB-hDS-Hw and GNB-hDS-Cw methods against the previously described state-of-the-art techniques.

The literature review in [4] listed the datasets used in studies in the hierarchical data stream classification area. However, most studies were false positives concerning the target area when considering the area constraints. As an outcome, the datasets used by the studies returned in the review do not necessarily represent hierarchical data stream sets [4]. Therefore, we limited our analysis to datasets with complete adherence to the hierarchical data stream classification area, resulting in three hierarchical data streams proposed in [24].

In addition, as previously introduced in Section 3, the authors in [3] introduced a new version of one of the datasets used in [24], with more instances and a formal definition of the concept drifts within the data stream.

While the authors initially presented this dataset as flat, it retained its inherent hierarchical structure, as originally proposed in [24], given that the instances still correspond to species of disease vector mosquitoes, which naturally adhere to an entomological taxonomy [26].

Table 3 depicts the resulting 14 hierarchical data stream sets used in the experiments, comprehending the three data sets proposed in [24] plus the 11 hierarchically labeled datasets resulting from the hierarchy incorporation on the Insects datasets proposed in [3].

Table 3: Description of hierarchical data stream sets used in the experiments.

Dataset	**Instances**	**Features**	**Classes**	**Labels per level**
Entomology [24]	21,722	33	14	4,6,9,14
Ichthyology [24]	22,444	15	15	2,6,10,15
Insects-a-b [3]	52,848	33	6	1,1,2,6
Insects-a-i [3]	355,275	33	6	1,1,2,6
Insects-i-a-r-b [3]	79,986	33	6	1,1,2,6
Insects-i-a-r-i [3]	452,044	33	6	1,1,2,6
Insects-i-b [3]	57,018	33	6	1,1,2,6
Insects-i-g-b [3]	24,15	33	6	1,1,2,6
Insects-i-g-i [3]	143,323	33	6	1,1,2,6
Insects-i-i [3]	452,044	33	6	1,1,2,6
Insects-i-r-b [3]	79,986	33	6	1,1,2,6
Insects-i-r-i [3]	452,044	33	6	1,1,2,6
Insects-o-o-c [3]	905,145	33	24	4,10,14,24
Instruments [24]	9,419	30	31	5,10,31

Note that, among the datasets proposed in [3], the "Insects-o-o-c" represents the main dataset, while other ones represent datasets built with different sampling strategies to simulate natural effects in insects' behavior and

induce distinct concept drifts on data. Also, in the same datasets, the words "Abrupt", "Balanced", "Gradual", "Imbalanced", "Incremental" and "Reoccurring" are reduced to their initials to increase readability.

The classifiers used in the experiments comprehend the methods proposed here and the state-of-the-art techniques described in Section 3. Table 4 lists these methods and summarizes the parameters used across all experiments. Note that all the classifiers were experimented with identical parameters when applicable. The comparison of Local and Global kNN-hDS used identical n and k parameters. Then, the comparison between the Local kNN-hDS and kNC-hDS and Dribble-hDS also used n and k, in addition to m used on both kNC-hDS and Dribble-hDS methods to perform the summarization strategy. Differently, the proposed adaptive GNB-hDS-Cw and GNB-hDS-Hw use a w parameter as the window size to perform a forgetting strategy in the statistical descriptors unrelated to the other methods.

Table 4: Hyper-parameter settings used on methods across experiments.

Method	**Parameter**	**Description**	**Experimented values**
Local kNN-hDS	n	Maximum number of instances to be stored in a node	$\{1, 5, 10, 15, 20\}$
	k	Nearest neighbors	$\{1, 3, 5\}$
Global kNN-hDS	n	Maximum number of instances to be stored in a node	$\{1, 5, 10, 15, 20\}$
	k	Nearest neighbors	$\{1, 3, 5\}$
kNC-hDS (Local and Global)	n	Number of centroids	$\{1, 5, 10, 15, 20\}$
	m	Maximum number of instances summarized in a centroid	$\{5, 10, 30\}$
	k	Nearest centroids	$\{1, 3, 5\}$
Dribble-hDS (Local and Global)	n	Number of CFs	$\{1, 5, 10, 15, 20\}$
	m	Maximum number of instances summarized in a CF	$\{5, 10, 30\}$
	k	Nearest CF means	$\{1, 3, 5\}$
GNB-hDS	-	-	-
GNB-hDS-Hw	w	Maximum number of instances summarized in the historical window	$\{10, 50, 100, 500, 1000, 5000\}$
GNB-hDS-Cw	w	Maximum number of instances summarized in the current window	$\{10, 50, 100, 500, 1000, 5000\}$

In addition, note that we also tested the original GNB-hDS method and

our two adaptive proposals with the previously described Incremental Yeo-Johnson Power Transformation [29] as an attached preprocessing step, resulting in additional variants specified in Table 5.

Table 5: GNB-hDS variants.

Method	**Data stream handling**	**Window strategy**	**Incremental Yeo-Johnson**
GNB-hDS	Incremental	No	No
GNB-hDS-iYJ	Incremental	No	Yes
GNB-hDS-Cw	Adaptive	Cw (only current window is considered)	No
GNB-hDS-Cw-iYJ	Adaptive	Cw (only current window is considered)	Yes
GNB-hDS-Hw	Adaptive	Hw (historical window weigthed by current window)	No
GNB-hDS-Hw-iYJ	Adaptive	Hw (historical window weigthed by current window)	Yes

All the experiments related to the Incremental Yeo-Johnson Power Transformation were performed using $N = 10^7$, $d = (0.95,\ 0.02)$, $p = 0.5$, $\alpha = \ 0.05$ and $\vec{l} = 1$, following the same protocol provided in [29].

Related to the evaluation metrics to be observed during the experiments, the predictive correctness of the classifiers was measured using the hierarchical F-Measure (hF) following a prequential evaluation method (interleaved test-then-train). The hF rates were computed and incrementally averaged for all incoming instances from the hierarchical data stream. The computational performance of all methods was measured by the previously described $inst/s$ metric by calculating the number of instances that each method can process and classify per second.

We retrieved the results obtained by the classifiers based on the two assessment view types previously introduced, i.e., the best average performance of all methods regarding prediction correctness (hF) and the best trade-off

performance, which considered the best trade-off between hF and $inst/s$ rates obtained by all methods separately resulting from an MCDM-WPM analysis.

To statistically validate the results, each assessment view type was submitted to significance tests of multiple comparisons considering a 95% confidence level according to the protocol provided in [34]. More specifically, the Friedman's hypothesis test [39] was used to make multiple comparisons in non-parametric data, assuming a null hypothesis that there is no significant difference between the results of all methods in terms of predictive (hF and performance ($inst/s$) rates. In the case of the null hypothesis being rejected, the Nemenyi *post-hoc* test [40] was applied to identify significant differences between two specific classifiers.

Finally, regarding the MCDM analysis, the weights of the hF and $inst/s$ rates were defined following ratios $w \in \left\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\right\}$, with hF and $inst/s$ assuming complementary values on five distinct scenarios representing different importance given to prediction correctness and speed performance.

6. Analysis

This subsection compares the proposed methods and provides a benchmark of the hierarchical data stream classification area.

We compared both GNB-hDS-CW and GNB-hDS-HW methods against the state-of-the-art related work. The following sections depict two analyses based on the best average performances regarding prediction correctness (hF) and the best trade-off performances of all methods. Next, Section 6.3 discusses both proposed methods regarding their adaptive learning strate-

gies. Finally, Section 6.4 portrays an overall analysis considering all methods together.

6.1. Best hF performance

Table 6 depicts the hierarchical F-Measure obtained by all methods in the hierarchical data stream sets (greater values highlighted in bold). These results represent the hF rates obtained by methods considering their best average performances regarding prediction correctness.

Table 6: Hierarchical F-Measure rates obtained by methods in best average performances regarding prediction correctness.

	Local kNN-hDS	**Global kNN-hDS**	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**	**GNB-hDS**	**GNB-hDS-iYJ**	**GNB-hDS-Cw**	**GNB-hDS-Cw-iYJ**	**GNB-hDS-Hw**	**GNB-hDS-Hw-iYJ**
	$n = 20$	$n = 20$	$n = 20$	$n = 5$	$n = 20$	$n = 5$						
			$m = 10$	$m = 30$	$m = 10$	$m = 30$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Datasets	$k = 1$	$k = 1$	$k = 3$	$k = 1$	$k = 3$	$k = 1$						
Entomology	51.51	51.48	57.38	53.61	**57.41**	53.71	48.63	52.82	50.08	50.02	50.41	51.59
Ichthyology	40.55	40.54	41.52	37.00	41.72	37.11	46.82	**49.72**	46.64	47.15	47.39	48.80
Insects-a-b	80.95	80.95	84.37	83.33	84.37	83.33	81.11	81.96	86.49	**86.97**	86.10	86.81
Insects-a-i	79.14	79.14	82.62	82.55	82.60	82.55	80.88	84.03	85.46	86.17	85.53	**86.90**
Insects-i-a-r-b	79.49	79.52	84.30	83.42	84.28	83.42	81.42	84.07	85.84	85.94	86.21	**86.59**
Insects-i-a-r-i	78.52	78.53	82.64	82.11	82.62	82.11	81.57	83.70	84.93	85.51	85.00	**86.22**
Insects-i-b	79.78	79.78	84.05	83.91	84.03	83.91	80.55	82.40	83.83	84.05	83.65	**84.44**
Insects-i-g-b	83.29	83.41	**88.02**	86.66	87.99	86.66	81.53	81.50	86.16	86.14	85.64	86.03
Insects-i-g-i	78.94	78.95	82.91	83.11	82.93	83.11	80.40	83.38	**86.93**	85.42	85.23	86.00
Insects-i-i	78.63	78.64	82.58	83.08	82.57	83.08	80.90	83.18	84.97	85.50	84.93	**86.14**
Insects-i-r-b	80.14	80.18	84.50	83.48	84.50	83.48	78.57	80.21	85.79	86.08	85.64	**86.18**
Insects-i-r-i	78.60	78.61	82.63	82.70	82.62	82.70	81.61	83.72	84.88	85.51	84.99	**86.25**
Insects-o-o-c	55.24	55.28	65.66	59.50	65.56	59.50	64.14	**69.38**	59.33	62.40	61.47	66.54
Instruments	**65.42**	65.06	55.04	56.53	55.59	56.68	48.31	49.48	40.06	36.91	45.93	42.36
Avg. *hF*	72.16	72.15	75.59	74.36	75.63	74.38	72.60	74.97	75.10	75.27	75.58	**76.49**
Avg. Ranking	10.32	9.96	5.39	6.96	5.71	6.75	9.50	6.07	5.36	4.18	5.07	**2.71**

The Local kNN-hDS and Global kNN-hDS methods obtained the lowest average hF rates and the lowest average rankings among all methods (10.32

and 9.96, respectively). Next, the GNB-hDS method achieved a ranking of 9.50. Most other methods obtained similar rankings between the fourth and seventh rankings. As an exception and highlight, the proposed adaptive GNB-hDS-Hw-iYJ method achieved the best average hF rate (76.49%) and the best average ranking (2.71), reaching the best overall results in 7 out of 14 hierarchical data streams.

The overall hF results were submitted to a Friedman test, which identified a significant difference between the methods (p-value $= 7.15 \times 10^{-10}$). Thus, a *post-hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 3 shows the resulting critical difference chart for the hF rates obtained by all methods. The four adaptive variants of GNB-hDS are significantly different from the Local and Global kNN-hDS methods. Besides, both adaptive GNB-hDS-Hw-iYJ and GNB-hDS-Cw-iYJ methods that use the incremental Yeo-Johnson Power Transformation are also significantly different from the primary incremental GNB-hDS.

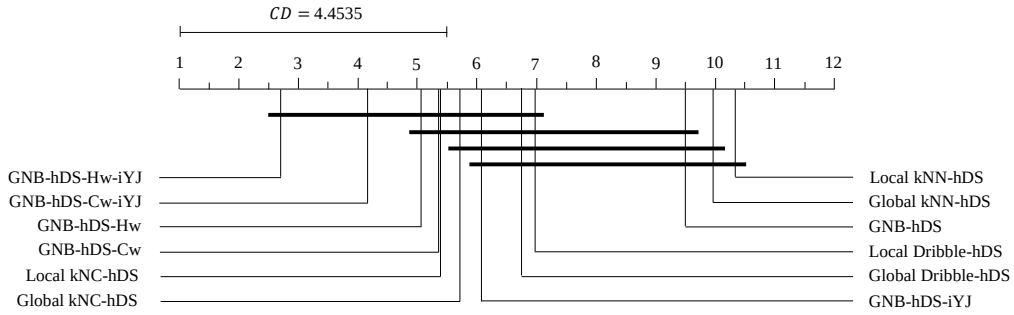

Figure 3: Critical differences chart for hF rates obtained by methods in best average performances regarding prediction correctness.

In addition to the overall results regarding hF rates, Table 7 depicts the overall instance per second ($inst/s$) rates obtained by all proposed methods. Likewise, these results represent the $inst/s$ rates obtained by methods considering their best average performances regarding prediction correctness.

Table 7: Instances per second rates obtained by methods in best average performances regarding prediction correctness.

	Local kNN-hDS	**Global kNN-hDS**	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**	**GNB-hDS**	**GNB-hDS-iYJ**	**GNB-hDS-Cw**	**GNB-hDS-Cw-iYJ**	**GNB-hDS-Hw**	**GNB-hDS-Hw-iYJ**
	$n = 20$	$n = 20$	$n = 20$	$n = 5$	$n = 20$	$n = 5$						
			$m = 10$	$m = 30$	$m = 10$	$m = 30$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Datasets	$k = 1$	$k = 1$	$k = 3$	$k = 1$	$k = 3$	$k = 1$						
Entomology	127	209	134	327	200	382	380	395	**397**	370	335	287
Ichthyology	157	223	186	286	291	380	395	**426**	398	394	366	338
Insects-a-b	151	383	152	452	353	**543**	490	472	469	437	374	325
Insects-a-i	153	384	151	467	354	**542**	495	473	472	444	378	325
Insects-i-a-r-b	153	386	152	456	354	**541**	496	467	475	441	374	320
Insects-i-a-r-i	153	374	150	468	351	**542**	492	465	476	442	376	323
Insects-i-b	148	363	152	456	345	**541**	501	503	500	454	402	334
Insects-i-g-b	158	385	165	459	372	**542**	483	469	466	441	371	323
Insects-i-g-i	154	385	155	466	359	**543**	496	470	475	444	377	326
Insects-i-i	152	377	151	467	354	**545**	497	479	473	445	385	323
Insects-i-r-b	153	388	152	458	356	**543**	491	459	474	439	363	324
Insects-i-r-i	153	386	151	469	353	**540**	494	451	476	443	377	323
Insects-o-o-c	75	135	78	247	127	275	**283**	277	280	271	237	212
Instruments	79	110	121	221	164	256	263	**284**	257	263	259	217
Avg. *inst/s*	140.43	320.50	146.54	407.00	309.40	**479.60**	446.80	434.95	434.77	409.01	355.35	307.20
Avg. Ranking	11.57	8.00	11.36	5.50	9.00	**2.00**	2.29	3.07	3.21	5.43	7.21	9.36

Local kNN and kNC methods obtained the lowest average $inst/s$ rates and the lowest average rankings among all methods (11.57 and 11.36, respectively). Next, the GNB-hDS-Hw-iYJ method achieved a ranking of 9.36. On the other ranking side, GNB-hDS-iYJ and GNB-hDS-Cw obtained similar rankings of 3.21 and 3.07. Also, GNB-hDS and Global Dribble-hDS achieved similar rankings (2.29 and 2.00, respectively), with Global Dribble-hDS in

first place in 10 out of 14 hierarchical data streams.

The overall $inst/s$ results were submitted to a Friedman test, which identified a significant difference between the methods (p-value $= 2.15 \times 10^{-24}$). Thus, a *post-hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4 shows the resulting critical difference chart for the $inst/s$ rates obtained by all methods. Both Dribble methods plus four variants of GNB-hDS (incremental ones and both adaptive with current window, Cw) are significantly different from the Local kNN-hDS method. Global Dribble-hDS, GNB-hDS, GNB-hDS-iYJ, and GNB-hDS-Cw are also significantly different from both kNN, both KNC and GNB-hDS-Hw-iYJ methods.

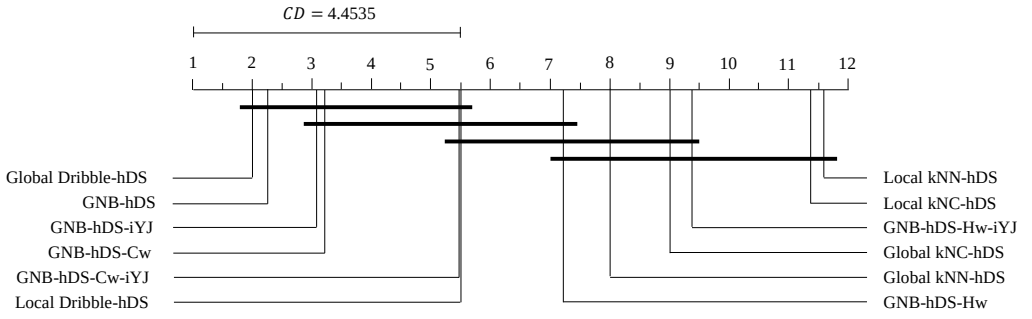

Figure 4: Critical differences chart for $inst/s$ rates obtained by methods in best average performances regarding prediction correctness.

Considering both hF and $inst/s$ rates together, one can observe that some methods obtained slightly opposite rankings in both analyses. For instance, GNB-hDS-Hw-iYJ achieved first place (ranking of 2.71) regarding hF rates but third to last place on $inst/s$ (ranking of 9.36). The same can be said about both kNC methods, with competitive hF rates but smaller

$inst/s$ rates.

Note that both Dribble methods take advantage of using a smaller data representation ($n = 5$) to obtain their average best performance. It ensured first place for Global Dribble-hDS among the $inst/s$ performances and still a competitive hF rate. A broader analysis regarding the averaged best performances is shown later in this section, with the Multi-Criteria Decision-Making (MCDM) technique.

6.2. Best trade-off performance

As described before, the MCDM analysis was applied to all classifier performances in an isolated view to retrieve the best trade-off performances of all methods considering both hF and $inst/s$ rates concomitantly.

The hF and $inst/s$ rates obtained by a method with each hyper-parameter setup were compared among each other concerning five different assignments of importance (weights, w) following the complementary percentage set $w \in \left\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\right\}$. In other words, each result set obtained with a specific hyper-parameter setup was evaluated with the MCDM-WPM analysis using complementary weighted hF and $inst/s$ rates (w_{hF} and $w_{inst/s}$) resulting in a WPM performance. Next, the WPM performances were ranked among each other to obtain the best result set for each pair ($w_{hF},\ w_{inst/s}$). Lastly, all sets of WPM rankings were averaged, and the best overall average was understood as the best trade-off performance of that method.

Table 8 depicts the hierarchical F-Measure obtained by all methods in the hierarchical data stream sets considering their best trade-off performance retrieved by the MCDM-WPM analysis (greater values per dataset are highlighted in bold). Note that the table also depicts the hyper-parameter setup

used by each method to achieve its best trade-off performance.

Table 8: Hierarchical F-Measure (%) rates obtained by methods on best trade-off performance.

	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
	$n = 5$	$n = 5$	$n = 5$	$n = 5$	$n = 5$	$n = 5$						
			$m = 30$	$m = 30$	$m = 30$	$m = 30$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Datasets	$k = 1$	$k = 1$	$k = 3$	$k = 1$	$k = 3$	$k = 1$						
Entomology	46.79	46.79	55.68	53.61	**55.71**	53.71	48.63	52.82	50.08	50.02	50.41	51.59
Ichthyology	36.66	36.66	39.44	37.00	39.53	37.11	46.82	**49.72**	46.64	47.15	47.39	48.80
Insects-a-b	77.63	77.64	84.05	83.33	84.07	83.33	81.11	81.96	86.49	**86.97**	86.10	86.81
Insects-a-i	75.32	75.32	82.29	82.55	82.30	82.55	80.88	84.03	85.46	86.17	85.53	**86.90**
Insects-i-a-r-b	76.89	76.96	84.00	83.42	83.96	83.42	81.42	84.07	85.84	85.94	86.21	**86.59**
Insects-i-a-r-i	74.98	75.00	82.31	82.11	82.29	82.11	81.57	83.70	84.93	85.51	85.00	**86.22**
Insects-i-b	76.43	76.42	83.62	83.91	83.62	83.91	80.55	82.40	83.83	84.05	83.65	**84.44**
Insects-i-g-b	79.97	80.12	87.61	86.66	**87.62**	86.66	81.53	81.50	86.16	86.14	85.64	86.03
Insects-i-g-i	75.62	75.65	82.22	83.11	82.20	83.11	80.40	83.38	**86.93**	85.42	85.23	86.00
Insects-i-i	75.01	75.03	82.40	83.08	82.40	83.08	80.90	83.18	84.97	85.50	84.93	**86.14**
Insects-i-r-b	77.33	77.42	84.33	83.48	84.29	83.48	78.57	80.21	85.79	86.08	85.64	**86.18**
Insects-i-r-i	75.05	75.06	82.31	82.70	82.31	82.70	81.61	83.72	84.88	85.51	84.99	**86.25**
Insects-o-o-c	49.99	50.21	65.00	59.50	64.90	59.50	64.14	**69.38**	59.33	62.40	61.47	66.54
Instruments	50.60	50.48	50.29	56.53	50.68	**56.68**	48.31	49.48	40.06	36.91	45.93	42.36
Avg. *hF*	67.73	67.77	74.68	74.36	74.71	74.38	72.60	74.97	75.10	75.27	75.58	**76.49**
Avg. Ranking	11.21	10.79	6.14	6.43	5.86	6.14	9.07	5.79	5.07	4.00	4.79	**2.71**

All kNN, kNC, and Dribble methods obtained their best trade-off performances with $n = 5$ probably because this hyper-parameter setup is the fastest possible with a reasonable data representation. Also, note that the average best performances and the best trade-off performances of all GNB-hDS methods are the same since the w parameter does not affect the processing speed.

All proposed methods (GNB-hDS-Hw-iYJ, GNB-hDS-Cw-iYJ, GNB-hDS-Hw, and GNB-hDS-Cw) obtained the best average hF rates and rankings regarding the trade-off performances. Next, kNC and Dribble methods ob-

tained similar average rates. The incremental GNB-hDS achieved third to last place, and both kNN obtained the lowest average rate among all methods, with the Local kNN-hDS method in the last position.

Both sides of the ranking (first and last positions) on both averaged best performances and best trade-off performances are the same. These rankings occur since the methods with the lowest trade-off performance rates could not obtain enough increases in hF rates even with more data representations. Also, the primary version of GNB-hDS does not differ significantly from both kNN regarding hF rates, but it does regarding $inst/s$ rates.

Following the same analysis protocol performed with the averaged best performances, the overall hF results of the trade-off performances were submitted to a Friedman test. As expected, it identified a significant difference between the methods (p-value $= 1.49 \times 10^{-12}$). Thus, a *post-hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 5 shows the resulting critical difference chart for the hF rates obtained by all methods in the trade-off performances.

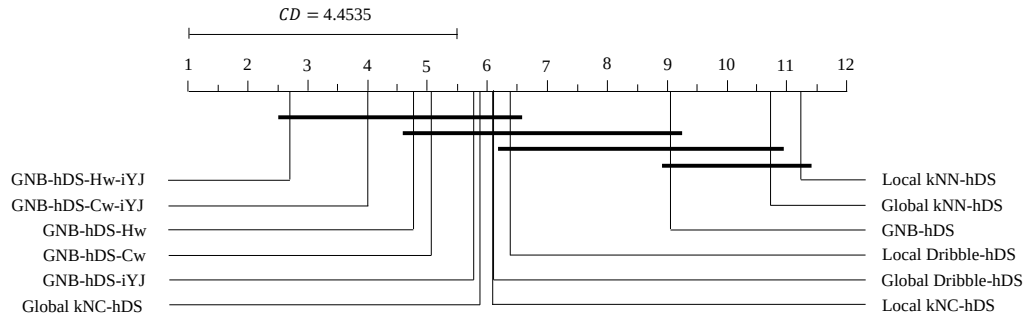

Figure 5: Critical differences chart for hF rates obtained by methods with on the best trade-off performance.

All proposed methods achieved significantly higher rates compared to the kNN methods. Also, the incremental GNB-hDS significantly differs (with lower rates) from its adaptive counterparts that use the incremental Yeo-Johnson Power Transformation.

Regarding speed comparison, Table 9 depicts the overall $inst/s$ rates obtained by all proposed methods on their best trade-off performances.

Table 9: Instances per second rates obtained by methods on best trade-off performance.

	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
	$n = 5$	$n = 5$	$n = 5$	$n = 5$	$n = 5$	$n = 5$						
			$m = 30$	$m = 30$	$m = 30$	$m = 30$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Datasets	$k = 1$	$k = 1$	$k = 3$	$k = 1$	$k = 3$	$k = 1$						
Entomology	283	370	265	327	335	382	380	395	**397**	370	335	287
Ichthyology	294	363	309	286	384	380	395	**426**	398	394	366	338
Insects-a-b	354	**568**	355	452	534	543	490	472	469	437	374	325
Insects-a-i	357	**570**	360	467	534	542	495	473	472	444	378	325
Insects-i-a-r-b	340	**583**	360	456	535	541	496	467	475	441	374	320
Insects-i-a-r-i	345	**582**	359	468	535	542	492	465	476	442	376	323
Insects-i-b	365	**584**	297	456	527	541	501	503	500	454	402	334
Insects-i-g-b	354	**579**	366	459	540	542	483	469	466	441	371	323
Insects-i-g-i	348	**581**	364	466	536	543	496	470	475	444	377	326
Insects-i-i	358	448	340	467	536	**544**	497	479	473	445	385	323
Insects-i-r-b	344	**576**	360	458	537	543	491	459	474	439	363	324
Insects-i-r-i	343	**581**	359	469	536	540	494	451	476	443	377	323
Insects-o-o-c	170	196	179	247	231	275	**283**	277	280	271	237	212
Instruments	178	215	187	221	230	256	263	**284**	257	263	259	217
Avg. *hF*	316.64	**485.54**	318.60	407.00	466.46	479.60	446.80	434.95	434.77	409.01	355.35	307.20
Avg. Ranking	11.00	3.50	10.50	7.29	4.07	**2.71**	3.64	4.43	4.57	6.93	8.21	11.14

Global kNN-hDS obtained the best average $inst/s$ rate (485.54), while Global Dribble-hDS obtained the best average ranking. Global kNN-hDS and GNB-hDS achieved close rankings, as well as Global kNC-hDS, GNB-hDS-iYJ and GNB-hDS-Cw. The slower performances resulted from local kNN and kNC methods and the GNB-hDS-Hw-iYJ.

The $inst/s$ rates of the trade-off performances were also submitted to a Friedman test. As with the hF rates, the Friedman test identified a significant difference between the methods (p-value $= 1.40 \times 10^{-19}$). Figure 6 shows the resulting critical difference chart for the $inst/s$ rates obtained by all methods in the trade-off performances after a *post-hoc* Nemenyi test.

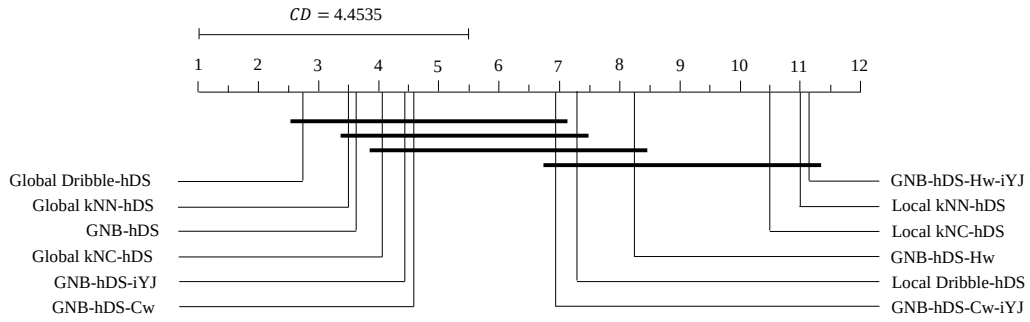

Figure 6: Critical differences chart for $inst/s$ rates obtained by methods on the best trade-off performance.

All global methods and both incremental GNB-hDS and GNB-hDS-Cw obtained significantly faster $inst/s$ rates than the related work Local kNN-hDS (and than Local kNN-hDS and GNB-hDS-Hw-iYJ). In the first and second places, Global Dribble-hDS and Global kNN-hDS significantly differ from their local counterparts and from both adaptive GNB-hDS variants with historical windows.

Note that the primary incremental version of GNB-hDS performs fewer steps than the proposed adaptive variants, resulting in a higher $inst/s$. In this sense, the incremental Yeo-Johson and the proposed forgetting strategy used on the adaptive GNB-hDS-Cw and GNB-hDS-HW constitute additional steps to the learning process and impact computational performance.

Thus, one can understand that GNB-hDS-iYJ and GNB-hDS-Cw perform one additional step than GNB-hDS; also, GNB-hDS-Cw-iYJ and GNB-hDS-Hw perform two additional steps; and, finally, GNB-hDS-Hw-iYJ performs three additional steps since the application of the historical window results

in the storage and processing of two distinct sets of statistical descriptors.

Besides, these additional steps represent opposite results in the hF and $inst/s$ rates. The methods that perform more additional steps achieved the highest hF rates and, consequently, have lower $inst/s$ rates. Likewise, the GNB-hDS method obtained the lowest average hF rate and the highest number of instances processed per second.

6.3. Adaptive Learning Analysis

To overview the impact of the window-weighted Gaussian probabilities on the prediction correctness of the classifiers, we measured the prequential hF rate over time along the data stream of both adaptive GNB-hDS-Hw and GNB-hDS-CW against the incremental GNB-hDS classifier.

Figure 7 compares GNB-hDS-Hw and GNB-hDS. Likewise, Figure 8 shows the comparison between GNB-hDS-Cw and GNB-hDS. The figures depict plots of the cumulative prequential hF rates obtained by the classifiers on each hierarchical data stream set using the same parameter settings of the results considering the best average performances regarding prediction correctness.

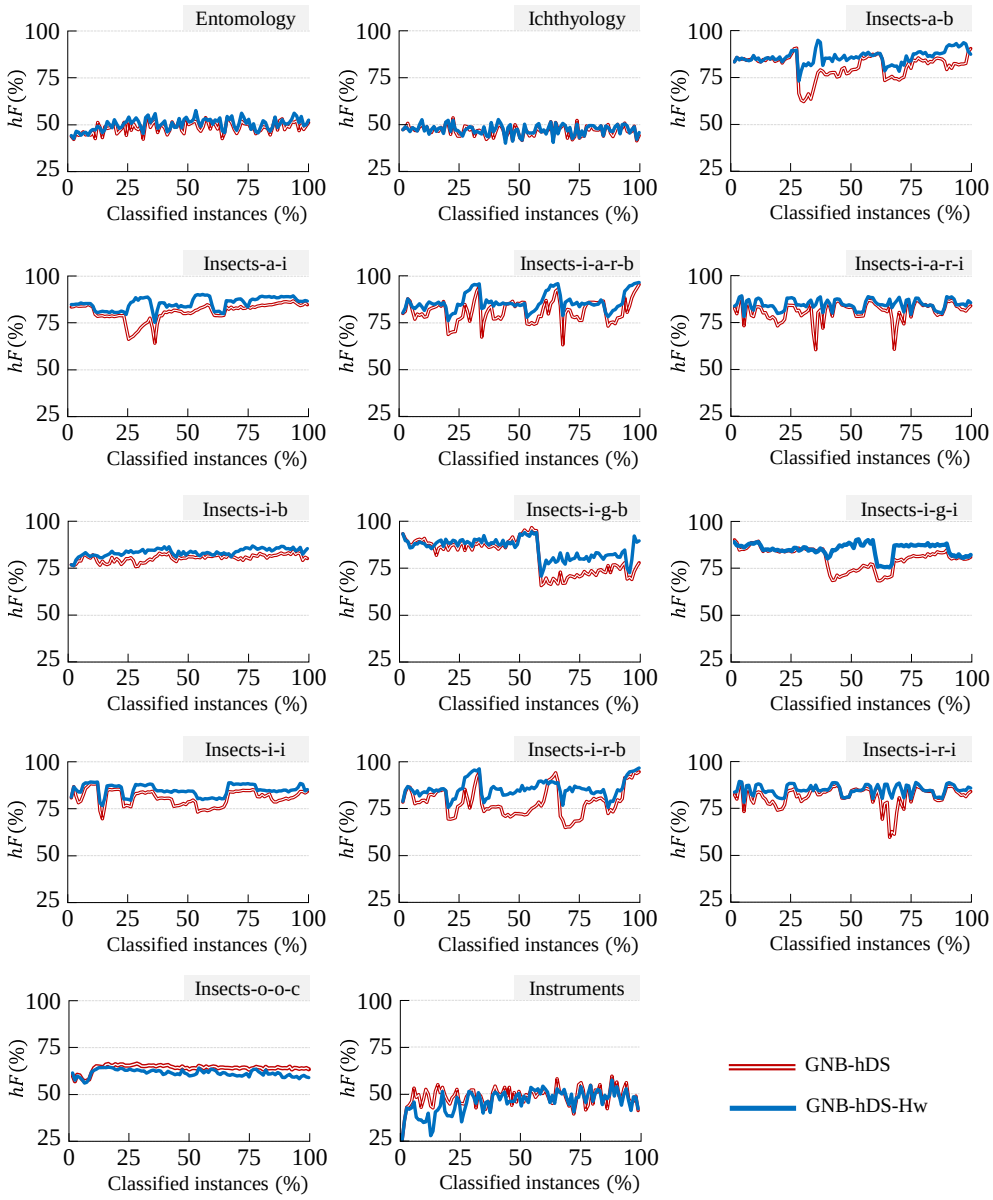

Figure 7: Hierarchical F-Measure (%) rates obtained by GNB-hDS-Hw and GNB-hDS methods along the hierarchical data stream.

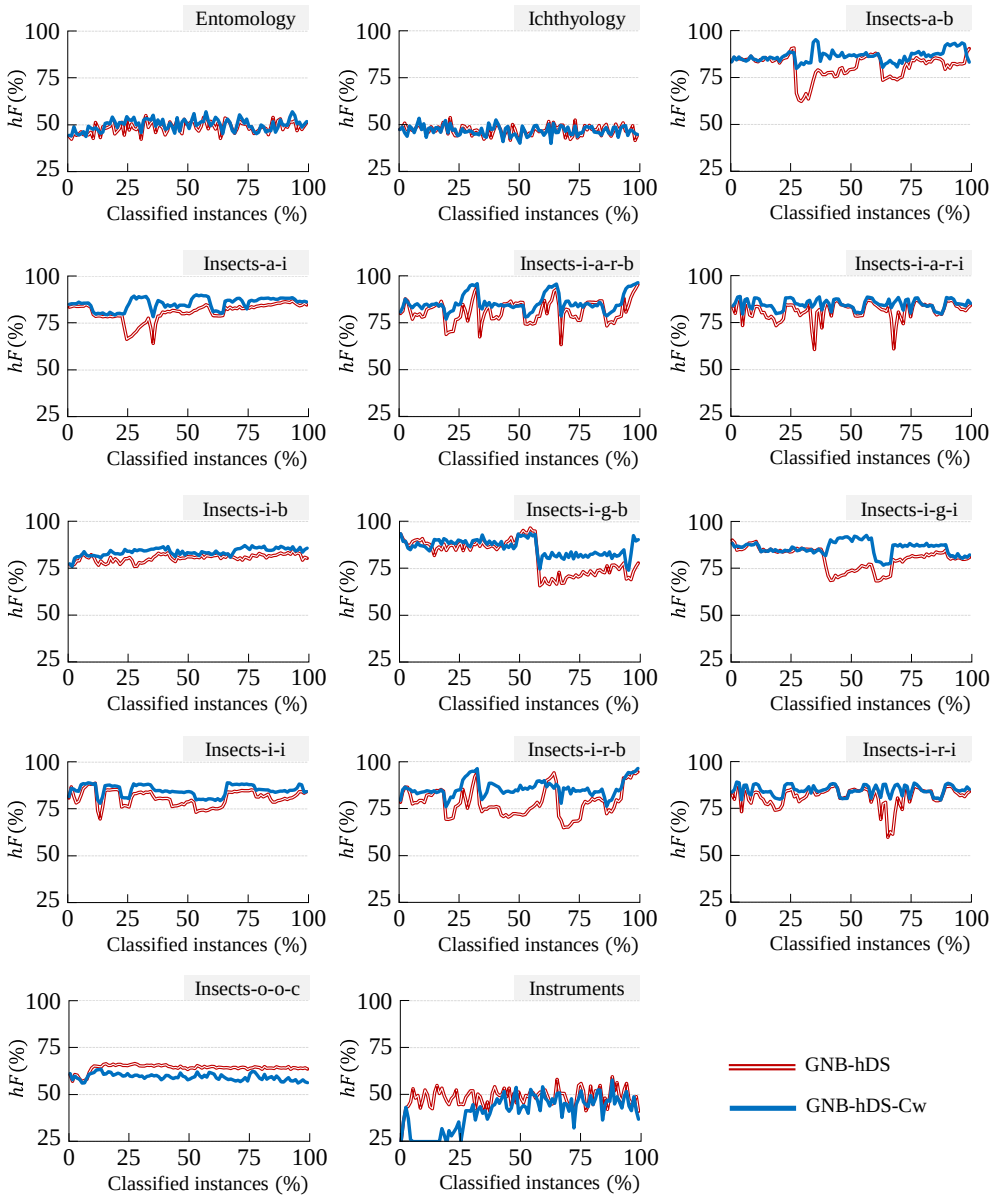

Figure 8: Hierarchical F-Measure (%) rates obtained by GNB-hDS-Cw and GNB-hDS methods along the hierarchical data stream.

Overall, we observe that both adaptive GNB-hDS-Cw and GNB-hDS-Hw classifiers could adapt themselves faster than the incremental GNB-hDS, especially on the datasets with well-described concept drifts [3]. For instance, on the "Insects-a-b" and "Insects-a-i" datasets, one can note that while the prediction correctness of the incremental classifier is affected by a concept drift and drops roughly from 80% to 65%, both adaptive classifiers adapt themselves swiftly enough to maintain and even obtain better hF rates on the same portion of the stream.

On average, this feature results in a higher prediction correctness rate for the adaptive classifiers when considering all hierarchical data streams together. However, in cases where the concept drifts are not well delimited or even nonexistent, the incremental classifier GNB-hDS uses the information of the entire data stream to achieve a more stable representation of the concepts and, consequently, a comparably or higher hF rate; this is the case in the "Entomology", "Ichthyology" and "Instruments" data stream sets, where no concept drift has been described yet [24]. In this sense, GNB-hDS-Hw is less affected by this stability than GNB-hDS-Cw since it maintains historical statistical descriptors only weighted by the current ones, whereas GNB-hDS-Cw completely discards the old information.

We also analyzed the behavior of the methods over variations of w. Table 10 compares the average hF (%) rates obtained by methods on each variation of the w parameter. Observe that this analysis considers only the proposed adaptive GNB-hDS-Cw and GNB-hDS-Hw and the variants using the Incremental Yeo-Johnson Power Transformation.

Table 10: Average Hierarchical F-Measure (%) obtained by GNB-hDS-Cw and GNB-hDS-Hw on each variation of w.

w	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
10	72.55	72.83	73.76	74.52
50	74.85	75.27	75.51	76.52
100	75.10	75.20	75.58	76.49
500	73.94	74.77	74.89	76.01
1000	73.92	74.84	74.66	75.87
5000	74.20	74.64	74.01	75.28
Avg. hF	74.09	74.59	74.73	75.78
Avg. Ranking	3.83	2.67	2.50	1.00

GNB-hDS-Hw-iYJ showed the best average results in all variations of w, resulting in a clear first ranking. Note that the average hF rates follow the same order as the one obtained in the previously described results considering the best average performances of the methods regarding prediction correctness. Also, the best results of all methods occur with $w \in \{50, 100\}$, suggesting that these values obtain a reasonable trade-off between historical and current data.

6.4. Overall Comparison

To portray an overview of both hF and $inst/s$ rates together, the MCDM-WPM analysis was also applied to the overall results of all methods, considering both the best average performances regarding prediction correctness and trade-off performances.

First, regarding the averaged best hF performance, Table 11 details the WPM performances for each variation of the hF criterion weight (w_{hF}). Note that, as previously described, the $w_{inst/s}$ is the complementary percent-

age of w_{hF}. The last row shows the average WPM performance concerning all w variations. Also, note that zero with decimal places is a rounding from a constant equal to 10^{-6} and represents the minimum value obtained with the criterion normalization.

Table 11: MCDM-WPM (values) of methods with best average performance regarding prediction correctness.

w_{hF}	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
1/6	0.0000	0.0590	0.0338	0.7310	0.5393	**0.8952**	0.6308	0.8273	0.8332	0.7792	0.6575	0.5535
2/6	0.0000	0.0066	0.0636	0.6799	0.5838	**0.8014**	0.4406	0.7883	0.8000	0.7668	0.6822	0.6230
3/6	0.0000	0.0007	0.1194	0.6323	0.6319	0.7174	0.3077	0.7510	**0.7681**	0.7546	0.7078	0.7012
4/6	0.0002	0.0001	0.2244	0.5881	0.6841	0.6422	0.2149	0.7156	0.7375	0.7425	0.7344	**0.7893**
5/6	0.0006	0.0000	0.4216	0.5470	0.7405	0.5749	0.1501	0.6818	0.7081	0.7307	0.7619	**0.8884**
Avg. WPM	0.0002	0.0133	0.1726	0.6357	0.6359	0.7262	0.3488	0.7528	**0.7694**	0.7548	0.7088	0.7111
Avg. Ranking	11.60	11.20	9.80	6.80	6.80	4.00	9.00	4.00	**2.60**	3.20	4.40	4.60

Additionally, Table 12 shows the rankings corresponding to the values shown in Table 11 regarding each w variation. The last row shows the overall average ranking of methods when considering their averaged best performance.

Table 12: MCDM-WPM (rankings) of methods with best average performance regarding prediction correctness.

w_{hF}	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
1/6	12.00	10.00	11.00	5.00	9.00	**1.00**	7.00	3.00	2.00	4.00	6.00	8.00
2/6	12.00	11.00	10.00	6.00	8.00	**1.00**	9.00	3.00	2.00	4.00	5.00	7.00
3/6	12.00	11.00	10.00	7.00	8.00	4.00	9.00	3.00	**1.00**	2.00	5.00	6.00
4/6	11.00	12.00	9.00	8.00	6.00	7.00	10.00	5.00	3.00	2.00	4.00	**1.00**
5/6	11.00	12.00	9.00	8.00	3.00	7.00	10.00	6.00	5.00	4.00	2.00	**1.00**
Avg. Ranking	11.60	11.20	9.80	6.80	6.80	4.00	9.00	4.00	**2.60**	3.20	4.40	4.60

The Local kNN-hDS method obtained the last combined ranking (11.60), followed by the proposed Global kNN-hDS method. Regardless of the weights given to hF and $inst/s$ rates, both methods did not perform well in any scenario as they have the lowest individual rates. On the other hand, the GNB-hDS-Cw method obtained the first combined ranking (2.60), obtaining the second place with smaller weights in the hF rate, the first place with equal weights in both rates and the third and fifth place with greater weights on the hF rate.

The GNB-hDS-Cw-iYJ method obtained the second-best combined ranking (3.20), with a worse performance than the GNB-hDS-Cw method, mainly in the speed comparisons. The Global Dribble-hDS and GNB-hDS-iYJ methods share the third-best combined ranking (4.00). The GNB-hDS-iYJ method proved competitive in all variations of w and even obtained better rankings than the Global Dribble-hDS method with higher weights in w_{hF}. Furthermore, it is noteworthy that Global Dribble-hDS achieved first place in the ranking when the weights favored the $inst/s$ rate.

The same general MCDM-WPM analysis was applied to the overall results of the methods regarding their best trade-off performance. Table 13 details the WPM performances for each variation of the hF criterion weight (w_{hF}). The last row shows the average WPM performance concerning all w variations.

Table 13: MCDM-WPM (values) of methods with best trade-off performance.

w_{hF}	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
1/6	0.0086	0.3984	0.0973	0.5885	0.8761	**0.9286**	0.7394	0.7336	0.7349	0.6113	0.3298	0.0000
2/6	0.0014	0.1587	0.1480	0.6188	0.8596	**0.8919**	0.6985	0.7512	0.7551	0.6546	0.4028	0.0001
3/6	0.0002	0.0632	0.2252	0.6506	0.8433	**0.8568**	0.6598	0.7693	0.7757	0.7010	0.4919	0.0010
4/6	0.0000	0.0252	0.3427	0.6841	**0.8273**	0.8230	0.6233	0.7878	0.7970	0.7506	0.6008	0.0100
5/6	0.0000	0.0100	0.5215	0.7194	0.8116	0.7905	0.5887	0.8068	**0.8188**	0.8037	0.7338	0.1000
Avg. *WPM*	0.0021	0.1311	0.2670	0.6523	0.8436	**0.8581**	0.6619	0.7698	0.7763	0.7042	0.5118	0.0222
Avg. Ranking	11.60	9.60	9.40	6.80	**1.80**	2.00	5.80	4.00	2.80	5.20	7.80	11.20

As well, Table 14 shows the rankings corresponding to the values shown in Table 13 regarding each w variation. The last row shows the overall average ranking of all methods when considering their best trade-off performances.

Table 14: MCDM-WPM (rankings) of methods with best trade-off performance.

w_{hF}	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
1/6	11.00	8.00	10.00	7.00	2.00	**1.00**	3.00	5.00	4.00	6.00	9.00	12.00
2/6	11.00	9.00	10.00	7.00	2.00	**1.00**	5.00	4.00	3.00	6.00	8.00	12.00
3/6	12.00	10.00	9.00	7.00	2.00	**1.00**	6.00	4.00	3.00	5.00	8.00	11.00
4/6	12.00	10.00	9.00	6.00	**1.00**	2.00	7.00	4.00	3.00	5.00	8.00	11.00
5/6	12.00	11.00	9.00	7.00	2.00	5.00	8.00	3.00	**1.00**	4.00	6.00	10.00
Avg. Ranking	11.60	9.60	9.40	6.80	**1.80**	2.00	5.80	4.00	2.80	5.20	7.80	11.20

As in the analysis concerning the best average performance of the methods, the Local kNN-hDS method also obtained the last place in the combined ranking considering the best trade-off performance. Even with smaller data representations (n = 5) and competitive $inst/s$ rates, the method could not maintain the hF rates obtained when using more data.

Next, the GNB-hDS-Hw-iYJ method obtained the second-to-last ranking

since it obtained the lowest overall $inst/s$ rate. Despite the best overall hF rate, the method did not achieve a good position in the combined ranking, as the gains in hF rates were not enough to offset the low $inst/s$ rates in the combined MCDM-WPM analysis.

On the other side of the combined ranking, Global kNC-hDS and Global Dribble-hDS methods obtained the first and second places, respectively (1.80 and 2.00). The Global kNC-hDS method achieved first place with $w_{hF} = \frac{4}{6}$, and second place in the other variations of w. The Global Dribble-hDS method achieved first place in the ranking when the weights favored the inst/s rate and with equal weights in both rates.

Furthermore, when the weights strongly favored the hF rate, the GNB-hDS-Cw method obtained first place, resulting in a third place in the combined ranking (2.80).

Overall, regarding averaged best performance analysis, all GNB-hDS variants, plus the Global Dribble-hDS method, presented competitive results with each other, with the Global Dribble-hDS method obtaining higher processing speed rates, GNB-hDS-Hw-iYJ obtaining better prediction correctness rates, and the other methods in between, with GNB-hDS-Cw obtaining the best equally weighted performance.

Regarding best trade-off performance analysis, Global kNC-hDS and Global Dribble-hDS methods stand out, with Global Dribble-hDS obtaining higher processing speed rates and Global kNC-hDS obtaining better prediction correctness, together with the GNB-hDS-Cw method.

Also, it is noteworthy that the analysis using the best trade-off performances presents less bias in the comparison of the methods since the analysis

using the best average performances regarding prediction correctness intrinsically gives greater relevance to the hF rate in the selection of the parameter configuration to be considered in the MCDM analysis, even before assigning weights to the hF and $inst/s$ criteria.

Finally, two key aspects related to the described MCDM-WPM analysis are noteworthy. First, it should be understood only as a guide for interpreting the results of the methods when compared together. Applying the MCDM-WPM method with different criteria and weight ranges can change the method ranking. In this study, the MCDM-WPM analysis was performed separately on each method (in order to remove the initial bias of the averaged best performances) and then in the methods together,normalizing the rates obtained by them. However, other MCDM protocols are possible and could not result in the same rankings and generate different interpretations.

Second, merging evaluation metrics is not straightforward nor trivial, and the resulting overall average ranking obtained in the MCDM-WPM analysis should not be understood as a single measure to be used instead of individual ones. The best learning model may depend on several external traits not measured by the hF and $inst/s$ metrics computed in the proposed protocol used in this study. The best learning model may also require specific solutions to problems not comprehended by this work, such as unusual data distributions, higher responsiveness to concept drifts, and different constraints regarding computational resources.

7. Conclusion

In this study, we introduced two novel adaptive Gaussian Naive Bayes classifiers, GNB-hDS-Hw (Gaussian Naive Bayes with Historical Window) and GNB-hDS-Cw (Gaussian Naive Bayes with Current Window), specifically designed to address the challenges of hierarchical data stream classification. These methods incorporate window-weighted probabilities into the Bayesian framework, leveraging both current data (GNB-hDS-Cw) and historical data (GNB-hDS-Hw) to calculate probabilities dynamically and adaptively.

A robust experimental testbed showed that this trait improved the adaptation capability of the classifiers, especially on dynamic data streams with well-known concept drifts. Results showed that GNB-hDS-Hw obtained overall higher prediction correctness, surpassing all state-of-the-art techniques. The method also achieved two best hF-weighted rankings when considering the averaged best performance. Also, GNB-hDS-Cw obtained the best overall combined ranking when considering the best average performance of the methods regarding prediction correctness and one best hF-weighted ranking when considering the best trade-off performance.

It is also relevant to highlight that the experimentation protocol encompassed existing algorithms for hierarchical data stream classification and datasets. A by-product of this experimentation yielded a unified protocol to assess and compare hierarchical data stream learning models using the hF and $inst/s$ literature metrics in a Multi-Criteria Decision-Making analysis, resulting in a reproducible testbed to future comparisons.

For further research, we are interested in designing a learning model based on a different learning paradigm than the ones described in this study, such

as decision trees. State-of-the-art learning models of the foundation areas – such as CLUS-HMC [41] on Hierarchical Classification, and Adaptive Random Forest [42] on Data Stream Classification – must be accounted for, used as a concept idea for the designing of a novel decision tree-based method fitted to the hierarchical data stream classification task, and benchmarked against state-of-the-art methods to understand its behavior regarding prediction correctness and computational performance on this new classification task.

Funding

This research was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] C. N. Silla, A. A. Freitas, A survey of hierarchical classification across different application domains, Data Mining and Knowledge Discovery 22 (1-2) (2011) 31–72.

[2] J. Gama, Knowledge discovery from data streams, Chapman and Hall/CRC, 2010.

[3] V. M. A. Souza, D. M. Reis, A. G. Maletzke, G. E. A. P. A. Batista, Challenges in benchmarking stream learning algorithms with real-world data, Data Mining and Knowledge Discovery (2020) 1–54doi:10.1007/s10618-020-00698-5.

[4] E. Tieppo, R. R. d. Santos, J. P. Barddal, J. C. Nievola, Hierarchical classification of data streams: a systematic literature review, Artificial Intelligence Review (2021) 1–40.

[5] A. Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin 106 (2) (2004) 58.

[6] M. Mermillod, A. Bugaiska, P. Bonin, The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, Frontiers in psychology 4 (2013) 504.

[7] S. Defiyanti, E. Winarko, S. Priyanta, A survey of hierarchical classification algorithms with big-bang approach, in: 2019 5th International Conference on Science and Technology (ICST), Vol. 1, IEEE, 2019, pp. 1–6.

[8] A. Freitas, A. Carvalho, A tutorial on hierarchical classification with applications in bioinformatics, in: Research and trends in data mining technologies and applications, IGI Global, 2007, pp. 175–208.

[9] Y. Lu, Concept hierarchy in data mining: Specification, generation and implementation, Ph.D. thesis, Theses (School of Computing Science)/Simon Fraser University (1997).

[10] F. Wu, J. Zhang, V. Honavar, Learning classifiers using hierarchically structured class taxonomies, in: International Symposium on Abstraction, Reformulation, and Approximation, Springer, 2005, pp. 313–320.

[11] S. Kiritchenko, F. Famili, Functional annotation of genes using hierarchical text categorization, Proceedings of BioLink SIG, ISMB (01 2005).

[12] R. Cerri, G. L. Pappa, A. C. P. Carvalho, A. A. Freitas, An extensive evaluation of decision tree–based hierarchical multilabel classification methods and performance measures, Computational Intelligence 31 (1) (2015) 1–46.

[13] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine learning 23 (1) (1996) 69–101.

[14] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, A survey on data stream clustering and classification, Knowledge and information systems 45 (3) (2015) 535–569.

[15] J. P. Barddal, H. M. Gomes, F. Enembreck, B. Pfahringer, A survey on feature drift adaptation: Definition, benchmark, challenges and future directions, Journal of Systems and Software 127 (2017) 278–294.

[16] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR) 46 (4) (2014) 44.

[17] J. Gama, R. Sebastião, P. P. Rodrigues, On evaluating stream learning algorithms, Machine learning 90 (3) (2013) 317–346.

[18] J. P. Barddal, H. M. Gomes, F. Enembreck, B. Pfahringer, A. Bifet, On dynamic feature weighting for feature drifting data streams, in: Joint european conference on machine learning and knowledge discovery in databases, Springer, 2016, pp. 129–144.

[19] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, L. Wan, Heterogeneous ensemble for feature drifts in data streams, in: Pacific-Asia conference on knowledge discovery and data mining, Springer, 2012, pp. 1–12.

[20] A. Naik, H. Rangwala, Large scale hierarchical classification: state of the art, Springer, 2018.

[21] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, J. Gama, Machine learning for streaming data: state of the art, challenges, and opportunities, ACM SIGKDD Explorations Newsletter 21 (2) (2019) 6–22.

[22] K. K. Wankhade, S. S. Dongre, K. C. Jondhale, Data stream classification: a review, Iran Journal of Computer Science 3 (4) (2020) 239–260.

[23] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, S. Maniu, Data stream analysis: Foundations, major tasks and tools, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 11 (3) (2021) e1405.

[24] A. R. S. Parmezan, V. M. Souza, G. E. Batista, Towards hierarchical classification of data streams, in: Iberoamerican Congress on Pattern Recognition, Springer, 2018, pp. 314–322.

[25] E. Tieppo, J. P. Barddal, J. C. Nievola, Adaptive global k-nearest neighbors for hierarchical classification of data streams, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2021, pp. 631–636.

[26] E. Tieppo, J. P. Barddal, J. C. Nievola, Automatic disease vector mosquitoes identification via hierarchical data stream classification, in:

Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 2022, pp. 1005–1012.

[27] E. Tieppo, J. P. Barddal, J. C. Nievola, Classifying hierarchical data streams using global classifiers and summarization techniques, in: The 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–8.

[28] E. Tieppo, J. P. Barddal, J. C. Nievola, Classifying potentially unbounded hierarchical data streams with incremental gaussian naive bayes, in: Brazilian Conference on Intelligent Systems, Springer, 2021, pp. 421–436.

[29] E. Tieppo, J. P. Barddal, J. C. Nievola, Improving data stream classification using incremental yeo-johnson power transformation, in: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2022, pp. 3286–3292.

[30] D. West, Updating mean and variance estimates: An improved method, Communications of the ACM 22 (9) (1979) 532–535.

[31] T. F. Chan, G. H. Golub, R. J. LeVeque, Algorithms for computing the sample variance: Analysis and recommendations, The American Statistician 37 (3) (1983) 242–247.

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

[33] Y. Wang, Z. Gong, J. Guo, Hierarchical classification of business information on the web using incremental learning, in: 2009 IEEE International Conference on e-Business Engineering, IEEE, 2009, pp. 303–309.

[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (Jan) (2006) 1–30.

[35] S. Zionts, Mcdm—if not a roman numeral, then what?, Interfaces 9 (4) (1979) 94–101.

[36] G. Nakhaeizadeh, A. Schnabl, Development of multi-criteria metrics for evaluation of data mining algorithms., in: KDD, 1997, pp. 37–42.

[37] E. Triantaphyllou, Multi-criteria decision making methods, in: Multi-criteria decision making methods: A comparative study, Springer, 2000, pp. 5–21.

[38] A. Çelen, Comparative analysis of normalization procedures in topsis method: with an application to turkish deposit banking market, Informatica 25 (2) (2014) 185–208.

[39] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.

[40] P. Nemenyi, Distribution-free multiple comparisons, in: Biometrics, Vol. 18, International Biometric Society, 1962, p. 263.

[41] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Machine learning 73 (2) (2008) 185–214.

[42] 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, Machine Learning 106 (9) (2017) 1469–1495.