Hindawi
Mobile Information Systems
Volume 2021, Article ID 5549300, 17 pages
https://doi.org/10.1155/2021/5549300

Research Article

ADES: A New Ensemble Diversity-Based Approach for Handling Concept Drift

Tinofirei Museba,[1] **Fulufhelo Nelwamondo**,[2] **and Khmaies Ouahada**[2]

[1]*Department of Applied Information Systems, University of Johannesburg, Johannesburg, South Africa*
[2]*Department of Electrical and Electronic Engineering Sciences, University of Johannesburg, Johannesburg, South Africa*

Correspondence should be addressed to Tinofirei Museba; tmuseba@uj.ac.za

Received 4 February 2021; Revised 26 April 2021; Accepted 11 May 2021; Published 1 June 2021

Academic Editor: Yugen Yi

Beyond applying machine learning predictive models to static tasks, a significant corpus of research exists that applies machine learning predictive models to streaming environments that incur concept drift. With the prevalence of streaming real-world applications that are associated with changes in the underlying data distribution, the need for applications that are capable of adapting to evolving and time-varying dynamic environments can be hardly overstated. Dynamic environments are nonstationary and change with time and the target variables to be predicted by the learning algorithm and often evolve with time, a phenomenon known as concept drift. Most work in handling concept drift focuses on updating the prediction model so that it can recover from concept drift while little effort has been dedicated to the formulation of a learning system that is capable of learning different types of drifting concepts at any time with minimum overheads. This work proposes a novel and evolving data stream classifier called Adaptive Diversified Ensemble Selection Classifier (ADES) that significantly optimizes adaptation to different types of concept drifts at any time and improves convergence to new concepts by exploiting different amounts of ensemble diversity. The ADES algorithm generates diverse base classifiers, thereby optimizing the margin distribution to exploit ensemble diversity to formulate an ensemble classifier that generalizes well to unseen instances and provides fast recovery from different types of concept drift. Empirical experiments conducted on both artificial and real-world data streams demonstrate that ADES can adapt to different types of drifts at any given time. The prediction performance of ADES is compared to three other ensemble classifiers designed to handle concept drift using both artificial and real-world data streams. The comparative evaluation performed demonstrated the ability of ADES to handle different types of concept drifts. The experimental results, including statistical test results, indicate comparable performances with other algorithms designed to handle concept drift and prove their significance and effectiveness.

1. Introduction

The assumption with most classification problems is that the data-generating mechanism is stationary and the data are drawn from a fixed and unknown probability distribution [1]. However, in the real world, data are streaming and evolving over time in dynamic and time-varying environments and the traditional assumptions of data independence and stationary distributions violate this assumption. In addition to the voluminous nature of the data, the streaming data are associated with concept drift. Concept drift in streaming data means that the target concept to be predicted by the learning model may evolve with time. The underlying data distribution or the target concept that the learning model is trying to predict from the streaming data is constantly changing. The changes in the underlying distribution of the streaming data can be abrupt, gradual, incremental, cyclical, and even intersected. Apart from drifting concepts, the volume of the streaming data is overwhelming and associated with high speed. For most streaming applications, changes in the target variable occur more often. The presence of concept drift in streaming data adversely affects prediction performance of the machine learning algorithm over time [2]. For classifiers to be capable of adapting to different types of concept drifts in dynamic environments, there is need to continuously revise and refine

approximation models and update machine learning algorithms as unseen observations become available. This makes the results obtained from experiments performed using incremental and batch processing to match those of online learning by exploiting the availability of relevant knowledge in streaming data [3]. For most real-world applications, concept drift can occur in a cyclic fashion. A typical example is that of weather prediction where a pattern can drift from one season to another but the pattern will eventually recur. Although previously learned concepts can be used to handle cyclically drifting concepts that return to previously visited states, past observations may become irrelevant and can even hurt the predictive performance of the learning algorithm. Classifiers that may be relevant for cyclically drifting concepts need to be stored and reused, thus allowing learning models to precisely capture time-evolving trends in dynamic environments to make critical predictions. For applications that exhibit concept drift, it is important to identify which learned information is useful for the currently occurring concept and to keep track of changes occurring in the underlying distribution of the data. Adaptive Diversified Ensemble Selection (ADES) introduces new perspectives to the problem of accurately and timely adapting to any type of concept drift in nonstationary environments with minimum computational overheads at any given time. The ADES approach basically involves selecting an ensemble that is optimal and representative of the current concept using the metrics of accuracy and diversity, and creating ensembles of high and low diversity since different types of drifting concepts require different amounts of diversity. The ADES algorithm stores previously learned concepts. This makes the ADES algorithm capable of adjusting its hypothesis to new concepts and make use of previously learned concepts to handle recurring concepts.

Current approaches to handle concept drift are concerned with updating ensembles of learning machines so that they can detect and quickly recover from concept drift and improve generalization of the learning model, and less work has been devoted to the investigation of what type of prediction model is most suitable to learn different types of concepts at any given time with minimum overheads.

To fill this gap, we propose an online heterogeneous ensemble that selects an optimal ensemble using diversity and accuracy as metrics, and creating ensembles of different levels of diversity. The ensemble-based machine learning algorithm can provide an optimal solution to problems of handling all types of concept drift regardless of the speed, severity, and possible recurrences of previously visited states. To validate the proposed hypothesis, we evaluated the prediction performance of ADES together with three well-known ensemble approaches to handle eight classification problems that exhibit concept drift using artificial and real-world data streams. Empirical experiments conducted demonstrate that ADES adapts to all types of drifting concepts through the selection of an optimal ensemble classifier consisting of high or low diversity classifier that is representative of the current concept. We show that ADES can perform well in concept drift scenarios with minimum overheads as classifiers with the same pattern recognition ability are separated. The experiments conducted on concept drift scenarios show that adaptation to different types of concept drift require different levels of diversity, and timeously adapting to recurring concepts requires the storage of previously learned knowledge.

The ADES machine learning algorithm was carried out using the Java programming language by customizing the Massive Online Analysis [4] framework [5].

The remainder of this work is organized as follows: Section 2 provides a formal description of the problem of concept drift. Section 3 provides a review of previous work on the concept of drift problem. Section 4 provides an overview of diversity, and Section 5 presents a description of selection metrics, Section 6 provides a description of the datasets used and experiments conducted, Section 7 provides a description of the drift detection method and the empirical experiments conducted.

2. Concept Drift

Ensembles of learning machines are often applied to real-world applications where data are streaming and often collected over an extended period of time. Typical examples include Cyber Security, Sentiment Analysis, Stock Market Prediction, and Human Activity Recognition. In time-varying and dynamic environments, data evolve over time and the process of performing data analysis must be performed as soon as data are obtained. Target variables to be predicted often change over time, and approximation learning models need to be refined and updated to adapt to changes in the underlying data distribution. For example, for credit card holders, the possibility of defaulting on payment may change due to an economic crisis. The changes in the underlying distribution of the data are known as concept drift [6]. In dynamic environments, concepts are not stable but change with time as characteristics of the environment change. For streaming data, concept drift is a prevalent property and makes learning data streams unpredictable. The presence of concept drift adversely affects the generalization performance of machine learning algorithms designed for classification and clustering purposes as the number of misclassification errors increase. To avoid making wrong decisions on the results obtained, it is important to identify changes in the streaming data to get accurate results and update the model accordingly [7]. For any type of concept drift that is currently occurring, a classification problem is well expressed by the use of prior probabilities and class-conditional probability distributions of the data that are sampled from it. When streaming data exhibit concept drift, prior probabilities and class-conditional probabilities are also considered to be a function of time. A model that learned the concept at a specified time t may change its target classes at step $t+1$, which reflects a change in the decision boundary separating the classes. This decision boundary will no longer correctly describe or be compatible with the full target concept space and the concept currently occurring at $t+1$. For most real-world applications, concept drift may occur abruptly. A drift is abrupt when the changes occur in a sudden way and the old concept

is replaced completely by the new concept in a single iteration. With gradual concept drift, a new concept slowly takes over the current concept and several steps are required for the impending concept to fully reflect. In incremental drift, the occurrence is gradual and the change from one distribution to another occurs smoothly and the change that occurs is continuous. Under some scenarios, concept drift occurs in a cyclic fashion. Recurring concept drift is a type of concept drift where previously observed states may appear in the future, describing a previously encountered concept. For applications associated with seasonal changes, cyclically drifting concepts exhibit a tendency to return to previously visited states. A typical example is that of weather forecasting where a pattern drifts from one season to another but at the end the pattern will eventually recur. The biggest challenge posed by concept drift in voluminous streaming data where the concept to be learned is not known in advance is that the time it takes to make a prediction may increase indefinitely thereby introducing a bottleneck. Learning models adaptation to new concepts is an important task for classification problems in dynamic and nonstationary environments.

2.1. Related Work. For many real-world applications associated with drifting concepts, a number of contemporary machine learning algorithms have been suggested to handle drifting concepts. Typical examples of applications that suffer concept drifts include information filtering systems that predict the user's reading preferences, a recommender or advertising system where the behavior of customers can change depending on the time of the year, inflation and new products made available, spam detection and Network intrusion detection.

A novel hybrid ensemble classifier called the Knowledge-Maximized Ensemble (KME) that handles different types of drift and combines components of online and chunk-based ensembles with limited labeled observations was proposed by Ren et al. [3]. The algorithm can only handle gradual and incremental drifts. The weighting mechanism makes it slow for the algorithm to reflect new concepts. The algorithm does not consider the importance of ensemble diversity, which is a cornerstone to the success of every ensemble classifier. Accuracy Weighted Diversity-based Online Boosting [8] is based on the concept of Diversity-Adaptation-based Online Boosting (ADOB) and other modifications. The algorithm introduces a new way of computing the weights of recently generated classifiers and calculates the sums of correctly and incorrectly classified instances by all classifiers. The algorithm has no pruning strategy and the pool of classifiers may grow indefinitely. Kappa Updated Ensemble (KUE) [9] aggregates online ensembles together with block-based ensembles and integrates the Kappa Statistic to perform dynamic weighting and to select the base classifiers. Base classifiers are trained using random feature subsets to achieve diversity, and new instances are used to update base classifiers with new observations using a probability that follows a Poisson distribution. KUE base classifiers can abstain and reduce the possibility of incompetent classifiers to make the final decision at the time of voting. The Iterative Boosting Streaming Ensemble (IBS) [10] applies boosting to previously unseen observations to maintain ensemble accuracy and adds learners to the ensemble with time. If the ensemble accuracy is low, IBS boosts performance adding classifiers and keeps a few classifiers in the pool if the prediction accuracy is high. The strategy helps the IBS algorithm to recover quickly in the event of concept drift. Handling recurring concepts with IBS is computationally expensive as it requires a large pool of classifiers. The Dynse framework [11] uses the dynamic selection of classifiers to handle drifting concepts. The frame is flexible allowing it to handle a variety of problems since it assumes that some supervised batches will be made available over time for training new classifiers and for estimating the classifier's competence. The framework uses Leveraging Bagging and the diversity from the pool of classifiers generated is limited, but different kinds of concept drift require different amounts of diversity. The Dynse framework removes models with the least accuracy considering the recent estimation window accuracy making it unable to handle recurrent and predictable drifts. The Diverse Discrimination Tracker (DEDT) [12] keeps a pool of diverse classifiers for all labeled samples and performs a query on labels that are disputed but informative. DEDT has no pruning strategy in place and no technique to control the amount of diversity for each type of concept drift. Bhatnagar et al. [13] suggested a linear classifier ensemble that uses accuracy and diversity of classifiers as metrics and the two metrics are the features attributed to the success of an ensemble classifier. The proposed approach has no mechanisms of introducing and controlling diversity. The Adaptive Online Learning Rule (SOAR) [14] handles drifting concepts and optimizes generalization performance by including different levels of classifier diversity. SOAR assigns each classifier a weight and uses an adaptive window in order to handle different types of concept drifts. Experimental results obtained show that SOAR handles sudden and gradual concept drifts well. SOAR can only handle two types of drifts and assigning weights to classifiers makes it slow to reflect new concepts. Idrees et al. [15] proposed a novel Heterogeneous Weighted Majority Algorithm (HDWM) (2018) that intelligently interchanges different classifiers in the ensemble to optimize the generalization performance of the ensemble in nonstationary environments. The algorithm maintains a dynamic pool of classifiers and inserts and deletes unseeded models according to their prediction performance. Classifiers are assigned weights and it takes them time to reflect new concepts. They not take into account the fact that different levels of diversity are required for each type of drift, taking into account the speed of change and the severity or magnitude of change.

Ensembles of classifiers have been used also for text classification in both static and dynamic domains. Onan [16] proposed a hybrid supervised clustering-based ensemble scheme for text classification that takes into consideration the feature of diversity. The clustering-based ensemble works well for static domains and cannot be transferred to dynamic environments. Onan et al. [17] proposed a hybrid ensemble pruning scheme based on clustering and randomized search for text sentiment classification. The

proposed approach is not suitable for handling recurrent or predictable drifts. Onan et al. [18] proposed a text classification ensemble algorithm called a multi-objective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. The ensemble was tested on applications such as credit risk modeling, spam filtering, and software defect prediction. The proposed approach is based on an optimization technique that uses a multi-objective differential evolution algorithm making the convergence to new concepts and recovery from concept drift slow.

Despite a plethora of existing ensemble methods to handle concept drifts in data streams associated with concept drift, none of the existing ensemble algorithms handle all types of concept drift and offer good generalization performance in both stable and unstable phases of the learning process. Existing ensemble algorithms rarely exploit ensemble diversity to improve generalization. Without loss of generality, the cornerstone to the success of an ensemble classifier is hugely attributed to ensemble diversity. No definition or measure of diversity is provided in previous works that exploit diversity to handle drifting concepts. Different types of concept drift, depending on the speed of drift, severity or magnitude of change, require different amounts of diversity. Although ensembles of learning machines have been used to handle concept drift, there exists no study of why they are capable of handling concept drifts and which of their features plays an important role in handling different types of drifts.

This paper proposes a new adaptive ensemble that exploits diversity to handle different types of concept drift at any given time using accurate and diverse classifiers representative of the current concept. Learning in nonstationary environments requires the creation of simple but quickly trained linear models that are very fast to evaluate. An avalanche of data-streaming applications makes the area of learning in nonstationary environments that exhibit concept drift increasingly important. Approaches to handle concept drifts in the literature are concerned with how to quickly detect and adapt to the current concept drift. Although various types of prediction models can provide a very different prediction performance depending on the problem being learned, not much work has been dedicated to the investigation of what type of prediction model is most adequate over time in nonstationary environments. In dynamic nonstationary environments associated with concept drift, time to perform resampling of the data is limited when training classifiers, generally precluding the use of bagging, boosting, or related methods that resample training data. Therefore, this paper proposes the Adaptive Diversified Ensemble Selection algorithm, an online ensemble learning algorithm for nonstationary environments. ADES automatically chooses the diverse and accurate base models to be used over time in nonstationary environments. This enables the algorithm to use diverse and accurate base models and use them to improve prediction performance to handle concept drift. Using minimum resources, ADES handles concept drift at any given time, recovers quickly from drift and converges to new concepts accurately.

3. Ensemble Diversity in Nonstationary Environments

Adaptation to drifting scenarios requires different levels of diversity given that some drifts maybe severe, intersected and, recurring. Furthermore, different levels of diversity are required at the start and after a drift occurred in order to improve the prediction performance and optimize convergence to both new and old concepts, thereby providing a faster recovery from concept drift. To accurately adapt to different types of concept drifts with ensemble classifiers, it is imperative to design ensembles that encourage different amounts of diversity for different types of concept drifts. Since drifting concepts can be gradual, abrupt, or recurring, adaptation to each type of drift requires different amounts of diversity depending on the speed, rate of change, and severity or magnitude of change. Unmeasured appropriation of amounts of diversity can hurt the performance of the ensemble learning model in nonstationary environments. Abrupt changes result in complete changes of the decision boundary separating the two classes for most classification problems. To encourage the generation of high and low diversity, the measure of diversity employed must be associated to the voting margin in order to also optimize the rate of adaptation to new concepts. Intuitively speaking, the key to the success of an ensemble classifier is that the base classifiers perform diversely [19]. In online learning, different levels of diversity can be explicitly encouraged in an ensemble classifier by using a modified version of Adaptive Boosting (Adaboost) [20]. Online Adaboost is an iterative algorithm that trains instances sequentially. The success of Adaboost is attributed to its ability to maximize the margin. If the margin is increased, the diversity also increases, and classification confidence increases and classification errors are greatly reduced. Online Adaboost computes an example weight in a way that corresponds to the weighting procedure of batch Adaboost instead of using the number 1 as the parameter to the Poisson distribution as in Online Bagging, thereby achieving a good approximation. The Adaboost algorithm uses a proxy parameter. The inclusion of the proxy parameter λ for the Poisson distribution instead of enforcing $\lambda = 1$ encourages different amounts of diversity.

The use of Adaboost and modifying the algorithm to include parameter λ for Poisson λ distribution generates higher or lower average values of λ to match the values of the Q statistic. With different amounts of diversity available for each concept drift, the rate of adaptation and convergence to new concepts is improved. Figure 1 provides a description of the Online Adaboost.

3.1. Ensemble Selection. Ensemble selection adopted for this work is based on the notion of Dynamic Ensemble Selection (DES). Given a pool of classifiers, the objective of the Dynamic Ensemble Selection is to select classifiers that are competent in a particular region of the feature space to learn unseen instances. Classifiers are selected based on their ability to predict a label using the validation dataset. An optimal ensemble with competent and diverse classifiers is

Input: Training dataset (x_i, y_i), W: Initial weights, H: Classifier WeakLearn, number of learners T

(1) $W_i \leftarrow W$
(2) $\xi_C \leftarrow \xi_w \leftarrow 0$
(3) For $t = 1$ to T do
(4) Trains $WeakLearn_t$ using Poisson (λ_t)
(5) Receives hypothesis h_t: $X \longrightarrow Y$
(6) $\xi c_i \leftarrow \xi c_t + w_t * [h_t(x_i)\, y_i]$
(7) $\xi w_t \leftarrow \xi w_t + w_t * [[h_t(x_i) \neq y_i]]$
(8) $CW_t \leftarrow \xi C_t + \xi W_i$
(9) $\xi_t \leftarrow \xi W_i / CW_i$
(10) if $\xi_t > (1/2)$ then
(11) | break
(12) end
(13) $\beta_t \leftarrow \varepsilon_t / (1 - \varepsilon_t)$
(14) $W_{t-1} \leftarrow (W_t \times \beta_t^{1-[h_t(x_i) \neq y_i]}) / ((\xi C_t / CW_i) \times \beta_t^1 + (\xi w_t / CW_t)\beta_t^0)$
(15) *end*
(16) Output: $h_f(x) \leftarrow \text{argmax}_{y \in Y} \sum_{t=1}^{T} (\log(1/\beta_t) * [[h_t(i) = Y]])$

Figure 1: Online adaboost.

created to predict the streaming data label. In concept drifting environments, models learned from previous concepts may have different accuracies and diversity levels when they are applied to the concept currently occurring. Ensemble selection is then used to average the prediction outputs of only those classifiers whose accuracy reflects the current concept and the diversity is better than other previously learned models. The rationale behind DES is to create a dynamic ensemble of classifiers that specialize in different kinds of concept drift.

For most machine learning algorithms, the combined prediction output of the ensemble classifiers usually adapts to a drifting concept well before all the models that constitute the ensemble are consulted, optimizing adaptation time and reducing computational overheads. For each test instance, the learning algorithm consults every model in the pool of classifiers, which is often time-consuming in an online dynamic environment associated with drifting concepts. Ensemble selection involves selecting models from a prebuilt library of models based on some criteria [21]. This is different from other ensemble approaches that maintain a single, dynamic ensemble of classifiers in memory. In contrast, our approach, ADES, selects the ensemble from the library based on accuracy and level of diversity based on the concept currently occurring, and this cache ensemble is applied to the current concept. To keep the pool of classifiers refreshed and representative of the current concept, ADES adds new classifiers from new data chunks in the pool in order to guarantee that the pool contains at least one classifier trained with the latest information received [15]. Since new classifiers are added to the pool over time, ADES incorporates a classifier pruning strategy in order to keep the pool of classifiers from increasing indefinitely. Originally envisioned for static domains, adaptation of ensemble selection to dynamic environments associated with drifting concepts is novel. The dynamic classifier selection strategy is based on the notion of group method for data handling (GMDH) [22] for dynamic classifier ensemble selection. ADES is a group of methods for data handling (GMDH) based on dynamic classifier ensemble selection that evaluates a fitness function composed of two important metrics in ensemble learning: accuracy and diversity [22]. The fitness function is evaluated on the nearest neighbors and is defined for ψ_x by

$$\text{fitness}(\psi_x) = d^2(\psi_x) + \lambda x\ DF_{av}(\psi_x), \tag{1}$$

where $d^2(\psi_x)$ measures the overall accuracy on the nearest neighbors and DF_{av} measures the average pairwise diversity within the sub-ensemble ψ_x and the estimated optimal solution corresponds to the optimal sub-ensemble. The approach tends to give nice performances especially when there is some presence of noise in the data and faster recovery from drifts and faster convergence to concept drift.

The pruning strategy used by ADES is based on the concept of One Nearest Neighbor (1NN), which is capable of maintaining a diverse pool that is able to adapt to different kinds of drifts. The One Nearest Neighbor (1NN) was selected to accurately estimate similarity between batches in the algorithm due to its high dependency on the feature space and class-conditional probability of its training set. The algorithm compares the training sets of each pair of classifiers in the pool and the classifier with the same discriminative capability with which the recent classifier in the pool is removed. The description of the algorithm is depicted in Figure 2 and parameters representing recent pool of classifiers, the most recent classifier created, and the maximum pool size are passed to the algorithm.

3.2. Detecting Concept Drift. A drift detection mechanism provides information regarding the type of drift currently occurring and the speed, magnitude, or severity of drift. The ADES algorithm generates different levels of diversity, that is, high- and low-level diversity. ADES is composed of two ensembles, one with high diversity and the other with low diversity. Before a drift is detected, we use the low-diversity

Input: Recent_Pool, Latest_Classifier, maxSize
Output: Trimmed_Pool TP
(1) $L \longleftarrow \varnothing$
(2) TP $\longleftarrow \varnothing$
(3) For classifier C_i in Recent_Pool do
(4) For each classifier C_j in Recent_Pool do
(5) If C_i != C_j != Latest_Classifier then
(6) One_Nearest_Neighbour_Classifier $\longleftarrow C_i$.trainSet
(7) Acc $\longleftarrow$ test(One_Nearest_Neighbour_Classifier, C_j,trainSet
(8) L $\longleftarrow L \bigcup$ {acc, C_i, C_j}
(9) End
(10) End
(11) End
(12) $t \longleftarrow$ Recent_Pool.Size − maxSize
(13) While $t > 0$ do
(14) {acc, C_i, C_j } $\longleftarrow$ highest_Accuracy(L)
(15) L $\longleftarrow L \varnothing$ {acc, C_i, C_j}
(16) If C_j not in TP then
(17) If C_i not in TP then
(18) TP $\longleftarrow$ TP $\bigcup C_j$
(19) $t \longleftarrow t - 1$
(20) End
(21) End
(22) End
(23) TP $\longleftarrow P \varnothing$ TP

Figure 2: The ADES pruning strategy.

ensemble to make predictions for it is more accurate on new concepts and converges faster to new concepts, thereby improving generalization performance. To detect concept drift and identify the speed, severity, or type of drift, ADES uses the Early Drift Detection Method (EDDM) [23]. EDDM is based on monitoring both low- and high-diversity ensembles. When a drift is detected, ADES creates ensembles of low and high diversity from the diversity measure. The algorithm starts to predict with an ensemble of low diversity previously learned to improve convergence to new concepts. Previously learned knowledge allows the process of learning recurring concepts faster. ADES does not apply a weighting mechanism to models since weights slow the reflection of new concepts. ADES maintains two ensembles of high and low diversity in memory in order to adapt to different types of concept drift. The presence of previously learned models in the pool makes ADES suitable to handle recurrent and predictable drifts. If a drift is detected, a change in the levels of diversity is required and this makes ADES more robust to false alarms as opposed to learning models that reset the entire system when drift is detected and brings with it a faster recovery from drifts and faster convergence to new concepts.

3.3. Adaptive Diversified Ensemble Selection (ADES). The Adaptive Diversified Ensemble Selection (ADES) trains classifiers over time with labeled available data and estimates the accuracy of the classifier for a test instance using an accuracy measure. To optimize convergence to new concepts, a diversity measure is introduced. The algorithm creates a small subset of an ensemble based on the accuracy and diversity of the classifier. Classifiers with the same discriminative capability are separated and one is removed from memory to reduce computational overheads. The Adaptive Diversified Ensemble Selection algorithm uses accuracy as a metric to evaluate classifier performance. The accuracy metric works well when evaluating classifier performance as it is a probability metric suitable when the output of a classifier falls within the interval [0,1] and the target class labels are also in the interval [0,1]. The success of an ensemble of classifier is based on the intuition that base classifiers perform diversely. To measure diversity, ADES uses the Yule's Q Statistic [24] to minimize the error of the ensemble. Yule's Q Statistic was selected for its simplicity and ease of interpretation [19].

We provide a description of the Adaptive Diversified Ensemble Selection (ADES) (Figure 3).

The ADES algorithm defines a simple procedure to exploit the diverse ensembles generated by the modified AdaBoost in nonstationary learning environments. ADES maintains two ensembles of high and low diversity and starts learning with a low ensemble and trains and updates both. If training is started with a high diverse ensemble, another concept drift might be detected before the high diverse ensemble is sufficiently trained. Starting to train with previously learned ensembles of low diversity improves convergence to new concepts. A change detection algorithm monitors the predictions of the low diversity ensemble and when a concept drift is detected by the measure of a drop in prediction of the subsequent data chunk, the drift detection mechanism is invoked and if real concept drift is confirmed, the low-diversity ensemble is replaced by the diverse

Input: D: Data Stream
: N: Learning algorithms
: h_t : Current ensemble
H_{low} Low diversity ensemble
H_{high} High diversity ensemble
DDM Drift Detection Method
(1) Stable_mode ← before drift
(2) For I = 1 to N do
(3) Prediction ← $h_t(d)$
(4) Compute accuracy and diversity on d_i
(5) Update H_{low} and H_{high}
(6) Eliminate duplicates
(7) Train H_{low} and H_{high} on (x_i, y_i)
(8) Compute accuracy
(9) If $\text{Acc}_{\text{after}} < \text{Acc}_{\text{before}}$ then
(10) DDM ← DetectDrift
(11) If change detected then
(12) H_{low} ← duplicate H_{high}
(13) Update H_{low} and H_{high}
(14) Endif
(15) Endfor
(16) Endfor

Figure 3: Adaptive diversified ensemble selection (ADES).

ensemble. To prepare for recurring concepts and to accommodate drifting concepts of different severity and magnitude, there is a need to ensure the availability of a trained diverse ensemble that can be invoked if change occurs.

4. Datasets

To evaluate the behavior of ADES in the presence of concept drift, we conducted experiments using four artificial data streams and four real-world data streams. These datasets are chosen because they are mostly used in the literature to evaluate ensemble algorithms designed to handle concept drifts. Out of 8 datasets, 4 are artificial datasets and 4 are real-world datasets. When analyzing the prediction performance of an algorithm, using real-world datasets, it is not possible to know exactly when or at what point a drift starts to occur, which type of drift is currently occurring, or even if drift exists in the data streams. Under such a scenario, it is difficult to perform a detailed analysis of the behavior of algorithms when they encounter drifting concepts and to know when a concept drift is currently occurring using only real-world datasets. To perform a proper analysis of the behavior of ADES in concept drifting scenarios and to assist the analysis of the generalization ability of ADES, we first used 4 artificial datasets. To reaffirm the prediction performance analysis of ADES, empirical experiments were conducted using 4 real-world datasets.

4.1. Artificial Datasets

(i) Moving Hyperplane Dataset [5]

The Hyperplane was firstly used to test the prediction performances of decision tree algorithms. The weights are modified to change the orientation and position of the hyperplane thereby simulating a drift. A dataset of 1,000,000 instances and 10 attributes was generated. Incremental drifts are simulated by changing the weight by 0.1 for every training example and adding 5% noise.

(iii) LED Dataset [25]

For the LED dataset, 1,000,000 instances were generated and the dataset exhibits sudden concept drifts. Important attributes are interchanged thereby simulating concept drift. The objective is to predict the digit displayed on a seven segment LED display.

(v) Waveform Dataset [25]

In the Waveform dataset, the idea is to predict the target of the Waveform types that consist of three sets. The Waveform dataset exhibits sudden concept drift.

(vii) STAGGER Dataset [26]

The Stagger dataset [26] generates instances with categorical values and uses rules to assign class labels. The dataset contains three nominal attributes namely: size = {small, medium, large}, color = {red, green, blue}, and shape = {circle, square, triangle}. Changing the items in the rules simulates concept drift.

4.2. Real-World Datasets. To affirm the analysis of HDES-AD, we performed experiments using four real-world datasets. These datasets are the widely used datasets in machine learning research to evaluate the effectiveness of algorithms designed to handle different kinds of drift.

(i) Airlines Dataset [4]

With the Airlines dataset, the goal is to predict which flight is delayed or not according to the scheduled departure. The dataset is composed of two classes, delayed or on schedule, and it contains 539383 instances with seven attributes.

(iii) Covertype Dataset [27]

The dataset consists of the observations extracted from the Resource Information system data. The dataset contains 581012 instances, 54 attributes, and no missing values. The task is to predict the type of forest cover based on cartographic variables such as elevation, slope, and soil type

(v) PokerHand Dataset [28]

The PokerHand dataset provides a description of a card using two ranks, namely, suit and rank. The prediction attributes add up to 10. In total, 1,000,000 instances are generated with a total of 11 attributes. The deck consists of 52 cards.

(vii) Amazon Web Services (AWS Prices) Dataset [29]

Amazon allows users on the Amazon Web Services to bid on a spare server capacity known as spot instances. Users can request for a spare server capacity commonly known as spot instances and the prices are much more affordable as opposed to normal on demand rates. The problem with the services offered is that the server can be terminated without notice if many people are willing to bid higher for the same spare capacity. The dataset consists of 27 410 309 instances gathered over a period of several months.

4.3. Experimental Test Configuration. All the empirical experiments conducted are evaluated in terms of time, computational resource utilization, and prediction performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were carried out on machines with Core i7 at 3.4 GHz, 4 GB of RAM, and experiments were presented in terms of CPU time.

The performance evaluation metrics used in this experiment are the accuracy, average accuracy over time plots, processing time measured in seconds based on the CPU time used for training and testing, and the Kappa performance evaluation measure for the dynamic environments. All experiments are run within the Massive Online Analysis (MOA) framework. To evaluate the statistical significance differences between the compared algorithms, nonparametric tests were performed based on the Friedman test [30–32]. For the statistical tests, the Friedman test was applied with $\alpha = 0.05$ based on the assumption that the null hypothesis is that no statistical differences exist between the compared algorithms.

A rejection of the null hypothesis means that the Nemenyi post hoc test is performed to locate the pairs of algorithms that are different from each other. To further validate the hypothesis, we performed the two Kappa evaluations for the experiment. The Kappa evaluation measure is widely used in nonstationary environments as it is capable of handling both multiclass and skewed datasets. A larger Kappa value indicates a more generalized classifier and negative Kappa value is an indication of low prediction accuracy. For the Kappa evaluation experiments, we only consider real-world data streams since artificial data streams work well under restricted environments and corresponding ranks are calculated and higher averages represent lower ranks.

The Neural Network used is the MLP, and it contains 10 hidden nodes each and were trained using backpropagation with one epoch and a learning rate of 0.01 and momentum of 0.01 so as to provide competition for previously published results. The ADES algorithm creates the first three classifiers from the first data chunk using the three base learning algorithms. The ensemble size for artificial datasets is fixed at 42 classifiers. For real-world datasets, the ensemble size is fixed at 100 classifiers.

The cross-validation techniques to measure model performance are not suitable as the streaming data originate from time-varying and dynamic environments. The prequential approach was used as it is a commonly applied estimation procedure in dynamic environments. The merit of the prequential approach is that all the instances are used in training and testing and therefore disregarding the need for a specific holdout set.

4.3.1. Analysis of Results. To analyze the behavior of ADES when it encounters different types of drifts using high- and low-diversity ensembles in dynamic environments that exhibit concept drift and perform empirical analysis of ADES, we first use artificial datasets and reaffirm the empirical analysis of ADES using real-world data streams. The analysis of the accuracy of ADES is implemented in the Java Programming language by extending the Massive Online Analysis (MOA) [5]. To derive the much needed diversity, Support Vector Machines, Neural Networks, and Decision Tree algorithm are used as base classifiers for the ADES algorithm. We also compared the computational costs regarding memory and runtime of ADES over the eight datasets. The predictive performance of ADES is compared with other representative algorithms designed to handle different types of concept drifts with minimal parameter tuning. The results obtained by the representative algorithms are presented and we used the default implementations and parameters associated with each algorithm available in the Massive Online Analysis framework. We perform experiments to compare the predictive performance of ADES with other algorithms that promote diversity such as Dynamic Selection Based Drift Handler (Dynse) [11], Diversity for Dealing with Drifts (DDD) [33], and Ensemble Selection in Dynamic Environments (ESDE) [21]. Dynse [11] is based on the concept of Dynamic Classifier Delection (DCS). The rational is that associating the time dependency to the nature of the concept drift, any neighborhood-based DCS approach can represent a natural answer to the concept drift problem,

regardless of the properties such as speed, severity, and the possible presence of recurrences. Dynse removes poorly performing classifiers making it unable to handle recurring concepts. The Diversity for Dealing with Drifts (DDD) [19] is an online ensemble approach that exploits diversity to handle concept drift. The algorithm maintains ensembles of low and high diversity and uses a drift detection mechanism. The approach cannot handle recurrent or predictable drifts. Ensemble Selection in Dynamic Environments (ESDE) [21] is an ensemble selection approach that selects models from a library and combines them to learn in time-varying domains. ESDE uses Support Vector Machines as the base learner making it homogeneous and is devoid of much needed diversity. The approach suffers from overfitting and has no pruning strategy.

4.3.2. Accuracy Results and Analysis. To ensure a fair comparison between the proposed method and the comparative methods, we adopt appropriate performance measures that can lead to fair and balanced conclusions such as accuracy and accuracy over time plots. Since accuracy can be misleading on datasets with class imbalance or temporal dependencies, Kappa M and Kappa Temporal were also used. The accuracy of the results and the ranks obtained for each of the algorithm conducted in MOA tested over both artificial and real-world datasets are presented in Table 1. The ranks are determined in such a way that higher averages represent lower ranks. Significant tests and post hoc comparisons on ranks are performed to determine significance level and critical differences. The predictive accuracies of ADES, Dynse, ESDE, DDD, and CPU time (s) consumption and memory consumption are shown in Tables 1 and 2, respectively. Real-world datasets used in the experiments have different features. The behavior of ADES is analyzed separately taking into account the features of the data. On the Covertype dataset, variables such as elevation and slope vary smoothly during the entire learning period. The variations represent different types of drifts. ADES demonstrates good behavior as it adapts accurately to different variations and performs comparatively well against DDD, ESDE, and Dynse. ADES showed a stable prediction performance in the presence of recurring, sudden, and gradual drifting concepts for all datasets. DDD performed poorly on recurring or predictable drifts. The computational complexity of ESDE to handle recurring concepts increases as the learning progresses. ADES exhibits the same behavior on the Amazon and Airlines datasets. DDD requires more time to conclude the classification process concerning run time. ADES is able to adapt to sudden and recurring concepts when compared to DDD, Dynse, and ESDE. ADES detects changes much more efficiently and captures different types of drifts as quickly as possible. ADES was able to recover faster than the three compared algorithms under different kinds of concept drifts.

To analyze the computational complexity of ADES, a data stream is partitioned into N data chunks and a window that contains current ensemble selected is created. The base classifiers of ADES are support vector machines, decision trees, and neural networks, and models are generated with a constant time per example. The training of ensemble members has complexity $O(N_i m_i)$, where m_i represents the number of labeled observations in a learning window. The Drift Detection Method (DDM) calculates the unsupervised statistics for each observation, and it requires $O(2N_u m_u)$ time to estimate the expected values of the sample mean and variance. Detecting recurring concepts requires the manipulation of a series of streaming data chunks and requires $O(Nm\ |Z^v|$ time, where m is the ensemble size and $|Z^v|$ represents the number of labeled instances in the validation set used to evaluate the equivalence level between two concepts based on supervised information. The time complexity of ADES in the training phase is $O(2N_i m_i) + Nm|Z^V|$. In the testing phase, the time consumption is dominated by the number of test instances. The prediction protocol of ADES is linearly proportional to the number of labeled instances in a dataset. The memory consumption of ADES is O(mdvc), where d is the number of attributes, v is the maximum number of values per attribute, and c is the number of classes. Data instances in the most recent data chunk are preserved to evaluate the relevance of previous components and to detect recurring concepts. As a result, $O(s_i d)$ memory is required and s_i represents the number of labeled data. Previously learned concepts are preserved to improve the generalization ability of ADES, and it consumes $O((m-1)s_i d)$ memory. The drift detection mechanism therefore requires O(4b) memory. Different variations of ADES were compared to evaluate its sensitivity to parameters such as severity of drift, warning thresholds, and ensemble size including variations of the algorithm that deactivate some of its modules such as drift detection, warning detection, and majority voting.

To analyze the benefits in terms of resource usage, we compare ADES, DDD, Dynse, and ESDE. We recorded an evaluation time of the algorithms in CPU seconds for all the datasets. From the observation, ADES requires the least amount of processing time, followed by Dynse and DDD, with ESDE requiring more processing time as a result of models that always reside in memory. In terms of memory consumption, ADES consumes the least amount of memory for almost all of the datasets. ESDE has no pruning mechanism and stores all models in memory. Dynse and DDD discard poorly performing models and poorly handle predictable drifts and recurrent concepts. Processing the statistics in both ADES and Dynse is not computationally heavy. Training base learners to constitute an ensemble in ADES takes time as the training is performed based on the Poisson distribution proxy parameter (λ). Table 2 presents the runtime consumption in seconds that ADES and the other three algorithms need to process each of the datasets.

Corresponding ranks are provided and higher averages represent lower ranks. ADES has the least average rank, which represents lower ranks. ADES has the least average rank, which represents a higher rank. ESDE has the highest average, a sign of the lowest rank. For datasets with more than 100 000 instances, ADES executes faster and for datasets that exhibit abrupt concept drifts and with instances less than 500 000, ADES executed faster than Dynse.

Table 1: Prediction accuracy and CPU time of the four algorithms on both artificial and real-world datasets.

Dataset	ADES		DDD		Dynse		ESDE	
	Accuracy	CPU-time	Accuracy	CPU-time	Accuracy	CPU-time	Accuracy	CPU-time
Hyperplane	92.12 (1)	33.2 (1)	74.27 (3)	39.6 (3)	81.6 (2)	36.48 (2)	71.33 (4)	48.34 (4)
Stagger	84.53 (2)	48.53 (1)	66.85 (3)	52.48 (3)	86.47 (1)	50.32 (2)	64.89 (4)	64.53 (4)
LED	79.34 (1)	63.27 (2)	75.38 (2)	53.16 (1)	71.29 (3)	49.34 (1)	68.43 (4)	128.45 (4)
Waveform	81.48 (3)	83.36 (2)	83.47 (1)	113.32 (3)	84.34 (2)	78.23 (1)	62.85 (4)	124.54 (4)
Covertype	87.38 (2)	73.39 (1)	81.43 (3)	124.63 (4)	89.39 (1)	93.47 (2)	71.23 (4)	85.67 (3)
Airlines	79.63 (1)	54.34 (2)	76.57 (2)	52.34 (1)	74.73 (3)	83.34 (4)	67.43 (4)	78.45 (3)
Amazon	89.37 (1)	47.53 (1)	76.89 (3)	64.39 (3)	84.63 (2)	57.46 (2)	72.34 (4)	69.48 (4)
Pokerhand	79.84 (1)	39.56 (2)	74.67 (3)	37.28 (1)	76.43 (2)	73.48 (4)	70.37 (4)	67.48 (3)
Average rank	**1.5**	**1.5**	**2.5**	**2.38**	**2.0**	**2.5**	**4.0**	**3.38**

Table 2: Memory consumption of the four algorithms on both artificial and real-world datasets.

Dataset	ADES Memory (MB)	DDD Memory (MB)	Dynse Memory (MB)	ESDE Memory (MB)
Hyperplane	1.24 (1)	1.32 (3)	1.29 (2)	4.82 (4)
Stagger	6.43 (3)	27.36 (4)	4.83 (1)	3.87 (2)
LED	0.82 (2)	0.61 (1)	0.69 (3)	0.92 (4)
Waveform	5.25 (1)	7.49 (2)	112.37 (3)	167.53 (4)
Covertype	3.16 (1)	28.43 (3)	18.73 (2)	49.73 (4)
Airlines	23.14 (2)	19.49 (1)	34.46 (3)	38.37 (4)
Amazon	48.36 (1)	112.23 (3)	98.74 (2)	128.68 (4)
PokerHand	12.36 (1)	58.23 (3)	43.68 (2)	87.25 (4)
Average ranks	1.5	2.5	2.25	3.5

4.3.3. The Pruning Impact. Learning drifting concepts using an infinite size is infeasible under a real-world scenario. ADES utilizes a pruning strategy to keep a flexible pool of 25 to 100 classifiers during the tests. The paradigm of ensemble pruning allows the construction of classifier ensembles with high predictive accuracy and efficiency. To validate the contribution of pruning on the ADES algorithm, we compare an infinite pool of ADES and a pruned ADES on all the datasets.

Table 3 contains the average accuracies achieved and memory consumption of the infinite pool and the pruned ADES. As can be observed, the infinite pool achieved the best accuracies in most cases considered. Despite the good prediction performance achieved by the infinite pool, it often leads to an unpractical amount of consumption of resources with time. The amount of memory consumed by the pruned pool configuration is less than the infinite pool.

In addition to the accuracies generated based on all datasets and the average ranks of the comparative algorithms, we generate accuracy over time plots for each dataset to accurately provide a description of the performance curves of all the comparative algorithms at each time step. The processed observations are represented by the x-axis and the average accuracy is denoted on the y-axis, giving more insight into the analysis of the adaptation capabilities of the comparative algorithms under concept drift scenarios. Figures 4–7 illustrate the accuracies of the compared algorithms under different types of concept drifts, including gradual, incremental, sudden, and recurrent drifts on artificial datasets.

Figure 4 shows the accuracy over time plot of the algorithms on the Hyperplane dataset, which includes an incremental concept drift and a gradual concept drift. ADES, DDD, and Dynse are the best performing algorithms. ESDE reacts poorly to abrupt changes and the average accuracies indicate mild drops in accuracy.

Figure 5 shows the predictive performances of the four algorithms on the Stagger dataset. For a Stagger data stream that exhibits abrupt real concept drift, the best performing algorithms are Dynse and ADES.

Figure 6 shows the predictive performances of the four compared algorithms on the Waveform dataset. ESDE performed poorly and Dynse and ADES continued to perform comparatively well on the dataset.

Figure 7 shows the accuracy over time plot of the algorithms on the LED dataset. For the LED dataset with abrupt and gradual drifts, the best performing algorithms are ADES and Dynse. Dynse and ADES recover quickly from sudden drifts.

The artificial datasets are used to explain the behavior and identify the types of drift for which it works better. Artificial datasets are typically designed for controlled environments. Synthetic datasets offer several advantages. They are easier to reproduce and bear low cost storage and transmission and most importantly, synthetic datasets provide an advantage of knowing the ground truth. For instance, we can know where exactly concept drift happens, what type of drift is, and the best classification accuracies achievable on each concept [34]. Several challenges emerge when dealing with classification problems. The major

Table 3: Average accuracy achieved and memory consumption for infinite pool and pruned ADES.

Benchmark	Infinite pool		Pruned ADES	
	Accuracy (%)	Memory (MB)	Accuracy (%)	Memory (MB)
Hyperplane	95.6 (5.7)	16.23	92.12 (6.4)	4.3
LED	89.3 (6.8)	12.08	79.34 (10.32)	8.9
Waveform	84.8 (8.3)	13.24	81.48 (8.5)	10.24
Stagger	91.4 (14.6)	8.9	84.53 (5.8)	5.9
Airlines	76.3 (4.3)	17.14	79.63 (12.34)	14.32
Covertype	74.6 (8.5)	14.17	87.38 (14.32)	12.43
PokerHand	73.4 (3.7)	10.24	79.84 (16.21)	16.34
Amazon	78.6 (4.6)	20.38	89.37 (18.34)	18.26

The numbers in parenthesis indicate the standard deviation between testing batches.

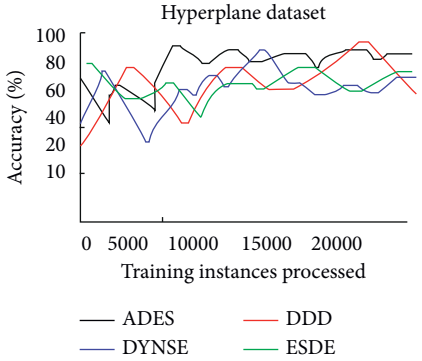

Figure 4: Accuracy over time plot of the four algorithms on the Hyperplane dataset.

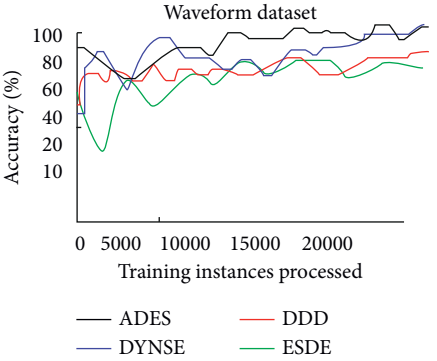

Figure 6: Accuracy over time plot of the four algorithms on the Waveform dataset.

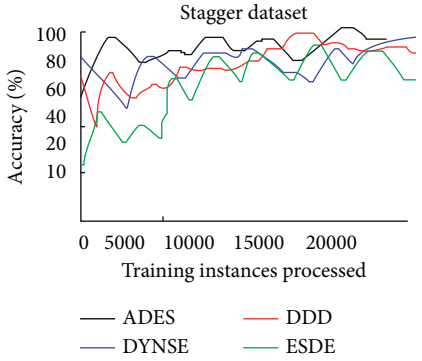

Figure 5: Accuracy over time plot of the four algorithms on the Stagger dataset.

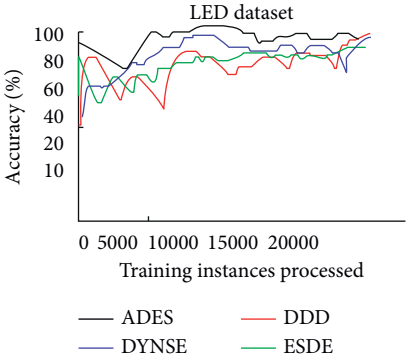

Figure 7: Accuracy over time plot of the four algorithms on the LED dataset.

problem is the identification and location of the concept drift. Accordingly, the drift handling capabilities of ADES is also evaluated on real-world data streams to reaffirm the analysis of ADES. To reaffirm the analysis of ADES, the Forest Covertype, Airlines, Amazon, and Poker Hand datasets are used in the experiments. These datasets have been used widely in recent literature to test the validity of new machine learning predictive models designed to handle concept drift in dynamic environments. The Forest Cover Type is used in Ghomeshi et al. [35] in the implementation of the evolutionary adaptation to drift and achieved 91.73% accuracy on the Forest Cover Type dataset. The MM-PRec system [36] used the Airlines dataset to evaluate the effectiveness of the algorithm to handle recurring concepts and achieved an accuracy of 64.85%. The Enhanced Concept Profiling Framework (ECPF) [37] used the Poker Hand dataset to test its validity to handle concept drift and achieved an accuracy of 74.6%. The relative density-ratio estimation [38] used the Amazon dataset to evaluate the performance of the algorithm to handle concept drift and achieved an accuracy of 82.73%. These real-world datasets are used to further confirm and validate ADES. When testing a new machine learning predictive model designed to handle all types of concept drift for classification problems, apart from verifying its behavior

through the use of artificial datasets, it is important to further verify if a good behavior is also attained using real-world datasets. As this serves as an extended evaluation in addition to the experiments using artificial datasets, the fact that it is not possible to know exactly what type of drift is being tested is not a problem [19].

Figures 8–11 show the accuracy over time plots of the four compared algorithms on real-world datasets. ADES and DDD recover quickly from sudden changes better than ESDE and Dynse. Figure 11 shows the performance accuracy of the four algorithms on the Covertype dataset. The accuracy over time of DYNSE and ADES are almost identical. As more instances are observed, DDD's accuracy over time improves as well.

Figure 9 shows the performance of the four algorithms on the Airlines dataset. Dynse and ADES adapt well to sudden and gradual changes. The performance accuracy of DYNSE and ADES is almost identical. ESDE performs the worst.

Figure 10 shows the performance of the four algorithms on the Amazon dataset where the task is to predict ratings of each review. ADES performs the best, followed by DYNSE. ESDE performs the worst.

Figure 11 shows the predictive performance of the four algorithms on the Poker Hand dataset. Dynse and ADES recover well from gradual and sudden drifts. DYNSE is second to ADES in terms of performance accuracy. DDD and ESDE perform the worst.

Previously learned high-diversity ensembles are used to learn concepts with low diversity in order to improve the algorithm's convergence to new concepts. Keeping previously learned high-diversity ensembles enables a better exploitation of diversity to handle recurrent or predictable drifts as previously learned information is also used to learn new concepts and helps the approach to be robust to false alarms. Combining weighting strategies and drift detection makes the algorithm to require some time to start reflecting new concepts.

4.3.4. Statistical Analysis of the Results. Despite the fact that some state-of-the-art drifting handling approaches performed well and close to ADES in some benchmark datasets, these approaches maybe more sensitive to the parameter tuning or to the concept drift properties featured in both the synthetic and real-world datasets. To validate the claim that ADES performs well in nonstationary environments associated with concept drift, we perform statistical tests on the four algorithms. To compare the results of the experiments, we used the F_r statistic on the nonparametric Friedman test [30, 31]. All methods are statistically equal based on the null hypothesis and if there is rejection, it means that a statistical difference exists in some of the methods. A post hoc is performed to ascertain whether there exist any differences. We use the Nemenyi test [39] to compare each method with other methods and the test uses critical difference as reference for the purposes of comparison. To carry out the Nemenyi test, we used KEEL 3.0 software [40]. Figure 12 graphically presents the results of the test referring to

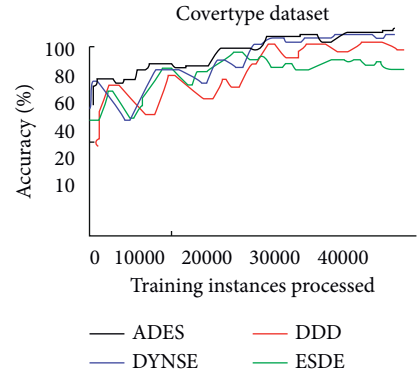

Figure 8: Accuracy over time plot of the four representative algorithms on the Forest Covertype dataset.

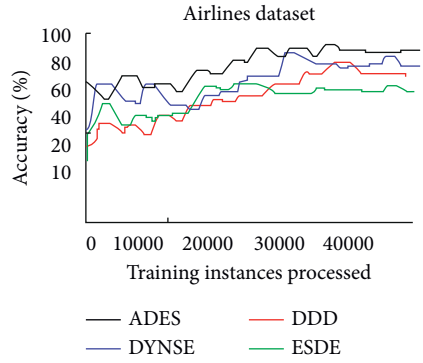

Figure 9: Accuracy over time plot of the four algorithms on the Airline dataset.

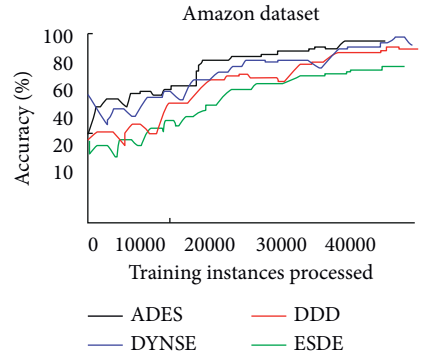

Figure 10: Accuracy over time plot of the four algorithms on the Amazon dataset.

Table 1. The critical difference is represented by a bar and the methods considered in the experiments connected to the base method by those bars are not statistically different. According to Figure 12, ADES is significantly better than other methods. The test also shows with 95% confidence that some state-of-the-art methods are not significantly different from the ADES algorithm as they are closely connected when it comes to generalization performance. Despite the fact that some of the ensemble algorithms surpassed the ADES algorithm on some benchmark datasets, these

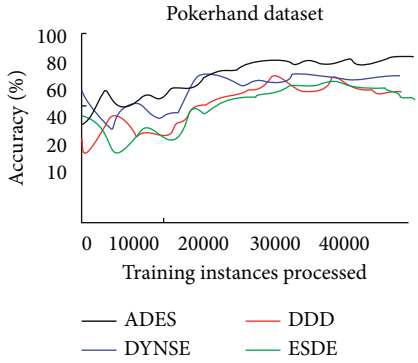

Figure 11: Accuracy over time plot of the four algorithms on the PokerHand dataset.

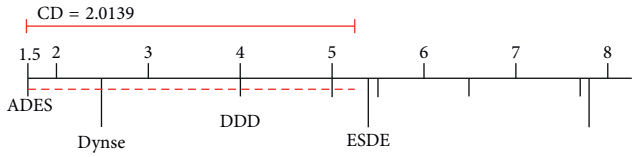

Figure 12: Average rank diagram of the compared algorithms.

approaches maybe sensitive to parameter tuning or to the concept drift properties featured in the datasets. To make it clear, we consider the average ranks and use a Bonferroni–Dunn test for $\alpha = 0.05$ to determine if the performance of ADES is statistically better than that of the other state-of-the-art algorithms, and it is demonstrated that ADES performs significantly better than Dynse, DDD, and ESDE. The approaches are connected with a line. Figure 12 shows the Bonferroni–Dunn test with a 95% confidence showing the ensemble approaches that are not significantly different from the ADES approach. Figure 12 shows the average rank diagram of the compared algorithms.

The ADES, DDD, Dynse, and ESDE approaches deemed as equivalent by the Bonferroni–Dunn test are further analyzed using pairwise comparisons, taking into account the hypothesis of equality between each pair of algorithms. A methodology proposed by Demsar [30] for the comparison of several algorithms over multiple datasets is followed. In the study, the nonparametric Friedman test is firstly used to determine if there is a statistically significant difference between the rankings of the compared techniques. The Nemenyi post hoc test is performed and algorithms that do not differ significantly are connected with a line. The Critical Difference (CD) is shown above the figure. As can be observed, ADES was considered to be significantly better than Dynse, DDD, and ESDE.

5. The Drift Detection Impact

The second experiment investigates the impact of drift detection mechanism on the predictive performance of the ADES algorithm. In the first empirical experiment conducted, an approach based on accuracy and diversity, that is, robust to false alarms and several types of drift without necessarily identifying the type of drift and having faster recovery from drifts were proposed. The next experiment analyzes the behavior of ADES when it is converted to an active heterogeneous ensemble by incorporating a drift detection mechanism in the algorithm, which is capable of identifying and locating concept drift. To properly investigate the behavior of the algorithm when it encounters different types of concept drift, we introduce the Drift Detection Method (DDM) in the algorithm. Drift Detection Method is based on a statistical process control, and it tracks the minimum error of an online learning model over time and its corresponding standard deviation by updating the variables whenever a new training example is received. Changes in the variables triggers a warning, and new training examples are used to update the base models and to store them in memory for future use.

To validate the impact of the drift detection mechanism on the predictive performance of ADES, we only consider real-world data streams since artificial data streams are typically designed for controlled environments and corresponding ranks are determined such that higher averages are representing lower ranks. The analysis of the ADES algorithm and the comparative algorithms are implemented in the Java programming language by extending and customizing the MOA software. The average predictive accuracies of ADES, Dynse, DDD, and ESDE are shown Table 3. Table 3 shows the prediction accuracy and the time taken in seconds of each algorithm together with corresponding ranks.

5.1. Kappa Evaluation Measures. Tables 4 and 5 provide the Kappa measures for the experiments. Table 5 shows the Kappa temporal values, and Table 6 provides Kappa M values for real-world datasets.

Kappa values for both temporal and M were all positive. The number of attributes in all of the data did not affect the classifier as it was more generalized, an indication of high predictive accuracy.

5.2. Accuracy Over Time Plots. The predictive performance of ADES is further evaluated using the Accuracy Over Time Plots against the representative algorithms. As shown in the accuracy over time plots, ADES performs comparatively well against other representative algorithms designed to handle concept drift and recovers quickly from sudden changes to reflect new concepts and optimize convergence to new concepts. Figure 13 shows the accuracy over time plot of the four algorithms on the Covertype dataset using drift detection. The performance of DYNSE and ADES is almost identical. DDD performs the worst and ESDE improves as more instances are observed.

Figure 14 shows the accuracy over time plot of the four algorithms on the Airline dataset using drift detection. As can be observed, DYNSE performs the best, followed by ADES. DDD is slightly less accurate and ESDE performs the worst.

Figure 15 shows the accuracy over time plot of the four algorithms on the Amazon dataset. The accuracy of DDD

Table 4: Prediction accuracy, CPU runtime, and corresponding ranks.

	ADES		DDD		Dynse		ESDE	
Dataset	Accuracy	CPU-time	Accuracy	CPU-time	Accuracy	CPU-time	Accuracy	CPU-time
Covertype	83.43 (2)	87.46 (1)	80.23 (3)	112.63 (4)	87.53 (1)	96.38 (2)	69.37 (4)	98.64 (3)
Airlines	74.53 (1)	74.34 (2)	71.37 (2)	63.34 (1)	69.83 (3)	105.34 (4)	66.33 (4)	96.45 (3)
Amazon	83.45 (1)	76.43 (1)	78.89 (3)	67.39 (3)	82.65 (2)	54.46 (2)	76.34 (4)	93.48 (4)
PokerHand	77.89 (1)	67.59 (2)	71.67 (3)	45.28 (1)	74.43 (2)	93.68 (4)	68.34 (4)	84.48 (3)
Average ranks	1.25	1.20	2.75	2.25	2.0	3.0	4.0	3.25

Table 5: Kappa temporal values for the real-world datasets.

	ADES	DDD	Dynse	ESDE
Covertype	36.89 (3)	73.44 (1)	48.93 (2)	31.23 (4)
Airlines	81.24 (1)	76.46 (3)	79.38 (2)	73.67 (4)
Amazon	73.47 (2)	68.34 (4)	76.47 (1)	70.34 (3)
PokerHand	76.12 (1)	67.84 (3)	73.46 (2)	66.48 (4)
Average ranks	1.75	2.75	1.75	3.75

Table 6: Kappa M values for real-world datasets.

	ADES	DDD	Dynse	ESDE
Covertype	44.65 (2)	71.26 (3)	63.27 (1)	74.36 (4)
Airlines	58.35 (1)	44.43 (2)	59.26 (3)	68.48 (4)
Amazon	85.68 (1)	54.37 (4)	75.32 (3)	81.36 (2)
PokerHand	63.82 (1)	42.54 (3)	58.37 (3)	38.63 (4)
Average ranks	1.25	3.0	2.50	3.5

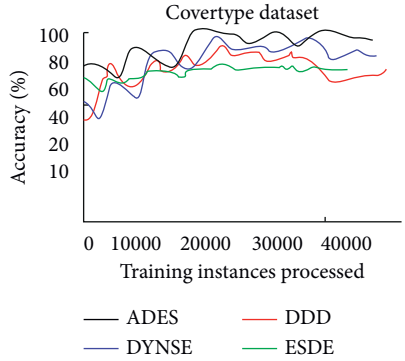

Figure 13: Accuracy over time plot of the four algorithms on the Covertype dataset using drift detection.

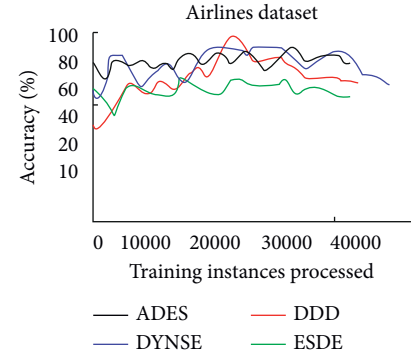

Figure 14: Accuracy over time plot of the four algorithms on the Airlines dataset using drift detection.

improves as more instances are observed with time. ADES performs slightly better than DYNSE although their accuracies are almost identical. As more instances are observed, ESDE's accuracy deteriorates.

Figure 16 depicts the accuracy over time plots of the four algorithms on the PokerHand dataset. As can be observed, the accuracy curves generated by all the algorithms demonstrate varying degrees of volatility, a sign that the dataset exhibits concept drift. In all cases, ADES recovers faster in all cases, followed by DYNSE. DDD improves adaptation as more instances are observed with time. ESDE recovers poorly from concept drift.

As can be observed, ADES performs comparatively well against the selected representative algorithms. ADES recovers well from sudden changes and converges to new concepts relatively fast. DDD handles recurrent and predictable drifts poorly. Dynse handles sudden and gradual drifts well but performs poorly in handling recurring concepts as it prunes poorly performing models. The inclusion of the drift detection algorithm in the ADES algorithm gives the algorithm more impetus in handling all kinds of concept drifts as it is now active to get more information about the type of drift currently occurring.

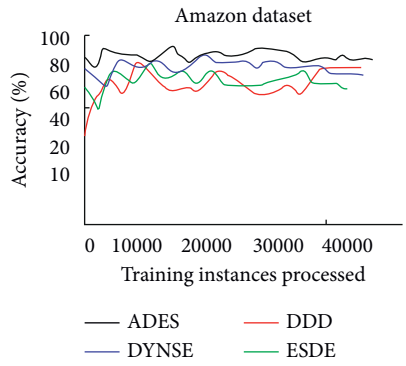

Figure 15: Accuracy over time plot of the four algorithms on the Amazon dataset using drift detection.

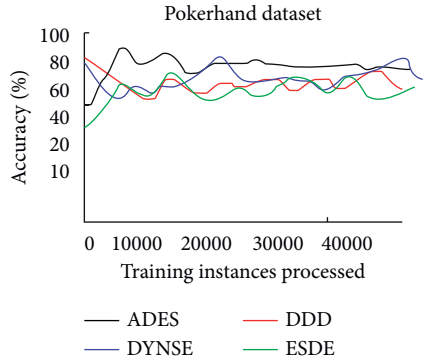

Figure 16: Accuracy over time plot of the four algorithms on the PokerHand dataset.

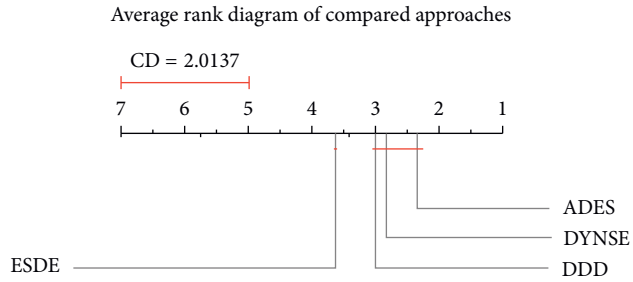

Figure 17: Average rank diagram of compared approaches using drift detection.

5.3. Average Rank Diagrams of Compared Approaches. We compare the accuracies achieved by the algorithms over all the datasets. We firstly apply the nonparametric Friedman test [30, 31] to determine if there is a statistically significant difference between the rankings of the compared techniques. The next step performs the Nemenyi post hoc test [41] with average rank diagram. To perform the Nemenyi post hoc test, the KEEL 3.0 software is used [40]. The algorithms that do not differ significantly are connected with a line. The Critical Difference (CD) is shown above the graph. As can be observed from the CD plot, ADES is ranked first. However, its performances are not statistically distinguishable from the performances of Dynse and DDD. Dynse handles gradual concepts well and DDD handles abrupt concepts well but poorly handles recurring concepts. Despite a slow convergence to new concepts, ADES comparatively handles sudden and abrupt drifts relatively better than both DDD and Dynse. Figure 17 shows the average rank diagram of compare approaches using drift detection.

6. Conclusion

This paper presented an Adaptive Diversified Ensemble Selection based on accuracy and diversity. The algorithm creates ensembles of high- and low-diversity ensembles for different types of concept drifts as they require different amounts of diversity. We evaluated the performance of the algorithm on four artificial datasets and affirmed the performances on four real-world datasets that exhibited sudden, gradual, incremental, and recurring concepts, and the datasets are the mostly used ones in the concept drift research area. We conducted empirical experiments and implemented them in Java programming language by customizing and extending the MOA framework. Based on the experimental results, ADES presented accuracy comparable to the other three representative algorithms designed to handle drifting concepts considering both artificial and real-world data streams. To further validate the performance of ADES, we used a statistic based nonparametric Friedman test to compare the four algorithms. The predictive performance of ADES was not distinguishable from the DDD algorithm and the Dynse algorithm. The second experiment investigated the impact of the drift detection mechanism on the ADES algorithm using real-world datasets. The drift detection revealed the type of drift occurring leading to the amount of diversity required. Results indicate that ADES still performed comparatively well against the other three algorithms on real-world datasets despite the fact that recovery from sudden and gradual drifts was slowed down. The main potential drawback of ADES is that as the volume of data increases, it becomes computationally costly than the compared methods. Future work involves the investigation of incorporating a drift detection method that is robust to false alarms and has the possibility of creating flexible high and low ensembles that can be increased or decreased automatically depending on the severity or magnitude of change. We will also investigate in future the prediction performance of ADES on more diverse and voluminous problems associated with a larger number of attributes.

Data Availability

The research used four artificial datasets: (1) Hyperplane, (2) Stagger, (3) LED Generator, and (4) Waveform Generator. The real-world datasets used are (1) Covertype dataset, (2) Airlines, (3) Amazon, and (4) Poker Hand dataset. The artificial and real-world data used to support the findings of this study have been deposited in the following repositories and sources: (1) [42] Hyperplane—A. Bifet and R. Kirky, "Introduction to massive online analysis (MOA)," 2010, http://moa.cms.waikato.ac.nz/datasets/PokerHand; (2) Stagger J. C. Schlimmer, R. H. Granger Jr., "Incremental learning

from noisy data," vol. 1, 1986, pp. 317–354, https://dx.doi.org/10.1007/BF00116895; (3) LED Generator—Cunningham P., Nowlan N., Delany S. J. and Haahr M., 2003, "A case-based approach to spam filtering that can track concept drift," In the Proceedings of ICCBR-2003 Workshop on Long-Lived CBR Systems; (4) Waveform Generator—Cunningham P., Nowlan N., Delany S. J. and Haahr M., 2003, "A case-based approach to spam filtering that can track concept drift," In the Proceedings of ICCBR-2003 Workshop on Long-Lived CBR Systems. (1) Covertype dataset—https://archive.ics.uci.edu/ml/datasets/Covertype; (2) Airlines dataset—https://moa.cms.waikato.ac.nz/datasets/airlines; (3) Amazon dataset—https://moa.cms.waikato.ac.nz/datasets/amazon; (4) PokerHand dataset—https://archive.ics.uci.edu/ml/datasets/Poker+Hand.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: a survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[2] T. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Systems with Applications*, vol. 82, pp. 77–99, 2017.

[3] S. Ren, B. Liao, W. Zhu, and K. Li, "Knowledge-maximized ensemble algorithm for different types of concept drift," *Information Sciences*, vol. 430–431, pp. 261–281, 2018.

[4] MOA, "Massive online analysis datasets," 2010, https://moa.cms.waikato.ac.nz/datasets.

[5] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, pp. 1601–1604, 2010.

[6] I. Zliobaite, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in *Big Data Analysis: New Applications for a New Society*, vol. 16, pp. 91–114, Springer International Publications, Cham, Switzerland, 2016.

[7] V. Mangat, V. Gupta, and R. Vig, "Methods to investigate concept drift in big data streams," *Knowledge Computing and its Applications*, Springer, Singapore, 2018.

[8] I. Baidari and N. Honnikoll, "Accuracy weighted diversity based online boosting," *Expert Systems with Applications*, vol. 160, 2020.

[9] A. Cano and B. Krawczyk, "Kappa Updated Ensemble for drifting data stream mining," *Machine Learning*, vol. 109, pp. 175–218, 2019.

[10] J. R. B. Junior and C. M. Nicoletti, "An iterative boosting-based ensemble for streaming data classification," *Information Fusion*, vol. 45, pp. 66–78, 2019.

[11] P. R. Almeida, L. S. Oliveira, A. S. Britto, and R. Sabourin, "Adapting dynamic classifier selection for concept drift," *Expert Systems with Applications*, vol. 104, pp. 67–85, 2018.

[12] k Meshgi, S. Ohu, and S. Ishii, "Efficient diverse ensemble for discriminative Co-tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4814–4823, Salt Lake City, UT, USA, June 2018.

[13] V. Bhatnagar, M. Bhardwaj, S. Sharma, and S. Haroon, "Accuracy-diversity based pruning of classifier ensembles," *Progress in Artificial Intelligence*, vol. 2, pp. 97–111, 2014.

[14] S. Ancy and D. Paulraj, "Online learning model for handling different concept drifts using diverse ensemble classifier on evolving data streams," *Cybernetics and Systems*, vol. 50, no. 7, pp. 579–608, 2019.

[15] M. M. Idrees, L. L. Minku, F. Stahi, and A. Badii, "A heterogeneous online learning ensemble for nonstationary environments," *Knowledge Based Systems*, vol. 188, 2020.

[16] A. Onan, "Hybrid supervised clustering based ensemble scheme for text classification," *Kybernetes*, vol. 46, no. 2, pp. 330–348, 2017.

[17] A. Onan, S. Korukoglu, and H. Bulut, "A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification," *Information Processing and Management*, vol. 53, no. 4, pp. 814–833, 2017.

[18] A. Onan, S. Korukoglu, and H. Bulut, "A multi-objective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification," *Expert Systems with Applications*, vol. 62, pp. 1–16, 2016.

[19] L. L. Minku and X. Yao, "DDD: a new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, 2012.

[20] N. C. Oza, "Online ensemble learning," Ph.D. thesis, The University of California, Berkeley, CA, USA, 2001.

[21] M. P. Procopio, J. Mulligan, and G. Grudic, "Learning in dynamic environments with ensemble selection for autonomous outdoor robot navigation," *IEEE/RSJ International Conference in Intelligent Robots and Systems, Acropolis Convention Center, Nice France, September*, vol. 22-26, 2008.

[22] J. Xiao, C. He, X. Jiang, and D. Liu, "A dynamic classifier ensemble selection approach for noise data," *Information Sciences*, vol. 180, pp. 3402–3421, 2010.

[23] A. Bifet, Campo-Avila, and R. Fidalgo, "Early drift detection method," in *Proceedings of the Fourth ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, pp. 77–86, Riva del Garda, Italy, September 2006.

[24] G. Yule, "On the association of attributes in statistics," *Philosophical Transaction Royal Society of London, Series A*, vol. 194, pp. 257–319, 1900.

[25] A. Asuncion and D. Newman, "UCI machine learning repository," 2007, http://www.ics.uci.edu/mlearn/MLRepository.

[26] J. Schlimmer and R. Granger, "An incremental learning from noisy data," *Machine Learning*, vol. 1, no. 3, pp. 317–354, 1986.

[27] M. Lichma, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, CA, USA, 2009, http://archive.ics.uci.edu/ml.

[28] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda, "Improving adaptive bagging methods for evolving data streams," *Asian Conference on Machine Learning*, Springer, pp. 23–27, November 2009.

[29] B. Visser, G. Henry, and K. Repository, https://www.kaggle.com, 2018.

[30] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[31] R. S. M. Barros, D. R. L. Cabral, P. M. Goncalves Jr., G. Silas, and T. C Santos, "RDDM: reactive drift detection method," *Expert Systems with Applications*, vol. 90, pp. 344–355, 2017.

[32] R. S. M. Barros, S. T. Garrido, S. de Carvalho, and P. M. Goncalves Junior, "A boosting-like online learning ensemble," in *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1871–1878, Vancouver, Canada, July 2016.

[33] C. W. ChiuL. L. Minku et al., "Diversity-based pool of models for dealing with recurring concepts," in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, July 2018.

[34] Y. Sun, Z. Wang, and H. Liu, "Online ensemble using adaptive windowing for data streams with concept drift," *International Journal of Distributed Sensor Networks*, pp. 1–9, 2016.

[35] H. Ghomeshi, M. M. Gaber, and Y. Kovalchuk, "EACD: evolutionary adaptation to concept drifts in data streams," *Data Mining and Knowledge Discovery*, vol. 33, pp. 663–694, 2019.

[36] A. M. Angel, G. J. Bartolo, and E. Menasalvas, "Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function," *Expert Systems with Applications*, vol. 46, pp. 87–105, 2019.

[37] R. Anderson, Y. S. Koh, and G. Dobbie, "Recurring concept meta-learning for evolving data streams," *Expert Systems with Applications*, vol. 138, 2019.

[38] B. Dong, Y. Gao, S. Chandra, and L. Khan, "Multistream classification with relative density ratio estimation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 3, 2019.

[39] R. S. M. de Barros and S. G. T. de Carvalho Santos, "An overview and comprehensive comparison of ensembles for concept drift," *Information Fusion*, vol. 52, pp. 213–244, 2019.

[40] I. Triguero, S. Gonzalez, J. M Moyano et al., "Keel 3.0: an Open Source Software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 1238–1249, 2017.

[41] R. S. M. Barros and S. G. T. C. Santos, "A large-scale comparison of concept drift detectors," *Information Sciences*, vol. 451-452, pp. 348–370, 2018.

[42] A. Bifet and R. Kirkby, "Tutorial 1, introduction to massive online analysis (MOA)," 2010.