A Heterogeneous Online Learning Ensemble for Non-Stationary Environments

Mobin M. Idrees [a], Leandro L. Minku [b], Frederic Stahl [a], Atta Badii [a]

[a] Department of Computer Science, University of Reading, Whiteknights, Reading, RG6 6AY, United Kingdom

[b] Department of Computer Science, University of Birmingham, Edgbaston, B15 2TT, Birmingham

Abstract

Learning in non-stationary environments is a challenging task which requires the updating of predictive models to deal with changes in the underlying probability distribution of the problem, i.e., dealing with concept drift. Most work in this area is concerned with updating the learning system so that it can quickly recover from concept drift, while little work has been dedicated to investigating what type of predictive model is most suitable at any given time. This paper aims to investigate the benefits of online model selection for predictive modelling in non-stationary environments. A novel heterogeneous ensemble approach is proposed to intelligently switch between different types of base models in an ensemble to increase the predictive performance of online learning in non-stationary environments. This approach is Heterogeneous Dynamic Weighted Majority (HDWM). It makes use of "seed" learners of different types to maintain ensemble diversity, overcoming problems of existing dynamic ensembles that may undergo loss of diversity due to the exclusion of base learners. The algorithm has been evaluated on artificial and real-world data streams against existing well-known approaches such as a heterogeneous Weighted Majority Algorithm (WMA) and a homogeneous Dynamic Weighted Majority (DWM). The results show that HDWM performed significantly better than WMA in non-stationary environments. Also, when recurring concept drifts were present, the predictive performance of HDWM showed an improvement over DWM.

Keywords: Heterogeneous Ensemble Classifier, Majority Algorithm, Concept Drift, Data Stream Mining

1 Introduction

Many real-world applications of machine learning operate in data streaming environments where additional data becomes available over time. Examples are Cyber Security [1][2][3][4], Sentiment Analysis [5][6], Human Activity Recognition [7][8] and Fraud Detection Systems [9]. The underlying probability distribution of such domains typically exhibits changes over time, i.e., these domains usually involve concept drift [10][11]. For example, in credit card approval [52][53] the likelihood of defaulting on payment may change due to an economic crisis.

The large number of data streaming applications makes the area of learning in non-stationary environments (i.e., environments where concept drift would occur) increasingly important. Several approaches to handling concept drift can be found in the literature [10][11][12]. Most studies in this area are concerned with how to quickly detect and/or adapt to concept drift. In particular, "Active" approaches use methods to explicitly detect concept drifts. If a drift is detected, new predictive models are typically created to learn the new concept thus helping the system to recover from the concept drift [13][14][30]. Passive approaches do not use concept drift detection methods. Instead, they usually maintain an ensemble of predictive models called "base models" and use weights in order to emphasise the models believed to best represent the current concept [19][20][28][29]. These approaches also typically create new base

* Corresponding author.
E-mail addresses: mobin_ksa@yahoo.com (M. Idrees), L.L.Minku@cs.bham.ac.uk (L. Minku), F.T.Stahl@reading.ac.uk (S. Frederic), atta.badii@reading.ac.uk (A. Badii).

models and enable the deletion of old base models to help in dealing with concept drifts.

Even though it is well known that various types of predictive models (e.g., Naïve Bayes, Hoeffding Trees, Multilayer Perceptron, etc.) can provide a very different predictive performance depending on the problem being tackled [15][16], little work has been dedicated to the investigation of what type of predictive model is most adequate over time in non-stationary environments. This could be a particularly important issue with regard to online learning [11], i.e., when each example is learnt separately upon arrival and then discarded.

For instance, when delivering online learning, it is difficult to know which type of machine learning algorithm would be best to use as a base model for an ensemble learning algorithm beforehand, due to the initially small amount of data available for evaluating base models. However, as more data is received, it is desirable that online ensemble learning algorithms automatically identify which types of base learners work best for the application domain. In addition, if the best type of base learner changes due to concept drift, online ensemble learning algorithms should also be able to automatically identify which types of models are best suited to the situation encountered after concept drift.

A good combination of different types of models can also sometimes lead to a better predictive performance than the use of a single type of model [17][18]. Therefore, it would be desirable for online learning algorithms applied to non-stationary environments not only to detect which is the best type of model maintaining the highest classification accuracies, but also to use a combination of different types of model if that is found to be beneficial.

Therefore, this paper proposes an online heterogeneous ensemble learning algorithm for non-stationary environments known as the Heterogeneous Dynamic Weighted Majority (HDWM). It aims to turn one of the most popular passive ensemble approaches, namely Dynamic Weighted Majority (DWM) [20], into a heterogeneous ensemble. HDWM automatically chooses or emphasises the best types of base models to be used over time in non-stationary environments. This enables it to keep different types of base models and use them to improve predictive performance to manage concept drift.

The HDWM algorithm was evaluated on artificially induced drift streams and real-world data streams. Its predictive performance was compared against existing well-known approaches such as the Weighted Majority Algorithm (WMA) and Dynamic Weighted Majority (DWM). The HDWM results show that it performs significantly better than WMA when there is concept drift in the data streams. Its heterogeneity and classifier switching mechanism make it independent of manually choosing the base classifier according to conditions. The results showed that despite the heterogeneity of WMA, no significant differences were found between DWM (Hoeffding Tree) and DWM (Naïve Bayes) with WMA. It is extremely difficult to choose the right type of base learner in the ensemble. HDWM overcomes this by intelligently switching its base learners and showed stability in a non-stationary environment.

This paper is further organised as follows. Section 2 presents related work. Section 3 describes the proposed approach. Section 4 outlines the experimental setup and provides an empirical evaluation of the developed algorithm. Section 5 analyses the results and Section 6 sets out concluding remarks.

2 Related Work

There is a rich literature on learning in non-stationary environments [10][11][12]. In addition to categorising existing algorithms into active and passive, it is also possible to categorise existing work into online and chunk-based approaches [11]. Online approaches process each new training instance separately and then discard it. Chunk-based approaches wait for a whole new batch of data to arrive, and then use this new batch for training before discarding it. We concentrate on online rather than chunk-based learning algorithms, because they are the main beneficiaries of an investigation of new heterogeneous ensemble approaches, as explained in the introduction.

A new Decision Tree (DT) ensemble was proposed [63] to increase the diversity of the ensemble by using different training sample numbers for different base DT classifiers. Another approach for multi-class and imbalanced data was presented [64] in which the binary classifiers are first created and then integrated in the ensemble by using majority voting to make predictions.

In terms of diversity the ensembles are broadly classified into homogeneous and heterogeneous, taking into consideration the drift handling approaches, the ensembles are categorised into active and passive approaches. Fig. 1 illustrated the categorisation of algorithms for the remainder of this Section. Please note algorithms mentioned in this figure are referred to in the following sub-sections. As shown in the Figure, HDWM is heterogeneous and sharing the features of both active and passive learning.

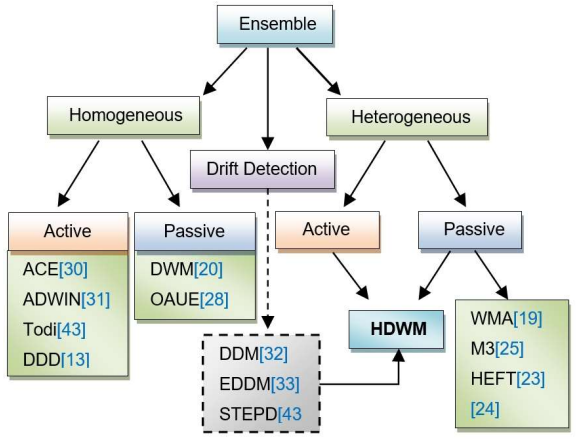

Fig. 1: Active and Passive approaches of Ensembles

2.1 Heterogeneous Ensembles

Most existing heterogeneous ensemble techniques rely on meta-learning [21][22]; this helps in deciding which learning techniques work well on what data. Nguyen et al. [23] proposed a general framework to integrate feature selection and heterogeneous ensemble learning for data stream classification. Cheng et al. [24] built a heterogeneous ensemble using three different tree-based ensembles (Random Forest, Rotation Forest, and Extremely Randomised Trees). It was shown that running heterogeneous/different, or homogeneous/similar data stream classification techniques over vertically partitioned data (data partitioned according to the feature space) resulted in comparable performance to batch and centralised learning techniques [51].

The Weighted Majority Algorithm (WMA) [19] uses fixed numbers of base learners $C=(C_1, C_2 \ldots C_L)$ with an initial weight 'w_i' equal to '1'. The weight is updated on each wrong prediction using $(w_i \leftarrow \beta w_i)$, where $(0 \leq \beta < 1)$ and the final prediction is made based on the weighted majority vote among the base learners C_i. The diversity of base learners has a significant effect in improving the performance on different streams. WMA base learners are heterogeneous, potentially helping to produce more diverse ensembles. However, it lacks the option to dynamically add new base learners. The algorithm has no explicit method to detect and handle concept drift thus being less effective in non-stationary environments.

The Modal Mixture Model (M3) [25] is a heterogeneous chunk-based ensemble for non-stationary environments. New classifier members are added to the ensemble at each data chunk and the weights are computed based on past performances. A weighting mechanism is used to deal with non-stationary environments. The algorithm continuously updates the models regardless of whether real drift occurs or not.

The Heterogeneous Ensemble with Feature drifT for Data Streams (HEFT-Stream) [23] is an online classifier that incorporates feature selection by applying the Fast Correlation Based Filter (FCBF) [26] algorithm that dynamically updates the relevant feature subsets for data streams. This is beneficial because non-stationary environments may present feature drift [23][41]. In high-dimensional datasets, not all features are significant for training a classifier and the relevance of a feature may grow or shrink over time. Given a set of p different classifier types, $M=\{M_1, M_2, \ldots M_p\}$, the ensemble is initialised with k classifiers of each model in M. It determines the most discriminative feature subset on a chunk using a sliding window. If the subset is different from the previous one, there is a feature drift. The approach then looks for the most accurate classifier having the smallest aggregated error and builds a new classifier. Finally, it removes the classifier with the least accuracy from the ensemble and adds the best classifier to the ensemble. However, after the initialisation stage, the algorithm never utilises the M models to create new classifiers. Therefore, there are chances that the ensemble may become homogeneous again in the future.

BLAST (short for best last) [42] introduced an Online Performance Estimation framework to weight the votes of (heterogeneous) ensemble members. Based on zero/one loss function, i.e. returns '1' on correct predictions and '0' otherwise, the weights are increased accordingly. Based on the performances on w (window size) it nominates one of its members to be

an active classifier and sets its weight to '1' and the weights of the remaining classifiers to '0'. The weights are updated on a predefined interval. The HEFT [23] and Online Accuracy Updated Ensemble (OAUE) [28] apply a similar approach in which worst performing models are replaced with new learners, unlike the BLAST that temporarily reduces the weights of a poorly performing member. However, it utilises a static ensemble size similar to WMA [19].

2.2 Active and Passive Homogeneous Approaches to Deal with Concept Drift

This section presents related work on passive and active online learning approaches for non-stationary environments which are not based on heterogeneous ensembles. Chunk-based approaches, could potentially use off-line procedures such as cross-validation to choose the best type of base learner for each new chunk of data, even though this has not been investigated so far. Therefore, this section will not cover chunk-based approaches. Sections 2.2.1 and 2.2.2 explain active and passive online ensemble approaches for non-stationary environments, respectively.

2.2.1 Active Approaches

Active approaches for dealing with non-stationary environments are typically based on single learners. They use concept drift detection methods to determine whether a concept drift has occurred. When concept drift detection occurs, methods for dealing with concept drift are triggered. A common strategy is to reset the single learner to learn the new concept from scratch [14][58]. Some drift detection methods used in active approaches are explained in Section 2.3.

A few ensemble-based active approaches are also available in the literature. Adaptive Classifiers-Ensemble (ACE) [30] is an active online ensemble that consists of one online learner, a set of offline classifiers trained on old data, and a method that uses the offline classifiers to detect concept drift. Ensemble predictions are based on a weighted majority vote across all classifiers. The classifier weights are based on their accuracy on the most recent training examples. ACE claims to be able to handle sudden, gradual and recurring concepts better than other systems. However, its integral drift mechanism restricts the algorithm to integrate with other drift detection methods.

Bifet et. al. [14] presented an algorithm that combines restricted Hoeffding trees using stacking and an ADWIN [31] change detector. They applied ensemble trees using a weighing mechanism based on combining the log-odds of their probability estimates using sigmoid perceptron. The learning rate of the perceptron is determined by using a change detector that is also responsible for resetting the weaker base learners. The algorithm uses the learning rate $\alpha = 2/(2+m+n)$ for 'm' attributes and 'n' instances in the data stream. However, choosing the learning rate is problematic on identically distributed data and results in slow adaptation of the perceptron. One option is to reset the learning rate when drift is detected which improves the learning curve (rate of accuracy over time) while keeping the learning rate relatively large.

Todi [43] is based on two online classifiers for learning and detecting concept drift; 'H_0' and 'H_1'. Drift detections are performed based on a statistical test of equal proportions to compare 'H_0's performance on recent and old training examples. When a concept drift is detected, 'H_0' is reset. 'H_1' is never reinitialised upon drift detection but can be replaced by 'H_0' when a concept drift is confirmed. Keeping the two classifiers can help to deal with false positive drift detections, as 'H_1' can be selected for prediction in the case that the reset 'H_0' classifier is inaccurate after the drift detection. The Todi predictions are the predictions given by the classifier with the best accuracy with the most recent training examples.

Diversity for Dealing with Drifts (DDD) [13] is an online active ensemble learning approach that creates different ensembles with different levels of diversity to achieve robustness for different types of concept drift. A drift detection method is used to activate very high diversity ensembles which are not helpful during stable concepts, but that can help to deal with slow drifts, or drifts that do not cause too many changes with respect to the current concept.

Even though these approaches are based on single learners rather than heterogeneous ensembles, their use of drift detection methods can inspire the proposal of novel heterogeneous ensemble approaches. In particular, our proposed heterogeneous approach makes use of a drift detection method, being classified as an active approach.

2.2.2 Passive Approaches

Most passive learning approaches (those that do not rely on drift detection methods) deal with concept drift by maintaining an ensemble of base models and use weights to emphasise the models believed to best represent the current concept [11].

Addictive Base learner Ensembles (AddExp) [27] adds a new base model (a.k.a. base learner) for every wrong classification given by the ensemble. The weight assigned to the new base model is equal to the total weight of the ensemble multiplied by the parameter $\gamma \in (0,1)$. The weight of each base model is updated by being multiplied by a pre-defined parameter $(\beta,\ 0 \leq \beta < 1)$, when it gives a wrong prediction. A pruning method eliminates the oldest base models for reducing the ensemble size. Alternatively, the base models whose weight is below a certain threshold can be deleted. The prediction given by the ensemble is the weighted majority vote of the predictions given by the base models.

The Online Accuracy Updated Ensemble (OAUE) [28] combines chunk-based and online ensemble methods. The weights of the base learners are calculated by estimating the prediction error on the last d examples. The window size is utilised to create a new base learner for a set of 'd' examples and periodically removes the weaker base learners from the ensemble. The output is predicted by aggregating the predictions of base learners using a weighted voting rule. However, the algorithm is highly dependent on the window size. It is likely therefore that a small window size may lose the sudden concept drift, while a larger window may result in false concept detection.

The Dynamic Weighted Majority (DWM) [20] is one of the most popular ensemble approaches to deal with concept drift. Each base learner is associated with a weight. Weights start with value one and are multiplied by a pre-defined parameter $\beta,\ 0 \leq \beta < 1$, when their associated learner gives a wrong prediction in a time step multiple of period ρ. This weighting mechanism of DWM is inspired by the WMA. The predictions are based on the weighted majority vote derived from the base learners. DWM enables removal and addition of base learners at every ρ time step. A new base learner is added whenever the ensemble prediction is wrong in a time step multiple of ρ. Removal of learners is controlled by a pre-defined weight threshold parameter θ. A base learner is removed if its corresponding weight is lower than θ in a time step multiple of ρ. In this way, new learners are created to learn new concepts and poorly performing learners, which possibly had learnt old concepts, are removed. The algorithm normalises the weights by uniformly scaling them such that the highest weight will be equal to one. This is done to prevent newly added base learners from dominating the decision-making of existing ones. However, despite using the WMA weighting mechanism, DWM does not exploit one of the key aspects of WMA - the use of different *types* of base models.

Existing passive ensembles can be seen as performing dynamic model selection approaches when they assign different weights to their base learners and when they decide to remove base learners from the ensemble. However, these approaches have not exploited the use of different types of base learners, i.e., they have not exploited the potential benefit of heterogeneous ensembles. Even though the weighting mechanism of DWM was inspired by WMA which is a heterogeneous ensemble, all its base learners in DWM are homogeneous, e.g., either all of them are Naïve Bayes or all of them are Hoeffding Trees.

2.3 Drift Detection Methods

Several drift detection methods have been proposed. An example of a drift detection method based on statistical process control is the Drift Detection Method (DDM). It tracks the minimum error p_{min} of an online learning model over time and its corresponding standard deviation s_{min} by updating these variables whenever a new training example is received. A warning that a concept drift may be occurring is triggered if $(p_i + s_i \geq p_{min} + 2 \times s_{min})$, where '$p_i$' is the current error rate and 's_i' is the current standard deviation [32]. When this happens, new training examples are used not only to update the base model, but also stored in a buffer for future use. A concept drift is detected if $(p_i + s_i \geq p_{min} + 3 \times s_{min})$. The base model is then deleted and a new one is created to replace it using all the examples stored in the buffer.

The Early Drift Detection Method (EDDM) [33] is similar to DDM but takes into consideration the distance between two error classifications instead of the error rate. The average distance between two errors is represented as $'p'_i'$ and its corresponding standard deviation is $'s'_i'$. The warning level is reached if $(p_i + 2$

$\times s'_i) / (p'_{max} + 2 \times s'_{max}) < \propto$ and the drift level is reached if $(p'_i + 2 \times s'_i) / (p'_{max} + 2 \times s'_{max}) < \beta$, where $\propto$ and β are pre-defined constants.

The Statistical Test of Equal Proportion to Detect concept drift (STEPD) [43] monitors the two predictive accuracies of a single online classifier, i.e. accuracy among the most recent examples and overall accuracy from the beginning of the learning. It detects significant decreases in these predictive accuracies by using a statistical test of equal proportions. If the accuracies are statistically similar, then it is assumed that there is no concept drift. If the accuracies are significantly different, then a concept drift is detected. STEPD uses significance levels for drifts and warnings. Like DDM and EDDM, it stores examples in a short-term memory during the warning period and re-builds the classifier on drift detection based on the stored examples.

Giacomo et al. [46] analysed two different approaches for building histograms in the context of change detection. When building histograms, nonparametric monitoring procedures were applied which implemented likelihood [47][48] and distance-based approaches [49][50]. Their results show that the combination of uniform density histograms and a distance-based method achieved the best results in change-detection performance.

As will be shown in Section 3, the HDWM algorithm can make use of any drift detection method in its framework.

3 The Proposed HDWM Algorithm

An overview of the proposed approach HDWM is shown in Fig. 2. HDWM maintains a dynamic list of learners. In Stage 1, the seed learners $\mathcal{E}_1$ to $\mathcal{E}_a$ are initialised. In Stage 2, the learners in the dynamic learners are prequentially tested on each instance in the data stream. In Stage 3, the same instance is used for training the dynamic list. In Stage 4, on globally wrong prediction, a best performing learner is cloned from the seed learners and added to the dynamic list. The max size of dynamic list is controlled using parameter B_{max}. The learners of the ensemble ($\mathcal{E}_m$) make their predictions use their corresponding weights w_m.

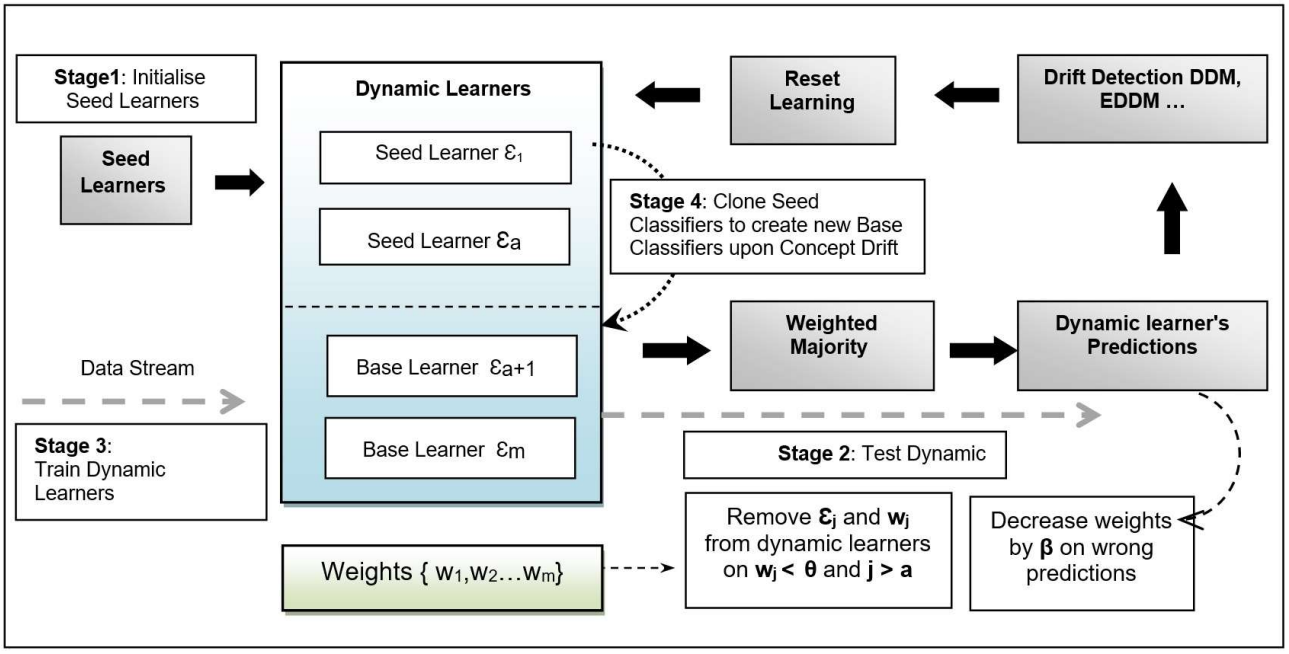

Fig. 2: Overview of HDWM

The global predictions on instances x_i for class label y'_i from a set of classes 'C' is based on the prediction made by 'm' base learners in dynamic list, $\mathcal{E}^j(x_i) \in C$. The ground truth for each example consists of pairs (x_i, y_i), and the aim was to combine the weighted predictions of each learner using their

corresponding weight w_j using Weighted Majority voting as shown in Eq. (1).

$$y_i' = \arg\max_{c \in C} \sum_{j \mid \varepsilon^j(\mathbf{x}_i)=c} w_i^j \quad (1)$$

Each learner in Ɛ is associated to a weight $\{w_1, w_2, \ldots, w_m\}$. The method to update the weights is similar as defined in DWM [20], i.e. by being multiplied by a factor β $(0 \leq \beta < 1)$ upon misclassifications at time-steps multiple of Period 'ρ', where ρ >= 1 is a pre-defined parameter.

HDWM implements both an active and passive approach for handling concept drifts, so that it is able to efficiently deal with different types of drift (gradual and abrupt). To implement a passive approach HDWM removes weaker learners and their associated weights from the dynamic list once their weights fall below the value predefined in parameter θ. After every 'ρ' time-steps, it performs the following tasks

1) When the global prediction of ensemble is wrong, a new learner is cloned from the "best" seed. The best seed is the seed corresponding to the base learner in Ɛ with the best weight.

2) Once the ensemble size exceeds a user pre-defined threshold B_{max}, the base learner which has the lowest weight is removed among $\mathcal{E}_j$, a+1 ≤ j ≤ m,.

These two approaches restrict the ensemble size to reduce the computational costs while enabling the ensemble to remain heterogeneous.

To implement an active approach, HDWM uses parameter δ to select a concept drift detection method e.g., DDM [32] or EDDM [33] and link it to each base learner in the ensemble. The predictions taken from the base learners are injected into their corresponding drift detection methods to detect concept drifts and warnings. To handle concept drifts HDWM has two options 1) Reset the learning of the seeds and their corresponding weights and re-train them. 2) Delete the weakest learners and create new learners of the same type as the best performing learner by cloning its seed.

The HDWM is outlined in Algorithm 1. Initially, the seed learners '$\mathcal{E}_1$ to $\mathcal{E}_a$ are initialised based on their base learning algorithm (line 2). Each learner in the dynamic list is assigned an equal weight 1.0 (line 3). Each base learner $\mathcal{E}_j$ in the dynamic list is asked for predictions on 'x_i' instances (Line 8), where *'i'* is the time-step and x is the vector representing attributes in the data-stream. Similar to the DWM rule [20] the weights of the learners are decreased upon incorrect predictions (Line 10-11). Over time when the ensemble grows, the base learners whose weights fall below θ are deleted while keeping intact the seeds in $\mathcal{E}$ for future use (Line 13-15), and set the flag $d = 1$ which indicates that the base learner has been deleted. By ensuring that at least one base learner of each type is maintained in $\mathcal{E}$, it is certain that a given type of base learner can repopulate the ensemble whenever it becomes beneficial, even if this follows a period of time when this type of base learner was not beneficial.

If no learner is deleted (line 17), the base learner's prediction is used to compute the weighted sum for each class (line 18). The maximum and minimum weights are stored in appropriate variables (line 19). The class with the most weight is then set as the global prediction (line 24). Weights are normalised using the DWM rule [20] (Line 26) and the parameter ρ is used to control the period for adding or removing the new dynamic learners.

An active drift detection method such as DDM [32] or EDDM [33] is invoked (line 22) and in the case of drift detection by any of the base learners, the Active Handle Drift (Algorithm 1.1) is invoked. The integration method for Active Drift Detection is explained in Section 3.1. On global wrong predictions (Line 27) the Passive Drift Handler (Algorithm 1.2) is invoked on (line 28). To control the ensemble size (line 30-32) parameter B_{max} is a user defined value to remove weaker learners from the dynamic learners list.

Algorithm 1: HDWM ($\{x,y\}_1^n$, β, θ, ρ)

Input: $\{x,y\}_1^n$: Stream of examples and class label

$\{\text{LearningAlgorithm}\}_1^a$: Set of Heterogeneous Seed Base Learning Algorithms

β: factor to decrease weights, $0 \leq \beta < 1$

θ: threshold to delete base learner

ρ: period between base learner removal, creation and weight update

$\{\mathcal{E},w,\delta\}_1^m$: Set of Seeds, Dynamic learners and Drift Detection Method

d {0,1}: base learner delete flag

B_{max}: Max size of ensemble

$c \in \mathbb{N}^*$: Number of classes, $c \geq 2$

$\wedge, \lambda \in \{1,\ldots,c\}$: global and local predictions

$\sigma \in \mathbb{R}^c$: sum of weighted prediction for each class

```
1  for seed = 1 to a                                   // Loop over seeds
2  |  Ε_seed ← Initialised_Seeds (LearningAlgorithm_seed)   // Clone seeds to Dynamic List
3  |  w_seed ← 1.0
4  end for
5  for i = 1 to n                                      // Loop over examples
6  |  for j = 1 to m                                   // Loop over ensemble of learners
7  |  |  d ← 0                                         // Learner's delete flag
8  |  |  λ = Classify (Ε_j, x_i )                      // Classify using both Seeds and dynamic learners in Ε
9  |  |  if (i mod ρ = 0) then
10 |  |  |  if (λ≠ y_i ) then
11 |  |  |  |  w_j ← β w_j                             // Update weight using DWM [20] rule
12 |  |  |  end if
13 |  |  |  if (w_j < θ and j > a) then                // j>a prevents deletion of seeds from Ε
14 |  |  |  |  {Ε_j, w_j } ← remove ({Ε_j, w_j }, θ)   // Delete learners whose Weights < θ
15 |  |  |  |  d ← 1;                                  // Set deleted flag to True
16 |  |  |  end if
17 |  |  |  if (d ≠ 1) then                           // If no learners are deleted
18 |  |  |  |  σ_λ ← σ_λ + w_j
19 |  |  |  |  w_min ← min(w), w_max ← max(w)
20 |  |  |  end if
21 |  |  end if
22 |  |  Call Active Drift Handler (λ , Ε, x_i ) (Algorithm 1.1)
23 |  end for
24 |  ∧ ← argmax_j σ_j
25 |  if (i mod ρ = 0) then
26 |  |  w ← normalize-weights ();                     // Using DWM [20] rule
27 |  |  if (∧ ≠ y_i) then                             // Global prediction is wrong
28 |  |  |  Call Passive Drift Handler (Algorithm 1.2)
29 |  |  end if
30 |  |  if size(Ε) = B_max then
31 |  |  |  { Ε,w} ← remove({Ε,w}, w_min)
32 |  |  end if
33 |  |  for i = 1 to m
34 |  |  |  Train (Ε_i, x_j )
35 |  |  end for
36 |  end if
37 end for
```

3.1 Drift Detection and Handling

Algorithm 1.1 outlines Active drift handling in HDWM. The seeds are reset upon the occurrence of drifts. The weight of the seeds are set to 0.5 instead of 1.0 (Lines 3-6) to prevent the domination of seeds over the new base learners. Finally, the seed learners are trained when the warning state is detected.

Algorithm 1.1 HDWM ActiveDrift Handling (λ, $\mathcal{E}$, δ, w, x_j)

Input: $\mathcal{E}$: Set of Seeds and Dynamic learners
λ: local predictions from base learners
w: ensemble weights
δ: Drift detection Method

1: $\delta_{local} \leftarrow$ DriftDetectionMethod(λ)
2: if (δ_{local} drift = true) // drift is detected
3: for seed = 1 to a
4: $\mathcal{E}_{seed} \leftarrow$ reset
5: $w_{seed} \leftarrow 0.5$
6 end for
7 end if
8: if (δ_{local} warning = true) // warning is detected
9: for j = 1 to a // Loop over seed learners
10: Train ($\mathcal{E}_i$, x_j)
11 end for
12: end if

Algorithm 1.2 implements the Passive drift handling mechanism in HDWM. In the case of globally wrong predictions the index position and the type of best seed learner is determined (line 1), a new classifier of a similar type is created (line 2) and added to the list dynamic learners from the seed learner (line 3). New learners are given weights 0.5 (line 5) to prevent new learners dominating over the existing ones.

Algorithm 1.2 PassiveHandleDrift ($\mathcal{E}$, w)

Input: $\mathcal{E}$: Set of Seeds and Dynamic learners
w: ensemble weights
w_{max} : maximum weight
m: size of the dynamic learners
$\{LearningAlgorithm\}_a^1$: Set of Seed Base Learners

1: Seed $\leftarrow$ bestLearner { $\mathcal{E}$, w_{max} }
2: $N_{seed} \leftarrow$ Initialised_Seeds {$LearningAlgorithm_{seed}$}
3: $\mathcal{E} \leftarrow \mathcal{E} \cup N_{seed}$ // append classifier to dynamic list
4: m $\leftarrow$ m+1
5: $w_m \leftarrow 0.5$

3.2 Maintaining the Heterogeneity

WMA [19] maintains heterogeneous ensembles, but is unable to deal with concept drifts due to its inability to create new learners and delete old learners. DWM [20] can deal with concept drift through the addition of new learners and deletion of inaccurate learners. However, it does not benefit from multiple types of base learners. Even if DWM was initialised with multiple types of base learners, because it deletes inaccurate base learners, it could become homogeneous over time and once it became homogeneous, it would not have any strategy to re-introduce other types of base learners if they become beneficial once again.

HDWM overcomes these problems presented by WMA and DWM. It enables the ensemble to deal with concept drift through the addition and removal of base learners, at the same time as it ensures that the ensemble can benefit from heterogeneity. It achieves that by ensuring that seed learners of any type can repopulate the ensemble whenever they become beneficial.

4 Experimental Setup

This Section investigates the HDWM algorithms and compares their accuracy and drift handling capabilities with WMA (due to its heterogeneity) and DWM (due to its ability to dynamically include and exclude base learners from the ensemble). Friedman tests with their corresponding post hoc tests are performed to support the comparison of several algorithms on multiple data streams.

Different variations of HDWM were compared to evaluate its sensitivity to parameters (e.g. drift and warning threshold, ensemble size) and variations of the algorithm that deactivate some of its characteristics (e.g. drift detection, warning detection, weighted vote). The second set of experiments concern the evaluation of computational resources usage (CPU time and RAM-Hours). Finally, experiments were presented comparing HDWM and other state of-the-art ensemble classifiers. Since accuracy can be misleading on data sets with class imbalance or temporal dependencies, Kappa M and Kappa Temporal were also used. Kappa M has advantages over Kappa statistic as it has a zero value for a majority class classifier [59]. Kappa Temporal is

applied since it replaces the majority class classier with the NoChange classifier [60]. This enables better estimations for data sets with temporal dependencies.

The evaluation metrics used are Prequential (P) Testing and Periodic Holdout (H). In Prequential Testing, each instance is used to test the model before it is used for training, and the accuracy is updated incrementally. The prequential accuracy is calculated based on the Massive Online Analysis (MOA) Windows Classification Performance Evaluator (WCPE) [34] with a window size of 1000. The Holdout method uses predefined partitions of train and test instances. However, it requires labelled test datasets which are difficult to obtain readily for real world applications. This method is applied in STAGGER (Drift) as pre-defined partitions of training and testing instances were used; the details are explained in section 4.1.

4.1 Data Streams

The artificial data streams used in the experiments are generated through the MOA workbench [34]. The details of the streams are given below, and the MOA commands to generate these streams are available in Appendix A. The characteristics and configuration of these data streams are summarised in Table 1.

- *RandomTrees (Recurring)* [34] generates a stream based on a randomly generated tree. The stream contains two sudden drifts. The first concept drift occurs at time step 25k and causes the first concept, which is described by 5 numerical attributes, to be replaced by 5 nominal attributes. At location 75k, the occurrence of a sudden drift re-introduces the first concept, which then lasts for 100k instances.
- *Hyperplane (Gradual Drift)* [34] is a flat *d*-dimensional space represented by $\sum_{i}^{d} w_i x_i = 0$, such that $\{x_i \ldots x_d\}$ are randomly generated instances and 'w' is the weight attribute. The instances are positive if $\sum_{i}^{d} w_i x_i \geq w_0$, where w_0 is the total weight. Gradual drift is introduced by slightly rotating the hyperplane by modifying w_i to 0.001 for each instance, and 5% noise is added in the stream.
- *Random Radial Basis Function (Gradual Drift)* [34] consists of a fixed number of randomly positioned centroids with a single standard deviation, class label and weight. New instances are generated by randomly choosing a centre. Gradual Drift is generated by choosing two centroids and gradually moving the centre at the speed level of 0.001 for each instance, and adding 5% noise in the stream.
- *SEA (Sudden and Gradual Drift)* [45] contains three attributes, function $x_i \in R$ and the value of x_i is between 1.0 and 10.0. The target concept is determined using the equation $y = [x_0 + x_1 + x_2 \leq \theta]$, such that $\theta \in \{7,8,9,9.5\}$. Two drifts are generated by changing the function x_1 to x_2. *Gradual Drift* appears at 25 for (width = 10k) and *Sudden Drift* at 75k for total 100k instances. For SEA *(Sudden)* two drifts are generated at the same location by using (width =1).
- *STAGGER (Sudden Drift)* [44] consists of three attributes, i.e. colour $\in$ {Red, Green, Blue}, size $\in$ {Small, Medium, Large} and shape $\in$ {Circle, Rectangle, Square}. The three concepts are [size = Small $\wedge$ colour = Red], [colour = Green $\vee$ shape = Circle] and [size; Medium $\vee$ Large]. The stream consists of 120 training instances, each concept is 40 instances long and sudden drifts appear at location 40 and 80. Each instance is evaluated on 100 test instances using Periodic Holdout (H).
- *LED (Sudden Drift)* [55] generates a stream defined by a 7-segment LED display and the task is to predict the digit (0-9). Such a stream was generated by emulating a sudden drift by combining two distributions. The first distribution was generated with the LEDGenerator and the second distribution was generated at location 50k using LEDGeneratorDrift and one attribute comprised a concept drift.
- *WaveForm (Sudden Drift)* [55] is a 3-class problem defined by 40 numerical attributes and shares its origin with the LED. The problem is to predict one of the three waveform types. The first distribution was generated with a WaveFormGenerator and the second distribution was generated at location 50k using WaveFormGeneratorDrift and setting 20 attributes with drift.
- *Sensor dataset* [54] deployed in the Intel Berkeley Research Lab, the sensor ID is used to label the class. The dataset consists of 220k instances; the input attributes include time-stamped topology information, along with humidity, temperature, light and voltage. The true drift locations are not known but gradual drifts exist as the light during working hours is generally stronger than at night, and the temperature readings of specific sensors may rise if there are meetings in the room [35].

- The *Spam email* dataset [36] contain input attributes that represent a gradual concept drift by the SpamAssassin collection. The dataset consists of 9,324 instances, 500 attributes and two target classes i.e. spam and legitimate. The attributes represent the presence of a given word in the email.
- The *Electricity* dataset [37] contains data consisting of 45,312 instances for a period of two years collected from the Australian New South Wales Electricity Market. Input attributes include day of the week, the NSW electricity demand, the Victoria electricity demand and the scheduled electricity transfer between states. The binary prediction task is to identify the change (up or down) of the price relative to a moving average. The concept drift appears due to changes in consumption habits due to unexpected events and seasonality.
- *The Forest Cover type* [57] dataset consists of the observation (30 x 30 meter cell) determined from the US Forest Service (USFS) Region 2 Resource Information System (RIS) data. The task is to predict the type of forest cover from cartographic variables such as Elevation, Slope, soil type etc.

Table 1: Characteristics of the Data Streams and Parameters Used in the Experiments

Stream	# Instances	# Features	Classes	# Drifts	Period	Freq.	Evaluation
SEA *(S)*	2500 K	3	2	2	50	1K	P
STAGGER *(S)*	12 K	3	2	2	1	1	H
RTree *R*	100K	10	2	2	50	1K	P
LED *(S)*		7	10	1			
Wave *(S)*		40	3	1			
Hyperplane *(G)*		10	2	3			
SEA *(G and S)*		3	2	2			
RRBF*(G)*		2	5	2			
Electricity	45,312	8	2	N/A	50	100	P
Spam	9,324	500	2			500	
Sensor	100K	5	58			1K	
Forest Cover	100K	54	7			1K	

[P] = Prequential Evaluation, [H] Periodic Holdout Evaluation, [R]= Recurrent Drift, [S] = Sudden Drift, [G]=Gradual Drift, Freq. and period are defined in Table 2.

4.2 Test Configuration

All the experiments are evaluated in terms of time and predictive performance. Processing time is measured in seconds and is based on the CPU time used for training and testing. All the experiments were performed on machines with Core i7 @ 3.4 GHz, 4 GB of RAM and experiments are presented in terms of CPU time. All experiments were executed within the MOA (Massive Online Analysis) framework.

The cross-validation techniques for measuring model performance are not suitable as the data streams originate from non-stationary environments. Therefore, the prequential method [62] was used, which is a commonly accepted estimation procedure in non-stationary environments. In this method each example is first used to test the model before it is used for training. The advantage of this method is that all the instances are used in training and testing, and therefore no specific holdout set is needed.

To determine the statistical significant differences between algorithms, non-parametric tests were carried out using Demsey's methodology [40]. For the statistical test the Friedman test was applied with α= 0.05 and the null hypothesis, "no statistical difference between the algorithms". If the null hypothesis was rejected, the Nemenyi post hoc test was used to identify which pairs of algorithms differ from each other.

The base learners used in DWM are NB (Naïve Bayes) and HT (Hoeffding Tree). HDWM and WMA are using four base learners, i.e. HT-MC (Majority Class at leaves), HT-NB (Naïve Bayes at leaves), HT-NBAdaptive and NB. The values β = 0.05 and θ = 0.01 are used as per the default values used in DWM [32]. Table 2 gives a description of the parameters used in the experiments.

Table 2: Parameters used in the experiments

Code	Description
β	Penalise learner's weight on wrong prediction
θ	Threshold of weights to remove base leaners
Period	The interval to create or remove base learners or to manipulate their weights
Freq.	The number of training examples between samples of learning performance

For the large data streams (size > 100K) and real-world datasets, the period is '50'. For small datasets, the period is '1'. *'Freq'* is the MOA sample frequency parameter corresponding to the number of training examples between samples of learning performance. Freq=1k is used for instances more than 100k and for smaller streams a lower value is applied.

To investigate the heterogeneity and its influence on active and passive drift handling approaches, a variant of HDWM, HDWM-P was developed which is heterogeneous although not utilising the Active Drift

handling option. This variant is used in the experiments in Section 5.3. The details of variants used in the experiments are described in Table 3.

Table 3: Variants used in the Experiments

Algorithms	Description of Algorithm
HDWM	HDWM uses Naïve Bayes and Hoeffding Tree; its Heterogeneous ensemble uses both Active and Passive Drift Handling.
HDWM – P	HDWM uses Naïve Bayes and Hoeffding Tree; its Heterogeneous ensemble uses only Passive Drift Handling, as used in Heterogeneity Analysis.

5 Evaluation of HDWM

This section investigates the proposed algorithm and compares its model switching capabilities, predictive accuracy and drift handling capabilities against the existing ensemble-based approaches WMA and DWM. We also investigated the effect of heterogeneity on the predictive performance and ensemble size in the presence of gradual, recurrent and sudden drifts on artificial data streams and real-world datasets.

5.1 Predictive Performance

The predictive capabilities of our new approach were tested on artificial data-streams and real-world datasets, corresponding ranks are determined such that higher averages are representing lower ranks. Significance tests and post hoc comparisons on ranks are performed to determine significance level and critical differences. The predictive accuracies of HDWM, DWM and WMA are shown in Table 4.

Table 4: Predictive Accuracies (%) of DWM-NB, DWM-HT WMA and HDWM

Streams	HDWM	DWM-NB	DWM-HT	WMA
SEA *(S)*	88.12 (1)	87.98 (2)	87.71 (3)	85.79 (4)
STAGGER *(S)*	82.8 (1)	81.82 (2)	81.26 (3)	55.08 (4)
RTree *R*	85.27 (1)	74.05 (4)	75.32 (3)	79.78 (2)
LED *(S)*	73.37 (3)	73.41 (1.5)	73.41 (1.5)	65.01 (4)
Wave *(S)*	82.16 (1)	80.31 (4)	80.34 (3)	80.65 (2)
Hyperplane *(G)*	88.12 (2)	88.08 (3)	88.19 (1)	80.54 (4)
SEA *(G and S)*	87.64 (1)	87.58 (2)	87.21 (3)	85.71 (4)
RRBF *(G)*	92.59 (3)	92.65 (2)	93.09 (1)	77.93 (4)
Electricity	89.4 (1)	79.73 (4)	84.06 (2)	80.92 (3)
Spam	90.54 (1)	87.83 (4)	88.39 (2)	88.04 (3)
Sensor	92.04 (1)	90.79 (3)	90.96 (2)	72.86 (4)
Forest Cover	91.03 (1)	82.92 (2)	79.33 (4)	80.65 (3)
Avg. Ranks	**1.42**	**2.79**	**2.38**	**3.42**

In both drift and real-world streams the χ^2_F statistic is 15.25 *(df =3, N = 12)* and the p-value 0.0016 shows significant differences at the level of significance of 0.05. The method to calculate chi-squared and p-value is described by Demsar [40]. The Nemenyi test [39] was applied for pairwise comparison. The critical difference [40] is 1.35. It is evident from the bar chart (green bars) in Fig. 3 that HDWM performed significantly better than DWM-NB i.e. (2.79 – 1.42 = 1.38 > 1.35) and WMA (3.42 – 1.42 = 2.0 > 1.35).

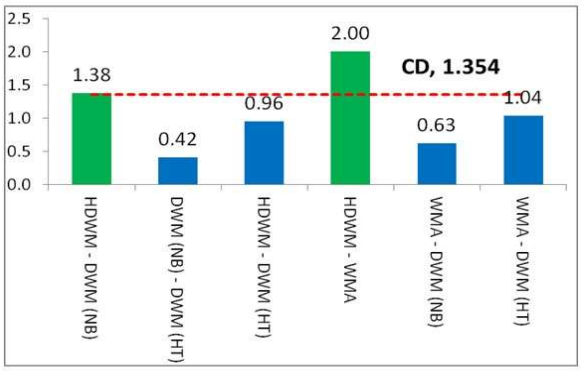

Fig. 3 Bar chart for pairwise comparisons between HDWM, DWM and WMA. Green bar indicates significantly different, and blue bars represent no significant difference

Tables 5 and 6 provide the Kappa measures for the experiments. The Kappa evaluation measure is widely used in data stream mining, it can handle both multi-class and imbalanced class problems. The larger the Kappa value, the more generalised the classifier, negative Kappa values indicate low predictive accuracy. Kappa values for Spam and Forest Cover datasets were negative in HDWM, DWM and WMA due to the large numbers of attributes in these datasets.

Table 5: Kappa Temporal DWM-NB, DWM-HT WMA and HDWM

Streams	HDWM	DWM-NB	DWM-HT	WMA
SEA *(S)*	73.81 (1)	73.47 (2)	72.87 (3)	68.84 (4)
STAGGER *(S)*	49.2 (1)	40.14 (2)	39.44 (3)	-19.43 (4)
RTree *R*	68.69 (1)	47.73 (4)	50.34 (3)	59.35 (2)
LED *(S)*	70.54 (1)	70.47 (2)	70.46 (3)	61.14 (4)
Wave *(S)*	73.36 (1)	70.41 (4)	70.46 (3)	70.94 (2)
Hyperplane *(G)*	75.05 (3)	76.14 (2)	76.37 (1)	61.07 (4)
SEA *(G and S)*	71.68 (3)	73.03 (1)	72.22 (2)	66.69 (4)
RRBF *(G)*	91.14 (2)	91.13 (3)	91.66 (1)	73.39 (4)
Electricity	16.91 (1)	-44.88 (4)	-14.83 (2)	-36.85 (3)
Sensor	92.5 (1)	90.78 (3)	90.95 (2)	72.84 (4)
Forest Cover	-153.1 (1)	-361.2 (3)	-163.1 (2)	-388.9 (4)
Avg. Ranks	**1.45**	**2.73**	**2.27**	**3.55**

Table 6: Kappa M for DWM-NB, DWM-HT WMA and HDWM

Streams	HDWM	DWM-NB	DWM-HT	WMA
SEA *(S)*	66.65 (1)	66.17 (2)	65.39 (3)	60.4 (4)
STAGGER *(S)*	13.09 (1)	0.59 (3)	0.72 (2)	-76.4 (4)
RTree *R*	66.0 (1)	43.27 (4)	46.2 (3)	55.98 (2)
LED *(S)*	69.99 (1)	69.92 (2)	69.91 (3)	60.39 (4)
Wave *(S)*	72.62 (1)	69.6 (4)	69.65 (3)	70.13 (2)
Hyperplane	74.46 (3)	75.58 (2)	75.81 (1)	60.1 (4)
SEA *(G and S)*	64.01 (3)	65.74 (1)	64.7 (2)	58.55 (4)
RRBF*(G)*	90.88 (2)	90.87 (3)	91.41 (1)	72.58 (4)
Electricity	71.84 (1)	50.91 (4)	61.26 (2)	53.36 (3)
Sensor	92.15 (1)	90.36 (3)	90.54 (2)	71.6 (4)
Forest Cover	64.45 (1)	38.85 (3)	61.99 (2)	37.55 (4)
Avg. Ranks	**1.45**	**2.82**	**2.18**	**3.55**

The statistical tests applied on Kappa Temporal on drift and real-world streams, with the χ^2_F statistic of 15.76 *(df =3, N = 11)* and the p-value of 0.0012 showed significant differences at the level of significance of 0.05. Statistical tests for Kappa M on both drift and real-world streams, the χ^2_F statistic is 15.10 *(df =3, N = 11)* and the p-value 0.0017 also shows significant differences at the level of significance of 0.05. The Nemenyi test [39] was applied for Kappa Temporal and Kappa M for pairwise comparison. The critical difference [40] is 1.41. HDWM performed significantly better than WMA.

Even though WMA is heterogeneous, it performed worst in most of the drift streams and real-world datasets, the reason is a lack of drift handling capabilities. Apart from this, there was no significant difference between DWM-NB and DWM-HT, DWM-HT and WMA and DWM-NB and WMA. This makes it extremely difficult to choose an optimal base classifier in DWM. We can conclude that HDWM is independent of deciding on which type of base classifier should be used.

5.2 Resources comparison

To analyse the benefits in terms of resources usage we compare HDWM, DWM and WMA. We recorded an evaluation time of HDWM in CPU seconds by setting max size of ensemble (B_{max}) to 25, 50,100 for all the data sets. It is expected that HDWM requires more processing time compared with WMA and DWM due to the seed learners that always reside in the ensemble. As shown in Fig. 4, the total CPU time is increasing by setting a larger value of B_{max}, however, the average predictive accuracies are not significantly affected.

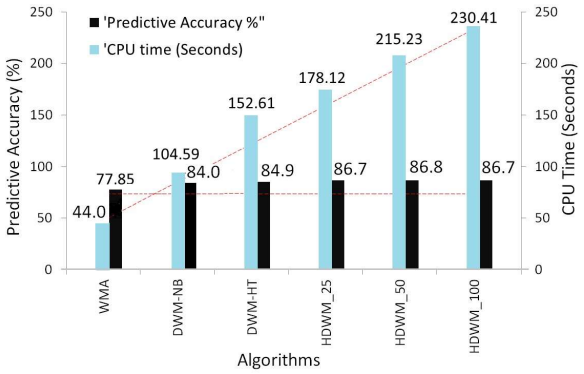

Fig. 4 CPU time (Seconds) and Predictive Accuracies of HDWM, DWM and WMA.

5.3 Analysis of Heterogeneity

The objective of this analysis is to investigate how the heterogeneity of an ensemble affects its predictive performance and whether the higher accuracy achieved in HDWM is due to its heterogeneity or due to its active drift handling capabilities. The results of these experiments are shown in Table 7.

Table 7: Heterogeneity Test, Predictive Accuracies (%)

Streams	HDWM -P	DWM (NB)	DWM (HT)
SEA *(S)*	87.73 (2)	87.98 (1)	87.71 (3)
STAGGER *(S)*	82.31 (1)	81.82 (2)	81.26 (3)
RTree *R*	75.51 (1)	74.05 (3)	75.32 (2)
LED *(S)*	73.44 (1)	73.42 (2)	73.41 (3)
Wave *(S)*	80.35 (1)	80.31 (3)	80.34 (2)
Hyperplane *(G)*	88.21 (1)	88.08 (3)	88.19 (2)
SEA *(G and S)*	87.26 (2)	87.58 (1)	87.21 (3)
RRBF*(G)*	93.04 (2)	92.65 (3)	93.09 (1)
Electricity	84.09 (1)	79.73 (3)	84.06 (2)
Spam	88.72 (1)	87.83 (3)	88.39 (2)
Sensor	90.98 (1)	90.79 (3)	90.96 (2)
Forest Cover	86.92 (1)	82.92 (2)	79.33 (3)
Avg. Ranks	**1.25**	**2.42**	**2.33**

For this experiment the DWM performance was compared with the Naïve Base and Hoeffding Tree as base learners in its ensemble and compared it with HDWM-P (a variant of HDWM without active drift handling) which is reliant on a passive approach similar to the DWM. The Friedman statistics [38] in a heterogeneity test, the χ^2_F statistic is 10.16 *(df=2, N = 12)* and the p-value 0.0062 indicates significant differences at the level of significance of 0.05. Post-hoc test using the Nemenyi test [39] was applied for pairwise comparison. The critical difference is 0.902. Box-plot in Fig. 5 shows that HDWM-P performed significantly better than DWM-NB i.e. (2.42 – 1.25 = 1.08 > 0.902) and DWM-HT (2.33 – 1.25 = 1.17 > 0.902). Given that the main difference between

HDWM-P and DWM is the heterogeneity, these results indicate that heterogeneity plays a key role in improving the HDWM accuracy over DWM. In particular, the model switching mechanism maintained the accuracy, making it independent of manually selecting base learners.

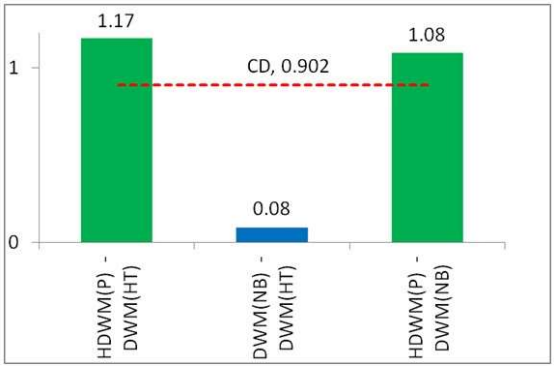

Fig. 5 Pairwise bar chart for Heterogeneity Test (Green bar) significantly different, (blue bars) No significant differences

5.4 Further Analysis on Artificial Drift Streams

In this section an in-depth analysis of the results achieved in the previous experiment are presented using the artificial drifts data streams. The predictive performances are analysed and the capabilities of each algorithm are graphically presented to investigate how these algorithms react to different type of drifts. The ensemble size was also analysed. The Ensemble Size in a dynamic base classifier is an important factor for balancing performance because a larger ensemble requires more processing time but may improve predictive accuracy.

5.4.1 Accuracy over Time

Fig. 6(a) represents RandomTree recurring concept drifts. HDWM (85.27%) and WMA (79.78%) handled the drift on a recurring concept at 75,000 instances. DWM-NB (74.05%) and DWM-HT (75.32) were unable to cope after the first sudden drift at 25,000. The base learners in DWM forgot the previous learnt concepts due to inclusion and removal of their base learners; unlike the WMA whose base learners are never deleted.

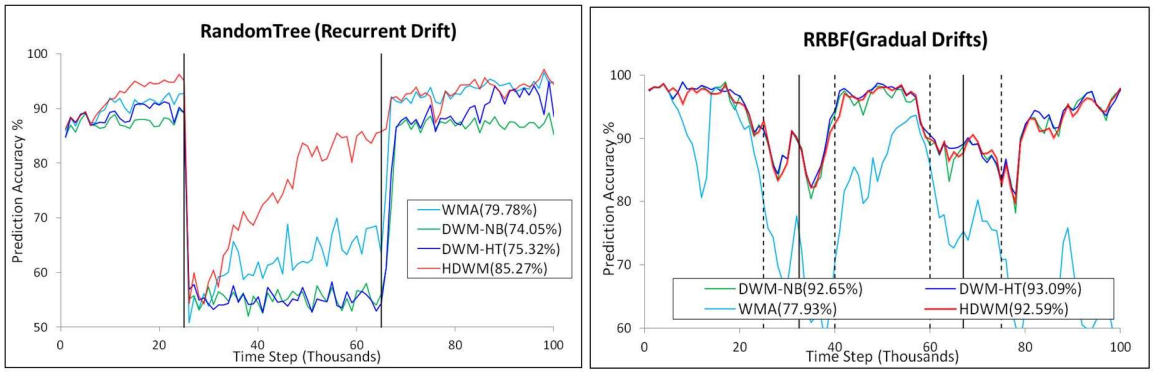

Fig. 6(a): Predictive Accuracies RandomTree (left) and RRBF (right) on Artificial Data Streams. Solid and dashed vertical black lines indicate the centre point of the drifts, and start/end of the drifts, respectively. The time steps between the start and end of the drift (inclusive) compose the drift window.

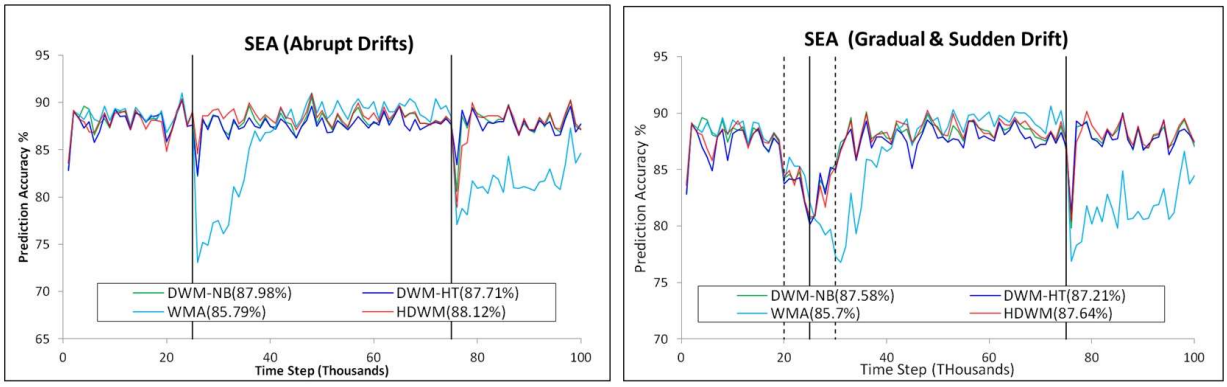

Fig. 6(b): Predictive Accuracies SEA Abrupt (left) and SEA Mixed (right) on Artificial Data Streams. Solid and dashed vertical black lines indicate the centre point of the drifts, and start/end of the drifts, respectively. The time steps between the start and end of the drift (inclusive) compose the drift window.

In HDWM the seeds are never deleted and retain the previously learnt concepts, this helps HDWM in appropriately dealing with recurring concept drifts. In RRBF Fig. 6(a), which represents gradual drifts, HDWM (92.59%) and DWM are able to deal with concept drifts appropriately due to periodically including new base learners while WMA does not; this being due to its static ensemble size. HDWM not only maintained the predictive accuracy of DWM but slightly improved it.

SEA Fig 6(b), represents abrupt drifts at 25,000 instances and 75,000 instances. HDWM and DWM handled these drifts appropriately, however, WMA failed to adapt to the new concept. SEA (Mixed) Fig. 6(b), represents gradual and sudden drifts. Gradual drift is centred around instance 25,000 with a window of 10,000 instances and is represented using a dotted line while the sudden drift occurs at 75,000 instances. DWM and HDWM both handled these drifts appropriately, but WMA reacted late on mixed concept drifts.

5.4.2 Ensemble Size

Due to the seed learners that always remain in the dynamic list, HDWM maintained a larger ensemble size (Average 27.6). HDWM in RTree $_{(R)}$ and Wave $_{(S)}$ utilised smallest ensemble (13.19 and 18.02) in achieving higher predictive accuracies (85.27% and 82.16%) compared with DWM and WMA. Table 8 represents average ensemble sizes and corresponding ranks achieved in HDWM and DWM; the lower averages representing higher ranks. The plots for the ensemble size in artificial data streams are shown in Fig 7(a) and (b).

Table 8: Average Ensemble Size in Artificial Data Streams

Streams	HDWM	DWM-NB	DWM-HT
SEA $_{(S)}$	61.39 (3)	35.72 (2)	25.38 (1)
STAGGER $_{(S)}$	12.18 (3)	7.73 (2)	7.07 (1)
RTree $_{R}$	13.19 (1)	28.37 (3)	16.69 (2)
LED $_{(S)}$	33.94 (1)	37.1 (2.5)	37.1 (2.5)
Wave $_{(S)}$	18.02 (1)	37.83 (3)	29.09 (2)
Hyperplane $_{(G)}$	22.91 (3)	14.28 (2)	13.52 (1)
SEA $_{(G\ and\ S)}$	43.56 (3)	37.89 (2)	25.6 (1)
RBF$_{(G)}$	16.26 (1)	8.76 (2)	10.48 (1)
Average	**27.6**	**25.9**	**20.6**
Avg. Ranks	**2.25**	**2.18**	**1.56**

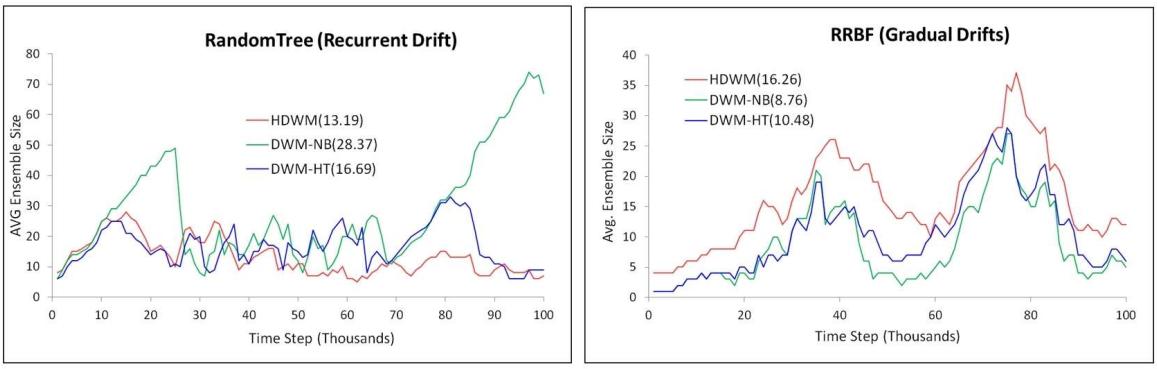

Fig. 7(a): Average Ensemble Size RandomTree (left) and RRBF (right) in Artificial Data Streams

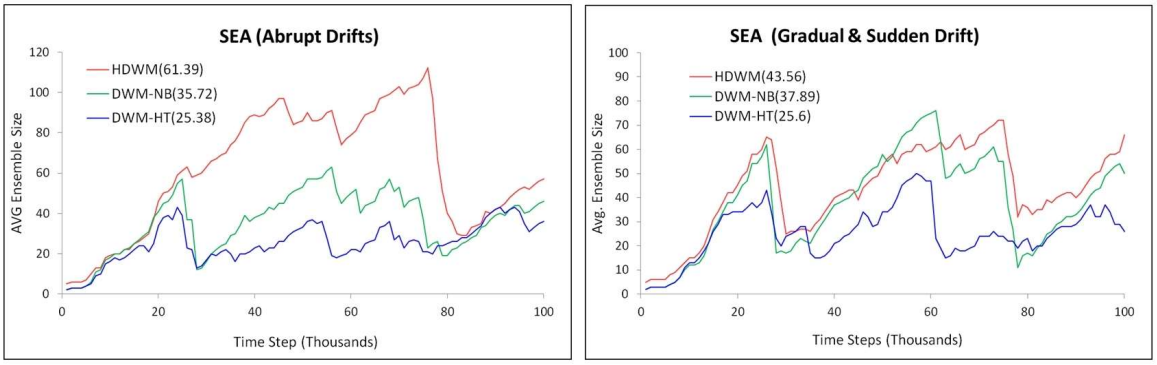

Fig. 7(b): Average Ensemble Size SEA Abrupt (left) and SEA Mixed (right) in Artificial Data Streams

5.5 Further Analysis on Real-World Datasets

Artificial data streams are typically designed for controlled environments. Several challenges emerge when dealing with real-world classification problems. The primary issues are the identification and location of the concept drifts. Accordingly, the HDWM was also evaluated on real-world data streams; namely: Electricity [37], Sensor [54], Forest Cover type [57] and Spam email dataset [36]. As there are only 4 datasets and thus 4 observations, no significance test was performed. However, the obtained results show improvements.

5.5.1 Accuracy over Time

As shown in Fig. 8, HDWM achieved the highest predictive accuracies on Spam email (90.54%), Electricity (89.4%), Forest Cover type (91.03%) and Sensor (92.04%). Overall the HDWM average ranking in real-world datasets is (1.0), DWM-HT (2.5) and DWM-HT and WMA (3.25).

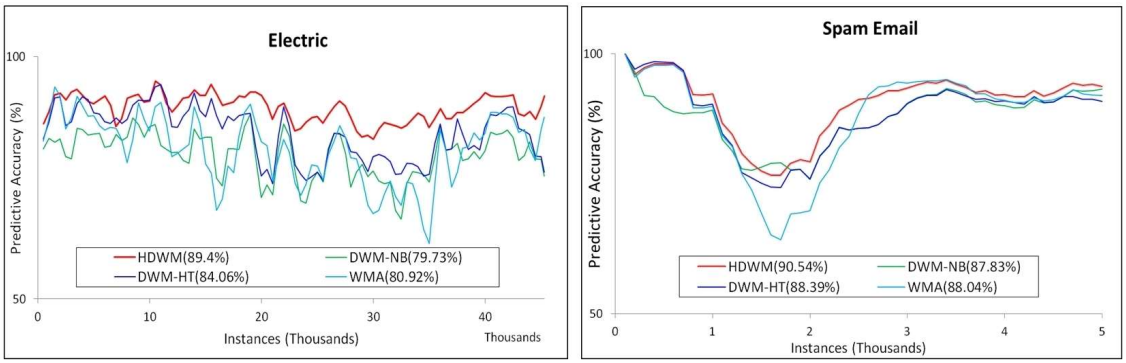

Fig. 8(a): (Left) Average Predictive Accuracies Electric dataset, (Right) Spam Email

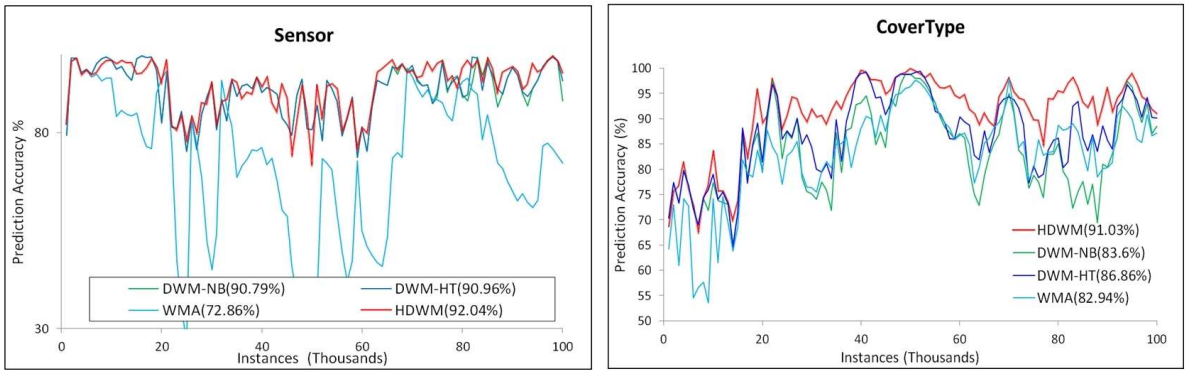

Fig. 8(b): (Left) Average Predictive Accuracies Sensor, (Right) Forest Cover

5.5.2 Ensemble Size

The ensemble sizes in DWM and HDWM are dynamic, i.e. growing and shrinking based on the predictive performance and the drift detections. HDWM achieved higher accuracy on the Sensor dataset (90.73%) using the lowest ensemble size (Average 8.04). Table 9 shows the average ensemble size and ranks in real world datasets with the lower averages representing higher ranks.

The plots in Fig. 9(a) and (b) show average ensemble sizes for real-world datasets. In general HDWM uses a slightly larger ensemble size (11.29) as compared with DWM (10.74), The reason for the larger ensemble in HDWM is that its base learners begin with 4 seed learners unlike DWM which uses a single base learner that evolves over time.

Table 9: Average Ensemble Size (%) and ranks of DWM-NB, DWM-HT WMA and HDWM, Real-world datasets

Streams	HDWM	DWM-NB	DWM-HT
Electricity	12.26 (3)	11.33 (1)	11.88 (2)
Spam	11.45 (3)	7.79 (1)	8.12 (2)
Sensor	8.04 (1)	8.58 (2)	9.06 (3)
Forest Cover	13.41 (2)	15.26 (3)	10.04 (1)
Average	**11.29**	**10.74**	**9.78**
Avg. Ranks	**2.25**	**1.75**	**2.00**

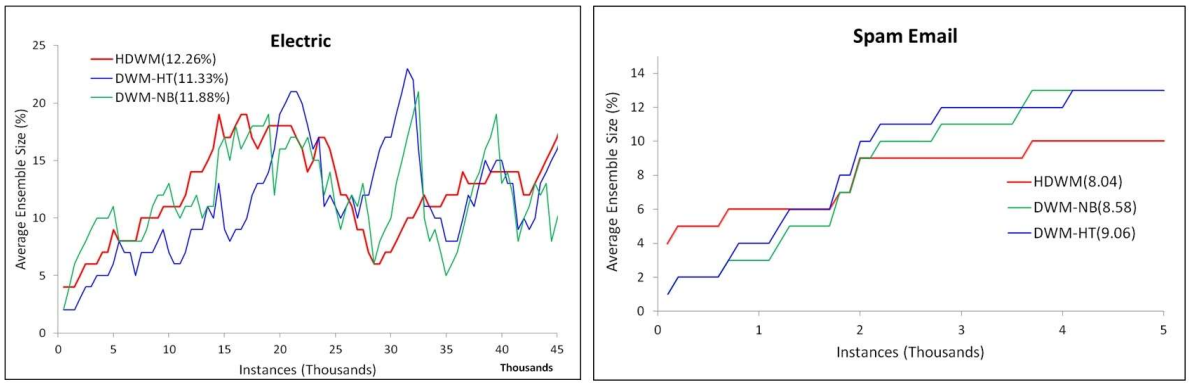

Fig. 9(a): Average Ensemble size in Electric (left) and Spam Email (right)

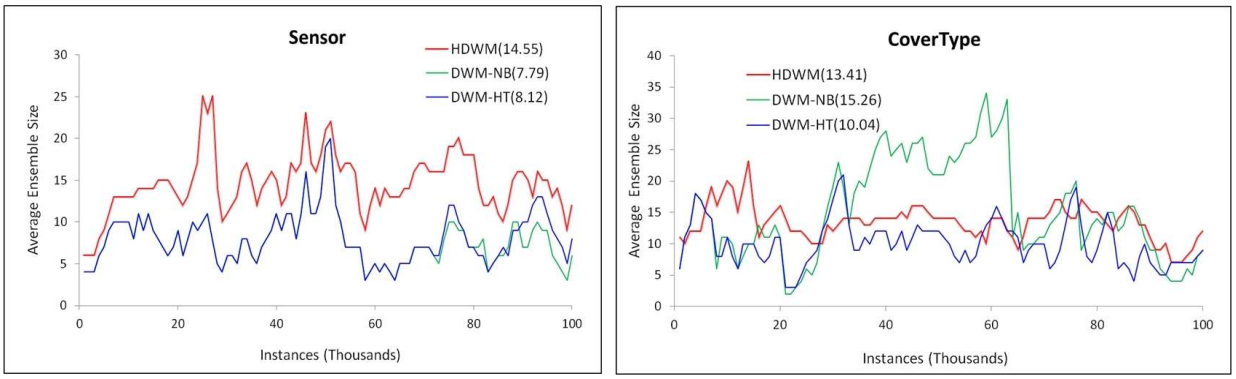

Fig. 9(b): Average Ensemble size in Sensor (left) and Cover Type (right)

5.6 Parameters Analysis

In terms of how to set the parameters in real world problems, the difficulty is that the best values may change over time. Potentially, one could run multiple versions of the approach with different parameter settings [61]. The parameters 'β', 'θ' and 'Period' were analysed and their effect on prediction accuracy, ensemble size and drift detections. The values for β ' and ' θ ' are randomly chosen between 0 and 1. While the period was also analysed on random values 1, 25 and 50; the period = 1 representing inclusion of all the instances in the data stream and then gradually increased by skipping 25 instances. The results on the 'effect of 'Period' on Predictive Accuracy and Drift Detection' is shown in Table 10. As evident from the table, the average prediction accuracy is gradually increasing while the number of drift detections is decreasing by applying a larger value of 'period'.

Table 10: Effect of 'Period' on Predictive Accuracies % & Drift Detection, β = 0.5 and θ = 0.01 (Fixed)

Streams	Period =1		Period =25		Period =50	
	Acc%	# Drifts	Acc%	# Drifts	Acc%	# Drifts
SEA (S)	84.0 (3)	0	87.9 (2)	2	88.1 (1)	2
STAGGER (S)	85.2 (1)	0	61.3 (2)	0	60.8 (3)	0
RTree R	76.5 (3)	6	82.5 (2)	1	84.4 (1)	1
LED (S)	54.8 (3)	4	72.2 (2)	1	73.4 (1)	1
Wave (S)	78.2 (3)	5	82.1 (2)	0	82.2 (1)	0
Hyperplane	77.5 (3)	4	85.5 (2)	0	87.5 (1)	0
SEA (G and S)	82.9 (3)	0	87.7 (1)	1	87.1 (2)	4
RRBF(G)	90.8 (3)	8	93.0 (1)	4	92.6 (2)	8
Electricity	89.4 (1)	8	89.3 (2)	2	88.4 (3)	2
Spam	93.9 (1)	2	89.9 (2)	0	89.7 (3)	0
Sensor	83.2 (3)	26	93.6 (1)	1	92.5 (2)	3
Forest Cover	90.7 (1)	10	90.4 (2)	0	89.7 (3)	0
Avg. (Ranks)	**82.3(2.2)**	**6.08**	**84.6(1.7)**	**1.0**	**84.7(2.0)**	**1.75**

The effect of 'β' on Predictive Accuracy and Ensemble Size is analysed by keeping a static value of 'Period = 50'. This value was chosen for subsequent experiments, as it achieved the highest accuracies in the experiments outlined in Table 10. In Table 11, the average ensemble size and accuracy is increasing by choosing a larger value of 'β'.

Table 11: Effect of 'β' on Predictive Accuracies % & Ensemble Size, Period = 50 and θ = 0.01 (Fixed)

Streams	β = 0.1		β = 0.5		β = 0.75	
	Acc%	Ensemble Size	Acc%	Ensemble Size	Acc%	Ensemble Size
SEA (S)	87.6 (3)	13.5	88.1 (1)	23.6	87.9 (2)	23.7
STAGGER (S)	60.8 (1)	4.0	60.8 (2)	4.0	60.7 (3)	4.0
RTree R	75.2 (3)	8.5	84.4 (2)	13.7	88.8 (1)	20.5
LED (S)	72.5 (3)	9.4	73.4 (1)	23.8	73.4 (2)	24.5
Wave (S)	80.3 (3)	13.3	82.2 (2)	17.4	83.5 (1)	24.5
Hyperplane	88.2 (1)	9.4	87.5 (3)	19.8	87.6 (2)	24.4
SEA (G and S)	87.4 (2)	12.3	87.1 (3)	17.2	88.3 (1)	23.2
RRBF(G)	92.6 (2)	8.6	92.6 (1)	14.8	92.5 (3)	20.3
Electricity	85.8 (3)	7.07	88.4 (2)	10.7	89.7 (1)	15.8
Spam	89.2 (3)	6.36	89.7 (2)	8.6	90.1 (1)	9.5
Sensor	93.0 (1)	6.74	92.5 (2)	10.3	91.7 (3)	13.8
Forest	85.7 (3)	7.17	89.7 (2)	11.5	91.3 (1)	16.5
Avg. (Ranks)	**83.2(2.2)**	**8.8**	**84.7(2.0)**	**14.6**	**85.5(1.7)**	**18.3**

Table 12: Effect of 'θ' on Predictive Accuracies % & Ensemble Size, Period = 50, β = 0.5 (Fixed)

Streams	θ = 0.01		θ = 0.05		θ = 0.1	
	Acc%	CPU time	Acc%	CPU time	Acc%	CPU time
SEA (S)	88.1 (2)	102.5	88.1 (1)	103.6	88.0 (3)	95.1
STAGGER (S)	60.8 (2)	0.04	60.8 (2)	1.0	60.8 (2)	1.0
RTree R	84.4 (1)	238.5	81.0 (2)	195.9	79.9 (3)	155.0
LED (S)	73.4 (1)	664.5	73.3 (2)	690.0	73.3 (3)	589.3
Wave (S)	82.2 (1)	1195.6	81.9 (2)	766.1	81.5 (3)	730.1
Hyperplane	87.5 (3)	508.6	87.8 (2)	429.3	88.2 (1)	343.1
SEA (G and S)	87.1 (3)	127.1	87.7 (1)	106.4	87.6 (2)	99.7
RRBF(G)	92.6 (2)	203.4	92.6 (1)	127.2	92.5 (3)	121.5
Electricity	88.4 (1)	148.4	88.3 (2)	153.5	87.9 (3)	127.6
Spam	89.7 (3)	148.9	90.0 (1)	155.6	89.9 (2)	128.1
Sensor	92.5 (2)	106.5	92.5 (3)	965.2	92.9 (1)	788.1
Forest	89.7 (1)	668.2	89.3 (2)	607.0	88.4 (3)	492.6
Avg. (Ranks)	**84.7(1.8)**	**442.5**	**84.4(1.8)**	**358.4**	**85.5(2.3)**	**305.9**

In another experiment, parameter 'θ' was analysed on predictive accuracies and CPU-time. Beta = 0.5 was fixed due to the moderate average ensemble size in the experiment outlined in Table 11. The results in Table 12 show that the CPU-time slightly decreased by increasing the value of θ. By increasing 'θ' the average ranks increased from 1.8 to 2.3. The lower ranks show a higher predictive performance.

6 Conclusion

The development of Heterogeneous Dynamic Weighted Majority (HDWM) algorithms revealed the ability to reduce human dependency on re-defining the best type of predictive models for a particular problem. The algorithm exhibited responsive adaptation; dealing appropriately with changing environments in a shorter period to increase the reliability and predictive accuracy of the model. It was also found that heterogeneity was a key enabler for the improved accuracy achieved by HDWM.

HDWM improved the predictive accuracies in the presence of different types of drifts, such as Gradual, Sudden and Recurring. It had been a key challenge in data stream mining, as some algorithms heavily rely on forgetting mechanisms while others retain previous learning. The HDWM seeding mechanism and dynamic inclusion of new base learners benefiting the use of both forgetting and retaining the models. In some of the data streams it performed in a similar way to DWM-HT and DWM-NB and the WMA, however the HDWM achieved these accuracies using a compact ensemble size and CPU time. The overall accuracy plots are representing the independence of choosing the right type of models in a given time and conditions.

As future work, we would like to investigate the HDWM performance on more diverse problems and in the presence of large number of attributes. We will also investigate to reduce its dependency on human pre-defined parameters.

References

[1] L. Watkins, et al "Using semi-supervised machine learning to address the Big Data problem in DNS networks," 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2017, pp. 1-6. DOI: 10.1109/CCWC.2017.7868376

[2] G. E. Melo-Acosta, F. Duitama-Muñoz, J. D. Arias-Londoño, "Fraud detection in big data using supervised and semi-supervised learning techniques," 2017 IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, Colombia, 2017, pp. 1-6. DOI: 10.1109/ColComCon.2017.8088206

[3] A. D. Gabriel, D. T.Gavrilut, B. I. Alexandru, and P. A. Stefan, "Detecting Malicious URLs: A Semi-Supervised Machine Learning System Approach," 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, 2016, pp. 233-239. DOI: 10.1109/SYNASC.2016.045

[4] D. Gavrilut, R. Benchea, C. Vatamanu, "Optimized zero false positives perceptron training for malware detection". In 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012, pages 247–253, 2012. DOI: 10.1109/SYNASC.2012.34

[5] E. Cambra., A. Hussain, Sentic Computing: "A Common-Sense-Based Framework for Concept-Level Sentiment Analysis", Springer, Cham, Switzerland, 2015. ISBN: 978-3-319-23654-4

[6] V. Iosifidis, E. Ntoutsi, "Large Scale Sentiment Learning with Limited Labels", In Proceedings of KDD '17, Halifax, NS, Canada, August 13-17, 2017. DOI: 10.1145/3097983.3098159

[7] S. Chernbumroong, A. S. Atkins, H. Yu, "Activity classification using a single wrist-worn accelerometer," 2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA) Proceedings, Benevento, 2011, pp. 1-6. DOI: 10.1109/SKIMA.2011.6089975

[8] B. Krawczyk., "Active and adaptive ensemble learning for online activity recognition from data streams", knowledge-Based Systems Volume 138, 15 December 2017, Pages 69-78. DOI: 10.1016/j.knosys.2017.09.032

[9] A. D. Pozzolo, G. Boracchi, et. al., "Credit card fraud detection and concept-drift adaptation with delayed supervised information," 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, 2015, pp. 1-8. DOI: 10.1109/IJCNN.2015.7280527

[10] G. Ditzler , M. Roveri, C. Alippi, R. Polikar, "Learning in Nonstationary Environments A Survey", IEEE Computational Intelligence Magazine, vol. 10, no. 4, pp. 12-25, Nov. 2015. DOI: 10.1109/MCI.2015.2471196

[11] B. Krawczyk, L. L. Minku, J. Gama , et. al., "Ensemble Learning for Data Stream Analysis: a survey", Information Fusion, v. 37, p. 132-156, January 2017. DOI: 10.1016/j.inffus.2017.02.004

[12] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, 2013. "A Survey on Concept Drift Adaptation". ACM Comput. Surv. 1, 1, Article 1 (January 2013), 35 pages. DOI: 10.1145/2523813

[13] L. L. Minku, X. Yao, "DDD: A New Ensemble Approach for Dealing with Concept Drift", in IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 4, pp. 619-633, April 2012. DOI: 10.1109/TKDE.2011.58

[14] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer, "Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking," in Proc. 2nd Asian Conf. Mach. Learn., vol. 13. 2010, pp. 1–16. DOI:10.1145/2089094.2089106

[15] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence – vol. 2 pp. 1137-1143, Aug 1995. ISBN:1-55860-363-8

[16] D. H. Wolpert, (1992), "Stacked generalization", Neural Networks 5, 241-259. DOI 10.1016/S0893-6080(05)80023-1

[17] S. Dzeroski and B. Zenko, "Stacking with multi-response model trees", In Proceedings of the 3d International Workshop in Multiple Classifier Systems. Springer, 2002. ISBN:3-540-43818-1

[18] C. J. Merz, (1999). "Using correspondence analysis to combine classifiers". Machine Learning, 36:1/2, 33–58. DOI: 10.1023/A:1007559205422

[19] N. Littlestone and K. Warmuth, (1994). "The Weighted Majority Algorithm". Information and Computation, 108:212–261. DOI 10.1006/inco.1994.1009

[20] J. Z. Kolter. and M. A. Maloof, (2007), "Dynamic weighted majority: An ensemble method for drifting concepts". The Journal of Machine Learning Research, 8:2755-2790. DOI: 10.1109/ICDM.2003.1250911

[21] J. N. Rijn, H. M. Gomes, B. Pfahringer. and J. Vanschoren, "Algorithm Selection on Data Streams" in Discovery Science, ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8777, pp. 325–336. DOI: 10.1007/978-3-319-11812-3_28

[22] A L. Debiaso, A. C. Ponce, C. Soares, B. Feresde, "MetaStream: A meta-learning-based method for periodic algorithm selection in time-changing data," Neurocomputing, vol. 127, pp. 52–64, 2014. DOI: 10.1016/j.neucom.2013.05.048

[23] H. Nguyen, Y. Woon, N. Wee-Keong, L. Wan, "Heterogeneous Ensemble for Feature Drifts in Data Streams", in Advances in Knowledge Discovery and Data Mining. Springer, 2012, pp. 1–12. DOI: 10.1007/978-3-642-30220-6_1

[24] W. X. Cheng, R. Katuwal, P. N. Suganthan, Q. Xueheng, "A heterogeneous ensemble of trees", 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, 2017, pp. 1-6. DOI: 10.1109/SSCI.2017.8285445.

[25] B.S. Parker, L Khan and A. Bifet., "Incremental Ensemble Classifier Addressing Non-Stationary Fast Data Streams." Data Mining Workshop (ICDMW), 2014 IEEE International Conference on. IEEE, 2014. DOI: 10.1109/ICDMW.2014.116

[26] Y. Lei and L. Huan, "Feature selection for high-dimensional data: A fast correlation-based filter solution". In the 20th ICML, pages 856–863, 2003. ISBN:1-57735-189-4

[27] J. Z. Kolter, M. A. Maloof, "Using additive base learner ensembles to cope with concept drift", in: Proceedings of the Twenty Second ACM International Conference on Machine Learning (ICML'05), 2005, pp. 449–456. DOI: 10.1145/1102351.1102408

[28] D. Brzezinski, J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams". Information Sciences, Vol 265, pp. 50-67, 2014. DOI: 10.1016/j.ins.2013.12.011

[29] G. Xiao-Feng, et al. "An improving online accuracy updated ensemble method in learning from evolving data streams." Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on. IEEE, 2014. DOI: 10.1109/ICCWAMTIP.2014.7073443

[30] K. Nishida, K. Yamauchi, "Adaptive classifiers-ensemble system for tracking concept drift", in: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC'07), 2007a, pp. 3607–3612. Honk Kong. DOI: 10.1109/ICMLC.2007.4370772

[31] A. Bifet and R. Gavald`a, "Learning from time-changing data with adaptive windowing". In SDM, 2007. DOI: 10.1137/1.9781611972771.42

[32] J. Gama, P. Medas, G. Castillo and P. Rodrigues, (2004). "Learning with Drift Detection," in Proc. of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04), pp. 286-295. DOI: 10.1007/978-3-540-28645-5_29

[33] M. Baena-Garcıa, J. D. Campo-Avila, R. Fidalgo, and A. Bifet, (2006). "Early Drift Detection Method," in Proc. of the 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams, pp. 77-86.

[34] A. Bifet and R. Kirkby, Tutorial 1. Introduction to MOA Massive Online Analysis http://sourceforge.net/projects/moa-datastream/files/documentation/Tutorial1.pdf (Accessed 10 Apr 17)

[35] Z. Xingquan, (2010). Stream Data Mining repository. Accessed on Jan 2012; Available from: http://www.cse.fau.edu/~xqzhu/stream.html [Accessed Feb 2017]

[36] The Apache SpamAssasin Project - http://spamassassin.apache.org/ [Accessed May 2018]

[37] M. Harries. "Splice-2 comparative evaluation: Electricity pricing". Technical report, The University of South Wales, 1999.

[38] M. Friedman, (December 1937). "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". Journal of the American Statistical Association. American Statistical Association. 32 (200): 675–701. DOI: 10.2307/2279372

[39] P. Nemenyi, (1963) "Distribution-free Multiple Comparisons". PhD thesis, Princeton University.

[40] J. Demsar, (2006). "Statistical comparisons of classifiers over multiple data sets". J. Mach. Learn. Res., ISSN 1532-4435

[41] J.P. Barddal, H. M. Gomes, F. Enembreck and B. Pfahringer, 2015c. "A survey on feature drift adaptation". In: Proceedings of 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1053–1060. DOI: 10.1016/j.jss.2016.07.005

[42] J. N. Rijn, H. M. Gomes, B. Pfahringer. and J. Vanschoren, "Having a Blast: Meta-Learning and Heterogeneous Ensembles for Data Streams". In 2015 IEEE International Conference on Data Mining, pages 1003-1008. IEEE, 2015. DOI: 10.1109/ICDM.2015.55

[43] K. Nishida, (2008). "Learning and Detecting Concept Drift", PhD thesis, Hokkaido University, Japan.

[44] J. C. Schlimmer, and R. H. Granger Jr., (1986). "Incremental learning from noisy data", 1: 317–354. DOI: 10.1007/BF00116895

[45] W. N. Street and Y. Kim. "A streaming ensemble algorithm (SEA) for large-scale classification", KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining n377-382 2001. DOI: 10.1145/502512.502568

[46] G. Boracchi, C. Cervellera, D. Macciò., "Uniform histograms for change detection in multivariate data," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 1732-1739. DOI: 10.1109/IJCNN.2017.7966060

[47] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD), 2007. DOI: 10.1145/1281192.1281264

[48] C. R. Blyth, "On the inference and decision models of statistics," Ann. Math. Statist., vol. 41, no. 3, pp. 1034–1058, 1970.

[49] E. L. Lehmann and J. P. Romano, "Testing Statistical Hypotheses". New York, NY: Springer Science+Business Media, Inc. Springer e-books,2005. ISBN: 978-0-387-98864-1

[50] G. Ditzler and R. Polikar, “Hellinger distance-based drift detection for non-stationary environments,” in Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011 IEEE Symposium on, April 2011, pp. 41–48. DOI: 10.1109/CIDUE.2011.5948491

[51] F. Stahl, M. M. Gaber, P. Aldridge, D. et. al., 2012. “Homogeneous and Heterogeneous Distributed Classification for Pocket Data Mining". In: Hameurlain, A., Küng, J. and Wagner, R., eds. Transactions on Large-Scale Data- and Knowledge-Centered Systems V (7100). Springer Berlin Heidelberg, pp. 183-205. DOI: 10.1007/978-3-642-28148-8_8

[52] S. Ghosh, D. L. Reilly, “Credit card fraud detection with a neuralnetwork", in: Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences., Vol. 3, IEEE, 1994, pp. 621-630. DOI: 10.1109/HICSS.1994.323314

[53] D. Sanchez, M. A. Vila, L. Cerda, J. Serrano, “Association rules applied to credit card fraud detection", Expert Systems with Applications 36 (2) (2009) 3630-3640. DOI: 10.1016/j.eswa.2008.02.001

[54] Intel Lab Data http://db.csail.mit.edu/labdata/labdata.html [Accessed May 2018]

[55] A. Asuncion & D. Newman, (2007). “UCI machine learning repository". http://www.ics.uci.edu/~mlearn/MLRepository.html.

[56] D. Brzezinski and J. Stephanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm,” IEEE Trans. Neural Netw. Learn. Syst., vol. 25, no. 1, pp. 81–94, Jan. 2014. DOI: 10.1109/TNNLS.2013.2251352

[57] Massive Online Analysis, datasets https://moa.cms.waikato.ac.nz/datasets/ [Assessed Jan 2019]

[58] H. M. Gomes., A. Bifet, J. Read., J. P. Barddal., et. al. “Adaptive random forests for evolving data stream classification". In Machine Learning, DOI: 10.1007/s10994-017-5642-8, Springer, 2017. DOI: 10.1007/s10994-017-5642-8

[59] A. Bifet., G. F. Morales., J. Read. G. Homes., and B. Pfahringer, “Efficient online evaluation of big data stream classifiers". In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 59–68. ACM, 2015. DOI: 10.1145/2783258.2783372

[60] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Homes, “Evaluation methods and decision theory for classification of streaming data with temporal dependence". Machine Learning, 98(3):455–482, 2015. DOI: 10.1007/s10994-014-5441-4

[61] L. L. Minku, A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation Empir Software Eng (2019). DOI: 10.1007/s10664-019-09686-w

[62] J. Gama, R. Sebastião, & P.P. Rodrigues, (2009). “Issues in evaluation of stream learning algorithms". In Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 329–338). ACM. DOI: 10.1145/1557019.1557060

[63] J. Sun., J. Lang, H. Fujita, L. Hui, “Imbalanced enterprise credit evaluation with DTE-SBD: decision tree ensemble based on SMOTE and bagging with differentiated sampling rates", Inf. Sci. 425 (2018) 76–91. DOI: 10.1016/j.ins.2017.10.017

[64] C. Zhang., B. Jingjun, X. Shixin, et. al,“Multi-Imbalance: An open-source software for multi-class imbalance learning", Knowl.-Based Syst. 174 (2019) 137–143. DOI: 10.1016/j.knosys.2019.03.001

Appendix A

SEA *(Sudden Drift)*
EvaluatePrequential -s (ConceptDriftStream -s (generators.SEAGenerator -f 4) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 2) -p 50000 -w 1) -p 25000 -w 1) -i 100000 -f 1000

SEA *(Gradual and Sudden Drift)*
EvaluatePrequential -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (ConceptDriftStream -s (generators.SEAGenerator -f 3) -d (generators.SEAGenerator -f 4) -p 50000 -w 1) -p 25000 -w 10000) -i 100000 -f 1000

HyperPlane *(Gradual Drift)*
EvaluatePrequential -s (generators.HyperplaneGenerator -k 10 -t 0.01) -i 100000 -f 1000

RandomTrees *(Recurring Drift)*
EvaluatePrequential -s (RecurrentConceptDriftStream -x 10000 -s (generators.RandomTreeGenerator -o 0) -d (generators.RandomTreeGenerator -u 0) -p 25000 -w 1) -i 100000 -f 1000

RandomRBF *(Gradual Drift)*
EvaluatePrequential -s (clustering.RandomRBFGeneratorEvents -n) -i 100000 -f 1000

LED *(Sudden Drift)*
EvaluatePrequential -s (ConceptDriftStream -s generators.LEDGenerator -d (generators.LEDGeneratorDrift -d 7) -p 50000) -i 100000 -f 1000

WaveFormDrift *(Sudden Drift)*
EvaluatePrequential -s (ConceptDriftStream -s generators.WaveformGenerator -d (generators.WaveformGeneratorDrift -d 20) -p 50000 -w 1) -i 100000 -f 1000