Enhancing Concept Drift Detection in Drifting and Imbalanced Data Streams through Meta-Learning

Gabriel Jonas Aguiar
Dpt. of Computer Science, College of Engineering
Virginia Commonwealth University
Richmond, VA, USA
aguiargj@vcu.edu

Alberto Cano
Dpt. of Computer Science, College of Engineering
Virginia Commonwealth University
Richmond, VA, USA
acano@vcu.edu

***Abstract*—One of the biggest challenges in learning from data streams is adapting the classification model to new data. Due to the evolving nature of data streams, they are subject to a phenomenon known as concept drift that makes previously learned knowledge and model outdated. Therefore, concept drift must be efficiently detected in order to adapt the classification model. While there exists a plethora of drift detectors, with different mechanisms, selecting the most suitable for a new stream is a difficult task, since apriori knowledge may not be available and changes over time can affect the performance of the detector. This paper proposes a framework that exploits statistical and temporal meta-features from sliding windows to dynamically recommend a suitable drift detector in real-time for unseen chunks of streams according to its properties using Meta-Learning. We performed experiments on 10 real-world data streams and 18 synthetic generated data streams that were subject to concept drift and class imbalance in order to evaluate the performance of the proposed framework. Experiments exposed that the proposed approach was able to enhance the concept drift detection in a variety of scenarios demonstrating robustness to class imbalance and the advantages of dynamically selecting the drift detector.**

***Index Terms*—Data streams, Machine Learning, Concept Drift, Meta-Learning**

I. Introduction

In recent years, there has been a remarkable advancement in our ability to collect and analyze data. The exponential growth has presented challenges for traditional machine learning methods, which were primarily designed to handle static data. On the other hand, modern data sources generate continuous streams of data characterized by high volume and velocity. This scenario is commonly referred to as a data streams [1], which can be defined as a potentially unbounded sequence of ordered instances that continuously arrive at a system.

The dynamic and evolving nature of data streams necessitates that classifiers adapt and learn from emerging concepts over time. This phenomenon is commonly referred to as concept drift [2]. If concept drift is not correctly detected and addressed, it can lead to a decline in predictive performance since the knowledge acquired from previous concepts may no longer be applicable to the current instances. Additionally, a significant challenge in data stream learning is the robust handling of class imbalance. This challenge is further amplified in the streaming scenario where class imbalance coexists with concept drift. Not only do the class definitions change, but the imbalance ratio becomes dynamic and may also switch [3].

Numerous solutions have been proposed in the literature to address these challenges [3], [4]. These solutions range from ensemble methods to explicit drift detectors, each with its own advantages and disadvantages. However, many of these approaches heavily rely on prior knowledge extracted from the data stream, which poses a challenge in non-stationary environments. Additionally, the *"No Free Lunch"* theorem [5] states that there is no single algorithm suitable for every dataset. One potential solution is to dynamically recommend the best algorithm for each specific problem [6]–[8]. In the context of data streams, the algorithm recommendation problem has been effectively addressed through Meta-Learning (MtL) [9], [10].

The core concept of MtL is to leverage knowledge gained from previous similar problems to recommend the most suitable algorithm for a new and unseen dataset [11]. Building upon this idea, our work introduces an online MtL framework that dynamically recommends drift detectors for drifting and imbalanced data streams. Our hypothesis is that MtL can be effectively applied to drifting data streams to dynamically select the most suitable drift detector for unseen chunks of a given stream, based on their varying characteristics. The main contributions of this paper can be summarized as follows:

- **Online Framework:** We propose an online Meta-Learning framework that can recommend the most suitable drift detector in real-time for an unseen and recently arrived chunk of data.
- **Robustness to class imbalance:** The proposed framework exhibits robustness to imbalanced drifting data streams without the need for additional mechanisms.
- **Generalization:** The Meta-Dataset utilized in our study consists of synthetically generated data streams, demonstrating the framework's capability for generalization.
- **Extensive experimental study:** We conduct a comprehensive experimental evaluation, employing a meticulously designed test bed that encompasses both real-world and artificial benchmarks.

This paper is organized as follows. Section II presents the theoretical foundation and related works. Section IV provides the experimental setup and evaluation methodology. Section V presents and analyzes the results of our study. Finally, Section VI discusses the conclusions and future work.

II. Background

This section reviews the background and foundations of our methodology. Firstly, we present the literature concerning data streams and explore their intrinsic challenges, such as class imbalance and concept drift. Moreover, we provide an overview of the literature on Meta-Learning and its applications.

A. Data stream

A data stream refers to a potentially unbounded sequence of ordered instances that arrive over time within a system. Learning from data streams imposes specific constraints on classifiers [1]. We can define a stream, denoted as S, as a sequence $< s_1, s_2, s_3, ..., s_\infty >$, where $s_i = (X, y)$. This stream can be processed either one instance at a time (online scenario) or in chunks (block scenario). Data streams possess four main characteristics [3], [12]: (i) Volume, (ii) Velocity, (iii) Veracity, and (iv) Non-stationarity, which introduce challenges to classifiers that must adapt accordingly.

Class imbalance stands as one of the most critical challenges in contemporary machine learning [13], [14]. It refers to the disproportionate distribution of instances across different classes, where certain classes are significantly underrepresented. Within the data stream mining domain [3], class imbalance emerges as a common issue. It manifests through class role switches, where the majority becomes the minority and vice versa, the introduction or disappearance of multiple classes, and the emergence of instance-level difficulties such as evolving class overlapping or clusters/subconcepts [15]. Moreover, it is important to note that in real-life scenarios, streams are often not predefined as balanced or imbalanced, and the imbalance may occur temporarily [16].

Data streams are also subject to a phenomenon known as concept drift [17], [18]. Each instance arrives at a specific time, denoted as t, and is generated according to a probabilistic distribution $\phi^t(X, y)$, where X represents the feature vector and y denotes the class label. If all instances in the stream are generated by the same probability distribution, the data is considered stationary, indicating that it originates from a consistent concept. However, in real-world applications, data rarely adheres to stationary assumptions [19]. Conversely, if two separate instances arriving at times t and $t + C$ are generated by $\phi^t(X, y)$ and $\phi^{t+C}(X, y)$ respectively, and if $\phi^t \neq \phi^{t+C}$, a concept drift has occurred. Moreover, when combined with class imbalance, concept drift introduces new and unique challenges [20]. This phenomenon affects various aspects of a data stream and thus can be analyzed from multiple perspectives. When analyzing and understanding concept drift, the following factors are considered [3], [21]:

Influence of the decision boundaries. Firstly, it is necessary to consider how concept drift affects the learned decision boundaries, distinguishing between real and virtual concept drifts. Virtual drift produces a change in the unconditional probability distribution $P(x)$, without affecting the learned decision boundaries. Although virtual drift does not impair learning models, its detection is necessary to avoid false alarms and prevent unnecessary, costly adaptations. In contrast, real concept drift modifies the decision boundaries, making them worthless to the current concept. Detecting and adapting to real concept drift is crucial for preserving predictive performance.

Speed of changes. Here we can distinguish three types of concept drift [22]: (i) incremental; (ii) gradual; and (iii) sudden concept drifts.

- **Incremental concept drift.** Incremental drift generates a sequence of intermediate states between the old and new concepts.We can formally define an incremental drift between distributions D_0 and D_1, occurring between the t_1-th and t_2-th instances, using the following equation:

$$\phi^t = \begin{cases} D_0(X, y), & \text{if } t < t_1 \\ (1 - \alpha_t)D_0(X, y) + \alpha_t D_1(X, y), & \text{if } t_1 \leq t < t_2 \\ D_1(X, y), & \text{if } t_2 \leq t \end{cases}$$

  where

$$\alpha_t = \frac{t - t_1}{t_2 - t_1}$$

- **Gradual concept drift.** Gradual drift oscillates between instances coming from both old and new concepts, with the new concept becoming more and more frequent over time, and can be defined as the following:

$$\phi^t = \begin{cases} D_0(X, y), & \text{if } t < t_1 \\ D_0(X, y), & \text{if } t_1 \leq t < t_2 \wedge \delta > \alpha_t \\ D_1(X, y), & \text{if } t_1 \leq t < t_2 \wedge \delta \leq \alpha_t \\ D_1(X, y), & \text{if } t_2 \leq t \end{cases}$$

  where δ is a random variable ranging between $(0, 1)$, and α_t denotes the same variable used in the context of incremental drift.
- **Sudden concept drift.** Sudden drift instantaneously switches between old and new concepts, leading to an instant degradation of the underlying learning algorithm.

$$\phi^t = \begin{cases} D_0(X, y), & \text{if } t < t_1 \\ D_1(X, y), & \text{if } t_2 \leq t \end{cases}$$

Recurrence. Changes in the stream can be either unique or recurring. In the latter case, the previously seen concept may reemerge over time, allowing us to recycle previously learned knowledge. The past knowledge can be used as an initialization point for the drift recovery.

Presence of noise. Noise can take the form of sporadic, insignificant variations within a stream that can be disregarded, or substantial corruption within the features or class labels that need to be dealt with in order to prevent the input of misleading or adversarial data into the classifier [23].

To address the challenges posed by concept drift, two approaches are commonly employed: (i) implicit and (ii) explicit. Implicit approaches manage drift adaptation through learning mechanisms embedded in the classifier, assuming its ability to adjust itself to new instances from the latest concept while gradually forgetting outdated information [24]. These approaches involve establishing appropriate learning and forgetting rates, utilizing adaptive sliding windows, or continually tuning hyperparameters. Conversely, explicit approaches delegate drift adaptation to an external tool known as

a drift detector [4]. Drift detectors continuously monitor stream properties (e.g., statistics) or classifier performance (e.g., error rates). They raise a warning signal when there are indications of impending drift and trigger an alarm signal when concept drift has occurred.

Drift detectors. Numerous explicit drift detectors have been proposed in recent years [25]. Like classifiers, there are three groups of methods that can be used to detect drift: supervised, semi-supervised, and unsupervised. The main distinction between them is the location where drift is detected. Whereas supervised drift detectors identify changes in class boundaries, unsupervised ones track changes in data distribution [2].

The group of supervised drift detectors that has gained the most popularity comprises techniques that rely on measuring the error or accuracy of classifiers over labeled instances, such as the Drift Detector Method (DDM) [26] or Early DDM (EDDM) [27]. These methods use statistical tests to determine whether or not to issue a warning for drift. For instance, the WSTD [28] method utilizes the Wilcoxon rank sum test, while the KSWIN [29] method is based on the Kolmogorov-Smirnov test. A well-liked group of supervised detectors are those that rely on metrics calculated within subwindows of a stream. The ADaptative WINdow (ADWIN) [30] is one such method, which uses an adaptive sliding window based on Hoeffding's inequality. This approach has inspired the development of several new detectors [31], [32].

Unsupervised drift detectors are concerned with identifying discrepancies in unlabeled data without any added supervision. These detectors often involve a statistical comparison between two sets of data from the older and recent chunk [33], [34]. More advanced unsupervised methods attempt to pinpoint the precise location in the feature space where the drift occurred. These detectors concentrate on a spatial search that utilizes different dissimilarity measures [35], [36].

When it comes to imbalanced data streams, there is a limited number of concept drift detectors specifically designed for this scenario. Most existing works on imbalanced data streams focus on changes in the underlying classifier, assuming its adaptability [3]. However, there are two notable dedicated drift detectors for skewed streams. PerfSim [37] monitors changes in the entire confusion matrix, while Drift Detection Method for Online Class Imbalance (DDM-OCI) [38] tracks recall for each class. Additionally, Korycki and Krawczyk [21] proposed the utilization of Restricted Boltzmann Machines to detect drifts in multi-class imbalanced data streams.

Despite the numerous drift detectors proposed in the literature, selecting the most suitable one for a given data stream requires a priori information, which is not feasible in the online scenario [4]. Furthermore, due to the evolving nature of data streams, changes in their characteristics over time may lead to the deterioration of previously selected drift detectors.

B. Meta-Learning

The main concept of Meta-Learning (MtL) [11], [39] is to leverage knowledge gained from previous similar problems to recommend the most suitable algorithm for a new dataset, encapsulated in the notion of *"Learning how to learn"* [7], [40]. Consequently, MtL utilizes Machine Learning (ML) to establish mappings between the characteristics of prior problems and algorithm performance. One prominent application of Meta-Learning lies in algorithm recommendation problems, originally introduced by Rice [41] to select an algorithm from a given set of options. Specifically, in the context of a set of algorithms $\mathcal{A}$, a set of datasets $\mathcal{P}$ composed of instances from distribution $\mathcal{D}$, and a performance measure $\mathcal{M} : \mathcal{P} \times \mathcal{A} \rightarrow \mathbb{R}$, the algorithm recommendation problem involves finding a function $m : \mathcal{P} \rightarrow \mathcal{A}$ that enhances the expected performance measure for the problems described by $\mathcal{D}$.

The core concept of Meta-Learning (MtL) is to exploit past learning experiences in a specific task and its corresponding solutions by adapting learning algorithms and data mining processes [11]. This is achieved by extracting *meta-features* from a dataset, which serve as representations of both the dataset itself and the performance of Machine Learning (ML) algorithms when applied to it. The relationship between meta-features and ML performance offers valuable insights for selecting the most appropriate algorithm for new datasets. Consequently, ML algorithms are employed on a *meta-dataset*, where each example is described in terms of *meta-features*, to induce a *meta-model*.

In the domain of traditional (or offline) Machine Learning (ML), Meta-Learning (MtL) has emerged as a pivotal approach for enhancing predictive performance by effectively addressing challenges related to ML algorithm recommendation for static [42], [43] and time-sensitive data [44], computer vision algorithm recommendation [45], [46], clustering [47], hyperparameter tuning [48], [49], and anomaly detection [50].

C. Meta-Learning for Data Streams

The application of Meta-Learning in the field of data streams research is a relatively recent development. One preliminary approach for time-sensitive data was proposed by Rossi *et al.* [9], where meta-learning is used to select different regressors for different time periods within a given time series. Unlike traditional meta-learning frameworks, the models are continuously retrained instead of remaining static until recalled by the meta-learner. Another notable application by van Rijn *et al.* [51] involves dynamically adjusting the weights of base classifiers within an ensemble using Meta-Learning. This approach not only improves predictive performance but also enhances ensemble diversity.

In the context of drifting data streams, the Enhanced Concept Profiling Framework (ECPF) [52] method dynamically selects new classifiers after detecting a concept drift using Meta-Learning. This work demonstrates how different classifiers can perform better in different segments of the same data stream. Additionally, Lacombe *et al.* [6] and Martins *et al.* [10] propose frameworks for hyperparameter tuning in data streams. The former optimizes drift detectors and classifier hyperparameters on-the-fly, while the latter selects the most suitable uncertainty threshold for active learning strategies to minimize labeling costs and maximize predictive performance.

In summary, most existing works focus on applying Meta-Learning to select the most suitable classifier (or regressor) or determine the optimal configuration for a given classifier. They typically address concept drift passively by relying on classifier adaptation or employing a fixed drift detector for the entire data stream. However, in order to effectively tackle concept drift, it is crucial to carefully select the most appropriate drift detector rather than naively assuming that the classifier will always adapt.

III. Proposed Framework

The goal of this paper is to evaluate the use of Meta-Learning (MtL) for dynamically recommending drift detectors in drifting and imbalanced data streams in real-time. The modeling task, as depicted in Fig. 1, involves leveraging knowledge acquired from similar tasks. On the other hand, Fig. 2 illustrates the recommendation step. Subsequent subsections provide detailed descriptions of each component. The implementation is public available in our GitHub repository [1].

In the modeling task, a collection of data streams is subjected to a group of drift detectors, and characterised by *"meta-features"*. Subsequently, each data stream's classification performance is evaluated. During the evaluation step, the *"meta-targets"* are defined, representing the labels to be predicted by the MtL recommender system, in this case, the most suitable drift detector. This process generates a *"meta-dataset"*. Finally, ML algorithms (*"meta-learners"*) are employed on a training subset of the meta-dataset to induce a *"meta-model"*. The meta-model establishes the relationships between the meta-features and the meta-target. It is important to note that this entire process is conducted offline.

As illustrated in Fig. 2, the meta-model can be applied to a new data stream, represented by the values of its meta-features, to recommend the most suitable drift detector. In this step, as the complete data stream is not available, we extract consecutive chunks of size w from the data stream to recommend the most suitable drift detector for a specific time period. It is worth mentioning that this step operates online and is applied in parallel with the classification process.

A. Meta-Dataset

We created a meta-dataset, published on the GitHub repository, consisting of $1,334$ synthetically generated data streams (meta-examples), using 5 different data stream generators: *Agrawal*, *Hyperplane*, *Mixed*, *SEA*, and *Sine*. They were selected due to their different intrinsic characteristics, which not only allowed us to increase the heterogeneity of the meta-dataset but also enabled us to leverage more knowledge from it.. To ensure a wide range of examples, we generated streams with 4 different sizes: $\{5,000;\ 10,000;\ 20,000;$ and $30,000\}$ instances. Each data stream contained 4 concept drifts occurring at quarter intervals of the total number of instances, with 3 different drift speeds: $\{1$ (sudden); $350;$ and $1,000\}$. Furthermore, to introduce data streams with varying imbalance ratios over time, we incorporated 3 different

[1] https://github.com/gabrieljaguiar/meta-drift-detection/

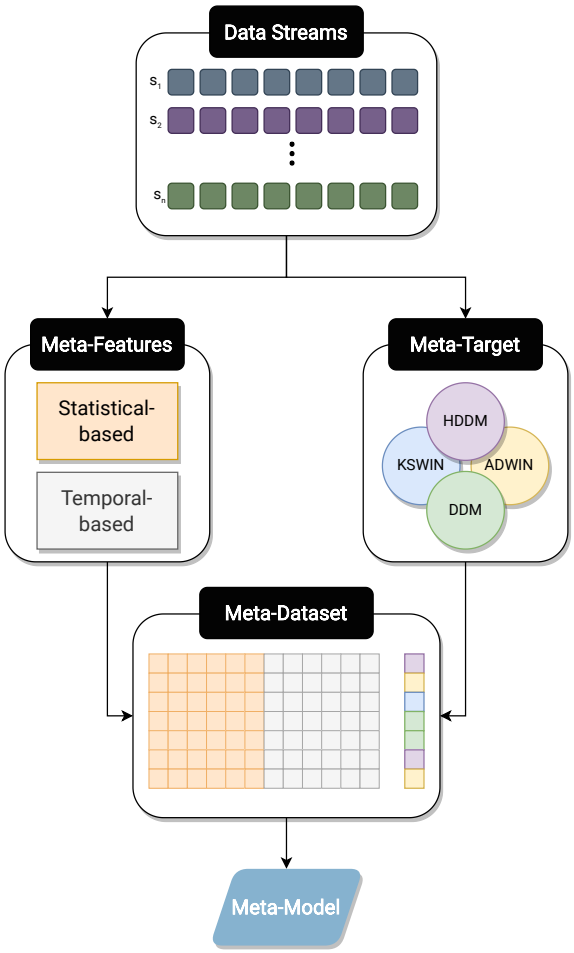

Fig. 1: General overview of the modeling (offline) phase of the framework to build the meta-dataset.

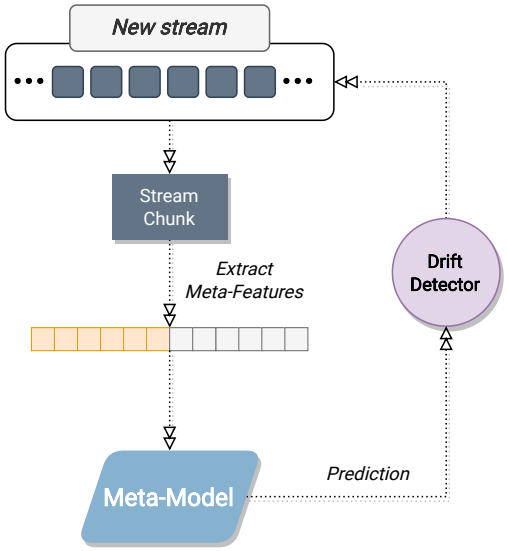

Fig. 2: Online meta-learning system for dynamic drift detector recommendation.

scenarios based on [3]: (i) `STABLE` with constant IR 1:1, (ii) `FLIPPING` with IR interleaving from 8:2 to 2:8, and (iii) `INCREASE-DECREASE` with IR increasing from 1:1 to 1:10 and then returning to 1:1.

The extensive variations of meta-examples in our meta-dataset were chosen to increase the diversity and representativeness of scenarios that can occur in real-world settings. By leveraging this knowledge, we can effectively address similar tasks and improve their performance.

B. Meta-Features

To characterize each data stream, we extract meta-features from the selected meta-examples. This is achieved by applying a function that captures relevant characteristics of the data stream. Our goal is to extract meta-features that effectively represent the behavior of drifting data streams and enable us to select the most suitable drift detector. The meta-features can be divided into two groups: Statistical and Temporal.

The Statistical group extracts information about the data stream without considering the temporal relationships between instances. Conversely, the Temporal group focuses on time-sensitive meta-features. Since the meta-features are computed based on each feature of the original stream, we further aggregate them using appropriate functions. The complete list of meta-features can be found in Table I. The extraction of meta-features is performed using the TSFEL library.

It is important to note that most of our meta-features are calculated in $\mathcal{O}(1)$ time, with only a few requiring $\mathcal{O}(\log n)$ time. Even with our aggregate set of meta-features, the computational costs remain relatively low, ensuring that our framework can respond promptly to concept drifts.

C. Meta-Target

We explored four widely used drift detectors: ADWIN, KSWIN, HDDM, and DDM. These detectors were selected based on their prominence in the literature [4], [52] and their diverse approaches to detecting concept drift.

To determine the performance of these drift detectors, we selected the Hoeffding Adaptive Tree (HAT) [53] as our base classifier. This classifier demonstrates excellent predictive performance but relies heavily on an effective concept drift detector [21], [53].

For evaluating the performance of the drift detectors, we employed the G-Mean metric [3] due to dealing with imbalanced data streams. Consequently, the drift detector that achieves the highest G-Mean value is selected as the meta-label for that particular meta-instance. The distribution of meta-classes (best-performing detector) is presented in Table II.

D. Meta-Learners

As our meta-learner, we opted for the Random Forest algorithm [54] due to its extensive usage and ability to induce models with high predictive accuracy. We constructed four distinct meta-models, one for each drift detector, to predict their relative rankings within the respective data streams. The highest predicted rank among the detectors is then selected as the final prediction.

TABLE I: Type, name, complexity and description of meta-features used in the experiments.

Group	Meta-Feature	Complexity
Statistical	ECDF	$\mathcal{O}(\log n)$
	ECDF Percentile	$\mathcal{O}(\log n)$
	ECDF Percentile Count	$\mathcal{O}(\log n)$
	Histogram	$\mathcal{O}(\log n)$
	Interquartile range	$\mathcal{O}(1)$
	Kurtosis	$\mathcal{O}(1)$
	Max	$\mathcal{O}(1)$
	Mean	$\mathcal{O}(1)$
	Mean absolute deviation	$\mathcal{O}(\log n)$
	Median	$\mathcal{O}(1)$
	Min	$\mathcal{O}(1)$
	Median absolute deviation	$\mathcal{O}(1)$
	Root mean square	$\mathcal{O}(1)$
	Skewness	$\mathcal{O}(1)$
	Standard deviation	$\mathcal{O}(1)$
	Variance	$\mathcal{O}(1)$
Temporal	Absolute energy	$\mathcal{O}(\log n)$
	Area under the curve	$\mathcal{O}(\log n)$
	Autocorrelation	$\mathcal{O}(1)$
	Centroid	$\mathcal{O}(1)$
	Entropy	$\mathcal{O}(\log n)$
	Mean absolute diff	$\mathcal{O}(1)$
	Mean diff	$\mathcal{O}(1)$
	Median absolute diff	$\mathcal{O}(1)$
	Median diff	$\mathcal{O}(1)$
	Negative turning points	$\mathcal{O}(1)$
	Neighbourhood peaks	$\mathcal{O}(1)$
	Peak to peak distance	$\mathcal{O}(1)$
	Positive turning points	$\mathcal{O}(1)$
	Signal distance	$\mathcal{O}(1)$
	Slope	$\mathcal{O}(\log n)$
	Sum absolute diff	$\mathcal{O}(1)$
	Total energy	$\mathcal{O}(1)$
	Zero crossing rate	$\mathcal{O}(1)$

TABLE II: Frequency of the best-performing drift detector on the meta-dataset used in the experiments.

Drift Detector	ADWIN	KSWIN	HDDM	DDM
Frequency	247	235	416	446
%	18.37	17.48	30.95	33.2

IV. Experimental Setup

The experiments were designed to evaluate the performance of the proposed framework under varied imbalanced scenarios and difficulties. We aim to understand in which scenarios meta-learning would improve the performance of the classifiers. The following research questions (RQ) were addressed:

- **RQ1:** Is Meta-Learning able to recommend the best drift detector for an unseen chunk of data?
- **RQ2:** Is the proposed Meta-Learning framework able to handle imbalanced data streams?
- **RQ3:** Is the proposed Meta-Learning framework able to handle real-world data stream difficulties?

A. Datasets

We selected a diverse set of benchmark data streams consisting of 10 streams from real-world domains and 6 streams generated artificially using the `river` environment [55]. This selection allowed us to evaluate the effectiveness of the proposed framework across a wide range of scenarios. By

TABLE III: Properties of artificial (top) and real-world (bottom) data stream benchmarks.

Dataset	Instances	Features	Classes	IR
RandomTree20	20,000	10	2	✓
RandomTree30	30,000	10	2	✓
RandomTree50	50,000	10	2	✓
RBF20	20,000	10	2	✓
RBF30	30,000	10	2	✓
RBF50	50,000	10	2	✓
amazon	8,000	30	2	6.13
coil2000	9,822	85	2	15.76
covtype	267,001	54	2	3.91
creditcard	284,807	30	2	577.87
electricity	45,312	8	2	1.35
nomao	34,465	118	2	2.50
poker	359,999	10	2	9.69
tripadvisor	18,569	30	2	2.76
twitter	9,090	30	2	5.36
weather	18,159	8	2	2.18

incorporating artificial data streams, we were able to control specific aspects such as class imbalance and concept drift injection. Considering the artificially generated streams, we selected two generators: Random Tree and Random RBF. For each generator, we created three streams with instance sizes of $20k$, $30k$, and $50k$ instances, resulting in a total of six streams. We then created nine possible scenarios for each stream by considering the same class imbalance and drift speeds configurations used to create the meta-dataset. These scenarios provided a diverse set of conditions for evaluating the proposed framework. Additionally, we included real-world streams that present challenging problems characterized by a mix of different learning difficulties [3]. Detailed properties of the data stream benchmarks can be found in Table III. The reported imbalance ratio is global, i.e., it does not consider fluctuations over time.

B. Experimental configuration

To implement our framework, it was necessary to determine the time window within which the meta-learner would predict the most appropriate drift detector. In our experimental setup, we utilized a fixed window size of $5,000$ instances for this purpose. The reported hyperparameters were chosen after empirical evaluation that can be verified in the github repository. The framework and respective classifiers were implemented using Python 3.8 and the `river` [55] package. Algorithms were run on a GNU/Linux cluster with 192 Intel Xeon cores, 6 TB RAM, and Centos 7.

C. Evaluation metrics and baselines

The evaluation process followed a similar approach to the definition of the meta-label. We assessed the performance of the Adaptive Hoeffding Tree (AHT) using the predicted drift detector, employing the G-Mean metric. The metrics were computed in a prequential manner [3], using a sliding window of 500 examples.

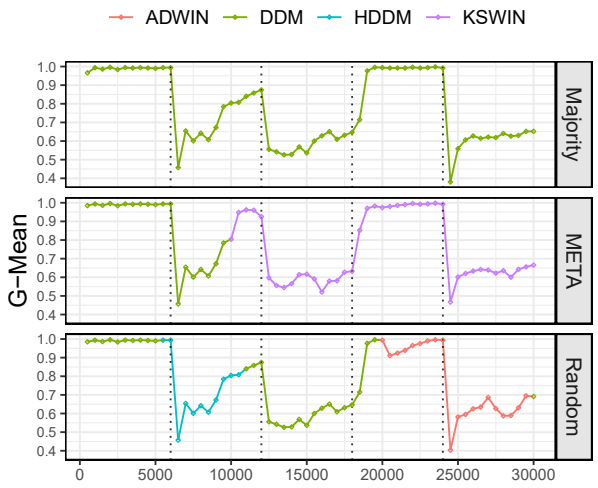

Fig. 3: Prequential G-Mean for the RandomTree generator in the `STABLE` scenario using three recommendation methods. Each line color corresponds to a different drift detector and dashed lines represent sudden concept drift points.

As baselines we used: (i) a model that always recommends the majority class for the whole dataset (`Majority`) and (ii) a model that provides random recommendations (`Random`) every $5,000$ instances. These baselines are widely used to endorse the need for a recommendation system [11].

V. Results

The results were organized by comparing the performance of the Meta-Learning framework (META) in specific generated scenarios and real-world data streams with the baselines of `Majority` and `Random` selection. This comparison allows us to gain insights into the scenarios where the evaluated recommendation models exhibit superior performance. Additionally, we present a meta-feature analysis to understand which characteristics of a data stream contribute to the selection of the most suitable drift detector.

Drift detector recommendation. Fig. 3 illustrates an example of how each recommendation method performs in practice. It demonstrates that META can effectively identify the most suitable drift detector for certain scenarios, while Random changes without any discernible criteria. The results for all methods on artificial stream generators are presented in Tables IV, V and VI. Each table represents a specific imbalance scenario, while within each table, the results for different drift speeds are shown.

In the `STABLE` scenario, it is observed that the Majority recommendation model outperforms the META and Random recommendation models only when the drift speed is 350, indicating a fast gradual drift. For slow gradual drifts, the Random recommendation model achieves the highest G-Mean value across all evaluated data streams. Turning to the `FLIPPING` scenario, the results obtained by all methods are generally

TABLE IV: STABLE scenario. Average G-Mean values for all streams and drift detector selection methods under different drift speeds. The best value for each drift speed is highlighted.

	Drift speed	1			350			1,000		
	Method	META	Majority	Random	META	Majority	Random	META	Majority	Random
Dataset	RandomTree20	**0.7545 ± 0.02**	0.7469 ± 0.02	0.7390 ± 0.03	**0.7487 ± 0.03**	0.7188 ± 0.01	0.7144 ± 0.05	0.7158 ± 0.03	0.7152 ± 0.01	**0.7194 ± 0.01**
	RandomTree30	**0.7786 ± 0.04**	0.7542 ± 0.01	0.7628 ± 0.04	0.7515 ± 0.05	**0.7606 ± 0.02**	0.7552 ± 0.02	**0.7498 ± 0.02**	0.7274 ± 0.06	0.7303 ± 0.03
	RandomTree50	0.7791 ± 0.02	0.7535 ± 0.05	**0.7907 ± 0.03**	**0.7804 ± 0.02**	0.7585 ± 0.05	0.7731 ± 0.02	0.7677 ± 0.03	0.7651 ± 0.04	**0.7694 ± 0.02**
	RBF20	0.7993 ± 0.05	0.8005 ± 0.06	**0.8014 ± 0.05**	0.7970 ± 0.04	**0.8038 ± 0.05**	0.7895 ± 0.05	0.7808 ± 0.03	0.7834 ± 0.04	**0.7894 ± 0.04**
	RBF30	0.8159 ± 0.04	0.8115 ± 0.04	**0.8252 ± 0.04**	0.8167 ± 0.05	**0.8178 ± 0.04**	0.8170 ± 0.02	0.8112 ± 0.05	0.8121 ± 0.05	**0.8134 ± 0.04**
	RBF50	**0.8341 ± 0.04**	0.8330 ± 0.04	0.8298 ± 0.04	0.8276 ± 0.04	0.8303 ± 0.05	**0.8409 ± 0.04**	0.8353 ± 0.05	0.8310 ± 0.05	**0.8357 ± 0.04**

TABLE V: FLIPPING scenario. Average G-Mean values for all streams and drift detector selection methods under different drift speeds. The best value for each drift speed is highlighted.

	Drift speed	1			350			1,000		
	Method	META	Majority	Random	META	Majority	Random	META	Majority	Random
Dataset	RandomTree20	**0.5045 ± 0.07**	0.4456 ± 0.09	0.4861 ± 0.09	**0.4933 ± 0.11**	0.4178 ± 0.15	0.4800 ± 0.07	**0.5165 ± 0.05**	0.4411 ± 0.11	0.4746 ± 0.11
	RandomTree30	0.4919 ± 0.08	0.4815 ± 0.07	**0.5290 ± 0.06**	0.4831 ± 0.08	0.4763 ± 0.11	**0.5431 ± 0.06**	0.4834 ± 0.08	0.4733 ± 0.09	**0.5601 ± 0.05**
	RandomTree50	0.5077 ± 0.07	0.5100 ± 0.07	**0.5309 ± 0.06**	0.4904 ± 0.08	0.4910 ± 0.07	**0.5844 ± 0.04**	0.4929 ± 0.07	0.5011 ± 0.07	**0.5476 ± 0.07**
	RBF20	**0.6476 ± 0.13**	0.6243 ± 0.13	0.6449 ± 0.11	0.6843 ± 0.07	0.6737 ± 0.09	**0.7136 ± 0.07**	0.7140 ± 0.10	0.7156 ± 0.08	**0.7500 ± 0.07**
	RBF30	0.6795 ± 0.10	0.6960 ± 0.11	**0.7186 ± 0.12**	0.7062 ± 0.09	0.6957 ± 0.09	**0.7335 ± 0.09**	0.7317 ± 0.11	0.7303 ± 0.11	**0.7520 ± 0.09**
	RBF50	0.7248 ± 0.11	0.6918 ± 0.09	**0.7508 ± 0.08**	0.7490 ± 0.13	**0.7582 ± 0.14**	0.7520 ± 0.09	0.7499 ± 0.11	0.7506 ± 0.11	**0.7707 ± 0.06**

TABLE VI: INCREASE-DECREASE scenario. Average G-Mean values for all streams and drift detector selection methods under different drift speeds. The best value for each drift speed is highlighted.

	Drift speed	1			350			1,000		
	Method	META	Majority	Random	META	Majority	Random	META	Majority	Random
Dataset	RandomTree20	**0.5707 ± 0.08**	0.5644 ± 0.09	0.5688 ± 0.12	**0.5997 ± 0.07**	0.5537 ± 0.12	0.5888 ± 0.08	**0.6090 ± 0.05**	0.5435 ± 0.08	0.5517 ± 0.13
	RandomTree30	0.6277 ± 0.06	0.6066 ± 0.06	**0.6332 ± 0.05**	0.5769 ± 0.06	**0.6007 ± 0.05**	0.5705 ± 0.07	**0.5961 ± 0.03**	0.5879 ± 0.05	0.5318 ± 0.06
	RandomTree50	**0.6335 ± 0.05**	0.6141 ± 0.08	0.6201 ± 0.03	0.6003 ± 0.08	0.6193 ± 0.07	**0.6461 ± 0.02**	0.6081 ± 0.05	0.6210 ± 0.05	**0.6516 ± 0.10**
	RBF20	**0.7135 ± 0.11**	0.7044 ± 0.08	0.7090 ± 0.12	0.7130 ± 0.10	0.7047 ± 0.08	**0.7275 ± 0.06**	**0.7103 ± 0.08**	0.6951 ± 0.12	0.6969 ± 0.11
	RBF30	0.7335 ± 0.10	0.7267 ± 0.08	**0.7520 ± 0.09**	0.7272 ± 0.09	0.7197 ± 0.10	**0.7306 ± 0.10**	**0.7305 ± 0.09**	0.7156 ± 0.08	0.7246 ± 0.10
	RBF50	0.7667 ± 0.09	0.7487 ± 0.09	**0.7944 ± 0.04**	0.7611 ± 0.09	0.7531 ± 0.09	**0.7718 ± 0.05**	0.7583 ± 0.09	0.7458 ± 0.10	**0.7742 ± 0.07**

worse compared to the previous scenario, highlighting its challenging nature. In this scenario, the Random recommendation model consistently exhibits the best average G-Mean value across most of the evaluated scenarios, indicating the instability of this particular scenario. For the INCREASE-DECREASE scenario, both the META and Random recommendation models yield the best results. When sudden drifts are present, the META model performs better in half of the evaluated scenarios, while the Random model performs better in the other half. Furthermore, for fast gradual drifts, the Random model displays superior performance, while for slow gradual drifts, the META model outperforms the others. In summary, the results demonstrate that both the Random and META recommendation models outperform the Majority model, suggesting that using different drift detectors in the same data streams, in scenarios with and without class imbalance, can lead to improved predictive performance compared to using a fixed detector. However, it is important to note that the average values for certain scenarios are similar, indicating the need for further analysis.

Furthermore, to evaluate the performance of each approach in different scenarios, we analyzed the distribution of rank frequencies obtained by each method across all evaluated scenarios. This analysis is presented in Fig. 4, with the Random Tree results shown on the left side and the RBF results on the right side. In the STABLE and INCREASE-DECREASE scenarios, META achieved the highest frequency as the best method for both generators. Additionally, in these scenarios, the Random recommendation outperformed the Majority approach, further highlighting the benefit of dynamically changing the drift detector. Concerning the FLIPPING streams, META achieved the highest rank position for nearly 50% of the time with the Random Tree generator. However, for the RBF generator, Majority performed the best followed by the Random recommendation. This discrepancy can be attributed to the inherent instability of the FLIPPING scenario.

Finally, to address **RQ1** and **RQ2**, we assessed the performance of META regarding the baselines by statistical tests. We used the Friedman test, with a significance level of $\alpha = 0.05$. Moreover, the Nemenyi post hoc test analyzes whether the performance of the two approaches is significantly different if their corresponding average ranks differ by at least a Critical Difference (CD) value. When multiple algorithms are compared, a graphic representation can be used to represent the results with the CD diagram, as proposed by Demšar [56].

META was compared to Random and Majority over all the datasets under different imbalanced scenarios. This analysis is shown in Fig. 5, using the results from the Nemenyi test. In this diagram, if the lines are connected, it means that they are similar, and there is no statistical difference. In the scenario without class imbalance, we observe a significant difference between META and the baselines, providing support for the effectiveness of the proposed method in improving performance under such conditions. Conversely, in the

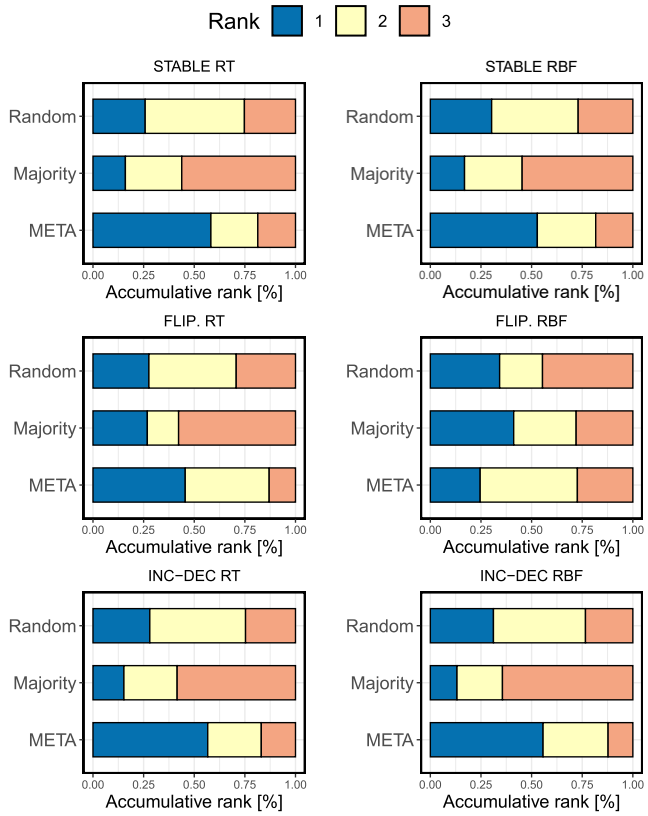

Fig. 4: Frequencies of ranks scored by each recommendation method in each of the evaluated scenarios.

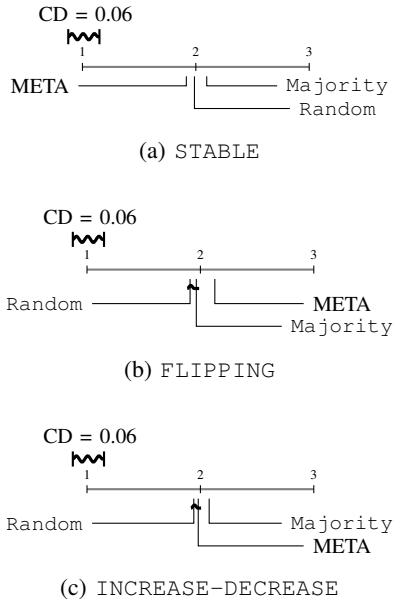

Fig. 5: Comparison of G-Mean of different recommendations algorithms according to the Nemenyi test in three class imbalance scenarios. Groups of algorithms that are not significantly different ($\alpha = 0.05$ and $CD = 0.06$) are connected.

FLIPPING scenario, Random and Majority outperformed META, which aligns with the earlier findings. Regarding the INCREASE−DECREASE scenario, there was no significant difference between Random recommendation and META. Nonetheless, both of these approaches performed better than the Majority baseline, further demonstrating the effectiveness of dynamically selecting drift detectors, even in imbalanced data streams.

Taking everything into consideration, we have successfully addressed **RQ1** by demonstrating that our proposed method effectively recommends the best drift detector for an unseen data chunk in scenarios without class imbalance. Furthermore, we can confidently state that META showcases the capability to handle certain imbalance scenarios without requiring a specific mechanism for skewed data. It outperforms a fixed drift detector when applied to the entire data stream, effectively addressing **RQ2**.

Real-world data streams. Firstly, it is crucial to distinguish between artificial and real-world imbalanced data streams. Generators rely on prior probabilities derived from the parametric imbalance ratio to generate instances, resulting in bounded windows where minority and majority instances appear. However, real-world datasets do not exhibit such clear probabilistic mechanisms as they are collected to model specific phenomena and do not adhere to strict priors. This presents unique challenges for classifiers, including the arrival latency of instances from a specific class and prolonged periods of instances from only one class.

With that in mind, Table VII presents the average G-Mean for each dataset and the respective drift recommendation model. It can be observed that META displayed the best results for six of the evaluated datasets. On the other hand, in contrast to the scenario with generators, Majority displayed better results than Random. This difference can be attributed to the unique challenges posed by real-world streams, such as noise that can be interpreted as concept drift and the difficulty in detecting well-defined moments of concept drift. This also highlights the robustness of DDM to these complexities.

Furthermore, to evaluate the performance of each approach in each dataset, we analyzed the distribution of rank frequencies obtained by each method across all evaluated scenarios. This analysis is presented in Fig. 7, comparing META to Majority (top) and to Random (bottom). When compared to Majority, META achieves the best G-Mean for at least 50% of the time in datasets like electricity and covtype, and more than half of the time in the weather dataset, even though Majority displays a higher average. In comparison to Random, META achieves less than 50% in only the tripadvisor dataset. These results demonstrate how META outperforms the baselines in almost every evaluated dataset, thereby addressing **RQ3**.

Meta-feature analysis. The importance of each meta-feature in the induction of the meta-models can be assessed using the RF Feature Importance metric. This metric is calculated by permuting the values of a feature in the Out-of-Bag (OOB)

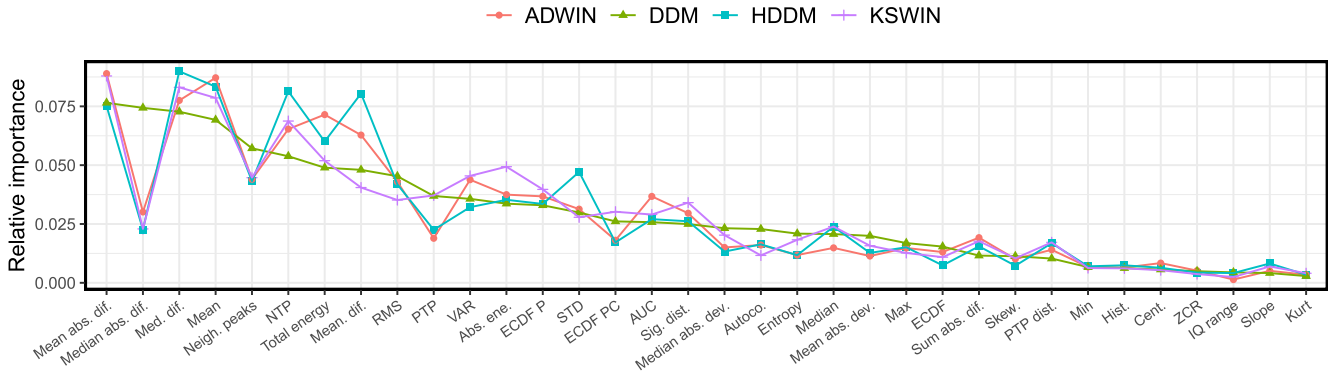

Fig. 6: Average relative importance of meta-features obtained from RF importance. Each line represents a model used to predict the ranking of the meta-labels. The X-axis is ordered by the importance of the Majority (DDM) model.

TABLE VII: Average G-Mean values for all real-world streams and drift detector selection methods. The best value for each dataset is highlighted.

Dataset	META	Majority	Random
amazon	**0.0538**	0.0263	0.0428
coil2000	**0.0890**	0.0332	**0.0890**
covtype	0.8160	**0.8443**	0.8226
creditcard	**0.2219**	0.0353	0.0547
electricity	0.8144	**0.8215**	0.8108
nomao	**0.3162**	0.3085	0.2947
poker-lsn-1-2vsAll	**0.1436**	0.1367	0.1389
tripadvisor	0.6408	0.6561	**0.6662**
twitter	**0.1053**	0.1006	0.0553
weather	0.6204	**0.6432**	0.6412

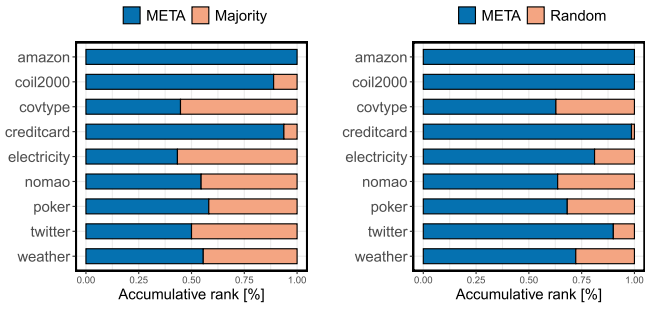

Fig. 7: Frequencies of ranks scored by each recommendation method in each of the real-world streams.

examples and reevaluating the OOB error. If replacing the values of a feature with random values leads to an increase in error, the feature is considered important. Conversely, if the error decreases, the resulting importance is negative, indicating that the feature is not important.

We employed the RF Feature Importance to examine the contribution of each feature in selecting the most suitable drift detector. Fig. 6 presents the importance of each meta-feature for each of the induced meta-models. The analysis reveals that the meta-models share a set of highly important features, including Histogram, Centroid, Median, Mean Difference, and Skewness. Conversely, features such as Autocorrelation, Absolute Energy, Peak to Peak distance, and ECDF are among the less important meta-features. This highlights the greater usefulness of Statistical-based characteristics of data streams in selecting a drift detector compared to Temporal-based features.

VI. Conclusion

In this paper, we addressed the challenges of learning from imbalanced data streams under concept drift while incorporating algorithm recommendations. We proposed an online Meta-Learning framework that dynamically recommends the most suitable drift detector for an unseen chunk of data. Through extensive experiments involving real-world and artificially generated data streams, we demonstrated the effectiveness of our framework compared to the baselines in a majority of the scenarios, both with and without class imbalance. The results of our experiments supported our hypothesis that dynamically changing the drift detector can significantly improve predictive performance, even when using a random recommendation. Notably, our Meta-Dataset consisted solely of synthetically generated data streams, highlighting the generalization capability of our framework.

These promising results motivate us to further investigate the integration of data streams and Meta-Learning. In our future work, we plan to explore the application of Meta-Learning for hyperparameter tuning, ensemble classifier selection, and adjusting Active Learning constraints. By expanding our research in these directions, we aim to enhance the practicality and performance of learning from imbalanced data streams under concept drift.

Acknowledgment

High Performance Computing resources provided by the High Performance Research Computing (HPRC) core facility at Virginia Commonwealth University (https://hprc.vcu.edu) were used for conducting the research reported in this work.

REFERENCES

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

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, 2014.

[3] G. Aguiar, B. Krawczyk, and A. Cano, "A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework," *Machine Learning*, 2023.

[4] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A survey on machine learning for recurring concept drifting data streams," *Expert Systems with Applications*, 2023.

[5] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, 1996.

[6] T. Lacombe, Y. S. Koh, G. Dobbie, and O. Wu, "A meta-learning approach for automated hyperparameter tuning in evolving data streams," in *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[7] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[8] B. Veloso, J. Gama, B. Malheiro, and J. Vinagre, "Hyperparameter self-tuning for data streams," *Information Fusion*, 2021.

[9] A. L. D. Rossi, A. C. P. de Leon Ferreira, C. Soares, B. F. De Souza *et al.*, "Metastream: A meta-learning based method for periodic algorithm selection in time-changing data," *Neurocomputing*, 2014.

[10] V. E. Martins, A. Cano, and S. B. Junior, "Meta-learning for dynamic tuning of active learning on stream classification," *Pattern Recognition*, 2023.

[11] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*, 2nd ed. Springer Verlag, 2009.

[12] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu, "Data stream analysis: Foundations, major tasks and tools," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2021.

[13] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[14] S. Wang, L. L. Minku, N. Chawla, and X. Yao, "Learning from data streams and class imbalance," 2019.

[15] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, 2016.

[16] A. Al-Shammari, R. Zhou, M. Naseriparsaa, and C. Liu, "An effective density-based clustering and dynamic maintenance framework for evolving medical data streams," *International Journal of Medical Informatics*, 2019.

[17] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, 2017.

[18] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[19] A. R. Masegosa, A. M. Martínez, D. Ramos-López, H. Langseth, T. D. Nielsen, and A. Salmerón, "Analyzing concept drift: A case study in the financial sector," *Intelligent Data Analysis*, 2020.

[20] Y. Sun, M. Li, L. Li, H. Shao, and Y. Sun, "Cost-sensitive classification for evolving data streams with concept drift and class imbalance," *Computational Intelligence and Neuroscience*, 2021.

[21] Ł. Korycki and B. Krawczyk, "Concept drift detection from multi-class imbalanced data streams," in *IEEE 37th International Conference on Data Engineering (ICDE)*, 2021.

[22] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, 2016.

[23] B. Krawczyk and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams," *Applied Soft Computing*, 2018.

[24] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational Intelligence Magazine*, 2015.

[25] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, 2018.

[26] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*, 2004.

[27] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *4th International Workshop on Knowledge Discovery from Data Streams*, 2006.

[28] R. S. M. de Barros, J. I. G. Hidalgo, and D. R. de Lima Cabral, "Wilcoxon rank sum test drift detector," *Neurocomputing*, 2018.

[29] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, 2020.

[30] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *SIAM International Conference on Data Mining*, 2007.

[31] D. T. J. Huang, Y. S. Koh, G. Dobbie, and R. Pears, "Detecting volatility shift in data streams," in *IEEE International Conference on Data Mining*, 2014.

[32] A. Pesaranghader and H. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases: European Conference*, 2016.

[33] A. Łapiński, B. Krawczyk, P. Ksicnicwicz, and M. Woźniak, "An empirical insight into concept drift detectors ensemble strategies," in *IEEE Congress on Evolutionary Computation*, 2018.

[34] P. Sobolewski and M. Woźniak, "Comparable study of statistical tests for virtual concept drift detection," in *8th International Conference on Computer Recognition Systems*, 2013.

[35] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[36] A. Liu, J. Lu, F. Liu, and G. Zhang, "Accumulating regional density dissimilarity for concept drift detection in data streams," *Pattern Recognition*, 2018.

[37] D. K. Antwi, H. L. Viktor, and N. Japkowicz, "The perfsim algorithm for concept drift detection in imbalanced data," in *IEEE 12th International Conference on Data Mining Workshops*, 2012.

[38] S. Wang and L. L. Minku, "Auc estimation and concept drift detection for imbalanced data streams with multiple classes," in *International Joint Conference on Neural Networks (IJCNN)*, 2020.

[39] J. Vanschoren, "Meta-learning: A survey," *arXiv preprint arXiv:1810.03548*, 2018.

[40] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. De Carvalho, "Hyper-parameter tuning of a decision tree induction algorithm," in *5th Brazilian Conference on Intelligent Systems (BRACIS)*, 2016.

[41] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976.

[42] I. Khan, X. Zhang, M. Rehman, and R. Ali, "A literature survey and empirical study of meta-learning for classifier selection," *IEEE Access*, 2020.

[43] G. J. Aguiar, E. J. Santana, A. C. de Carvalho, and S. B. Junior, "Using meta-learning for multi-target regression," *Information Sciences*, 2022.

[44] A. L. D. Rossi, C. Soares, B. F. de Souza, A. C. P. de Leon Ferreira *et al.*, "Micro-metastream: Algorithm selection for time-changing data," *Information Sciences*, 2021.

[45] G. F. C. Campos, S. M. Mastelini, G. J. Aguiar, R. G. Mantovani, L. F. d. Melo, and S. Barbon, "Machine learning hyperparameter selection for contrast limited adaptive histogram equalization," *EURASIP Journal on Image and Video Processing*, 2019.

[46] G. J. Aguiar, R. G. Mantovani, S. M. Mastelini, A. C. de Carvalho, G. F. Campos, and S. B. Junior, "A meta-learning approach for selecting image segmentation algorithm," *Pattern Recognition Letters*, 2019.

[47] I. Gabbay, B. Shapira, and L. Rokach, "Isolation forests and landmarking-based representations for clustering algorithm recommendation using meta-learning," *Information Sciences*, 2021.

[48] R. G. Mantovani, A. L. Rossi, J. Vanschoren, B. Bischl, and A. C. Carvalho, "To tune or not to tune: recommending when to adjust svm hyper-parameters via meta-learning," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[49] R. G. Mantovani, A. L. Rossi, E. Alcobaça, J. Vanschoren, and A. C. de Carvalho, "A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers," *Information Sciences*, 2019.

[50] G. M. Tavares and S. B. Junior, "Process mining encoding via meta-learning for an enhanced anomaly detection," in *European Conference on Advances in Databases and Information Systems*, 2021.

[51] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Having a blast: Meta-learning and heterogeneous ensembles for data streams," in *IEEE International Conference on Data Mining*, 2015.

[52] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Systems with Applications*, 2019.

[53] A. Bifet and R. Gavalda, "Adaptive learning from evolving data streams," in *Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis*, 2009.

[54] L. Breiman, "Random forests," *Machine learning*, 2001.

[55] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem *et al.*, "River: machine learning for streaming data in python," *The Journal of Machine Learning Research*, 2021.

[56] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, 2006.