Novel Technique(s) for Concept Drift Detection and Handling

*A **Thesis***

submitted in partial fulfillment of the requirements for the award of degree of

Doctor of Philosophy

by

Kanu Goel

(901603007)

under the guidance of

Dr. Shalini Batra

Associate Professor

Computer Science and Engineering Department

Associate Dean (Academic Affairs)

Thapar Institute of Engineering and Technology

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala -147004, INDIA

October 2021

Contents

List of Figures

List of Tables

List of Abbreviations

ACE	Accuracy Classifier Ensemble
ADACC	Anticipative and Dynamic Adaptation to Concept Changes
ADES	Abstained Dynamic Ensemble Selection
ADWIN	Adaptive Windowing
ARF	Adaptive Random Forests
ASHT	Adaptive-Size Hoeffding Trees
AUE	Accuracy Updated Ensemble
AWE	Accuracy Weighted Ensemble
BOLE	Boosting-like Online Learning Ensemble
CD	Critical Difference
CFB	Circulating Fluidized Bed
CONDOR	Concept Drift via Model Reuse
CPF	Concept Profiling Framework
CUSUM	Cumulative Sum
CVFDT	Concept Adapting Very Fast Decision Tree
DA-DDE	Dynamically Adaptive and Diverse Dual Ensemble
DCS	Dynamic Classifier Selection
DDD	Diversity for Dealing with Drifts
DDM	Drift Detection Method
DDSVM	Dynamic Dual Selective Voting Mechanism
DESDD	Dynamic Ensemble Selection for Drift Detection
DWM	Dynamic Weighed Majority
ECDD	EWMA for Concept Drift Detection
ECPF	Enhanced Concept Profiling Framework
EDDM	Early Drift Detection Method
En-ODDD	Ensemble Based Online Diversified Drift Detection
EWMA	Exponentially Weighted Moving Average
FHDDM	Fast Hoeffding Drift Detection Method

GA	Genetic algorithm
GPS	Global Positioning System
HAT	Hoeffding Adaptive Tree
HT	Hoeffding Tree
KME	Knowledge Maximized Ensemble
KUE	Kappa Updated Ensemble
LevBag	Leveraging Bagging
MDDM	McDiarmid Drift Detection Method
MLP	Multilayer Perceptron
MOA	Massive Online Analysis
MSE	Mean Square Error
NB	Naive Bayes
NSE	Non-Stationary Environment
OAUE	Online Accuracy Updated Ensemble
PHT	Page-Hinkley Test
RBF	Random Basis Function
RDDM	Reactive Drift Detection Method
SEA	Streaming Ensemble Algorithm
SPC	Statistical process control
TBE	Trigger Based Ensemble
TLP-EnAbLe	Two-Level Pruning based Ensemble with Abstained Learners
VFDT	Very Fast Decision Tree
WMA	Weighted Majority Algorithm
WUDCDD	Weighted classification and Update algorithm of Data stream based on Concept Drift Detection

Certificate

I hereby certify that the work which is being presented in this thesis entitled **"Novel Technique(s) for Concept Drift Detection and Handling"**, in partial fulfillment of the requirement for the award of degree of **"Doctor of Philosophy"** submitted in Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala (India), is an authentic record of my own work carried out under the supervision of **Dr. Shalini Batra** and refers other research works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Kanu Goel)

Regn. No. 901603007

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Shalini Batra)

Associate Professor

Computer Science & Engineering Department

Associate Dean (Academic Affairs)

Thapar Institute of Engineering and Technology (Deemed University)

Patiala, 147004

Punjab, INDIA.

Acknowledgements

First and foremost, I would like to thank **Almighty**, for being my strength and pillar (as a humble being) to complete this task successfully. He has given me an opportunity to believe in my passion and pursue my dreams. I could never have done this without his divine blessings.

I would like to express my sincere gratitude to my supervisor, **Dr. Shalini Batra**, Associate Professor, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala (India), for being a pillar of support and encouragement throughout my research work. Her guidance and constructive comments helped me in all the time of research and writing this thesis. Her experience, strength, motivation, tenderness and willfulness, has taught me valuable lessons of life, which are going to be of immense help to me in taking decisions in going forward.

My sincere thanks are due to **Dr. Maninder Singh**, Professor and Head, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala (India), for providing me the necessary administrative assistance in completion of the work. I am thankful to my Ph.D. committee members, **Dr. Rajesh Khanna**, Professor, Department of Electronics and Communication Engineering, and **Dr. Maninder Kaur**, Associate Professor and **Dr. Shreelekha Pandey**, Assistant Professor, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala (India), for their constructive comments and regularly ensuring the progress of my research work. I am thankful to all the **faculty** and **staff members** of Computer Science and Engineering Department, Thapar Institute of Engineering and Technology (Deemed University), Patiala.

I offer my deepest gratitude to my loving mother, **Mrs. Meena Rani** and father, **Mr. Arun Kumar Goel** whose unconditional love and affectionate blessings have always motivated me to work hard and pursue my goals.

My dear husband, **Mr. Puneet Garg**, thank you for your presence, support, inspiration, love, and care throughout this and other chapters of my life.

I also acknowledge the cooperation and encouragement extended to me by my friends, especially **Vipul**, **Sanatan**, **Shefali** and **Shiwani**, and all members in my family.

Patiala

Feb, 2021

Kanu Goel

(Kanu Goel)

Abstract

In today's world, learning in the presence of dynamic environments, where continous change and development are evident, is a challenging task. Recent advances in technology have witnessed an increase in the number of real world applications which include spam filtering, fraud detection, weather forecasting, sensors, smart cities, health monitoring *etc.* Data generated from such sources in form of streams tends to evolve with the course of time. The predictive models which are trained using such data tend to become obselete with time, resulting into poor adaptabilty to the underlying drifting distributions. In terms of machine learning and data mining, the change in the statistical properties of data is known as *concept drift.* Such changes causes degradation in the performance of the learning systems since the models that were built on the old data are no longer consistent with the new data.

To address the problem of concept drift, efficient learning models which can monitor the evolving distributions and update themselves regularly are required. These models should detect the drifts and handle them timely by using adaptive learning techniques, to overcome the deteriorating performance. Various learning methods which include single learners as well as ensemble based modelling which utilize drift detectors, are used in literature to handle evolving data streams.

This thesis proposes three techniques for concept drift detection and handling. First one, a hybrid diversity based ensemble approach, called Ensemble Based Online Diversified Drift Detection (En-ODDD), combines explicit drift detection and adaptive techniques deal with drifting distributions. In second approach, Two-Level Pruning based Ensemble with Abstained Learners (TLP-EnAbLe), similarity based pruning strategy has been proposed for adapting to all types of drift patterns. The third approach, Dynamically Adaptive and Diverse Dual Ensemble (DA-DDE) utilizes the characteristics of both online and block-based ensemble techniques for concept drift handling. It proposes a dual ensemble mechanism for

separetely handling abrupt and gradual drifts. It is based on usage of novel Dynamic Dual Selective Voting Mechanism (DDSVM) for ensemble selection and hypothesis generation.

Performance of the proposed approaches has been evaluated by conducting comparative analysis with existing concept drift techniques and through the standard evaluation parameters which include classification accuracy, kappa statistic, train time, test time, memory consumption *etc.* Experiments conducted using several real datasets and artificially generated streams of data, with variety of drift patterns, indicate that all the three approaches handle the concept drift scenarios effectively giving better classification results.

Chapter 1

Introduction

The growth in the wide range of applications related to Data science and Big data have led to generation of huge amounts of digital data. Different application domains, such as health-care, network monitoring, climatology, banking transactions, social media, stock-market analysis *etc.* record data in massive amount which may contain hidden knowledge [2, 3, 4]. To capture useful information from this ever increasing data, need is to access and process this data intelligently. The rapid growth of various software and tools over the past few years have enabled data mining to be used more widely. Various data mining techniques have been proposed by researchers to extract knowledge out of such data [5, 6, 7]. However, these techniques were earlier generally applied to static datasets where a fixed data set was collected and processed, but now the data needs to be analyzed in real time.

1.1 Data streams

Data streams are unbounded sequence of instances which are generated continously from an input source. Real-time data from GPS signals, sensor readings, device logs, digital transactions, network monitoring, banking *etc.* are all application domains of data streams.

Figure 1.1: Data Stream Mining Flow

Two major characteristics of data stream, the speed at which data is arriving *i.e.* velocity and amount of data *i.e.* volume, makes it difficult to store the incoming data. Moreover, the applications that process and analyse such streams must process the data sequentially, one data packet at a time. *Data stream mining*, an approach of processing data streams, is a popular research field that studies algorithms and methods to process and extract knowledge from streaming data [8, 9].

1.1.1 Characteristics of Data Stream Mining

Following are the characteristics of data stream mining:

- Continuous stream of data: The dataset, generated from various sources, in unknown and voluminous data is produced in form of an infinite stream [10].
- Volatility of data: In data stream mining, once the data is analysed, it is either summarized or discarded since the system has limited resources and can not store the entire data received [11].
- Concept Drifting: The underlying data evolves or changes with time.

1.1.2 Offline vs Online Learning

In initial stages of data mining, data was collected and processed in offline mode. Learning models were provided complete dataset for training and the same was then applied to predict output target classes for unseen data. In contrast to this mode, online learning models process the incoming data sequentially. They train a model with the available data and

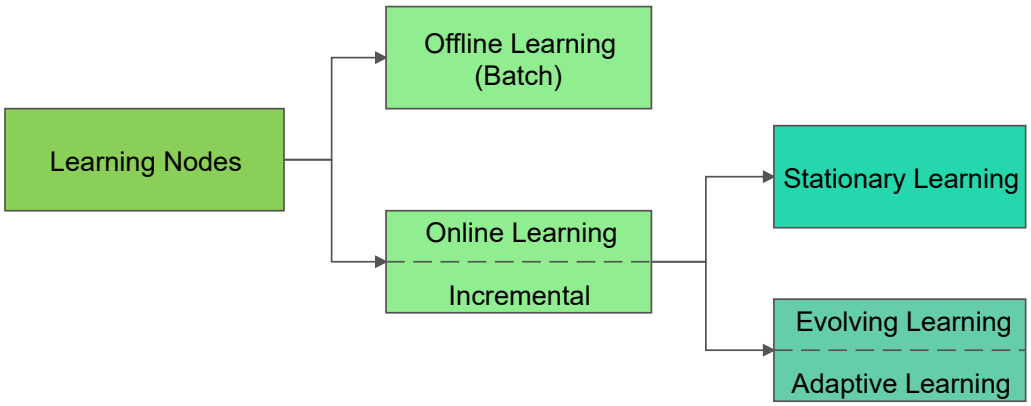


Figure 1.2: Types of Learning Modes

use it for testing purpose. When new instances arrive, the existing trained model is updated with the latest data [12].

1.1.3 Incremental Learning

Incremental learning models process instances one by one *or* in a batch mode. They update the underlying decision model after every instance. In contrast to online learning models where the instances are discarded after processing, in incremental models data may be stored for future use and may be randomly accessed when required. When the new data arrives, the update operation takes place, based on the existing one. Streaming algorithms work on online learning models which process high speed data [13]. Figure 1.2 illustrates types of learning modes.

1.1.4 Characteristics of Learning Model

Offline data mining and machine learning algorithms, where complete dataset is available at once in memory, are not suitable for handling data stream classification tasks. Online processing of data is required for data stream classification [14]. The learning model should possess following characteristics:

- One instance should be processed at a time:

  The instances in a data stream keep arriving one after another. In such scenario, it becomes crucial to process them in the sequence of their arrival. Random access to instances is not possible. Once an instance is processed, it is discarded.

- Work within the limited memory:

  Since the size of incoming data is quite large as compared to available memory, it is not possible to store such huge amount of data in limited memory. Since the training classification models work incrementally, memory bounds must be defined initially.

- Work within limited time:

  Learning models should be able to process the incoming instances as they arrive. Since the data is not stored in online processing, there is a need to decide beforehand the amount of time that will be allocated for analysing instances.

- Model must be ready to classify at any given point of time:

  Since the instances are available dynamically, learning models must follow anytime prediction strategy to predict unseen data.

1.2 Concept Drift

Advances in data stream mining techniques for processing streaming data has led to huge demand for online learning in the field of predictive analytics. One of the major challenges for stream analytics in real world applications is that, the domains where machine learning models are deployed, often evolves with time [15]. Thus, the models built to analyze such data become obselete over time. Such scenarios require continuous learning methods which can adapt to the changing concepts in the incoming streams of data. Due to the dynamic

nature of data, underlying distributions tend to drift with time, called *concept drift*. In a continuously changing environment, the phenomenon of concept drift exists in majority of data streams. The statistical properties of the incoming data, which the learning model tries to process, can change in unpredictable ways over the time. In other words, the properties of the target variable we want to predict, changes with time.

Such unpredictable changes are generally visible in the incoming learning instances and lead to deterioration in the predictive accuracy of algorithms that are trained from past instances [16]. The reason for this is the shift that happens at the data level. The data distribution of the source data which is used to train the machine learning model, is different from the distribution of target data.

Several factors which include evolving user preferences, population change, adverse activities may lead to concept drift [14] *e.g.* in a problem of analysing and filtering emails of a user account for anti-spam content, the features, namely the set of particular words used to characterize a spam email can change over time. This situation can be considered as concept drift and an anti-spam filter is required to detect these changes and handle them to adapt to new patterns of spam. A learning model which can continuously monitor the incoming emails must implement detection and handling of such type of drifts. There are particular situations in the real world where the impending change can also be known apriori with respect to the occurrence of particular environmental events. Incremental learning algorithms process the input instances one-by-one and update the model statistics with these instances. Such algorithms need to continuously monitor the incoming data ditributions to detect possible changes and adapt themselves whenever a drift occurs [14].

Adaptive learning techniques use incremental data processing and employ drift detection strategies to detect drifts and adapt themselves to the changing concepts. Classification is an integral machine learning and data mining task and requires predicting target classes

and labels in many streaming applications [14, 17, 18]. After training in labelled instances, classification algorithms predict the class of new unlabelled instances. Due to concept drift, the predictive performance of such classification algorithms tend to fall. To enhance the classification accuracy, various concept drift handling techniques such as sliding windows, drift detectors and adaptive ensembles have been proposed by the researchers [19].

1.2.1 Concept Drift: Definition

In machine learning, the classification model is built with the objective of predicting the target class label y of the incoming data instance X. An instance in the learning model (M) is described as a pair of (X,y). At every time step t, the model analyses labeled training instances of $X = (x_1,x_2,x_3,....x_t)$ while an incoming instance x_{t+1} is treated as the testing instance. In case of training, the instances of form (X,y) are used to build model, where X and y are known. When this trained classification model is applied for prediction task, class y of input instance X is determined.

According to the Bayesian Decision Theory [20], a classification model is described by the prior probabilities of the classes $p(y)$ and class conditional probability density functions $p(X|y)$ for classes $y \in 1,2,3....m$ where m is the number of classes predefined for classification task. The classification is done according to posterior probabilities of classes, representing y as:

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)} \tag{1.1}$$

In the above equation $p(X) = \sum_{y=1}^{m} p(y)p(X|y)$.

Concept drift refers to the scenario in which statistical properties of target concept or underlying conceptual data distribution changes over time. Concept drift is registered whenever there is a any change in joint distribution between the set of input variables X and target

class variable y at time step t_0 and t_1 [14].

$$\exists X : P_{t0}(X,y) \neq P_{t1}(X,y) \tag{1.2}$$

In the above equation, P_{t0} and P_{t1} denote joint distribution at time t_0 and t_1. Using the above definition of concept drift, changes in data can be due to changes in following components of the given relation [14]:

- the prior probabilities of classes $p(y)$ may change,
- the class conditional probabilities represented as $p(X|y)$ may change, and
- as a result, the posterior probabilities of classes $p(y|X)$ may change, affecting the prediction.

Given the above possibilities of changes, following two types of concept drift can be described: real and virtual drift.

- Real drift

  Real drift refers to changes in $p(y|X)$. These changes can occur with or without change in $p(X)$. It is also called as concept shift and conditional change. Real drifts directly affect the decision boundary leading to decrease in predictive performance of the learners.

- Virtual drift

  Virtual drift is defined as change in $p(X)$ or $p(y)$ that do not cause any affect of $p(y|X)$ [14]

Since our work is focussed on studying the effect of concept drift on classification domain, we shall be dealing with real concept drifts. From the perspective of prediction, adaptation

Table 1.1: Characteristics of concept drift

Type of characteristic	Examples
Drift Source	Changes in preferences, adverse activities, population change
Concept Drift Type	Incremental, gradual, sudden, reoccurring
Drift Expectation	Predictable, unpredictable
Drift Visibility	Statistical hypothesis, visual inspection, direct, indirect

is required once a real concept drift occurs since the current decision boundary becomes outdated for newly arriving data in such scenario.

1.2.2 Characteristics of Concept Drift

Some of the general characteristics of concept drift are presented in Table 1.1.

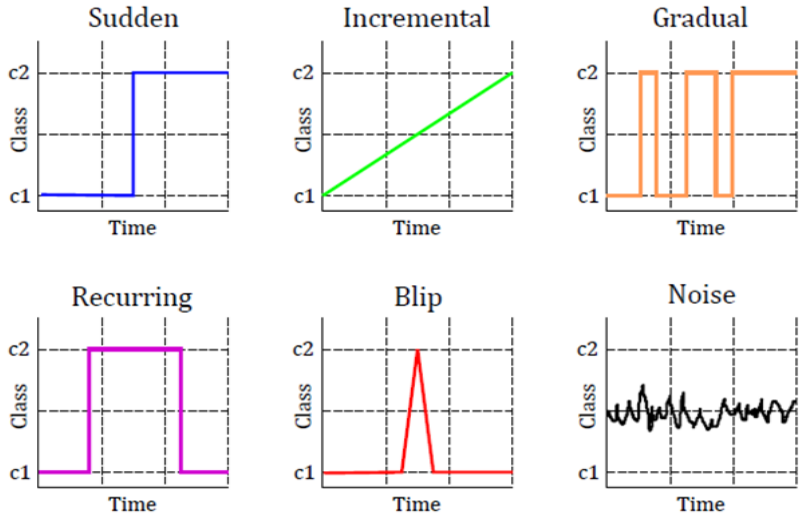


Figure 1.3: Patterns of concept drift over the time [1]

1.2.3 Concept Drift Patterns

The changes in the data distribution over the course of time could manifest in variety of forms. Figure 1.3 depicts six basic types of changes that may occur.

- *Sudden or abrupt drifts*

  These are the irreversible and instant changes in the variables where a concept sud-

denly switches from one to another for *e.g.* a major catastrophe occurs and that particular topic rapidly trends all over the news and social media. Another example is replacing a sensor with a new one in a chemical plant which has a completely different calibration [21, 22].

- *Incremental drifts*

  Incremental drift takes place when the variables slowly change their values over time and there are many intermediate concepts in between. An example of incremental drift is growing price of market products due to inflation. In another example of sensors, a working sensor may produce less accurate reading due to wearing off [23, 24].

- *Gradual drifts*

  These drifts are often determined by the slow change in the target values. A gradual transition is visible from one concept to another. However, during transition phase, we might experience instances from both the concepts. Slowly changing definitions of the user-interesting or spam news feeds are examples of gradual changes [25, 26].

- *Recurrent drifts*

  The drifts may introduce new concepts which were not seen in earlier times but were witnessed previously or may reoccur after some time intervals, such drifts are called recurrent drifts [27]. Such drifts are also regarded as local drift. An example of reoccurring drifts can be a reader who is currently interested in mystery books. For some duration of time he might switch to reading adventure books and then after a couple of months he again starts reading mystery books. This drift can be attributed to the change in user preferences [28, 29].

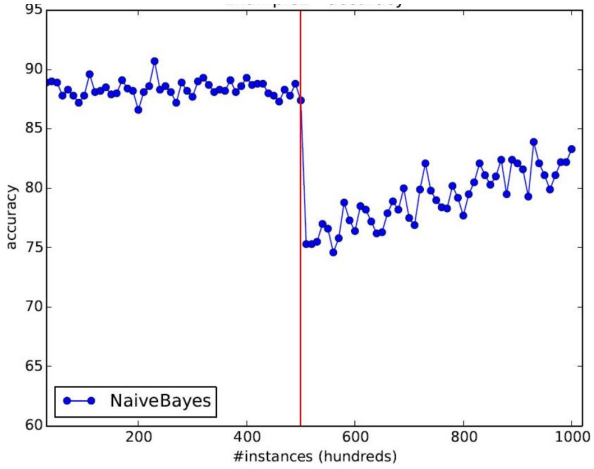
- *Blip*

  This represents a *'rare event'*, that can be regarded as an outlier in static distribution.

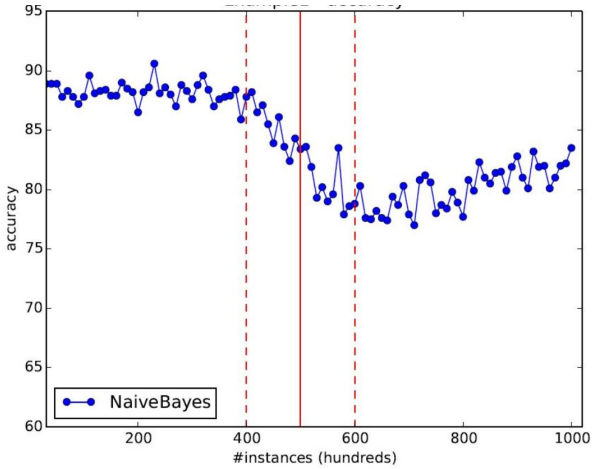
In streaming data, the blips detected can be ignored as the change they represent has random nature. Examples of blips include anomalies in network intrusion, fraudulent card transactions and landfill gas emission [30].

- *Noise*

  Noise should not be considered as drift in concept because it is an insignificant fluctuation that is not connected with any change in source distribution.



(a)



(b)

Figure 1.4: Effect on accuracy of a) abrupt drifts b) gradual drifts

Figure 1.4(a) and Figure 1.4(b) illustrate the drop in accuracy of the learner (Naive Bayes in this example) due to abrupt and gradual drift respectively in case any drift

adaptation technique is not implemented. The graph is between classification accuracy and the number of instances processed. Former case is an example of an abrupt drift at 50,000 instances and latter is the case with one gradual drift at 50,000 instances with a drift window of 20,000 instances.

1.3 Applications of Concept Drift

Concept drift is a well-studied topic in various branches of the machine learning and data mining community. It has been studied in both supervised and unsupervised settings as it is an important component in many applications [31, 32, 33, 34]. Along with the characteristics of each domain we present real life problems and discuss the sources of drift in the context of these problems.

1. Monitoring systems: Monitoring systems are characterized by large data streams that need to be analyzed in real time. The typical task of a monitoring system is to distinguish unwanted situations from *'normal behaviour'*. This includes recognizing adversary actions and issuing alerts before the critical system state arises.

    - *Network security:* The detection of unwanted computer access, also called intrusion detection, is one of the typical monitoring issues. Intrusion detection systems filter incoming network traffic in search of suspicious behaviour [35, 36, 37].

    - *Telecommunications:* Intrusion and fraud detectors are also an important part of telecommunication systems. The goal is to prevent fraud and stop adversaries from accessing private data. The objective is to track the adversary behaviour as well as change in the behaviour of legitimate users [38, 39].

- *Finance:* Streams of financial transactions can be monitored to signal for possible credit card and Internet banking frauds. Stock markets use data stream mining techniques to prevent insider trading. In both cases the data labeling may be imprecise due to unnoticed frauds and misinterpreted legitimate transactions [40]. Fraud detection directly relates to evolving customer habits leading to concept drift. Handling drifts in credit-card frauds has received a lot of attention from the researchers [41, 42, 43].

2. Personal assistance

   Personal assistance systems do not react in real time, but are usually affected by more than one source of drift. This can include individual assistance for customer profiling, personal use, text classification, sentiment classification and recommendation systems [44].

   - *Spam filtering:* It is a complicated information filtering which opens to adversary actions such as spamming. Adversaries are adaptive and change spam content rapidly to overcome filters. The amount and types of illegitimate mail are subject to seasonality and drift irregularly over time [31]. Various factors like change in user preferences where user may alter their attitude towards some of the particular categories of emails to classify them as spam. Spammers tend to show an adaptive behaviour to overcome classification of emails to spams [45, 46].

   - *Sentiment classification:* This is a popular application of concept drift where the vocabulary used to predict the sentiment of an expression may evolve with time. With the change in particular set of negative and positive sentiment words, concept drift may arise [47, 48].

- *News feeds:* Most individual assistance applications that are related to textual data aim at categorizing articles and classifying news feeds. Concept drift techniques can help in analyzing the cause of drifting user interests [31, 49].

3. Decision Support: Decision support with concept drift handling includes diagnostics and evaluation of credit worthiness. The true answer whether a decision is correct is usually delayed in these systems. The cost of mistakes in these systems is large, thus the main challenge is high accuracy [50, 51].

    - *Biomedical and Healthcare applications:* This domain presents an interesting field of concept drift research due to the adaptive nature of microorganisms. As microorganisms mutate, their resistance to antibiotics changes [52]. Patients treated with antibiotics when it is not necessary, can become 'immune'to their action when really needed. Effects of classification drift caused by hidden context such as new light sources, image background, and rotation are a major area of research [53, 54].

4. Environmental applications: This set of applications include broad variety of dynamic and stationary systems specifically those interact with changing environments [55].

    - *Drivable Terrain colour Prediction:* This is an application of classification task where the Stanley team [56] proposed an adaptive approach to classify the terrain as drivable or not. Changing lighting conditions led to abrupt changes in surface colours, *e.g.* shifting from a paved to unpaved road. In such scenario, gradual adaptation depending upon the road conditions was required for handling drifts.

The above mentioned domains of applications highlight the need of adaptive learning systems which can detect and handle concept drifts [57]. Application oriented research has

been carried out by Delany *et al.* [58] where they presented a case based technique for tracking the concept drift in spam filtering. They explored the benefit of periodically redoing the feature selection process for bringing new features into play.

Beyene *et al.* [59] proposed an algorithm called Trigger based Ensemble (TBE) for handling concept drift in surgery prediction scenario. They gave an ensemble based approach to cater to the drifts in the referrals due to clinical practices and scientific developments.

Pechenizkiy *et al.* [60] considered the problem of learning an accurate predictor to estimate the mass flow with the explicit detection of drifts that were abrupt along with the noise handling mechanisms. They emphasized the importance of having domain knowledge concerning considered case. Scaling sensors were located under the container which provided real time streaming data. They demonstrated the performance of concept drift handling methods by showing their effect on the accuracy of the online mass flow prediction with real datasets collected from the CFB boiler laboratory. They also highlighted the need of adaptive systems for monitoring raw sensor signals to handle the changes to feeding and burning components. Apart from these, several other real world prediction applications focus on the importance of handling concept drifts [61, 62, 63, 64].

1.4 Thesis Organization

The thesis has been organized in the following chapters:

Chapter 1: Introduction

This chapter provides an overview of Data Streams, characteristics of data stream mining and various learning modes. It also introduces Concept drift along with its characteristics and explains its patterns. Further, different application domains which specifically witness drifting scenarios are discussed in this chapter.

Chapter 2: Literature Review

This chapter provides a comprehensive review of concept drift detection and handling techniques. Various online and block based approaches, in single learner and ensemble based modes have been reviewed. Factors that play a significant role in the performance of ensemble learning are also discussed. This chapter concludes with the motivation, problem formulation and objectives of the thesis.

Chapter 3: Ensemble Based Online Diversified Drift Detection (En-ODDD)

In this chapter, the proposed hybrid diversity based ensemble approach has been discussed in detail. Characteristics of both explicit drift detection and adaptive techniques are combined to overcome the drifting distributions. Experiments conducted to examine the effectiveness and reliability of En-ODDD on artificial and real datasets have been described. Besides this, empirical study has been conducted to compare ten existing online and block based algorithms by inducing majority of drift patterns including gradual, abrupt, recurring. En-ODDD has been experimentally evaluated on various parameters like classification accuracy, train time, test time, memory utilization, kappa statistic.

Chapter 4: Two-Level Pruning based Ensemble with Abstained Learners (TLP-EnAbLe)

This chapter discusses similarity based pruning approach for concept drift handling. The performance of TLP-EnAbLe has been compared with various state-of-the-art concept drift handling techniques and results achieved have been discussed. Statistical tests have also been conducted on the compared techniques to validate the performance under drifting scenarios.

Chapter 5: Dynamically Adaptive and Diverse Dual Ensemble (DA-DDE)

In this approach drift handling is achieved by combining utilizing characteristics of both online and block-based ensemble techniques. A novel Dynamic Dual Selective Voting Mechanism (DDSVM) for hypothesis generation is proposed which dynamically selects the best

ensemble based on the correctness of previous chunk in real time. This chapter discusses the effectiveness of the proposed approach in improving the ensemble adaptability to both gradual and recurrent drifts. In this chapter apart from the artificially simulated drifting streams and real datasets, two data stream applications namely Electricity pricing prediction and Spam filtering are considered to demonstrate the effectiveness of the proposed approach.

Chapter 6: Conclusion and Future Scope

Thesis concludes with this chapter by highlighting the contributions made through the proposed research work. It also provides an insight into the future directions for working in this area.

Chapter 2

Literature Review

This chapter provides an extensive literature survey on various concept drift detection and handling techniques that have been proposed by the researchers. It also discusses, in detail, the factors that impact the ensemble based learning while handling drifting scenarios.

2.1 Concept Drift: Detection and Handling techniques

Several concept drift detection and handling techniques have been proposed which can be roughly classified into three categories: drift detector based techniques, single learner based and ensemble based.

2.1.1 Drift Detectors

The drift detector based approaches have single learners which have a specific mechanism to monitor the incoming data and detect the onset of concept drift. The detection techniques can be categorized into three types: (i) statistical process control; (ii) sequential analysis techniques; (iii) monitoring two different time windows.

Statistical Process Control (SPC)

These techniques primarily use the statistical methods for monitoring as well as controlling a continuous process. The methods that adapt from SPC consider learning as a process and thereby monitor the evolution of the process with time.

Some of the SPC algorithms are listed below.

- Drift Detection Method (DDM): DDM, proposed by Gama *et al.* [65], is among the most popular drift detector for abrupt drifts where an online classifier is employed to predict the output class of the current learning instance. True value or false value, is associated with each prediction which indicates whether the instance is correctly classified or not. A miss-classification error rate is calculated using Binomial distribution. In DDM, p_i denotes the probability of a false prediction and s_i is the standard deviation, calculated as Eq. 2.1.

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}} \tag{2.1}$$

Two values p_{min} and s_{min} are calculated which are used to generate warning and error signals when predefined threshold is exceeds.

$$p_i + s_i \geq p_{min} + \alpha . s_{min} \tag{2.2}$$

$$p_i + s_i \geq p_{min} + \beta . s_{min} \tag{2.3}$$

When the warning level is signalled, instances thereafter are stored in a separate window. In case the alarm level is reached, previously trained classifier is replaced by a newly created one. This new classifier is trained from the instances of warning window. In the equations Eq. 2.2 and Eq. 2.3, α and β denote the confidence levels for

triggering warning and alarm signals respectively.

- Early Drift Detection Method (EDDM) [66]: Baena-Garca *et al.* proposed EDDM [66] which keeps track of the distance between two consecutive errors and raises alarm when predefined threshold values exceed. If $\hat{p}_i$ denote the average distance between two consecutive mis-classifications and $\hat{s}_i$ is standard deviation, then Eq. 2.4 and Eq. 2.5 denote the warning and alarm conditions respectively. p_{max} and s_{max} are the maximum values of p_i and s_i. EDDM particularly works well for gradual drifts.

$$\frac{\hat{p}_i + 2.\hat{s}_i}{\hat{p_{max}} + 2.\hat{s_{max}}} \leq \alpha \tag{2.4}$$

$$\frac{\hat{p}_i + 3.\hat{s}_i}{\hat{p_{max}} + 3.\hat{s_{max}}} \leq \beta \tag{2.5}$$

- EWMA for Concept Drift Detection (ECDD) [67]: ECDD works by calculating estimates of probability of wrongly classifying an instance with two measures. This drift detector is based upon the popular forgetting strategy [68]. In one case more weight is given to recent instances and in second one equal emphasis is given to old and recent data. This detector signals drift when a particular threshold value exceeds.

- Reactive Drift Detection Method (RDDM) [69]: RDDM is proposed as an improvement to DDM by periodic recalculation of DDM warning and drift statistics. It mainly addresses the problem of loss in prediction performance of DDM when, in case of large concepts, the detector becomes less sensitive to drifts. Apart from the two statistics p_i and s_i, and warning and drift constants α and β, it introduces three new variables: the maximum size of the current concept *max*, reduced size of a stable concept *min* and *warnLimit* to store maximum number of instances in warning window are introduced in RDDM. In contrast to DDM, it discards older instances if the total number of

instances is too large to evaluate. Eq. 2.6 is used to signal warning level.

$$p_i + s_i \geq p_{min} + \alpha * s_{min} \tag{2.6}$$

The drift is signalled if any of the three conditions specified by Eq. 2.7, Eq. 2.8 or Eq. 2.9 is satisfied.

$$p_i + s_i \geq p_{min} + \beta * s_{min} \tag{2.7}$$

$$num_{inst} \geq max \tag{2.8}$$

$$num_{warn} \geq warnLimit \tag{2.9}$$

Sequential Analysis Techniques

In such techniques, the drift is diagnosed if the computed statistics exceeds a pre-defined threshold. The accuracy of such detection methods is often deteriorated by the false alarm rate and the mis-detection rate. Some of the frequently discussed drift detectors in this category are:

- CUSUM test [70]: The cumulative sum approach (CUSUM) works by detecting a change based upon the value of a drift parameter of a probability distribution. It indicates when the change is significant. The expected value of the parameter is used while considering the classification error, estimated on the basis of the labels of incoming instances of the data stream.

- Page-Hinkley Method [70]: This detector is a variant of CUSUM, popularly used in change detection for signal processing [71]. A test variable used in this technique is the cumulative difference between observed values and their mean value until the latest moment. Observed value is taken as classifier's error rate for the drift detection.

When drift occurs, the classifier may incorrectly classify the incoming instances and accuracy may decrease and the mean accuracy till current moment also decreases. The cumulative difference (U_t) and the minimum difference (m_t) are computed for both values. When the difference between both these computed values exceeds a specified threshold λ, drift is signaled. Higher values of threshold λ may lead to lesser false alarms, but might miss or delay some drifts.

Monitoring Two Different Time Windows

These methods use a fixed reference window which summarizes the past information and a sliding detection window over the most recent examples. Here the distributions over two windows are compared with the implementation of statistical tests where the null hypothesis states that the distributions are equal or not. In case of rejection of the null hypothesis, a drift is declared at the start of the detection window.

- Sliding Windows: The concept of sliding window is used by many concept drifting approaches [72, 73, 74]. This strategy controls the number of instances given to the classifier for training while removing those which are part of older concepts. The latest instance is added to the current window and then the classifier is trained with the given window. Models detect the concept drifts by using the forgetting mechanisms [73, 75]. Very Fast Decision Tree (VFDT) [9] is an induction algorithm which modifies the decision tree without storing instances once they have been used to train the model. Further CVFDT [76] was proposed which had fixed-sized windows to locate aged nodes. However, these approaches depend largely on the selection of optimum window size to give good accuracy. If a smaller window is considered for building the classifier, the detector will react quickly to changes. But, in case of stable periods it may lead to decrease in accuracy. Whereas, considering a larger window will not help in adaptation

to quickly changing concepts. So, deciding optimum window size is crucial in such algorithms. Algorithm 1 shows the working of a windowing approach.

Algorithm 1 Windowing Algorithm

Input: $\hat{S}$: stream of instances, $\hat{W}$: window of instances
Output: C: classifier trained on instances of $\hat{W}$

1: Initialize $\hat{W}$
2: **for** each instance x_t ε $\hat{S}$ **do**
3: Add x_t to the window $\hat{W}$
4: Delete outdated instances from $\hat{W}$ as required
5: Update C by training with instances of $\hat{W}$
6: **end for**

- Weighted Windows: Weighted windows are used to make the forgetting process of older instances dynamic. Here a weight, which signifies the importance, is assigned to each instance of the incoming stream. Older ones get lesser weight and are considered as less important. Finally, while generating the hypothesis for output prediction, aggregated weighted results are calculated based upon different decay functions [77].

- Adaptive Windowing (ADWIN) [73]: This drift detection method uses concept of sliding windows of different sizes. The size of window W increases and decreases depending on occurrence or non occurrence of change. Two sub-windows, W_1 and W_2, with distinct averages μ_1 and μ_2 are located. If averages differ, it signifies they correspond to different expected values. In such case older instances and latest instances of two sub-windows correspond to different concepts resulting in dropping of older window. Such methods help in calculating precise change point. One of their major drawbacks is memory requirement as they need to store the incoming data in two windows.

Apart from the drift detector techniques discussed above, several other detectors have been proposed in the literature [78]. In an approach called Weighted Classification and Update Algorithm (WUDCDD) [79], a dual detection mechanism based on Mahalanobis

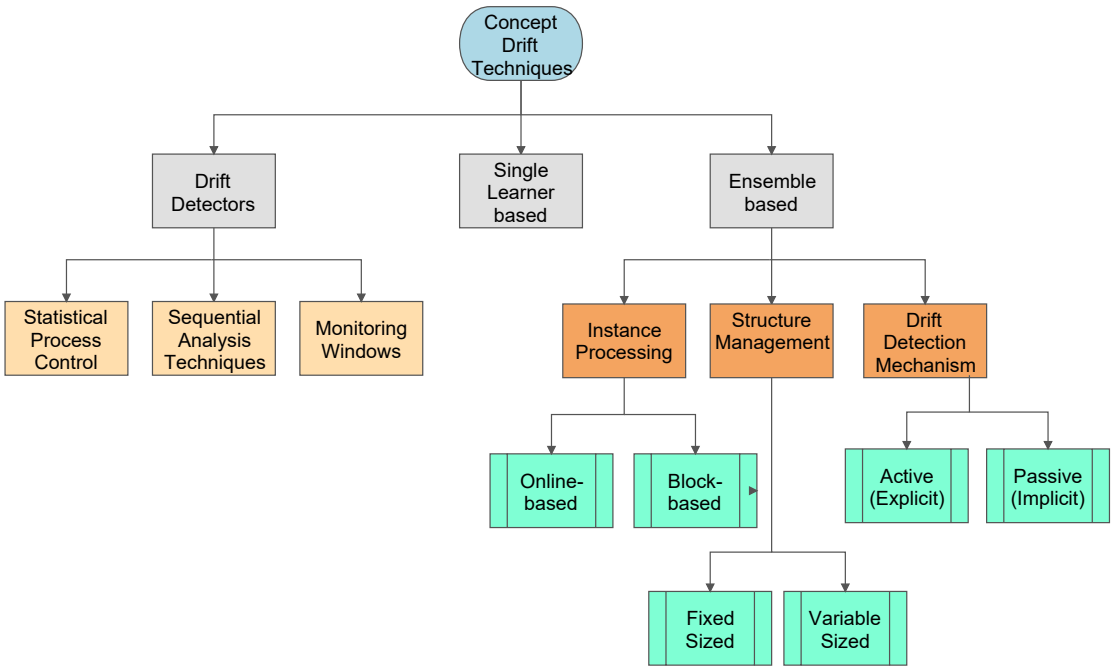


Figure 2.1: Taxonomy of Concept Drift Techniques

distance and a test static is explored. It periodically detects the changes in streams using index of classification error. Further techniques like MDDM [80], FHDDM [81] are also proposed to detect drifts. A detailed comparison of the performance of all drift detectors is provided by Barros and Santos [82]. Figure 2.1 depicts the overview of taxonomy of concept drift techniques.

2.1.2 Single Learner based Concept Drift Handling

Single learners based techniques are known from static learning which can be adapted to cope with the evolving data streams as well. They are hybrid of the online learners and forgetting mechanisms. Forgetting methods help in eliminating the data examples coming from the old concept distributions, thereby keeping the models updated. Some of the popular single learners which can be modified to react to changing concepts are:

- Naive Bayes

Naive Bayes is a bayesian prediction based learner which makes naive assumption that all input variables are independent. Based on Bayes' theorem, it computes the class-conditional probabilities for every new instance. It is a simple learner having low computational cost associated with learning. For a given number of classes, n_c, a trained Naive Bayes learner predicts a class C, to which an unlabelled instance belongs with highest probabilty. To deal with concept drifting problems forgetting mechanism is added to the learner and is used with sliding window to remove the older instances. A single Naive Bayes learner often forms a component of decision trees used for data streams [83].

- Decision Trees : Hoeffding Tree

  Decision Trees were one of the first static learning algorithms which were adapted using a Hoeffding bound for data streams. A Hoeffding tree (HT) also called as Very Fast Decision Tree (VFDT) [9] is an incremental algorithm also called anytime decision tree which is specifically designed for stream setting. The learning process of VFDT is based upon a splitting attribute for which a small sample is enough to chose its optimal value. It depends upon a mathematical bound value, Hoeffding bound, which quantifies the number of instances (observations) required to estimate the goodness of an attribute. It states that with a probabilty 1 - δ, the true mean of a random variable with range R shall not vary from the estimated mean after n independent observations (no. of instances) by more than ε. Eq. 2.10 specifies this value.

$$\varepsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{2.10}$$

  Using Hoeffding bound it can be shown that its output is asymptotically nearly identical to the output given by a non-incremental learner which uses infinite instances.

Originally VFDT had no forgetting mechanism as it was designed for stationary data streams. An improved version of VFDT known as Concept-adapting Very Fast Decision Tree (CVFDT) was proposed by Hulten *et al.* [76], which dealt with the problem of changing data streams. Outdated instances were forgotten and tree node statistics were updated in the sub-trees rather retraining the whole learner. Hoeffding trees are being used by most of the data stream algorithms as base learners for dealing with concept drifting environments [84, 85, 86].

- Neural Networks

  Neural networks are based on learning via neurons which are connected with each other and a weight is associated with each relationship. An activation function defines the output generated by every neuron. Apart from usage in static setting, these networks are also used in stream based input processing. Since every instance is seen just once, neuron weights are updated in nearly constant time, thereby fulfilling suitability for stream handling. The underlying networks change according to the new instances which help in reacting to concept drifts [87, 88].

2.1.3 Ensemble based Concept Drift Handling

Ensembles provide a way to adapt to the changes by modifying components (experts) or their method of aggregation. Modularity in ensembles provide adaptability to change by either changing their structure, retraining ensemble components or by updating the rules for decision making. Decisions of the single classifiers are combined to generate the final prediction usually by voting approach. This combined output is more accurate than produced by the single experts [89]. Many ensemble-based algorithms have been discussed in literature which are used to handle concept drifts [90, 91, 92, 93, 94]. Compared to the single

learner based techniques they provide better adaptability to drifting streams as they capture the dynamic concept under non-stationary conditions.

The ensemble based approaches for concept drift handling can work in two modes with respect to the method of processing the incoming data: *online mode* and *block-based mode*.

Online Ensembles

These ensembles techniques learn incrementally while processing instances one by one. Here each instance is separately processed upon arrival. Applications which deal with large amounts of incoming data and having stringent memory and time constraints, are processed in online mode [95, 96].

- Online Bagging

  Online Bagging Algorithm also known as OzaBag was introduced by Oza and Russel [97] as an online version of popular bagging technique. In OzaBag it is assumed that a new instance can be replicated zero, one or more times in the updating process while training the base learner. Each learner in the ensemble is updated with k copies of the new incoming instance. In this algorithm, value of k is obtained from the Poisson distribution with $k \sim Poisson(1)$. The underlying learners provide their decisions for output class which are combined using majority voting scheme.

- Leveraging Bagging and Adaptive-Size Hoeffding Trees

  With the aim of generating more randomization in the input sample space, two modifications namely Leveraging Bagging (LevBag) [98] and Adaptive-Size Hoeffding Trees (ASHT) [99] were introduced by Bifet *et al.*. In LevBag the sampling from Poisson distribution has been modified to Poisson (λ) from Poisson (1) and ASHT can synchronously grow trees of variable sizes. In the former algorithm λ is a user defined

value which can lead to diverse set of sample space.

- Online Boosting

  This is another online ensemble based technique for drift handling with incremental learning. An ensemble with fixed set of learners is maintained which are updated with each incoming instance. Initially weight $\lambda = 1$ is assigned to each instance. The first learner is updated with this learner k = Poisson(λ) times. Based upon the prediction by the learners, the instances which are misclassified are given more weight as compared to those which are correctly classified. Other boosting variant techniques like Online Coordinate Boosting [100], AdaBoost [101] were also proposed to generalize the weighting procedure.

Block based Ensembles

In block based ensembles the processing of the instances is done in blocks of data, also called chunk based methods. Fixed or variable block size is defined for a particular approach which may require multiple iterations over the training instances in each block [102, 103, 104].

- Streaming Ensemble Algorithm

  Streaming Ensemble Algorithm (SEA) [105] is one the earliest chunk based approaches for drift handling. In this algorithm, a new learner is created with each incoming batch/chunk of instances and added to the ensemble until maximum ensemble size is reached. The new learner replaces one of the existing learners with lowest quality score. The quality scores are assigned considering accuracy and diversity of learners. It uses simple majority voting for making final predictions for ensemble. As compared to single classifiers, SEA attains better recoverability from concept drifts. However, in

some cases where older learners performed better than the new ones, SEA algorithm may result in outdated concepts.

- Accuracy Weighted Ensemble

  Accuracy Weighted Ensemble (AWE) [84] is another algorithm having similar ensemble restructuring as SEA. It is based upon the principle of assigning weights to underlying learners which depend upon their predictive performance on the recent chunk of instances. A novel weighing funtion based upon mean square error is calculated for every learner. To maintain ensemble size, weight based pruning is employed and in every iteration a newly trained learner on latest chunk is added. The newly added learner is evaluated using k-cross validation which incurs huge computational cost. However, AWE works well for recurring concepts and is best suited for larger streams.

- Accuracy Updated Ensemble

  Accuracy Updated Ensemble (AUE) [106] is a chunk based algorithm which supports incremental update of its learners with each incoming chunk. Novel weighing functions along with accuracy based pruning help to achieve better computational cost than its peers. AUE2 [85] was proposed to emphasize handling of both abrupt and gradual drifts. This algorithm applies a separate weighing function for newly added and existing learners. Weights are recalculated while a new chunk of instances is processed which leads to continuous adaptation to the changing concepts. High accuracy is obtained while handling gradual drifts as re-weighting helps in evolving over changes.

- Knowledge Maximized Ensemble

  Knowledge Maximized Ensemble (KME) [107] is a data stream classifier based approach which leverages supervised and unsupervised knowledge of underlying concepts in blocks to detect concept drift. It evaluates weights of ensemble members

using a piecewise exponential function particularly well performing on sudden drifts. It also proposes a concept drift detection system to monitor the prediction capability of learners and react to conditional changes. Pairwise comparisons are conducted especially to deal with recurrent concepts.

Online based ensembles vs Block based ensemble techniques

Online concept drift approaches react to abrupt changes quickly whereas block-based approaches are efficient in dealing with gradual and reoccurring drifts since they are equipped with continuous adaptive and update techniques. Online learners are less stable than block-based approaches as the latter require much more data for building their models and updating. Online methods which have drift detector embedded in them, when used in isolation, do not give good results as they do not remember the historic concept knowledge. This makes them less favourable for gradual drifts. But when combined with adaptive block-based ensembles, they have a good scope of generating better results in dealing with multiple types of drifts [108]. Block based ensembles are in handling concept drifts as they enclose various learners trained on different learners over a long period of time. However, there are some limitations associated with these techniques. It is difficult to tune the block size as sometimes it leads to a compromise between quick reaction to incoming drifts and high accuracy. If the block size is too large, it may by pass some of the abrupt drifts as they are usually of short duration. On the other hand, keeping a smaller block size may result in less stable period of learning and also lead to increase in computational cost.

Further, the ensembles techniques may or may not use explicit drift detecting mechanism based upon which they are categorized into two types: *Active* or *Passive* techniques.

Active Ensembles (Explicit Drift Detection)

These techniques are equipped with explicit drift detection method and observe the stream to search for changes and warn the learners to take specific action on the onset of drift. They are useful when existence of concept drift needs to be notified. They ensure swift reaction to drifts but may produce false alarms as well [109, 110].

- Accuracy Classifier Ensemble

  Accuracy Classifier Ensemble (ACE) is a hybrid online approach proposed by Nishida [111]. Algorithms like SEA and AWE are dependent on chunk size. Smaller chunk sized learning deteriorates performance of underlying learners whereas larger chunks lead to poor response to abrupt drifts. ACE overcomes these limitations as it uses a combination of batch learners, drift detector and an online learner. With each incoming example the online learner is trained. The drift detector checks the accuracy of the blocks and signals a change when the best performing component falls outside the confidence level. Upon detection of a change, a new classifier is added and the online classifier is reset. ACE does not limit the number of component classifiers in the system. It also reacts well to recurring concepts by utilizing already existing learners. Due to presence of explicit drift detector, it shows better response to abrupt drifts.

- Diversity for Dealing with Drifts

  Diversity for Dealing with Drifts (DDD) [112] is an example of active online ensemble based learning approach. It works on the principle that ensembles high on diversity (whose components produce very different predictions from each other) are likely to have poor predictive performance under the stationary conditions, but might be useful when used in concept drifts. It maintains a set of low and high diversity ensemble, both before and after the drift is detected. Once the drift is detected new ensembles of low

and high diversity are created. This approach achieves better predictive accuracy under various types of concept drift as compared to other ensemble techniques. However, due to use of multiple ensembles high computational cost is incurred.

Passive Ensembles (Implicit Drift Detection)

Passive drift detection based ensembles have implicit mode of drift detection which consider that drift may occur constantly or occasionally and continuously update themselves with the data. They focus more on the adaptive strategy to handle concept drift. Passive approaches have different techniques of assigning weights to component classifiers of ensemble. While reacting to concept drifts, specific algorithms are implemented for adding and removing classifiers from the underlying ensemble to maintain appropriate size.

- Dynamic Weighed Majority

  Dynamic Weighed Majority (DWM), an extension of Weighted Majority Algorithm (WMA) is one of the popular online ensemble algorithms. DWM considers the dynamic nature of streams and alters the ensembles members based on the global performance of the entire ensemble and the performance of individual members. It consists of a set of incremental classifiers whose weights are updated according to their accuracy after each incoming example. If any component fails to accurately predict an incoming stream, its weight is decreased by a user specified parameter β ($0 \leq \beta \leq 1$). The system is evaluated at regular interval p and a new classifier is added, if required [113]. Selecting optimum value of p is important as large value may result into slower adaptation to concept drift.

- Online Accuracy Updated Ensemble

  Online Accuracy Updated Ensemble (OAUE) [108], inherits few positive solutions

which come from its predecessor Accuracy Updated Ensemble (AUE) [106] where periodical incremental updates of components using blocks of examples is a key factor in achieving high accuracy. OAUE updates its component classifiers along with learning new classifiers at constant time steps. A cost effective approach that achieves a good trade-off between memory usage, predictive accuracy and processing time is proposed.

Other concept drift approaches:

Roberto *et al*. [114] proposed an approach which increased the accuracy of online boosting methods for handling abrupt and frequent drifts. They investigated the effect of changing the drift detection method by substituting ADWIN for DDM. Raquel *et al*. [115] proposed a windowing scheme which compares the data distributions using the concept of fading histograms to detect the drifts in the data.

In some of the ensemble approaches like BLAST [116], the weight of the low performing learners is lowered temporarily rather than replacing them. In an approach named Concept drift via model reuse (CONDOR) [93], concept of model reuse is discussed where weights of learners are adaptively adjusted as per their performance. Overview of popular data streaming techniques for detecting and handling concept drift is given in Table 2.1. Compilation of the various concept drift techniques reported has been presented in Table 2.2 and Table 2.3.

2.2 Role of Diversity in Ensemble Learning

Though diversity measures have been studied extensively for static conditions, few algorithms have employed diversity based pruning for ensemble selection. In concept drift domain, few algorithms have been proposed which incorporate advantages of diversity. Diversity for Dealing with Drifts (DDD) [112] maintains two ensembles of varying diversity

Table 2.1: Overview of popular data streaming techniques for detecting and handling concept drift

Category	Acronym	Algorithm's name	Features	Shortcomings/Challenges
Drift detector based	DDM [65]	Drift Detection Method	Error-rate based trigger drift detection methods; signal various drift levels.	Usually handle only a particular type of drift; lack continuous updating of learners.
	EDDM [66]	Early Drift Detection Method		
	ECDD [67]	EWMA for Concept Drift Detection		
	RDDM [69]	Reactive Drift Detection Method		
Windowing based	ADWIN [73]	Adapative Windowing	Compare statistical properties of standard variables between consecutive window of instances and signal drifts on crossing threshold values.	Trade-off between window size and predictive accuracy of the model
	VFDT [9]	Very fast decision tree		
Ensemble based (Block-based)	AWE [84]	Accuracy Weighted Ensemble	Constitute trained base learners; Associate weight with learners; Updation of ensemble members; continuous updating; can be block-based or online.	Complex weight updating of underlying learners; obsolete learners sometimes; Select optimum set of updated learners for decision making
	AUE2 [85]	Accuracy Updated Ensemble 2		
	KME [92]	Knowledge-Maximized Ensemble		
	ECPF [117]	Enhanced Concept Profiling Framework		
Ensemble based (Online)	ARF [91]	Adaptive random forests	Process the stream instance by instance; pruning based updating techniques.	Single-pass high computation cost as they work on instance-level; work under time and memory constraints
	WMA [118]	Weighted Majority Algorithm		
	DWM [113]	Dynamic Weighted Majority		
	HDWM [119]	Heterogeneous Dynamic Weighted Majority		
	OzaBag [97]	Online Bagging		
	LevBag [98]	Leveraging Bagging		
	ACE [111]	Accuracy Classifier Ensemble		
Diversity based	DDD [112]	Diversity for Dealing with Drifts	Use diverse ensemble members to provide generalization of concepts.	Determining diversity generation strategies.
	DDWM [120]	Diversified dynamic weighted majority		
	DESDD [121]	Dynamic Ensemble Selection for Drift Detection		
	DRED [122]			

Table 2.2: Overview of popular Concept Drift techniques

Approach	Abbrevation	Drift detection		Drift Type Focussed				Instance Processing		Classifiers		Diversity	Reference
		Implicit	Explicit	Abrupt	Gradual	Recurr	Mixed	Online-based	Chunk-based	Single	Ensemble		
CUSUM	Cumulative Sum		√	√	√			√		√		None	E. S. Page [1954] [70]
PHT	Page-Hinkley Test		√	√	√			√		√		None	E. S. Page [1954] [70]
WMA	Weighted Majority Algorithm	√		√	√			√			√		Avrim Blum [1997] [118]
AWE	Accuracy Weighted Ensemble	√		√	√				√		√	None	Wang et al. [2003][84]
DDM	Drift Detection Method		√	√	√			√		√		None	Gama et al. [2004] [65]
ACE	Accuracy Classifier Ensemble		√	√	√				√		√	None	Nishida et al. [2005] [111]
OzaBag	Online Bagging	√		√	√			√			√	Instance level	Nikunj C Oza [2005] [97]
EDDM	Early Drift Detection Method		√	√	√			√		√		None	Garcıa et al. [2006] [66]
ADWIN	Adapative Windowing		√	√	√			√		√		None	Bifet and Gavalda[2007] [73]
DWM	Dynamic Weighted Majority	√		√	√			√			√	None	Kolter and Maloof [2007] [113]
AUE1	Accuracy Updated Ensemble 1	√		√	√				√		√	None	Brzezinski and Stefanowski [2011] [106]
DDD	Diversity for Dealing with Drifts		√	√	√			√			√	Instance level	Minku and Yao [2012] [112]

Table 2.3: Overview of popular Concept Drift techniques (cont.)

Approach	Abbrevation	Drift detection		Drift Type Focussed				Instance Processing		Classifiers		Diversity	Reference
		Implicit	Explicit	Abrupt	Gradual	Recurr	Mixed	Online-based	Chunk-based	Single	Ensemble		
ECDD	EWMA for Concept Drift Detection		√	√				√		√		None	Ross et al. [2012] [67]
ADACC	Anticipative and dynamic adaptation to concept changes	√		√	√	√		√			√	None	Jaber et al. [2013] [123]
AUE2	Accuracy Updated Ensemble 2	√		√	√	√	√		√		√	None	Brzezinski and Stefanowski [2014] [85]
OAUE	Online Accuracy Updated Ensemble		√	√	√	√	√		√		√	None	Brzezinski and Stefanowski [2014] [108]
BLAST	BLAST heterogeneous ensembles	√		√	√			√			√	None	Rijn et al. [2015] [116]
BOLE	Boosting-like Online Learning Ensemble	√		√	√			√			√	None	Barros et al. [2015] [114]
RDDM	Reactive Drift Detection Method		√	√	√			√		√		None	Barros et al. [2017] [69]
ARF	Adaptive random forests		√	√	√			√		√		Instance level	Gomes et al. [2017] [124]
AES	Adaptive Ensemble Size	√		√	√			√			√	None	Olorunnimbe et al. [2017] [125]
KME	Knowledge-Maximized Ensemble		√	√	√				√		√	None	Ren et al. [2017] [92]
DESDD	Dynamic Ensemble Selection for Drift Detection		√	√	√			√			√	Instance level	Albuquerque et al.[2019] [126]
ECPF	Enhanced Concept Profiling Framework		√	√		√		√			√	None	Anderson et al [2019] [117]
KUE	Kappa Updated Ensemble	√		√	√	√		√			√	Instance level	Cano and Krawczyk [2020] [94]

levels but it does not preserve the older models in the system. Algorithms like Adaptive Random Forests (ARF) [127], Dynamic Ensemble Selection for Drift Detection (DESDD) [126], Kappa Updated Ensemble [94] generate diverse learner ensembles by employing modified versions of Online Bagging [97]. ARF simulates randomization in streaming environment where each underlying tree grows on different subset of instances.

Majority of the above mentioned techniques employ diversity at instance level not at learner level. There is a huge scope of improvement in prediction performance of incremental approaches like Accuracy Weighted Ensemble (AWE) [84], Accuracy Updated Ensemble (AUE2) [85] *etc.* which consider only weight based pruning. None of the above mentioned approaches employ diversity although it has a huge impact in ensemble learning.

Similarity between learners *i.e.* ones which classify given instances similarly is one of the important components of diversity. Techniques like Concept Profiling Framework (CPF) [128] and Enhanced Concept Profiling Framework (ECPF) [129] use conceptual equivalence to find and reuse similar learners. In ECPF, one of the similar learners with better accuracy is retained while the other is deleted. However, this may lead to loss of a high performing learner which may serve as a good predictor in consecutive chunks. To avoid such scenarios, appropriate strategies must be adopted to prevent removal of desired learners [130, 131].

2.3 Selection of Ensemble Components

Many approaches have been proposed in the past to choose the competent learners which play important role in decision making [114, 132, 133]. Another important aspect of ensembles learning is Dynamic Classifier Selection (DCS) where the use of meta-learning for selecting underlying classifiers is emphasized. Local region based methods choose most promising learners which can classify the instances in the local region of the test instance

correctly [134]. Some approaches consider majority voting of all the learners instead of employing any selection technique; incurring huge computation [84, 106]. An approach based on abstaining the use of uncertain learners was introduced by Krawczyk and Cano [135] based upon a certain threshold where the less confident learners were forced to abstain from contributing in final decision making. However, such removal based upon local performance may result into non-relaible selection of learners in global aspect. Thus, it becomes important to analyse the learners carefully before any abstaining strategy is adopted.

2.4 Role of Ensemble Size in Adaptability

Ensemble size plays a great role while ensuring adaptability to drifting concepts. Ensemble based techniques can be *fixed* size or *variable* size.

- Fixed sized ensembles

  These techniques maintain a constant size of the ensemble by apriori limiting the number of constituent learners. To manage the ensemble size, the base learners are periodically evaluated for their predictive performance. In the common accuracy based pruning strategy the poorly performing learner is removed and replaced by a newly trained one. Most of the concept drift handling approaches use fixed sized ensembles since it helps in maintaining ensemble complexity [84, 85, 113].

- Variable sized ensembles

  Variable sized ensemble based techniques have recently emerged in the field of ensemble learning. Ensemble size automatically adapts based upon the updating process. The underlying learners are not removed from the ensemble pool, rather some of them are dynamically selected to participate in decision making. To select the learners,

weight based strategy is implemented for reducing the misclassification rate. While handling recurring drifts, the preserved knowledge of existing learners plays an important role [136].

2.5 Research Gaps

Based on the literature review conducted, following research gaps were identified:

- *Majority of existing ensemble pruning is based upon accuracy:* In most of the algorithms that have been proposed, pruning of the component classifiers in the ensemble is based on the accuracy of the classifier. Systems need to be designed in a way that they leverage other pruning strategies like diversity-based, age-based *etc.* Using the pruning strategies in combination rather than using them in isolation needs to be explored for improving performance of ensembles.

- *Techniques required which consider diversity of ensembles:* While preparing models for analyzing drift detection, updating the ensemble with the latest data stream is the usual practice. This methodology increases the accuracy of the ensembles but training them on similar examples reduces their diversity. Literature review indicates that ensemble diversity is not explored much while handling streams.

- *Combination of drift detectors and adaptive methods is required:* Various approaches have been proposed for drift detection like block based drift detectors, window based, using an incremental learner, *etc.* So far, in most of the existing techniques, these methods have been used in isolation. However, methods which are amalgam of explicit drift detectors and adaptive techniques need to be explored.

- *Techniques required which can handle multiple types of drifts in single stream:* Drift

handling methods which can adapt to various types of drifts such as gradual, sudden or recurrent along the different data streams'lengths under combined scenarios need to be considered. Most of the current research concentrates on single or separate drifts. Combinations of drifts such as gradual and sudden are worth analyzing.

2.6 Problem Formulation

The motivation to study time-changing high-speed data streams comes from the emergence of temporal applications such as signal processing, sensor networks, time series analysis, fraud detection, automatic control, user modeling, real time-monitoring in industrial and biomedicine processes, safety of complex systems and many others.

Due to dynamic nature of the data, some of the properties of a problem may change over the time, *i.e.* the target concept on which data is obtained might shift from time to time, after some minimum permanence. Learning algorithms which model time-changing underlying processes should track dynamic behavior and adapt the decision model accordingly. As old observations (which reflect the behavior of nature in past) become irrelevant to the current state, the prediction models need to be retrained after the occurrence of a drift so that it can accurately predict underlying data. However, considering dynamic processes, it is a big challenge for researchers to propose appropriate algorithms for concept drift detection and handling.

After conducting an exhaustive survey on concept drift techniques, it was realized that existing learning methods usually focus on handling a particular type of drift pattern, *e.g.*, only abrupt or only gradual change. Learning models are required which can handle multiple drift forms at the same time and in a single algorithms are required.

While generating models for handling drifting streams, updating the ensembles while

incorporating diversity of ensembles need to be explored. Drift detectors and adaptive techniques are treated as different aspects/domains and hence used in isolation. Their combinations can be helpful in dealing with different drift types in a single algorithm. Further, adaptive systems need to be designed in a way that they leverage the combination of accuracy and diversity based pruning for ensemble selection. Research is required to analyze the evolving nature of processes that demand huge amounts of data processing within defined memory and time constraints. The intent of this thesis is to propose adaptive learning systems that can detect and handle concept drift in evolving data distributions.

2.7 Research Objectives

Following are the major objectives of our research work:

- To study existing drift detection and handling techniques and identify shortcomings of the existing solutions to concept drift and adaptive systems.
- To propose and implement novel technique(s) for drift detection and handling concept drift.
- To test and validate the proposed technique(s) using various existing concept drift detection and handling parameters.

Chapter 3

Ensemble Based Online Diversified Drift Detection (En-ODDD) for Concept Drift

This chapter discusses ensemble based techniques for concept drift detection and handling. Proposed technique introduces an explicit trigger based drift detection mechanism in the dynamically updated block based ensemble framework. It obtains high prediction performance in varied data stream settings.

3.1 Introduction

While preparing learning models for analyzing concept drift, updating ensemble with latest data stream is the usual practice. During stable period (when there is no concept drift) of ensemble, the diversity between components is marginal. Therefore, the ensemble members trained on similar data are almost the same after long periods of stability. It has been observed that diverse ensembles adapt relatively better to new concepts.

Incremental systems like Online Bagging [97] or Leverage Bagging [98] provide diversity and react well to abrupt drifts but due to lack of periodic weight-updating mechanism

their performance degrades in analyzing gradual drifts. In adaptive concept drift handling approaches, high computational cost is incurred due to constant updating of ensemble members even in the absence of drift. On the other hand, drift detector based techniques are sometimes not effective as they lack perfectly updated component classifiers. They might face catastrophic forgetting especially in case of recurrent changes.

Hence it is imperative to combine characteristic features of both groups such that a better adaptation to all kinds of drifts is achieved. Systems need to be designed to accommodate diversity of ensembles such that they adapt well to majority of drift patterns such as gradual, abrupt, recurring, mixed, *etc*. To enhance the true detection capability, algorithms should consider diverse effects of statistical changes corresponding to specific drift patterns. In changing environment, the ensembles need to be generalized and this can be achieved by replacement of weaker member with new classifier trained on the most recent example or by using aggregation techniques (*eg.* updating weights in the voting formula). Further, ensemble based techniques should retain the simplified schema of component classifiers while weighting and updating themselves when new instances are added.

Considering these motivations, a novel concept drift detection and handling technique, Ensemble Based Online Diversified Drift Detection (En-ODDD) has been proposed. En-ODDD is a hybrid diversity based approach which is in line with the characteristic features of both the explicit detector based and adaptive techniques.

3.2 Ensemble Based Online Diversified Drift Detection (En-ODDD)

In the proposed technique, ensemble experts are built using the most recent data chunk and are timely pruned to deal with deteriorating performance of overall ensemble and cope

Table 3.1: Summary of the main notations in En-ODDD

Symbol	Description	Symbol	Description
$\hat{D}_t$	Data distribution	$P_t(X, y_i)$	Joint probability at time t
$\hat{C}_n$	Current data chunk	$\lvert\hat{C}_{max}\rvert$	Maximum chunk limit
$\hat{X}_e$	Base model expert	λ	Poisson distribution value
ε_{cut}	Drift threshold	$\lvert\hat{E}\rvert$	Current ensemble size
L	Maximum ensemble size	ω_b	Weighted instance
MSE_r	Mean square error of randomly predicting expert	δ	Split confidence of learner
t_i	Tie-threshold	$n(min)$	Grace period value
θ	Minimum fraction weight	β	Penalizing factor
F_f	Friedman value	$\bar{R}$	Average ranks
${\chi_F}^2$	Friedman statistic	N	Number of datasets
$p(y)$	Probability of class y	k	Number of algorithms for comparison
$p_y^k(x)$	Probability of instance x in class y	$\hat{H}$	Final hypothesis function
λ_l	Low Diversity level	λ_h	High Diversity level
CD	Critical difference	σ_w^2	Variance of W elements
m	Harmonic mean of elements of both sub-windows	δ_c	confidence level

up with drifting data. The usual incremental training is augmented by deployment of online bagging approach at the time of creation as well as updating of ensemble experts, which introduces diversity and randomization to the input instances. An ensemble of experts which use various subsamples of training data achieve high accuracy and generalization in this approach. This ensures effective learning of underlying models even during stable period. Figure 3.1 provides the block diagram of the proposed technique. Summary of the main notations used in En-ODDD are given in Table 5.1. In the following section, details of the proposed algorithm have been discussed along with its characteristic features.

3.2.1 Ensemble Based Online Diversified Drift Detection (En-ODDD): Algorithm

The algorithm En-ODDD incorporates the online drift detection in a diversified adaptive setting while pruning the non-performing experts at fixed intervals. Further, different levels

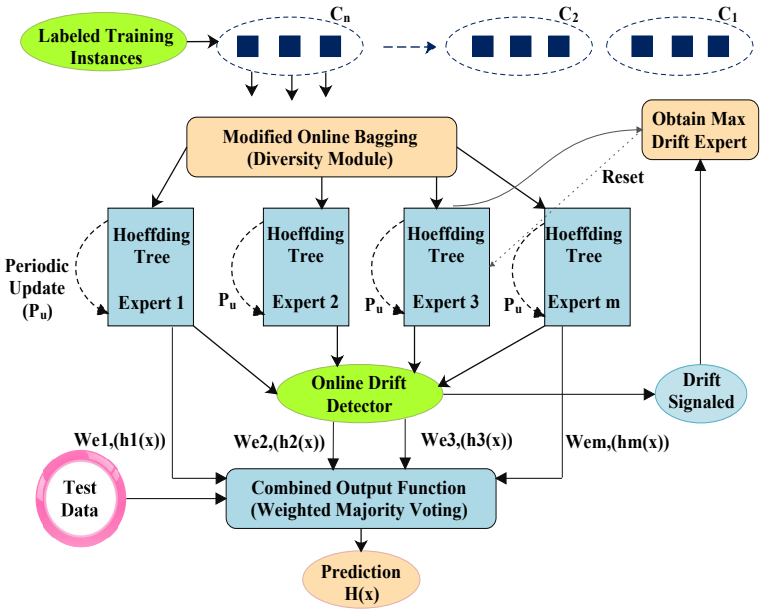


Figure 3.1: Block Diagram for En-ODDD.

of randomization have been applied to produce best prediction results in the above setting.

En-ODDD uses ensemble based model where Hoeffding Tree is used as base expert for building the ensemble, Figure 3.1. Initially, each incoming instance is added to $\hat{C}_n$, the current chunk, until the maximum chunk limit $|\hat{C}_{max}|$ is attained (Algorithm 1: lines 1-3). Online Bagging, which manipulates the input instance, induces diversity in the base experts. The base expert $\hat{X}_e$ is updated $\hat{k}$ times (obtained from $Poisson(\lambda)$ distribution) with the current instance (lines 5-8). As stated by Oza and Russell [97] when the number of instances used for training tend to infinity, binomial distribution of $\hat{k}$ value tends to $Poisson(\lambda)$ distribution for $\lambda = 1$. Each expert has a separate drift detector which constantly monitors the drift error rate produced during classification of the current instance (Algorithm 2). The detector uses $DriftErrWin$, a sliding window of variable length, for storing the prediction of the current expert.

$$\varepsilon_{cut} = \sqrt{\frac{2}{m}.\sigma_w^2.\ln(\frac{2n}{\delta_c})} + \frac{2}{3m}\ln(\frac{2n}{\delta_c}) \tag{3.1}$$

Whenever the difference of distinctive average values between two sub-windows is greater

than the threshold ε_{cut} obtained using Eq. (3.1) [73], a change in class distribution is signaled (line 12). To accommodate this changing scenario, the least performing expert (one having the maximum drift error) is identified, reset and its drift error value is reinitialized to stabilize the ensemble with current distribution. Once the predefined chunk limit (500 in the proposed scheme) is obtained, the weights of existing experts are updated with the current chunk using Eq.(3.3) where the probabilities of all input classes are considered [85]. $p_y^k(x)$ denotes the probability that an expert $\hat{X}_k$ classifies x as the instance of class y. Weight of experts ($w_{nonLinear}$) depends upon mean square error of their misclassification and that of a randomly predicting expert (Eq.(5.2)), calculated on the current chunk $\hat{C}_n$. In En-ODDD, a new expert $\hat{X_{en}}$ is added and one of the existing experts, with minimum weight, is pruned to maintain the consistency of ensemble size (lines 21-25).

$$MSE_r = \sum_y p(y)(1-p(y))^2 \tag{3.2}$$

$$w_{nonLinear} = \frac{1}{\frac{1}{|C_n|}\sum\limits_{\{x,y\}\varepsilon C_n}(1-p_y^k(x))^2 + MSE_r + \varepsilon} \tag{3.3}$$

$$w_{newExpert} = \frac{1}{MSE_r + \varepsilon} \tag{3.4}$$

Weight assigned to $\hat{X_{en}}$, given by Eq.(3.4), depends only on current data distribution. A small value ε is added to Eq.(3.4) for avoiding divide by zero error. After replacement of expert, all the existing experts are updated using Online Bagging and $\hat{C}_n$ is reinitialized (lines 26-30). In due course of time, after a stable duration, most of the ensemble experts become identical since they are trained on similar data instances. Here, diversity has been introduced by employing bagging to provide higher generalization accuracy among the experts.

Algorithm 1 Ensemble based Online Diversified Drift Detection (En-ODDD)

$\langle x_t^i, y_t^i \rangle$: i^{th} training instance at time t with the feature vector $\langle x^i \rangle$ and class label y^i from Dataset D
$\hat{C}_n$: current chunk of instances
$\hat{\text{E}}$: ensemble of experts $\hat{X}_e$
$DriftErrWin$: drift error window corresponding to the ensemble experts
DetectDrift: method to detect drift
$|\hat{C}_{max}|$: max size of chunk
$|\hat{E}|$: current size of ensemble
L: maximum ensemble size where L $\in$ N

for every instance b^i $\langle x_t^i, y_t^i \rangle$ in D **do**
  $driftSignal \leftarrow$ false
  add b^i to $\hat{C}_n$
  for all experts $\hat{X}_e$ in ensemble $\hat{\text{E}}$ **do**
    $\hat{k} = Poisson(\lambda)$
    if $\hat{k} > 0$ **then**
      weightedInstance ω_b = weight(b^i) * $\hat{k}$
      update the expert $\hat{X}_e$ with ω_b
    end if
    $driftSignal \leftarrow DetectDrift(b^i, DriftErrWin, \hat{X}_e)$
  end for
  if $driftSignal$ == true **then**
    $\hat{i_d} = LocateMaxDriftErrorExpert(\hat{E})$
    reset the learning expert $\hat{X}_e[\hat{i_d}]$
    reset the $DriftErrWin$ for the expert $\hat{X}_e[\hat{i_d}]$
  end if
  if b^i mod $|\hat{C}_{max}|$ == 0 **then**
    for each $\hat{X}_e$ in $\hat{\text{E}}$ **do**
      $\hat{w_e} = W_{nonLinear}$ using Eq.(3.3)
    end for
    construct the new expert $\hat{X_{en}}$ on $\hat{C}_n$ and calculate its weight using Eq.(3.4)
    if $|\hat{E}| \geq$ L **then**
      remove poorest expert with min $\hat{w_e}$ from $\hat{\text{E}}$
    end if
    add $\hat{X_{en}}$ to the $\hat{\text{E}}$
    for each $\hat{X}_e$ in $\hat{\text{E}}$ **do**
      TrainWithBagging($\hat{X}_e, \hat{C}_n$)
    end for
  end if
  Reinitialize the $\hat{C}_n$
end for
Output final hypothesis $\hat{H}$**:** $\mathbf{X} \rightarrow \mathbf{Y}$
Predict with $\hat{h}_k(x_t^i) : \delta[h_k(x_t^i) = y_t^i]$
return $\hat{H}(x_t^i)$: arg max$_{y_t^i} \sum_{k=1}^{L} \hat{w}_k \hat{h}_k(x_t^i)$

Algorithm 2 DetectDrift(b, DriftErrWin ,$\hat{X}_e$))

```
1: Initialize Width, Variance and Total
2: bit_var = correctlyClassify(b, X̂_e)        /* obtain the prediction bit for instance b by X̂_e */
3: insertElement(bit_var, DriftErrWin)        /* insert prediction bit into drift error window */
4: for every split of DriftErrWin into DriftErrWin_0.DriftErrWin_1 do
5:    μ̂_DriftErrWin_0 ← Average of prediction in DriftErrWin_0
6:    μ̂_DriftErrWin_1 ← Average of prediction in DriftErrWin_1
7:    if |μ̂_DriftErrWin_0 − μ̂_DriftErrWin_1| > ε_cut then
8:       driftSignal ← true
9:       return driftSignal
10:   end if
11: end for
```

3.3 Experimental Evaluation

In this section the datasets, methodology and experiments to perform comparison of different concept drift handling techniques are discussed.

3.3.1 Datasets

All the datasets, artificial and real, used to analyze the proposed approach have been described briefly in Table 5.3. Twelve artificially simulated datasets have different variations of drifts are used in experimentation. These datasets are created using generators available in MOA framework [137, 138]. Three real datasets: *Covertype*, *Poker* and *Weather*, commonly used in the concept drift domain, have been considered for experimentation and evaluation purpose.

Poker is generated by varying the combination of suits and ranks of the five playing cards drawn from a standard deck consisting of 52 cards [139]. It has ten predictive attributes (5 cards $\times$ 2 attributes-rank and suit) along with an attribute known as poker hand. This value is inferred after identification of value of five cards in game. 25,000 instances are produced

from this dataset.

Table 3.2: Datasets and their characteristics

Datasets	#Instances	#Attributes	#Classes	Type of drift	Generator	Drift description
Agr_{mix}	1M	9	10	mixed	Agrawal [137]	three drifts each after 250k instances
Mix_{incr}	1M	4	2	incremental	Mixed [137]	one drift after 500k instances
Mix_{grdl}	1M	4	2	gradual	Mixed	one drift after 500k instances
Mix_{abr}	1M	4	2	abrupt	Mixed	one drift after 500k instances
RBF_{grdl_rec}	1M	20	4	gradual recurring	RBF [85]	four drifts each after 125k instances
RBF_{mix}	1M	20	4	mixed	RBF	four alternating sudden and gradual drifts each after 125k instances
$Sine_{grdl}$	1M	2	2	gradual	Sine [66]	one gradual drift with concepts switched at angle of drift 45° after 500k instances
$Tree_{grdl\_rec}$	1M	5	4	gradual recurring	Random Tree [9]	four drifts each after 200k instances
$Tree_{abr\_rec}$	1M	5	4	abrupt recurring	Random Tree	four drifts each after 200k instances
$Wave_{grdl}$	400k	40	3	gradual	Waveform [137]	three drifts each after 100k instances
$Wave_{abr}$	400k	40	3	abrupt	Waveform	three drifts each after 100k instances
$Wave_{mix}$	1M	40	3	mixed	Waveform	three alternating sudden and gradual drifts each after 100k instances
Poker	25k	10	10	unknown	real	unknown
Covertype	581k	13	7	unknown	real	unknown
Weather	18k	8	2	unknown	real	unknown

Covertype dataset is based on cover type information of forest obtained from US Forest Service region Resource Information System data. 53 cartographic variables define the examples of this dataset. Instances may belong to one of the seven cover types based on the cartographic variables. 581k instances and 54 attributes represent this dataset [92].

Weather is a classification dataset which has an extensive range of weather patterns collected from US National Oceanic and Atmospheric Administration of 9,000 station from all over the the world [140]. Features like temperature, wind speed, *etc.* help to predict long

term weather conditions where different variations lead to drifts in the data. Diverse weather patterns make it a popular prediction/classification problem.

3.3.2 Experimental Setup

Experiments were performed using MOA framework [137], a tool extensively used in data stream domain to analyze the streaming approaches. Machine equipped with Intel(R) Core(TM) and i7-6700 CPU @3.41 GHz Processor having 8 GB of RAM has been used. An initial study was conducted, which indicates that using large number of classifiers do not increase the accuracy, instead, increase the time requirements. Taking this point into consideration, the number of learners considered in ensemble based approaches are 10, with Hoeffding tree as base learner of ensembles with $\delta = 0.01$, n(min) = 100 and $t_i = 0.05$. Chunk size of $|d| = 500$ is used for all datasets since this size value is considered as the minimal suitable size for the block-based ensembles. The ensemble experts are trained in parallel using separate individual threads which reduced training time considerably. For a meaningful comparison between different algorithms, same parameter values have been set as stated above.

3.3.3 Evaluation using Different Diversity Levels

To leverage the bagging performance, En-ODDD was tested by introducing different levels of diversity. As λ is the parameter that largely influences the diversity, its impact on predictive accuracy was verified by tuning it on training data. Table 3.3 presents the average accuracies obtained by performing 8 preliminary executions using $\lambda = 0.01, 0.1, 0.5, 1, 1.5, 2, 2.5, 3$ on each dataset. Additionally, the plot in Figure 3.2 depicts that prediction accuracy in most of the datasets considered increase until $\lambda = 1.5$. However, the performance of En-ODDD tends to converge for $\lambda > 1.5$. Thus, for analyzing the performance of proposed

Table 3.3: Average classification accuracy (%) of En-ODDD using various values of λ.

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 1.5$	$\lambda = 2$	$\lambda = 2.5$	$\lambda = 3$
Agr_{mix}	80.61	86.34	90.84	91.56	92.61	92.82	92.78	92.46
Mix_{incr}	91.75	95.86	98.40	98.97	99.12	98.90	99.33	99.31
Mix_{grdl}	91.43	95.74	98.43	98.96	99.10	98.90	99.31	99.33
Mix_{abrpt}	91.96	95.88	98.59	98.91	99.14	98.95	99.30	99.33
RBF_{grdl_rec}	76.56	90.02	94.64	96.18	96.68	96.40	96.94	96.92
RBF_{mix}	69.85	85.04	91.66	94.05	94.94	94.55	95.62	95.60
$Sine_{grdl}$	87.14	95.83	98.52	99.08	99.19	99.10	99.21	99.27
$Tree_{grdl\_rec}$	49.74	70.29	81.18	85.80	88.50	87.45	90.92	91.51
$Tree_{abr\_rec}$	60.68	77.51	86.14	89.33	91.06	90.04	92.74	93.07
$Wave_{grdl}$	77.46	79.65	82.50	83.84	84.02	83.67	83.42	82.98
$Wave_{abr}$	78.45	80.91	83.25	84.42	84.46	84.28	83.89	83.63
$Wave_{mix}$	80.82	80.96	83.34	84.65	84.77	84.52	84.32	83.90
Poker	48.18	48.51	50.76	54.96	56.62	51.91	58.81	59.04
Covertype	75.77	80.64	84.02	84.96	85.30	84.80	85.51	85.51
Weather	64.36	70.43	73.57	75.24	75.23	74.68	76.44	76.60

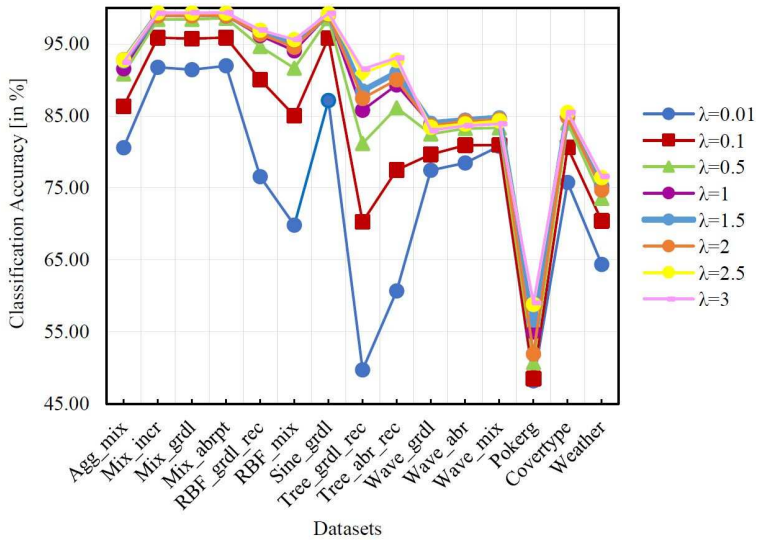


Figure 3.2: Average classification accuracy of En-ODDD using different values of λ.

approach, value of $\lambda = 1.5$ has been considered in all experiments.

3.3.4 Evaluation using Interleaved Test-then-Train Method

In this methodology, every instance is used first for evaluating the existing classifier before using it for updation process. However, the classifier is always tested on the unseen instances [16]. Plots between the number of processed instances and classification accuracy

are drawn to examine the effect of the underlying classifiers to concept drift. Accuracy has been evaluated as percentage of instances classified correctly over the total number of instances. Table 3.4, 3.5 present results of the average accuracy and training time of the algorithms evaluated respectively.

3.3.5 Parametric Configuration

En-ODDD has been compared with both online and block based algorithms by analyzing various performance metrics as given in Table 3.6. To implement the ensemble based approaches, the default values of the parameters were used as per their configuration. In DWM, factor to penalize experts (β) is set to 0.5, minimum fraction weight (θ) is set as 0.01 and value for period between removal of expert (p) is set to 50. In AWE and AUE2, the number of classifiers to learn is set as 10 with a block size of 500. Various values of block sizes which include 250, 750 and 1000 were also tested for the same but 500 provided best average accuracy. The DDD algorithm is chosen for experiment with $\lambda_l = 1$ and $\lambda_h = 0.1$, as it is the important approach considering diversity of ensembles. LearnNSE, Online Bagging and Leverage Bagging are considered as they are efficient representatives of the ensemble based online approaches.

3.4 Results and Discussion

The performance of different algorithms under varying drift patterns for the datasets condidered for evaluation is presented.

Experiments with Wave Generator: Figure 3.3(a) shows the accuracy achieved on the $Wave_{abr}$ dataset. The best performing algorithms here are En-ODDD, DDD and Oza, closely followed by AUE2 and LevBag. ARF, WMA and LearnNSE have relatively shown loss in

Table 3.4: Average classification accuracy in percentage [%]

Dataset	DDD	Oza	AUE2	ACE	DWM	WMA	NSE	LevBag	ARF	En-ODDD
Agr_{mix}	89.53	89.43	88.11	90.36	87.1	88.64	83.5	91.18	87.4	92.63
Mix_{incr}	98.64	98.01	98.38	85.10	97.76	94.93	91.6	98.37	98.18	99.11
Mix_{grdl}	98.63	98.02	98.38	84.31	97.65	94.97	91.62	98.25	98.19	99.13
Mix_{abr}	98.62	98.03	98.38	85.22	97.77	94.97	91.61	98.25	98.18	99.12
RBF_{grdl_rec}	96.11	96.25	94.69	84.04	89.6	88.56	77.99	96.12	86.08	96.72
RBF_{mix}	93.94	94.07	91.68	83.97	86.52	83.66	73.77	95.09	85.34	95.13
$Sine_{grdl}$	98.83	98.35	98.57	87.36	98.06	95.55	89.32	98.2	95.84	99.14
$Tree_{grdl\_rec}$	80.29	67.08	79.68	53.29	65.47	59.24	44.27	82.26	59.48	88.21
$Tree_{abr\_rec}$	84.99	72.4	84.73	61.48	78.32	66.46	57.67	86.42	65.3	90.87
$Wave_{grdl}$	83.87	83.87	82.5	74.66	79.57	79.75	78.84	81.04	79.35	84.16
$Wave_{abr}$	83.95	83.95	83.04	75.83	81.02	79.83	80.4	81.68	79.61	84.52
$Wave_{mix}$	84.55	84.55	83.36	75.96	81.04	81.58	80.47	81.89	80.27	84.65
Poker	52.95	51.48	49.64	45.75	48.58	49.66	49.19	57.21	55.2	56.38
Covertype	65.75	82.21	84.28	49.07	82.54	76.79	77.85	83.52	84.82	85.32
Weather	61.18	75.1	73.34	73.00	72.34	72.8	73.2	75.7	77.38	75.81

Table 3.5: Average chunk training time in deciseconds [S*10]

Dataset	DDD	Oza	AUE2	ACE	DWM	WMA	NSE	LevBag	ARF	En-ODDD
Agr_{mix}	1.014	2.58	0.147	0.175	0.328	0.029	2.413	9.141	0.662	0.278
Mix_{incr}	0.151	0.202	0.044	0.085	0.032	0.011	1.447	0.796	0.154	0.091
Mix_{grdl}	0.146	0.193	0.045	0.084	0.029	0.009	1.446	1.222	0.15	0.088
Mix_{abr}	0.14	0.193	0.041	0.085	0.028	0.008	1.438	1.218	0.146	0.091
RBF_{grdl_rec}	1.987	1.763	0.566	1.719	0.295	0.102	20.12	4.822	0.457	0.62
RBF_{mix}	0.895	0.654	0.657	1.795	0.353	0.099	5.342	2.554	0.457	0.667
$Sine_{grdl}$	0.187	0.306	0.068	0.147	0.089	0.014	1.859	2.089	0.245	0.111
$Tree_{grdl\_rec}$	0.917	5.295	0.226	0.462	0.259	0.048	5.698	10.364	0.529	0.302
$Tree_{abr\_rec}$	0.863	3.554	0.222	0.484	0.241	0.044	5.481	10.029	0.524	0.301
$Wave_{grdl}$	2.23	1.903	1.047	2.337	1.617	0.161	12.863	12.869	0.786	1.393
$Wave_{abr}$	2.335	1.893	1.05	2.199	1.615	0.159	12.877	8.388	0.753	1.345
$Wave_{mix}$	4.463	4.171	1.017	2.231	1.625	0.157	31.697	14.785	0.756	1.344
Poker	0.768	0.482	0.482	1.016	1.979	0.104	0.104	5.742	2.552	0.99
Covertype	2.612	4.594	1.124	1.974	1.37	0.239	11.754	2.968	1.377	1.454
Weather	2.261	0.478	0.57	0.827	0.882	0.129	0.386	4.063	1.967	0.882

Table 3.6: Data streams learning techniques used in comparative analysis

Algorithm	**Learning Process**	**Data Processing Mode**	**Drift handling mechanism**	**Detection Mechanism**	**Parameters Considered**
DDD [112]	Ensemble	Block	Explicit	EDDM	λ_l: to maintain ensemble with low diversity λ_h: to maintain high diversity ensemble γ: drift detection parameter
LearnNSE(NSE) [140]	Ensemble	Block	Implicit	–	a: sigmoid slope b: sigmoid infliction point
Oza [97]	Ensemble	Online	Implicit	–	l : base learner option
DWM [113]	Ensemble	Online	Implicit	–	β: factor for decreasing weights of experts($0 \leq \beta < 1$) p: period between expert removal θ: threshold for deleting experts
ACE [111]	Ensemble	Online	Explicit		α: confidence level factor μ: Adjustment factor of ensemble S_a: short term memory size.
WMA [118]	Ensemble	Online	Implicit	–	l: learner option list β: penalty factor for experts γ: minimum fraction of weight per model p: pruning factor
AUE2 [85]	Ensemble	Block	Implicit	–	n: maximum no. of component classifiers in ensemble c: chunk size m: maximum byte size of ensemble memory
LevBag [98]	Ensemble	Online	Implicit	–	λ_l: to maintain diversity ensemble l: base learner option
ARF [91]	Ensemble	Online	Explicit	ADWIN & PHT	λ_l: Poisson distribution parameter GP: Value of grace period taken for split test heuristics m: maximum no. of features that are evaluated per split δ_w: drift warning threshold δ_d: drift alarm threshold c(.): drift detection method

performance. First drift at 100k instance point has abruptly changing concept. It has a major influence on the accuracy which seems to stabilize later. The use of explicit drift detector in En-ODDD works best with abrupt changes and is the reason that performance of rest of the algorithms may not be as good as this hybrid technique. ACE has shown poor results despite the presence of drift detector which can be attributed to the fact that it lacks pruning of poorly performing classifiers which are not updated time to time. As seen in Figure 3.3(b), En-ODDD and Oza show most accurate results for mixed dataset $Wave_{mix}$ comprising of two gradually moving concepts separated by an abrupt drift at 500k instances. Since AUE2 is not well equipped with any explicit drift detection, it performs less compared to other diversity based approaches like DDD, ARF and En-ODDD. However DWM and LearnNSE handle this change without largely affecting performance due to their adaptive nature. ACE shows

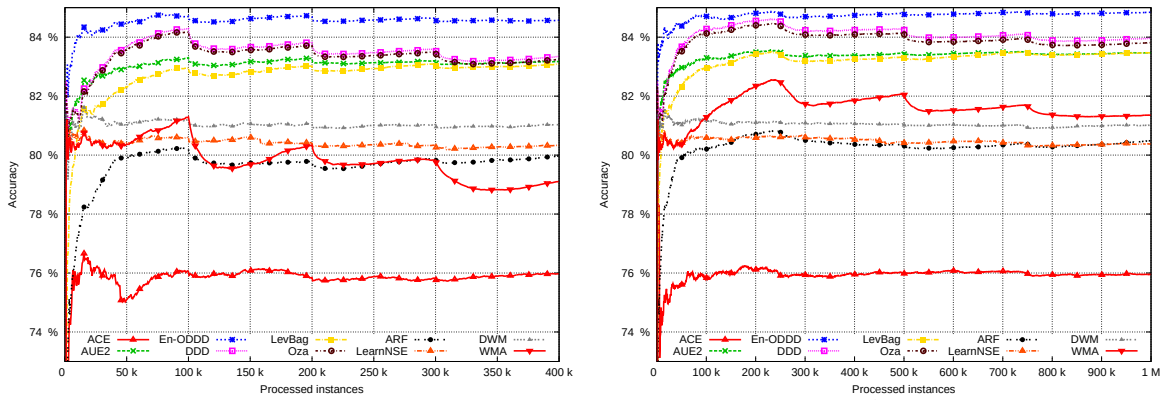


Figure 3.3: Predictive accuracy on recurring drifts a) $Wave_{abr}$ b) $Wave_{mix}$

a steady performance to mixed drifts without much rise or fall near drift points.

Experiments with Agrawal Generator: En-ODDD performs well at all the three consecutive drift points *i.e.* at 250k, 500k and 750k instances with accuracy of LevBag being slightly lower than it. The capability to sustain accuracy by En-ODDD and LevBag is attributed to the presence of diverse ensemble components. The explicit drift detector in En-ODDD and ACE helps them to recover from the drop in accuracy after first drift. ACE shows a steady accuracy near the drift point. DDD handles itself efficiently as compared to others but most of the algorithms like LearnNSE, DWM and even AUE2 are severely affected by the first drift point. Even ARF which usually stabilizes itself after drift points, has shown a fall in accuracy largely after first drift.

Experiments with RBF Generator: In this dataset, there is a case of four alternating drift points at 125k, 250k, 500k, 750k. En-ODDD, DDD, LevBag and Oza have performed in similar manner in recovering from these drifts. WMA and DWM failed to adapt to mixed drifts due to absence of explicit drift detectors. Despite the difference in accuracy levels of ARF and LearnNSE being large, both have shown a similar recovery pattern after the drifts, because of strong time similarity in data being used for classification. In case of RBF_{grdl_rec} there is a slight difference in accuracy of diversity based online algorithms DDD, LevBag,

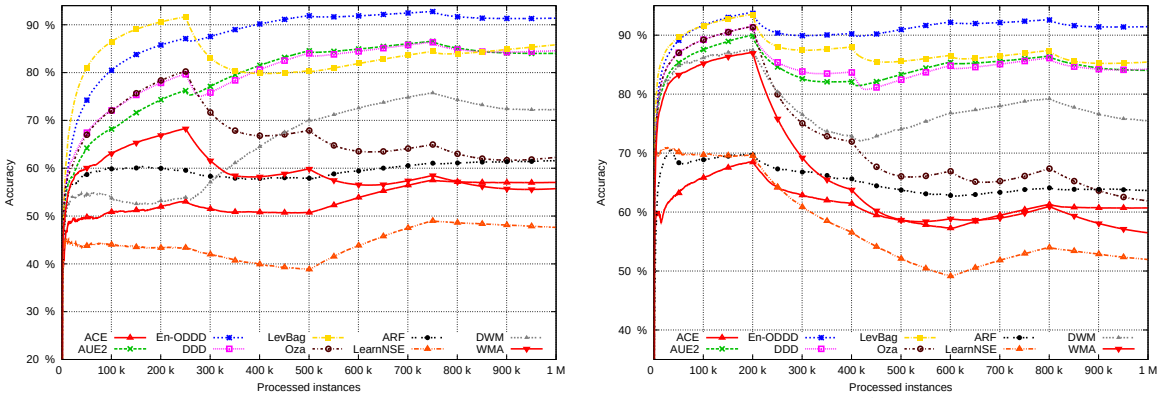


Figure 3.4: Predictive accuracy on recurring drifts a) $Tree_{grdl\_rec}$ b) $Tree_{abr\_rec}$

Oza and En-ODDD. In this scenario, there is no single best performing approach. ARF has shown a severe drop in performance after 500k instances which clearly shows it is not suitable in gradually recurring drift scenarios.

Experiments with Tree Generator: Figure 3.4(a) and Figure 3.4(b) illustrate the performance of classifiers on the $Tree_{grdl\_rec}$ and $Tree_{abr\_rec}$ datasets respectively. In the $Tree_{grdl\_rec}$ scenario the speed of recurring changes play an important role. Although DWM has shown better adaptation to drift, En-ODDD performs quite well by achieving better accuracy as compared to DDD. Interestingly, LevBag outperformed others before the first drift point but had a major drop in accuracy after that. AUE2 performs similar to En-ODDD due to removal of buffer classifiers. Both Oza and LevBag do not adapt themselves to the recurring drifts after every 250k instances as they lack any pruning mechanism. Results indicate that the diversified ensemble without any drift control strategy is not enough to handle such situations. LearnNSE, WMA and ACE do not react well to recurring changes irrespective of speed of change. In $Tree_{abr\_rec}$ dataset, abruptly recurring drifts are simulated after every 200k instances. En-ODDD closely followed by LevBag algorithm performs efficiently in this case. DDD and Oza failed to respond to recurring drifts well. The absence of drift detector in Oza is a reason for poor adaptation to suddenly recurring concepts. WMA has

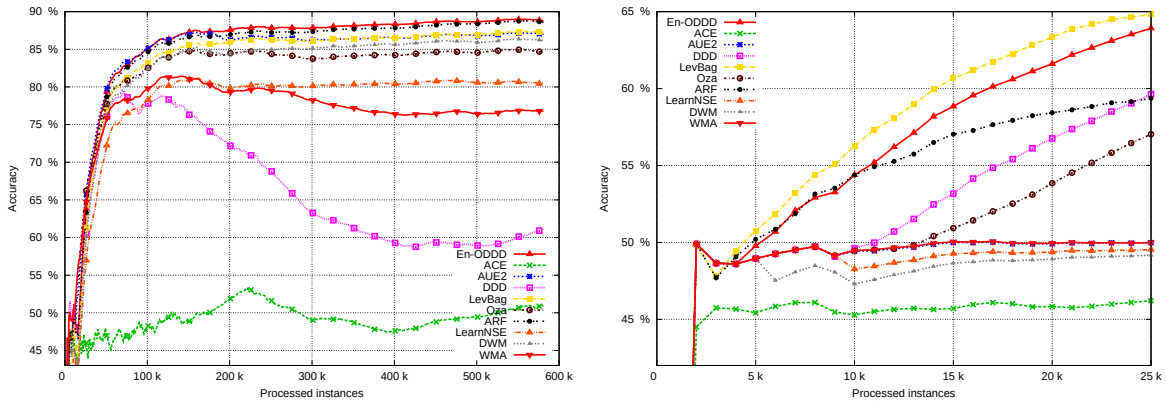


Figure 3.5: Predictive accuracy on real datasets a) Covertype b) Poker

shown drastic decrease in accuracy. Further, algorithms like LearnNSE and ACE which do not prune their weak performing components, show decrease in accuracy.

Experiments with Real datasets: In all three real datasets (*Poker*, *Weather*, *Covertype*) En-ODDD performs consistently well than all the algorithms considered for comparisons. Efficient performance is accomplished because of the generalization in classification error produced due to diverse components. It is evident from Figure 3.5(a) that the adaptive approaches like AUE2, LevBag, ARF, LearnNSE, Oza, DWM perform relatively better in this case as compared to simulated drifts. The combination of online and adaptive approach has helped En-ODDD achieve best results. A significant drop in accuracy is observed in the DDD algorithm. The combination of low and high diversity ensembles does not cater to drifts in this real dataset. As in most of the artifical datasets, ACE continues to be a poor performer. Figure 4.6(a) analyzes the *Poker* dataset. There is a sudden increase in accuracy of En-ODDD and LevBag algorithms after 10k instances. ARF and DDD are also closely following them. However, other approaches like WMA and AUE2 show a consistent performance with no increase in accuracy at any point of time. DWM, LearnNSE, ACE are least performing algorithms on this dataset.

Table 3.7: Average classification accuracy of En-ODDD obtained over 10 iterations in percentage[%]

DataSet	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean ± s.d.
Agr_{mix}	92.62	92.66	92.56	92.65	92.66	92.46	92.51	92.41	92.98	92.74	92.62 ± 0.16
Mix_{incr}	99.10	99.13	99.09	99.10	99.13	99.09	99.11	99.11	99.09	99.16	99.11 ± 0.02
Mix_{grdl}	99.14	99.13	99.09	99.18	99.16	99.13	99.13	99.18	99.10	99.09	99.13 ± 0.03
Mix_{abr}	99.13	99.12	99.14	99.14	99.11	99.11	99.16	99.13	99.07	99.11	99.12 ± 0.02
RBF_{grdl_rec}	96.80	96.72	96.64	96.75	96.69	96.71	96.81	96.72	96.76	96.60	96.72 ± 0.06
RBF_{mix}	95.06	95.28	95.15	94.96	95.20	94.99	95.25	95.34	95.11	94.95	95.13 ± 0.14
$Sine_{grdl}$	99.16	99.18	99.12	99.15	99.12	99.13	99.08	99.15	99.15	99.13	99.14 ± 0.03
$Tree_{grdl\_rec}$	88.37	87.86	88.07	88.39	87.82	88.28	88.30	88.70	88.24	88.09	88.21 ± 0.26
$Tree_{abr\_rec}$	90.83	90.85	90.91	90.98	90.84	90.68	90.98	90.96	90.97	90.73	90.87 ± 0.11
$Wave_{grdl}$	80.29	80.33	79.75	80.44	80.27	80.24	80.34	79.79	80.20	80.24	80.19 ± 0.23
$Wave_{abr}$	82.59	82.57	82.62	82.58	82.55	82.58	82.53	82.61	82.50	82.61	82.57 ± 0.04
$Wave_{mix}$	82.61	82.67	82.61	82.66	82.67	82.72	82.68	82.62	82.61	82.68	82.65 ± 0.04
Poker	57.03	56.69	56.04	56.33	55.73	57.05	56.76	55.64	56.01	56.50	56.38 ± 0.51
Covertype	85.41	85.31	85.39	85.28	85.14	85.36	85.28	85.37	85.31	85.34	85.32 ± 0.08
Weather	75.20	74.74	75.43	73.99	74.10	75.27	73.12	74.96	74.28	75.34	74.64 ± 0.75

3.4.1 Interpretability of the Proposed Approach

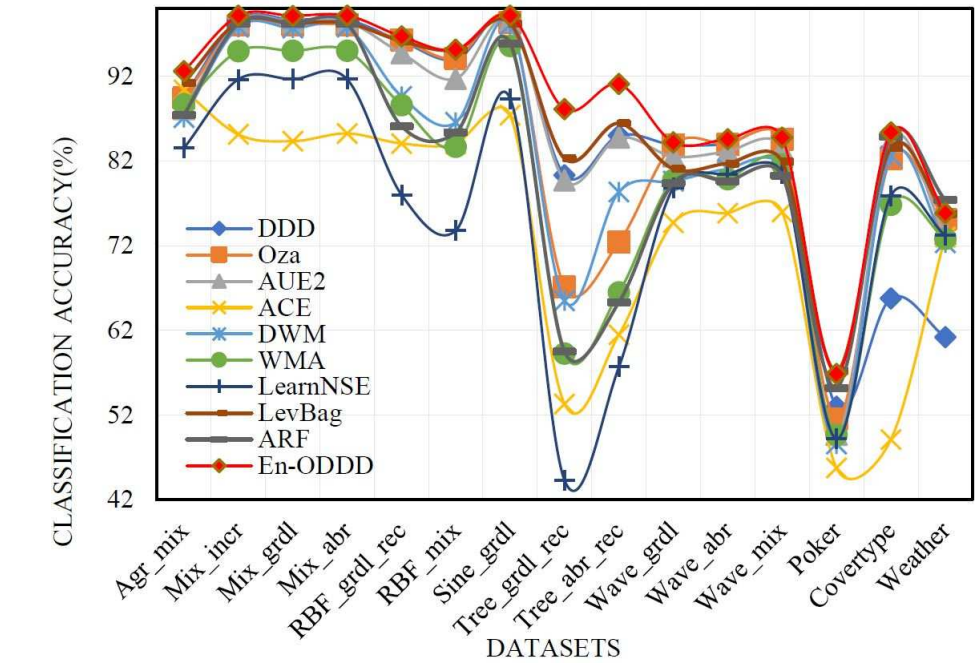
In the proposed approach bagging is employed which provides higher generalization accuracy to the ensemble system. It creates varied training sets of random sub-samples for continous improvement and outputs weighted aggregated result. However, as the process involves randomization, sometimes it makes experiments less interpretable in single execution. To verify its correctness and relaibility, ten repetitions of En-ODDD were done for each dataset. Table 3.7 presents the results of average accuracy along with mean± standard deviation obtained over multiple runs to check interpretabilty. Further, statistical tests were also performed for analyzing the varied performance over multiple datasets. It can be concluded that the deviation among multiple runs is not significant, making the approach relaible and

trustworthy.

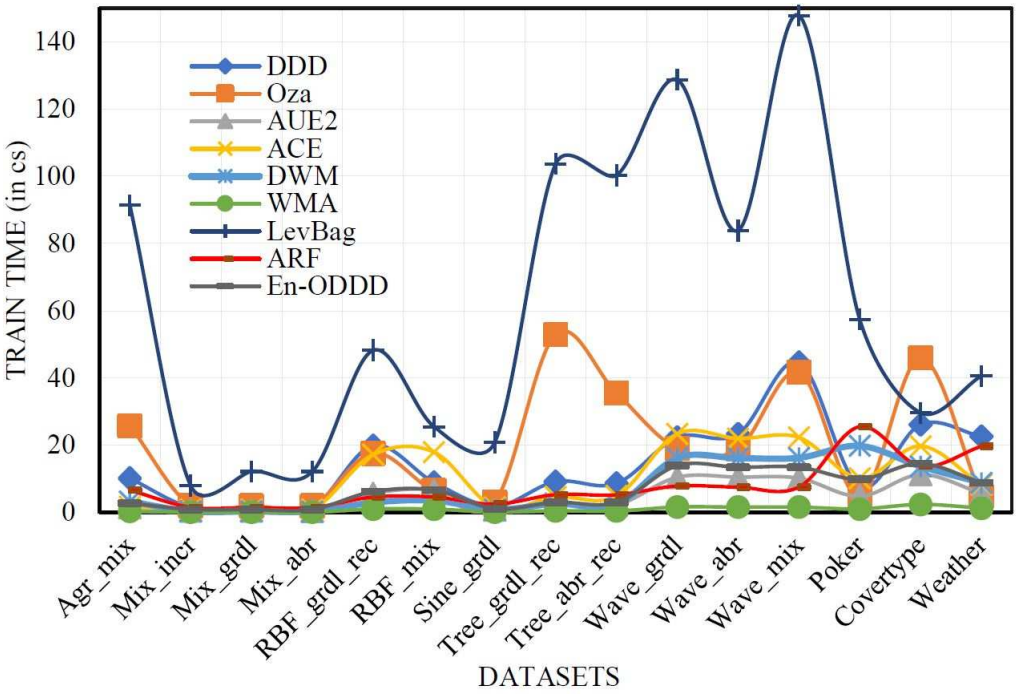
3.4.2 Trade-off Analysis

Introduction of bagging in our approach increases the predictive accuracy of the underlying ensemble as better training is achieved due to diverse learners. Figure 3.6(a) demonstrates that En-ODDD achieves best accuracy among all the algorithms compared for all the datasets. However, in Figure 3.6(b) we have compared the training time that was obtained for all approaches. It can be seen that for ARF, ACE, Oza, DDD, NSE and LevBag, training time is much more than the proposed approach. Three algorithms WMA, DWM and AUE2 take less time in comparison to En-ODDD but accuracy achieved by our approach is more than all these. Though bagging encounters some overhead in random sub-sampling the increase in accuracy is much more significant as compared to it. Further time is reduced by the usage of multithreading where base learners are trained in parallel while updation.

In terms of computational complexity, in EnODD updation of T classfiers incurrs $O(T)$ time complexity which includes logarithmic component of $O(logW)$ for window of length W processing per item. Bagging involves sub-sampling but does not impact the overall complexity too much. Instead, other techniques like LevBag, LearnNSE and DDD have huge training time due to high cost computations involved. DDD uses four ensembles simultaneously giving time complexity of $O(4*T)$ which is much higher than En-ODDD. However WMA and DWM took less training time than En-ODDD, since they do not continously update their existing ensembles rather just use pruning mechanism. Hence, they provide very low accuracy for all drifting streams (Table 3.8). AUE2 lacks diversity generation strategy, therefore take slightly less time than our approach but at the cost of accuracy. ACE in online setting has inbuilt detctor which accounts for major training time. ARF has hyper-paramter tuning which is a time consuming process. Thus, En-ODDD manages to maintain a balance



(a)



(b)

Figure 3.6: a) Predicitive accuracy of all the algorithms b) Train time of all algorithms

between achieving high accuracy and feasible train time. Figures 3.7(a), 3.7(b), 3.8(a) and 3.8(b) depict the evaluation time taken by various algorithms for gradual, abrupt, mixed and

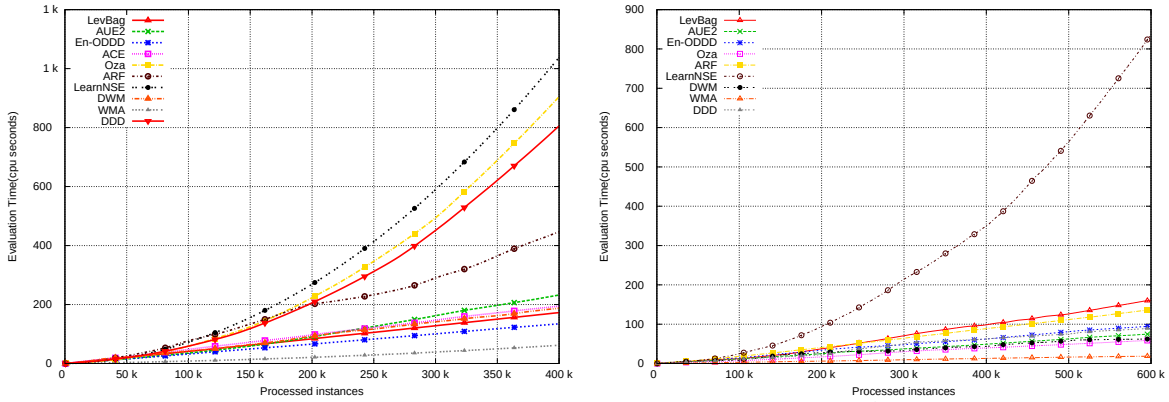


Figure 3.7: Evaluation time for a) gradual drifts b) abrupt drifts

real drifts respectively. It is observed that with increase in processed instances, En-ODDD adopts linear increase in time which is comparatively less than other approaches. Constant updating and weight based detection system majorly helped En-ODDD to encounter concept drift of all types which is otherwise difficult to handle. Since the online concept drift problems have major focus towards achieving higher accuracy, it can be concluded that En-ODD is trustworthy.

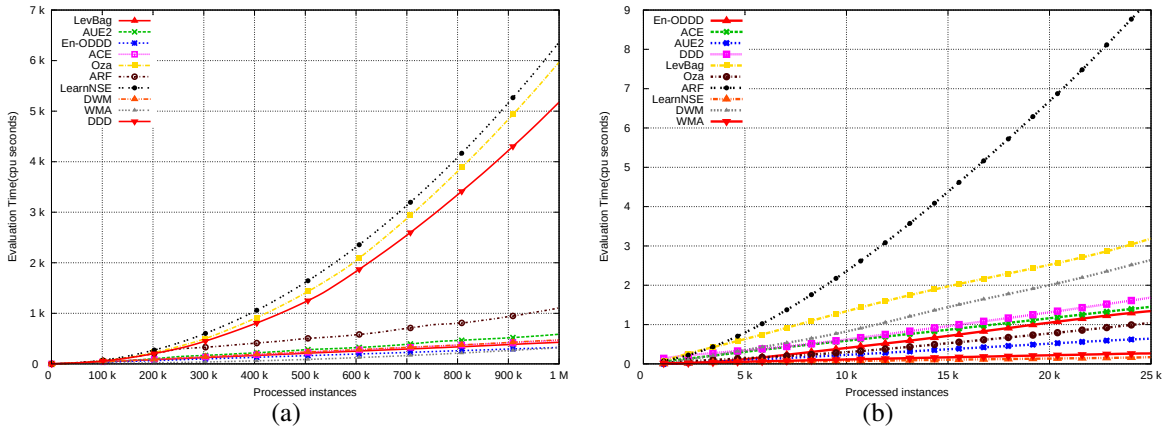


Figure 3.8: Evaluation time for a) mixed drifts b) real dataset

3.4.3 Statistical Analysis

To compare various algorithms and to show if there exist significant differences among them, it is essential to give statistical tests support. This work investigates usage of Friedman and Wilcoxon tests for machine learning methods [91, 92, 139]. The null hypothesis for experimental design suggests that there exists no significant difference between the prediction performances of algorithms tested. Post-hoc analysis using the Bonferroni-Dunn test [141] is performed in case null-hypothesis is rejected. F_f statistic value ranks separate methods based upon the average results [142]. The lowest rank is given to the best performing approach and vice-versa. As stated by Eq. 3.5 [142] average ranks ($\bar{R}$) and Friedman statistic (${\chi_F}^2$) are computed (using N datasets and k algorithms).

$$ {\chi_F}^2 = \frac{12N}{k(k+1)}[\Sigma\bar{R}^2 - \frac{k(k+1)^2}{4}] \tag{3.5a} $$

$$ F_f = \frac{(N-1){\chi_F}^2}{N(k-1)-{\chi_F}^2} \tag{3.5b} $$

Table 3.8 presents the average ranks of the algorithms that were analyzed earlier to compare accuracy, train time and test time. Computing the F_f for accuracy, value = 32.21 was obtained which is greater than the critical value 1.95 obtained by F-distribution at 95% of confidence indicating that the null hypothesis gets rejected. Post-hoc analysis results indicate that performance of En-ODDD is better than Oza, ACE, DWM, WMA, ARF and LearnNSE as the Critical Difference (CD) = 3.13. For algorithms like AUE2, LevBag and DDD, Wilcoxon Test was performed since their average accuracy ranks were higher than that of En-ODDD. The p-values obtained were: $p_{DDD} = p_{AUE2} = 0.0006$ and $p_{LevBag} = 0.0011$. These values indicate that En-ODDD is better in terms of accuracy as compared to all other algorithms considered.

For the train time, the F_f statistic returned 32.87, indicating rejection of hypothesis.

Table 3.8: Average algorithm ranks obtained from Freidman tests

	En-ODDD	DDD	OzaBag	AUE2	ACE	DWM	WMA	LevBag	ARF	NSE
Accuracy	1.13	3.76	4.36	4.26	9	6.8	7.4	3.4	6.133	8.733
Train Time	4.77	6.93	6.833	2.966	5.533	3.76	1.033	9.4	5.26	8.5
Test Time	6	6.77	6.77	5.96	2	4.63	2.46	6.73	4	9.66

By comparing average ranks obtained in Table 3.8 and performing Wilcoxon test, it can be concluded that En-ODDD is faster than DDD, Oza, LevBag and LearnNSE ($p_{DDD} = 0.001, p_{Oza} = 0.008, p_{LevBag} = 0.0005, p_{LearnNSE} = 0.001$) but slower than AUE2, DWM and WMA. The tested En-ODDD algorithm has outperformed in classification accuracy in presence of all possible drift scenarios.

3.5 Threats to Validity

This section discusses various potential threats to the validity of this study along with some mitigations taken to reduce their impact on our work. Though the performance of proposed approach has been proved on a reasonable number of instances ($\sim$1M), it can be further validated by increasing more number of instances. Therefore, the analysis on increasing the scalability of this approach is left for future work. Another threat could be misinterpretation of the actual relationship between predicted variable and predictors which is caused because of non-evaluation of statistical results [143]. However, this threat is removed in this study by using two non-parametric tests, Wilcoxon and Friedman at confidence level of 95% to statistically validate the results obtained. Finally, although the proposed technique exhibited encouraging prediction performance on various drift patterns like gradual, abrupt and recurring, it requires further investigation on combination of drift scenarios where multiple types of drifts co-exist.

3.6 Conclusion

En-ODDD is suitable for concept drift handling specifically when dealing with heterogeneous drifts. Incremental learning ensures timely reaction to concept drift by using ensemble of diversified experts embedded with an active drift detector. Further combination of majority weighting mechanism and online drift detector in En-ODDD covers all possible drift scenarios: gradual, abrupt, recurring, mixed. Introduction of diversity by modifying the incoming training instances using Online Bagging and diversified-update mechanism, are primary reasons of En-ODDD outperforming most algorithms in terms of accuracy. An explicit drift detector enables En-ODDD in identifying abrupt drifts quickly, without waiting for the chunk cycle to complete. Moreover, the updating of experts at regular intervals of time helps the model to adapt better to gradual drifts. The impact of diversity parameter λ is also investigated by testing En-ODDD with different values. Best results were obtained when $\lambda = 1.5$. After analyzing results of various pruning strategies it can be concluded that substituting the least performing expert is the best option. The least performing expert is identified based on the weight associated with the experts. The statistical tests and empirical study suggest that En-ODDD achieves higher accuracy under different drifting streaming conditions.

Next chapter elaborates the second proposed technique, *i.e.*, TLP-EnAble, which is also used to handle concept drift using two-level pruning mechanism.

Chapter 4

Two-Level Pruning based Ensemble with Abstained Learners (TLP-EnAbLe) for Drifting Distributions

In this chapter, a novel concept drift handling technique named as Two-Level Pruning based Ensemble with Abstained Learners (TLP-EnAbLe) is proposed. It augments chunk based method where prime consideration is to make the system as diverse as possible.

4.1 Introduction

A popular strategy to tackle challenges of concept drifting data streams is to use ensemble methods which combine the output of many learning models [144]. Since the streaming data arrives continuously, it is important to add models which are trained on the latest concept, while at the same time it is crucial to maintain the ensemble size [145].

To address this issue majority of the techniques employ two pruning modes: age based or accuracy based. In the former, the old learners are periodically removed from the system

[132, 140] and in the latter, low performing ones are pruned [85, 106]. In accuracy based pruning, a weight, calculated based upon a learner's predictive performance on chunk of data, is monitored. After a certain period of time, weights associated with individual learners are checked to locate the least performing learner. Diversity *i.e.* how each learner is different from one another, plays an integral role in shaping the final prediction. In classification context, using diversity of learners while pruning can play a vital role for success of ensemble.

In TLP-EnAbLe diversity of learners is combined with accuracy-based pruning to facilitate diverse learning environment and improve performance of learners. It removes learners with least performance *i.e.* learners generating lesser diverse ensemble are removed. Further, deferred removal of redundant learners based on conceptual equivalence [146] is incorporated to make the ensemble as diverse as possible. A learner is removed after it is experimentally identified that it is incapable of predicting current instances effectively. Such delay can lead to a reliable selection and retention of efficient ensemble members.

Another important aspect while dealing with ensembles is to choose the best performing learners while formulating final set for hypothesis generation. To improve the prediction accuracy, final decision making process should include a flexible and efficient learner selection method. TLP-EnAbLe enhances the existing learner selection strategies by adding another level of abstaining based upon average weights of learners. With such method, a globally efficient learner is allowed to participate in decision even if it has a low confidence on the current testing instance. This leads to improved dynamic selection of overall confident learners and also achieve adaptability over noisy streams as well.

4.2 Proposed Approach: TLP-EnAble

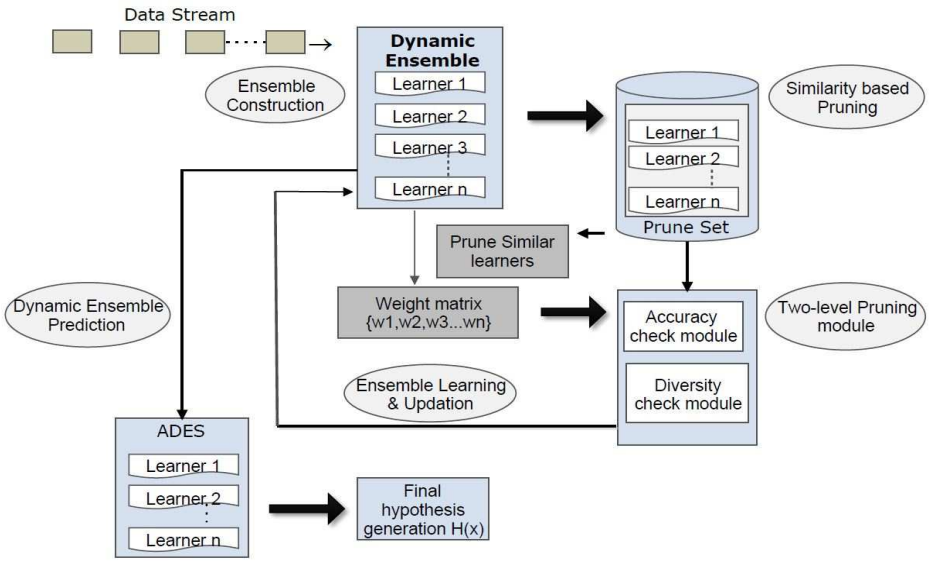


Figure 4.1: Block diagram of TLP-EnAbLe (the proposed approach).

The proposed approach TLP-EnAble considers a dynamic ensemble construction which constantly updates the underlying learners with latest information. Block diagram for TLP-EnAble is depicted in Figure 4.1 TLP-EnAbLe has following three components.

1. First component is similarity based pruning based on conceptual equivalence to identify similar models. One of the similar learners is pruned in the next iteration rather than the current one. This deferred pruning ensures removal of relatively less accurate similar models once it is assured that it is under performing and redundant.

2. Second is two-level pruning mechanism which considers diversity along with accuracy while pruning the ensemble learners leading to maintenance of diverse set of learners at all times.

3. Third is a dynamic selection scheme which abstains learners in decision making. The proposed approach abstains the low performing learners from the final hypothesis generation considering their local and global performance.

Following sections provide the details about each phase in the proposed approach. A

Table 4.1: Summary of main notations used in TLP-EnAble

Symbol	Description	Symbol	Description
w_{new}	Weight of newly added learner to ensemble	x_i	Current instance
w_k	Weight of existing learner in ensemble	$[E^k_{div}]$	Set of k poor learners
MSE_r	Mean square error of randomly predicting learner	div(E)	Diversity associated with ensemble E
$p(y)$	Prior probability of class y in current chunk .	l_{new}	Newly built learner
ε	Small positive value	$y(x_k)$	Prediction associated with instance x_k
$p^k_y(x)$	Probability given by learner l_k that x is instance of y	α_c	threshold confidence factor
c_n	Current chunk	$\hat{H}(x_t)$	Final hypothesis function
E	Ensemble	$FLAG_{prune}$	Flag value associated with pruning
$\hat{M}$	Size of ensemble E	d_k	Diversity of ensemble E after removing learner l_k
l_m, l_n	Learners m and n under comparison	CTR	Learner to remove
s	Threshold similarity margin	[W]	Weight Matrix
w_m	Weight associated with learner l_m	w_{avg}	Average weight of learners
w_n	Weight associated with learner l_n	α_{sig}	Significance level
$[C_{worst}]$	Set of n worst learners	N	Number of datasets
$[MTR_i]$	Pruning buffer set in current iteration i	k	Number of algorithms in comparison
$[MTR_{i-1}]$	Pruning buffer set in previous iteration i-1	F_f	Friedman static
$[SP_r]$	Final pruning set	CD	Critical difference
δ	Magnitude of change	θ	Angle of drift

summary of all the notations used in the approach are stated is Table 4.1.

4.2.1 Deferred Removal of Similar Learners

In TLP-EnAbLe, an ensemble of $\hat{M}$ learners is built over the incoming stream of instances. They are assigned weights according to their performance on chunks of data received. Concept of weighted learners given in AUE2 [85] is used for voting since it leads to best overall performance in ensemble learning [85]. New learner is constructed once the predefined chunk size is achieved and it is assigned a weight w_{new} according to the distribution of the current chunk c_n using Eq. (4.2). MSE_r, the mean square error of a randomly predicting learner is calculated and ε is a very small positive value considered to avoid the divide by zero error in Eq. (4.2). The weights w_k of the existing learners in the ensemble are updated considering their prediction error over the chunk (Eq. (4.3)).

$$MSE_r = \sum_y p(y)(1-p(y))^2 \tag{4.1}$$

$$w_{new} = \frac{1}{MSE_r + \varepsilon} \tag{4.2}$$

$$w_k = \frac{1}{\frac{1}{|c_n|} \sum\limits_{\{x,y\} \varepsilon c_n} (1 - p_y^k(x))^2 + MSE_r + \varepsilon} \tag{4.3}$$

Here $p_y^k(x)$ is the probability given by learner l_k that x is instance of class y. Once the

Algorithm 4 DEFERRED_SIM_PRUNING

Input: Ensemble E, Weight Matrix [W], Current chunk c_n, MTR_{i-1}

1: **for** every pair (m,n) $\in$ E **do**
2: Compare l_m and l_m
3: **for** each instance x_i in c_n **do**
4: Calculate prediction over x_i using Eq. (4.4)
5: **end for**
6: Calculate *SimIndex* using Eq. (4.5) over c_n
7: **if** SimIndex(c_n,l_m,l_n) $\geq$ s **then**
8: **if** ($w_m < w_n$) and m $\in$ [C_{worst}] **then**
9: Add l_m to set [MTR_i]
10: **else**
11: **if** ($w_n < w_m$) and n $\in$ [C_{worst}] **then**
12: Add l_n to set [MTR_i]
13: **end if**
14: **end if**
15: **end if**
16: **end for**
17: Final Prune Set [SP_r] $\Longleftarrow$[MTR_i] $\cap$ [MTR_{i-1}]
18: Remove learners in Final Prune Set [SP_r] from E
19: Update E $\Longleftarrow$ E–SP_r
20: Update [MTR_i] $\Longleftarrow$ [MTR_i] $- SP_r$
21: return E, [MTR_i]

learners are built, similarity based deferred pruning mechanism is followed to restrict the size of ensemble. Algorithm 4 explains all the steps involved in this phase. The learners are compared with each other and similar learners are identified using conceptual equivalence. Two learners are said to be similar if they classify the given instances similarly. Pairwise comparisons are made among all the existing learners in the ensemble and a similarity matrix is constructed based upon the predictions of these learners on the current chunk of instances.

While comparing two learners say l_m and l_n, their prediction value *Prediction* is calculated over all the instances x_i of the current chunk c_n as per Eq. (4.4) (line 4). Similarity index *SimIndex* lying between [0,1] is calculated using Eq. (4.5)(line 6). The learners are marked as similar if the value of *SimIndex*(c_n, l_m, l_n) exceeds the threshold similarity margin s where $s \in [0, 1]$ (line 7). Here, s is a user set parameter, defined as the proportion of number of instances two learners are classifying in same manner while considering same concept. If the learners l_m and l_m are similar, one of them is removed from the ensemble as they are likely to be trained in similar manner and represent same concept. A general approach could be to remove the less accurate one among both say l_p where p is either m or n. But there could be possibility of removing an overall well performing learner from the ensemble. To avoid such scenario, first a set of n worst learners $[C_{worst}]$ among the $\hat{M}$ ensemble members are identified. Next, the less accurate similar learner is removed, if it is amongst the n worst learners. In the proposed approach the concept of deferred pruning is introduced where l_p (included in the list of non-performing learners) is added to the pruning buffer $[MTR_i]$ but not removed until next chunk arrives (line 9, 12). In the next iteration, $i+1$, $[MTR_{i+1}]$ is calculated and set of learners $[SP_r]$ which form part of both $[MTR_i]$ and $[MTR_{i+1}]$ are removed (line 17). Thus, the learners which are liable to be removed in consecutive chunks are pruned (line 19). This step ensures reliable pruning of only low performing similar learners, leading to construction of diverse and accurate ensemble.

$$Prediction(x_i, l_m, l_n) = \begin{cases} 1, & \text{if } l_m(x_i) = l_n(x_i) \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

$$SimIndex(c_n, l_m, l_n) = \frac{\sum Prediction(x_i, l_m, l_n)}{\sum_{i=1}^{c_n} x_i} \tag{4.5}$$

4.2.2 Two-Level Pruning Mechanism

When the data stream is processed, the ensemble based methods require selection of best learners which can be updated with the latest incoming chunk. This requirement is accomplished by focusing on aggregating diverse and accurate learners. In general, majority of algorithms implement accuracy based pruning where the worst performing is pruned from ensemble [85][92]. In the proposed approach, diversity of base learners is considered along with their predictive performance for pruning. Two-level pruning strategy is incorporated: first set $[E^k_{div}]$ of k poorly performing learners are selected out of $\hat{M}$ total learners in the ensemble E. This is done by calculating weights of respective learners on current chunk. The learner with lowest weight is considered least performing in terms of accuracy. Next $[E^k_{div}]$ is passed onto diversity check module wherein diversity of ensemble E, *div(E)* is calculated by removal of each learner $l_k \in [E^k_{div}]$ where the learner k, is the one whose removal leads to maximum diversity in the system. Finally, the selected learner is replaced with the new learner l_{new} built over the current chunk (Algorithm 9).

Table 4.2: Representation of classification output by the learners

Representation (N^{mn})	**Classification output (m) by learner** l_i	**Classification output (n) by learner** l_j
N^{00}	Incorrect	Incorrect
N^{01}	Incorrect	Correct
N^{10}	Correct	Incorrect
N^{11}	Correct	Correct

The diversity check module can be configured with any of the proposed diversity measures [147]. In the proposed work three popular diversity measures: Q-static *(Q)*, double fault *(DF)* and disagreement *(DS)* have been considered. In k-dimensional binary vector data stream where $X = x_1, x_2 .... x_k$ are labeled data instances and $Y = [y(x_1), y(x_2) ... y(x_k)]$; value of $y(x_k)$ = 1 if a learner l_v correctly predicts the class label of instance x_k and 0 if it incorrectly classifies it. N^{mn} represents the number of training samples whose prediction is m and

n for learners l_i and l_j respectively where m,n $\in \{0,1\}$. Representation for N^{mn} is provided in Table 4.2.

Three diversity measures Q-static *(Q)*, double fault *(DF)* and disagreement *(DS)* are calculated, using Eq. (4.6), Eq. (4.7) and Eq. (5.9) respectively:

$$Q(l_i,l_j) = \frac{N^{00}N^{11} - N^{01}N^{10}}{N^{00}N^{11} + N^{10}N^{01}} \tag{4.6}$$

$$DF(l_i,l_j) = \frac{N^{00}}{N^{00}N^{11} + N^{10}N^{01}} \tag{4.7}$$

$$DS(l_i,l_j) = \frac{N^{01} + N^{10}}{N^{00}N^{11} + N^{10}N^{01}} \tag{4.8}$$

4.2.3 Abstaining Learners in Decision Making

It is important to check the efficacy of the underlying learners after applying diversity based pruning. Since TLP-EnAbLe considers accuracy as an important parameter, dynamic selection of high performing ensemble is proposed. Unlike other approaches where the weighted majority vote of all learners in the ensemble is considered, in TLP-EnAbLe a selective approach is proposed to allow only a set of best performing learners to participate in final decision making. Algorithm 6 describes the Abstained Dynamic Ensemble Selection (ADES) in detail where TLP-EnAbLe implements a dynamic abstaining principle which considers the performance of a certain learner at two levels. At first level the weight w_i of the learner l_i on the current chunk c_i is calculated (line 5) and compared with the average weight w_{avg} of all the members of the ensemble (line 6). If w_i is more than w_{avg}, the learner is allowed to participate in the voting (line 8). If it is less than w_{avg}, the second level of performance of l_i is evaluated. Lesser value indicates that l_i is not as competent as the other members of the ensemble. The prediction vote v_i of the learner l_i on the current testing

Algorithm 5 Proposed Approach (TLP-EnAbLe)

Input: $(x_1, x_2, x_3 ... x_i)$: Data stream DS , Ensemble model E, confidence factor α_c, Ensemble Size SIZE

Output: Hypothesis : $\hat{H}(x_t)$

if $|c_n| <$ CHUNK SIZE **then**
  add current instance x_i to current chunk c_n
else
  Construct a new learner l_{new} on c_n
  Calculate w_{new} using MSE_r for l_{new} using Eq. (4.2)
  for each learner l_k in E **do**
    Update w_k using Eq. (4.3) on c_n
  end for
  if $|E| < \hat{M}$ **then**
    Add l_{new} to E
  else
    Set $FLAG_{prune}$ = TRUE
    Sort all l_k in E according to w_k
    Add worst t learners from E in set $[C_{worst}]$
    $[MTR_{i-1}] = [MTR_i]$
    DEFERRED_SIM_PRUNING(E, [W], c_n, $[MTR_{i-1}]$, $[C_{worst}]$) //Algorithm 4
  end if
  Sort E according to [W]
  Update SIZE $\Longleftarrow |E|$
  k = min(SIZE/2, 3)
  $[E^k_{div}] \Longleftarrow$ Select poorest k learners from E
  for each learner l_k in $[E^k_{div}]$ **do**
    Calculate d_k = div(E-l_k) using Eq. (4.6)
  end for
  Update CTR $\Longleftarrow$arg max(d_k)
  if CTR $\in MTR_i$ **then**
    Update $MTR_i \Longleftarrow MTR_i$-CTR
  end if
  Replace CTR with l_{new}
  Update all $l_k \in$ E on c_n
end if
Obtain Final ensemble E_f from ADES(E, [W], α_c) //Algorithm 6
Output final hypothesis $\hat{H}(x_t)$: $\mathbf{X} \rightarrow \mathbf{Y}$
Predict with $\hat{h}_k(x_t) : \delta\ E_f[h_k(x_t) = y_t]$
return $\hat{H}(x_t)$: arg $\max_{y_t} \sum_{k=1}^{L} \hat{w}_k \hat{h}_k(x_t)$

instance x_t is calculated which lies in the range [0,1] (line 10). This vote value is compared with a pre-defined threshold confidence factor α_c (line 11). If the obtained vote value exceeds this confidence factor, the learner l_i is allowed to participate in final hypothesis generation else it is abstained from voting (line 12). Smaller values of α_c indicate more chances of participation. v_i depends on the prediction confidence on the latest testing instance. By applying two-level abstaining strategy, the less confident and under performing learners are excluded from decision making. The confidence threshold is adjustable and can change its value according to the drifting scenario so as to promote maximum suited learners for drift recovery.

Algorithm 6 Abstained Dynamic Ensemble Selection (ADES)

Input: Ensemble E, Weight Matrix [W], threshold confidence factor α_c, Final Ensemble E_f
Output: Final Ensemble E_f

1: Initialize w_{avg}
2: **for** each learner l_i in E **do**
3: $w_{avg} \Longleftarrow w_{avg} + w_i$
4: **end for**
5: $w_{avg} \Longleftarrow w_{avg} / \mid E \mid$
6: **for** each learner l_i in E **do**
7: **if** $w_i > w_{avg}$ **then**
8: $E_f \Longleftarrow l_i \cup E_f$
9: **else**
10: $v_i \Longleftarrow$ get prediction vote on x_t
11: **if** $v_i \geq \alpha_c$ **then**
12: $E_f \Longleftarrow l_i \cup E_f$
13: **end if**
14: **end if**
15: **end for**
16: return E_f

4.3 Experimental Setup and Results

In this section, various experiments performed to evaluate the performance of the proposed approach have been discussed. The experiments have been divided into four sections.

In the first section, the classification performance of the proposed approach has been compared against state-of-the-art algorithms on artificially generated streams. Massive Online Analysis (MOA) framework [137], which is popular for streaming data, is used for running all experiments to check the robustness of the proposed approach. Various drift patterns such as abrupt, gradual, recurring along with the complex combination of these have been considered for evaluation. To further validate proposed approach, ten real world datasets used in majority of streaming classification problems, are considered for comparison with other algorithms. The proposed approach is then evaluated on three diversity measurement variants to check how these options affect its behavior. To evaluate the influence of two important parameters, similarity index, s, and abstain confidence factor, α_c on the performance of proposed approach, experiments have been conducted by varying their values.

The evaluation methodology used for experimentation is *EvaluateTestThenTrain* [148]. Each instance is first used to test the underlying ensemble model before using it for training. Experiments have been carried out on Intel Core i7-7700HQ 2.80GHz Windows system with 8 GB of RAM. Various artificially created data streams available in MOA are used for experimentation (details are provided in Section 4.3). Performance metrics used for comparison are classification accuracy, kappa statistic [94, 144] and evaluation time over all the datasets. For supporting the experimental results, statistical analysis has also been performed over multiple data streams.

Comparison approaches:

For comparison with the proposed approach, eleven ensemble-based techniques have been considered, mentioned in Table 4.3 along with their base learners and main parameters. AWE, AUE2 and BLAST are popular techniques which use accuracy based pruning. Our approach enhances the accuracy based pruning by incorporating diversity so these techniques are used as baseline for comparison. Since ARF, ECPF and OZA are frequently considered

Table 4.3: Description of parameters of compared algorithms

Acronym	Algorithm's name	Parameters
AWE [84]	Accuracy Weighted Ensemble	Base Learner: Hoeffding tree, Ensemble Size: 10, Buffer size: 30
OZA [97]	Online Bagging	Base Learner: Hoeffding tree, Ensemble Size: 10
OC [100]	Online coordinate boosting	Base Learner: Hoeffding tree, Ensemble Size: 10
ADACC [123]	Anticipative and dynamic adaptation to concept changes	Base Learner: NaiveBayes, Ensemble Size: 20
AUE2 [85]	Accuracy Updated Ensemble 2	Base Learner: Hoeffding tree, Ensemble Size: 10
BLAST [116]	BLAST heterogeneous ensembles	Base Learner: Hoeffding tree, Ensemble Size: 5
BOLE [114]	Boosting-like Online Learning Ensemble	Base Learner: Hoeffding tree, Ensemble Size: 10
ARF [127]	Adaptive Random Forests	Base Learner: ARFHoeffding tree, Ensemble Size: 10
AES [125]	Adaptive Ensemble Size	Base Learner: Hoeffding tree, Ensemble Size: 10
ECPF [129]	Enhanced Concept Profiling Framework	Base Learner: Hoeffding tree, Ensemble Size: 10
KUE [94]	Kappa Updated Ensemble	Base Learner: Hoeffding tree, Ensemble Size: 10
TLP-EnAbLe	Two-Level Pruning based Ensemble with Abstained Learners	Base Learner: Hoeffding tree, Ensemble Size: 10, Similarity index: 0.95, Abstain confidence factor: 0.95

ensemble approaches which use concept of diversity and similar learners, they have been included to test performance of the proposed approach. OC, ADACC, BOLE, AES and KUE are also popular ensemble based concept drift handling algorithms. The experiments are conducted with default parameter settings as per MOA implementation for all algorithms under comparison. Base learners are constructed using Hoeffding tree [9], popularly used in classification problems and ensemble size is set to 10 [84, 85, 139].

Artificial Datasets:

Streams of data are artificially generated using MOA in which the drift positions, number of drifts and their drift patterns are incorporated to cover all possible scenarios of concept drift. For conducting fair comparison, chunk size is kept to 500 instances for all algorithms considered in the study. Table 4.4 provides summary of these artificial datasets.

1) Agrawal: This dataset, introduced by Agrawal *et al.* [149], consists of stream of nine attributes: three categorical and six numerical. A hypothetical loan application is described by this dataset. Ten defining functions generate binary class labels to determine if the loan should be approved or not. $Agrawal\_(Grdl)$ and $Agrawal\_(Abrpt)$ having gradual and abrupt drifts respectively at interval of 100k are generated. A mixed distribution $Agrawal\_(Mix)$

Table 4.4: Artificial concept drift datasets

Name	Inst	Attr	Cls	Drift Type	Drift Points	Drift value	Noise
Hyper_(Slow)	1M	15	5	Gradual		δ= 0.001	5%
Wave_(Grdl)	400k	40	3	Gradual	100k,200k,300k	width=40k	Extra 19 attr
Agrawal_(Grdl)	1M	9	2	Gradual	100k,200k,300k, 400k,500k,600k, 700k,800k,900k	θ=π/9, π/6	n=0.05
Gaussian_(Grdl)	1M	15	2	Gradual	250k,500k,750k	θ=π/6	
RTree_(Grdl+Slow)	1M	15	5	Gradual(Slow moving)	250k,500k,750k	θ=π/18, π/12,π/9	
Hyper_(Fast)	1M	15	5	Abrupt(Fast moving)		δ= 0.1	5%
Wave_(Abrpt)	400k	40	3	Abrupt	100k,200k,300k	θ=π/2	Extra 19 attr
Agrawal_(Abrpt)	1M	9	2	Abrupt	100k,200k,300k, 400k,500k,600k, 700k,800k,900k	θ=π/2	n=0.05
Gaussian_(Abrpt)	1M	15	2	Abrupt	250k,500k,750k	θ=π/2	
Mixed_(Abrpt)	500k	4	2	Abrupt	250k	θ=π/2, width=10	
Stagger_(Abrpt)	400k	3	2	Abrupt	100k,200k,300k	θ=π/2	
Sine_(Abrpt_Fast)	400k	5	2	Abrupt(Fast moving)	100k,200k,300k	θ=π/2	
RTree_(Abrpt+Fast)	1M	15	5	Abrupt(Fast moving)	200k,500k	θ=π/2	
Agrawal_(Mix)	1M	9	2	Mixed (Gradual+Abrupt)	250k,500k,750k	width=10k,1,10k	n=0.05
Wave_(Mix)	1M	40	3	Mixed (Gradual+Abrupt)	250k,500k,750k	width=10k,1,10k	Extra 19 attr
RBF_(Grdl+Recc)	1M	15	5	Gradual Recurring	125k,250k,375k,500k	θ = π/4	
RTree_(Grdl+Recc)	1M	15	5	Gradual Recurring	250k,500k,750k	θ = π/6	
RTree_(Abrpt+Recc)	1M	15	5	Abrupt Recurring	200k,400k,600k,800k	θ = π/2	
Mixed_(Abrpt+Recc)	100k	4	2	Abrupt Recurring	20k,40k,60k,80k	θ=π/2, width=100	

comprising of three drift positions at consecutive locations of 250k is also generated to show alternating abrupt and gradual drifts.

2) Waveform: It is a three class problem where 40 numerical attributes define the prediction of one of the three types of waveform. Two version of the generator: wave40 having 19 extra attributes, wave21 including 21 attributes are produced [150]. Three drifting distributions are generated in this study: $Wave\_(Grdl)$ for gradually moving four concepts, $Wave\_(Abrpt)$ for abrupt drifts, both having drift points at difference of 100k instances. Finally, $Wave\_(Mix)$ has combination of two gradual drifts and one abrupt drift.

3) Random Radial Basis Function (RBF): In this generator, instances are randomly generated by choosing a center. The RBF function comprises of a certain number of randomly positioned centroids with a single standard deviation, weight and class label [151]. In this work, one dataset $RBF\_(Grdl + Recc)$ which has gradually recurring concepts is generated.

4) RandomTree: It creates a random decision tree after choosing attributes for splitting

[152]. A random class label is given to each leaf of the tree after split. This labeling is done after traversal from tree to leaf. Two streams $RTree\_(Grdl + Recc)$, $RTree\_(Abrpt + Recc)$ are generated with recurring concepts in both gradual and abrupt fashion respectively. Two datasets $RTree\_(Grdl + Slow)$ and $RTree\_(Abrpt + Fast)$ have been created. The former has three drift points at 250k, 500k and 750k whereas the latter has two drift points at 200k and 500k each. In $RTree\_(Grdl + Slow)$ the concepts evolve gradually with a slow speed and in $RTree\_(Abrpt + Fast)$ the change in more rapid.

5) Hyperplane: This generator defines simulation of time-changing concepts which is done by altering position and orientation of hyperplane [151]. The relative size of weights w_i assigned to points x_i is changed by adding drift, d, to each weight attribute. The hyperplane is defined by Eq. (4.9). $Hyper\_(Fast)$ is fast changing stream at a rate of 0.1 and $Hyper\_(Slow)$ represents a slowly drifting stream drifting at a rate of 0.001 with every instance.

$$\sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i \tag{4.9}$$

5) Gaussian: This generator generates attributes which follow Gaussian distribution where the mean value keeps on changing to create drifts [140, 153]. Concepts are switched between two classes to produce two streams, $Gaussian\_(Grdl)$ and $Gaussian\_(Abrpt)$, having 1M instances with gradual and abrupt drifts occuring after every 250k instances.

6) Mixed: The mixed generator has two numeric attributes (a,b) and two boolean attributes (x,y) for binary classification problem. To classify as a positive class, two out of the three conditions need to be fulfilled: x, y and $b < 0.5 + 0.3\sin(3\pi * a)$. This rule gets reversed on the onset of a drift point. Drifting concepts are created by gradually selecting examples from both new and old concepts in a way that the probability of selection from the new concept is greater than that of old thus marking a new concept. In the proposed work,

Table 4.5: Results on artificial and real datasets with respect to predictive accuracy (in %)

DataSet	AWE	OZA	OC	ADACC	BLAST	KUE	BOLE	ARF	AUE2	AES	ECPF	TLP-EnAbLe
Hyper_(Slow)	65.09	89.09	84.39	59.09	82.13	91.28	87.49	81.22	91.91	89.09	64.39	92.43
Wave_(Grdl)	81.42	82.59	52.49	72.25	80.72	82.75	80.81	78.48	82.50	82.59	78.67	83.12
Agrawal_(Grdl)	81.04	88.05	89.29	74.13	86.95	93.67	92.25	82.61	93.89	88.05	72.37	94.24
Gaussian_(Grdl)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.92	100.0	100.0	100.0
RTree_(Grdl+Slow)	47.60	60.02	28.08	34.39	60.12	78.48	66.08	50.77	76.11	60.02	49.62	82.75
Hyper_(Fast)	65.08	89.09	84.08	59.08	82.13	92.08	88.27	80.74	91.88	89.09	64.39	92.42
Wave_(Abrpt)	82.91	82.59	52.82	74.54	82.14	83.63	81.65	78.72	83.04	82.59	80.14	83.76
Agrawal_(Abrpt)	79.82	88.10	89.37	77.66	86.76	93.54	92.05	82.51	93.88	88.10	78.54	94.19
Gaussian_(Fast)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Mixed_(Fast)	91.89	94.56	95.24	88.73	97.02	97.59	97.66	98.95	97.43	94.55	91.86	97.73
Stagger_(Abrpt)	99.83	99.67	99.83	99.83	99.78	100.0	99.84	99.83	99.82	99.67	99.86	99.83
Sine_(Abrpt_Fast)	95.94	93.13	96.36	93.28	98.09	98.60	98.38	98.25	98.23	93.13	91.12	98.65
RTree_(Abrpt+Fast)	53.90	81.30	19.83	36.76	79.16	87.28	79.91	52.38	87.21	81.30	56.32	89.88
Agrawal_(Mix)	87.44	92.53	91.53	79.63	91.51	93.92	92.75	90.68	94.13	92.53	86.46	94.00
Wave_(Mix)	83.31	83.86	52.88	74.48	82.62	84.20	81.98	79.74	83.36	83.86	80.11	84.07
RBF_(Grdl+Recc)	80.19	94.12	44.83	63.28	98.08	94.47	90.84	96.33	94.65	94.12	74.48	95.44
RTree_(Grdl+Recc)	47.60	60.04	28.07	34.41	60.12	77.08	66.03	50.78	76.12	60.04	49.62	83.31
RTree_(Abrpt+Recc)	52.10	62.82	25.93	37.20	63.10	80.38	70.26	46.17	78.80	62.82	54.42	80.30
Mixed_(Fast+Recc)	89.95	79.79	87.70	87.99	94.64	91.03	91.39	92.43	91.38	79.79	90.17	92.24
Poker	31.45	49.66	44.91	38.32	48.80	49.32	47.91	50.82	49.67	49.66	41.11	49.62
Electricity	63.02	78.87	76.12	70.07	76.65	78.94	55.58	79.62	78.04	78.87	53.51	78.09
Weather	68.87	73.33	70.07	71.76	77.49	74.56	64.74	77.33	73.34	73.33	67.86	74.07
Shuttle	96.82	97.19	75.28	92.51	98.79	99.42	98.18	99.38	97.44	97.19	92.44	98.78
Powersupply	14.18	14.92	3.01	6.52	14.26	15.44	4.17	12.75	15.13	14.92	14.35	14.23
BNG_lymph	85.15	85.90	54.76	81.35	86.73	85.50	86.42	88.11	86.93	85.90	85.90	86.32
BNG_wine	92.13	93.31	67.34	88.06	92.81	93.28	91.78	93.51	93.22	93.31	92.03	93.51
BNG_zoo	92.45	93.55	58.86	87.67	93.52	94.23	91.48	94.68	92.93	93.55	92.83	93.82
Covertype	77.27	79.37	70.67	59.43	84.71	84.00	52.20	80.52	83.83	79.37	56.40	84.05
Sensor	64.75	54.74	1.64	76.91	67.48	72.33	80.88	68.56	79.83	54.74	78.36	78.77
Ranks	8.6	6.2	9.5	10.1	5.8	3.3	6.3	5.8	4.3	6.3	8.9	2.8

two datasets $Mixed\_(Abrpt)$ and $Mixed\_(Abrpt+Recc)$ are considered both having abruptly changing concepts. The former has only one major drift at center of dataset with a width of change as small as only ten instances whereas the latter has repeating concepts after every 20k instances.

7) Stagger: This generator has three attributes namely color, size and shape which can give either of two class values: postive or negative. Abruptly switching concepts are created each after 100k instances using the stream $Stagger\_(Abrpt)$.

8) Sine: This dataset works by identifying the position of x and y co-ordinates on two different contexts: Sine1 and Sine2 [66]. The former classifies an instance below the curve $y = \sin(x)$ as positive and the latter requires the instances to satisfy $y < 0.5 + 0.3\sin(3\pi x)$.

Table 4.6: Results on artificial and real datasets with respect to kappa statistic (in %)

DataSet	AWE	OZA	OC	ADACC	BLAST	KUE	BOLE	ARF	AUE2	AES	ECPF	TLP-EnAbLe
Hyper_(Slow)	27.13	77.44	67.84	14.05	63.19	81.96	74.30	60.70	83.28	77.44	24.98	84.40
Wave_(Grdl)	72.14	73.88	28.64	58.37	71.08	74.14	71.23	67.72	73.75	73.88	68.00	74.69
Agrawal_(Grdl)	62.07	76.10	78.58	48.25	73.90	87.35	84.50	65.23	87.77	76.10	44.74	88.49
Gaussian_(Grdl)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RTree_(Grdl+Slow)	30.03	46.66	5.01	12.42	46.80	71.28	54.73	34.26	68.11	46.66	32.68	76.98
Hyper_(Fast)	27.12	77.43	67.19	14.08	63.19	83.64	75.89	59.58	83.18	77.43	24.99	84.36
Wave_(Abrpt)	74.37	73.89	29.13	61.82	73.21	75.45	72.48	68.08	74.57	73.89	70.21	75.65
Agrawal_(Abrpt)	59.64	76.21	78.74	55.32	73.52	87.08	84.09	65.02	87.76	76.21	57.08	88.38
Gaussian_(Fast)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mixed_(Fast)	83.79	89.11	90.49	77.45	94.04	95.18	95.33	97.90	94.87	89.11	83.72	95.47
Stagger_(Abrpt)	99.66	99.34	99.66	99.66	99.57	100.0	99.69	99.66	99.64	99.34	99.72	99.66
Sine_(Abrpt_Fast)	91.87	86.26	92.72	86.56	96.18	97.20	96.75	96.51	96.58	86.26	82.23	97.30
RTree_(Abrpt+Fast)	38.32	75.56	6.71	16.85	72.76	83.38	73.67	34.57	83.30	75.56	41.46	86.77
Agrawal_(Mix)	74.87	85.07	83.06	59.27	83.03	87.84	85.51	81.36	88.25	85.07	72.93	88.01
Wave_(Mix)	74.98	75.79	29.24	61.73	73.92	76.30	72.98	69.61	75.05	75.79	70.18	76.11
RBF_(Grdl+Recc)	73.46	92.12	27.60	50.57	97.43	92.52	87.70	95.08	92.83	92.12	65.71	93.90
RTree_(Grdl+Recc)	30.02	46.69	5.01	12.46	46.80	69.41	54.66	34.26	68.14	46.69	32.68	77.72
RTree_(Abrpt+Recc)	35.54	49.98	4.83	15.62	50.38	73.60	59.95	27.17	82.14	49.98	38.60	73.48
Mixed_(Fast+Recc)	79.90	59.58	75.40	75.98	89.28	82.06	82.78	84.87	82.76	59.58	80.35	84.48
Poker	0.03	0.00	0.06	−0.13	0.96	−0.21	−0.59	4.51	0.02	0.00	0.37	0.02
Electricity	27.47	55.32	49.75	40.37	51.26	55.69	10.63	56.91	53.74	55.32	12.95	53.90
Weather	31.72	35.39	26.23	32.84	39.93	27.42	24.49	39.93	31.73	35.39	27.64	33.42
Shuttle	90.72	92.09	38.74	76.58	96.60	96.38	94.82	98.26	92.85	92.09	79.68	96.52
Powersupply	10.45	11.22	−1.21	2.46	10.53	11.76	0.00	8.96	11.43	11.22	10.63	10.50
BNG_lymph	72.64	74.02	3.26	65.07	75.04	74.63	74.95	77.40	75.79	74.02	74.07	74.73
BNG_wine	88.06	89.84	47.73	81.85	89.09	89.79	87.52	90.14	89.71	89.84	87.90	90.15
BNG_zoo	90.17	91.57	43.21	83.99	91.54	92.45	88.86	93.04	90.77	91.57	90.67	91.92
Covertype	60.45	63.81	43.74	27.97	73.71	72.51	16.87	65.21	72.23	63.81	22.90	72.44
Sensor	64.01	53.77	0.24	76.42	66.77	71.73	80.46	67.88	79.39	53.77	77.89	78.31
Ranks	8.4	6.5	9.5	9.9	5.7	3.5	6.2	5.8	4.3	6.5	8.6	3.1

Both attributes denoted by x and y are uniformly distributed between the interval $[0,1]$. The drift is simulated by interchanging the requirements between the two curves Sine1 and Sine2. This dataset uses five attributes to generate a fast moving stream $Sine\_(Abrpt)$ with three abrupt drift points at 100k instances each. The concepts change very rapidly at drift point.

***Real Datasets* :**

The real datasets used in experimentation to evaluate the performance are:

1) Electricity: The New South Wales Electricity Market [112] in Australia has generated this data comprising of eight attributes which predict the changes in the prices of the electricity. The dataset is broadly classified into 2 classes: up and down.

2) Poker: It is a popular machine learning dataset having ten predictive attributes (5

Table 4.7: Real Datasets

Name	Instances	Attr	Cls	Drift Pt.
Poker	100k	10	10	Unknown
Electricity	45,312	8	2	Unknown
Weather	100k	8	2	Unknown
Shuttle	58,000	9	7	Unknown
Powersupply	29,928	3	24	Unknown
BNG_lymph	100k	19	4	Unknown
BNG_wine	1M	14	3	Unknown
BNG_zoo	1M	17	7	Unknown
Covertype	5,81,012	54	7	Unknown
Sensor	1,40,000	5	54	Unknown

cards * 2 attributes- suit and rank) [139]. The ten attributes have five ordinal features and five numerical features for every instance.

Other datasets namely Weather, Covertype (described in Section 3.3.1), Shuttle, Powersupply, BNG_lymph, BNG_wine, BNG_zoo, Covertype and Sensor are also popular real datasets which are used in the study for experimentation. These are described in Table 4.7 along with their characteristics.

4.3.1 Experiments on Artificial Datasets

Table 4.5 and Table 4.6 show the results for the classification predictive accuracy and kappa static obtained for all the datasets. It clearly depicts that the diversity component and conditional abstaining introduced in the proposed approach is effective and can adapt to drifting distributions. Table 4.8 provides the evaluation time taken by all these approaches.

Gradual Drifts

Four artificial streams have been considered for analysis of slowly changing concepts. Figure 4.2(a) and Figure 4.2(b) show performance of all approaches on Hyperplane and Wave datasets respectively. For *Hyperplane* stream, proposed approach maintains high ac-

Table 4.8: Results on artificial and real datasets with respect to evaluation time (in centiseconds)

DataSet	AWE	OZA	OC	ADACC	BLAST	KUE	BOLE	ARF	AUE2	AES	ECPF	TLP-EnAbLe
Hyper_(Slow)	4.83	1.15	2.95	2.82	6.28	1.58	2.23	8.93	1.84	0.99	0.11	2.69
Wave_(Grdl)	24.96	5.65	11.58	19.38	17.31	7.77	31.85	11.62	12.48	5.34	1.14	17.08
Agrawal_(Grdl)	4.00	0.69	1.75	2.80	5.34	1.17	1.82	5.28	1.68	0.71	0.10	1.90
Gaussian_(Grdl)	3.82	0.45	1.28	4.45	8.63	0.88	1.47	0.44	1.54	0.58	0.09	1.52
RTree_(Grdl+Slow)	5.43	1.51	3.85	3.10	5.51	2.61	4.52	6.18	2.61	1.53	0.21	3.76
Hyper_(Fast)	4.79	1.11	2.94	2.83	6.27	1.60	2.11	9.03	1.85	1.18	0.12	2.65
Wave_(Abrpt)	24.79	5.61	11.74	19.22	17.61	7.30	34.03	11.56	12.09	4.55	1.02	17.24
Agrawal_(Abrpt)	3.78	1.05	2.55	2.90	5.57	1.14	1.72	8.02	1.22	0.98	0.17	1.80
Gaussian_(Fast)	3.84	0.45	1.31	4.97	8.58	1.02	1.49	0.45	1.57	0.61	0.09	1.53
Mixed_(Fast)	2.24	0.24	0.98	1.92	4.20	0.61	0.97	0.96	0.75	0.36	0.05	1.25
Stagger_(Abrpt)	0.78	0.15	1.01	0.72	2.57	0.00	0.22	0.38	0.19	0.14	0.01	0.24
Sine_(Abrpt_Fast)	1.69	0.36	1.00	1.34	2.32	0.63	0.66	1.62	0.62	0.24	0.06	0.78
RTree_(Abrpt+Fast)	6.57	1.59	4.00	3.57	5.64	1.72	4.71	5.78	3.70	1.69	0.18	3.03
Agrawal_(Mix)	3.92	0.90	2.48	3.06	5.71	1.05	1.76	6.43	1.24	0.64	0.14	1.77
Wave_(Mix)	25.14	5.81	11.58	21.09	17.68	7.51	31.21	11.88	11.82	4.79	1.02	16.90
RBF_(Grdl+Recc)	15.79	3.45	8.33	24.91	8.43	4.24	11.19	4.00	7.58	3.47	0.75	8.26
RTree_(Grdl+Recc)	5.42	1.48	3.82	3.07	5.52	1.94	4.46	6.24	2.61	1.31	0.22	3.72
RTree_(Abrpt+Recc)	5.39	1.48	3.91	3.13	5.54	1.91	4.58	6.19	2.65	1.25	0.27	3.79
Mixed_(Fast+Recc)	2.18	0.31	1.44	1.78	4.44	0.79	1.75	3.38	0.85	0.38	0.05	1.40
Poker	8.85	3.78	6.12	9.51	46.48	2.86	11.33	12.50	6.25	2.99	1.17	4.43
Electricity	10.80	1.99	6.39	21.73	30.33	8.52	55.68	25.36	10.51	3.05	1.07	10.01
Weather	11.03	2.94	6.80	14.34	33.09	6.43	50.37	28.49	10.11	2.94	2.02	10.48
Shuttle	22.32	3.18	10.83	24.72	34.49	5.52	21.37	5.36	14.40	5.08	1.62	10.49
Powersupply	30.25	5.47	9.82	25.56	18.86	12.05	23.10	23.88	29.46	7.14	2.12	25.78
BNG_lymph	19.92	4.36	11.68	14.71	41.64	8.74	22.41	9.44	10.35	3.38	0.57	19.95
BNG_wine	42.80	6.53	11.26	41.54	32.12	10.07	65.04	21.12	24.55	7.80	0.79	19.49
BNG_zoo	33.61	4.55	10.73	44.50	32.33	8.56	32.68	11.88	21.36	7.00	0.75	19.31
Covertype	59.36	8.51	17.53	156.44	50.77	22.05	159.45	27.20	40.32	11.45	3.22	39.46
Sensor	224.82	20.35	47.05	341.46	30.01	52.18	123.29	19.74	114.34	22.10	12.61	165.83
Ranks	10.2	2.6	6.7	9.2	10.4	4.2	8.9	8.2	6.4	2.7	1.0	7.3

curacy level across all the instances. As the concept changes, slowly with magnitude of 0.001; ECPF, AWE and ADACC maintain constant accuracy. TLP-EnAbLe, AUE2, KUE result in prolonged accuracy adaptation. However, after a stable initial increase these three algorithms give high accuracy with TLP-EnAbLe being the best performer. In *Wave* stream, TLP-EnAbLe gives a relatively higher performance in comparison to majority of algorithms but KUE outperforms all the others. Addition of diversity based pruning to weight based pruning helps TLP-EnAbLe to perform better than competitive algorithms like AUE2, ARF, BLAST *etc*. It is seen that at 200k instance drift point ADACC, ECPF, ARF and AES show a significant drop in performance and struggle to achieve consistent accuracy after these drifts.

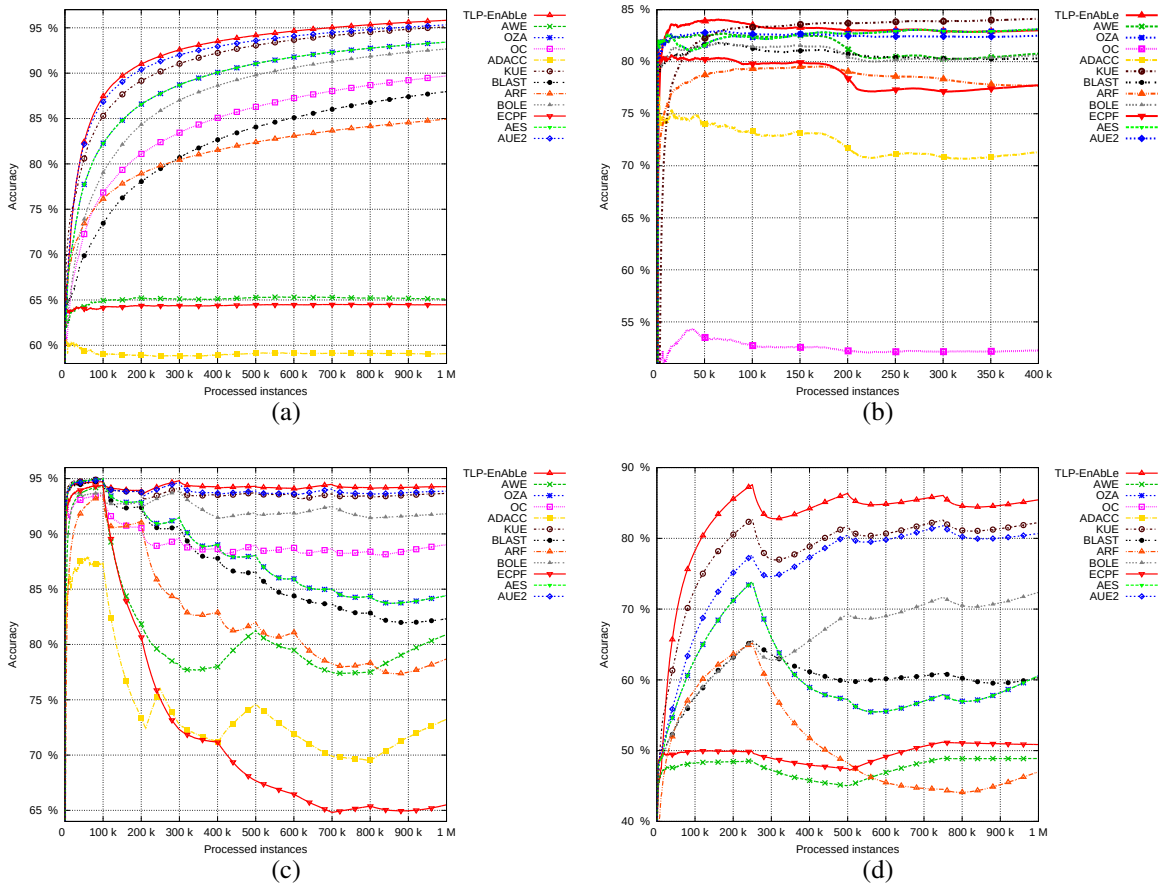


Figure 4.2: Predictive accuracy on gradual drifts a) Hyperplane b) Wave c) Agrawal d) Random Tree

Agrawal_(Grdl) is shown in Figure 4.2(c) is a plot of consecutive gradual drifts, each at a distance of 100k instances. Approaches like ECPF, ADACC, AWE have shown a significant drop in accuracy at all drift points. ARF which has diverse component configuration, fails to cope up with multiple drifts. Only TLP-EnAbLe, AUE2, KUE have adapted themselves to this challenging drifting scenario with TLP-EnAbLe being the best performer. Dual level pruning of learners can be attributed the reason behind such adaptability. Figure 4.2(d) depicts the case of slowly changing concepts for RandomTree generator. This dataset comprises of four concepts changing each at a distance of 250k instances. TLP-EnAbLe witnesses highest overall accuracy in all the instances. KUE and AUE2 show nearly similar pattern of adaptablity as they both exhibit the property of adding only one learner after each

chunk. However, AES, OZA, and even ARF do not perform well.

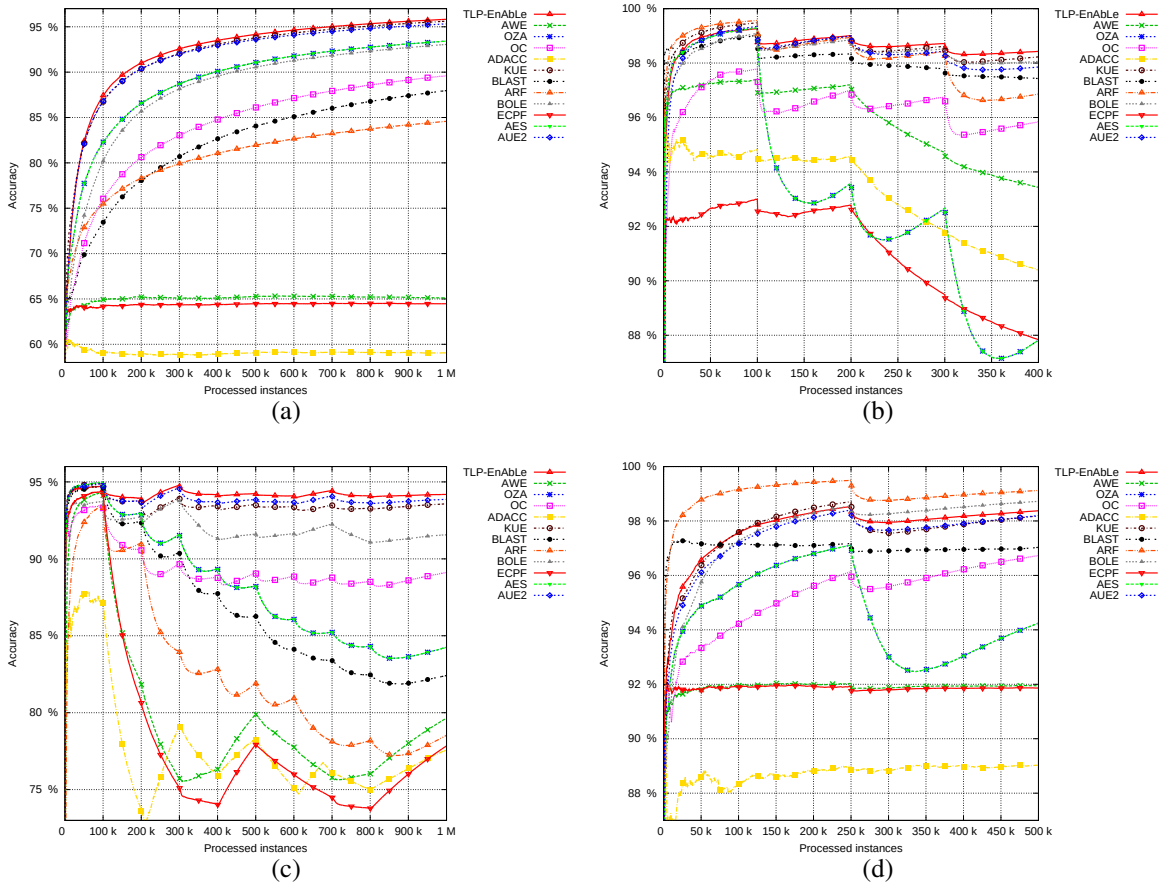


Figure 4.3: Predictive accuracy on abrupt drifts a) Hyperplane b) Sine c) Agrawal d) Mixed

Abrupt Drifts

This section discusses performance of algorithms on some of the datasets which exhibit abrupt drifts. Figure 4.3(a) and Figure 4.3(b) show the scenario of suddenly shifting concepts in two generators *Hyperplane* and *Sine*. In *Sine*, as it can be seen in Figure 4.3(b) TLP-EnAbLe gives better performance than all; is closely followed by KUE and AUE2. However, *Sine* stream shows a very interesting plot where sudden drop in accuracy levels is seen at the three drift points: 100k, 200k and 300k. ECPF, ADACC, OC, AES and OZA show major drops in accuracy at all drift points this leading to an overall lower average accuracy. As

compared to TLP-EnAbLe and KUE, ARF is unable to adapt itself to the abruptly changing streams indicating that its diversification is insufficient to cope up with such scenario. TLP-EnAbLe maintains higher accuracy and shows no major drop at drift points.

In Figure 4.3(c) a scenario of multiple drifts with short gap of merely 100k instances is depicted. With majority of algorithms showing continous decrease in performance only three approaches TLP-EnAbLe, KUE and AUE2 seem to adapt well. The high performing learners in these can be the reason behind their performance. Though KUE did not adapt well to initial two drifts but it shows stabilty later. Online boosting relatively performed better in this dataset than in others. Figure 4.3(d) depicts a case of simple concept drift with just one major drift at 250k instances. The *Mixed* generator emphasizes the point of change. ARF stood out than others in terms of overall accuracy. Usually good performing algorithms TLP-EnAbLe and KUE give slight drop in performance at the drift point. However, TLP-EnAbLe has reduced the error in classification in very less instances. Both BOLE and TLP-EnAbLe witness a faster recovery as compared to the algorithm KUE.

Recurring Drifts

Figure 4.4(a), represents repeating concepts with gradual drifts at 250k, 500k and 750k instances for *RandomTree* generator. TLP-EnAbLe, KUE and AUE2 handle these changes appropriately, however AWE, ECPF, OZA, OC, ADACC fail to adapt themselves to repeating concepts. Accuracy level achieved by TLP-EnAbLe is significantly more than others. The proposed approach witnessed a drop in performance at the drift points but soon it manages to recover resulting into overall highest average accuracy. Figure 4.4(b) represents a plot over five different concepts distinguished by four drift points at a distance of 200k instances each. Since TLP-EnAbLe does not remove any learner till next iteration, it can handle recurring concepts quite efficiently. Along with maintenance of accuracy pattern of KUE and AUE2,

TLP-EnAbLe has also improved it to great extent, as seen in the plot. On the other hand, due to lack of continuous pruning strategy, ARF shows decreasing accuracy with growing instances. Approaches like OC, AWE, ADACC ECPF show fall in accuracy levels.

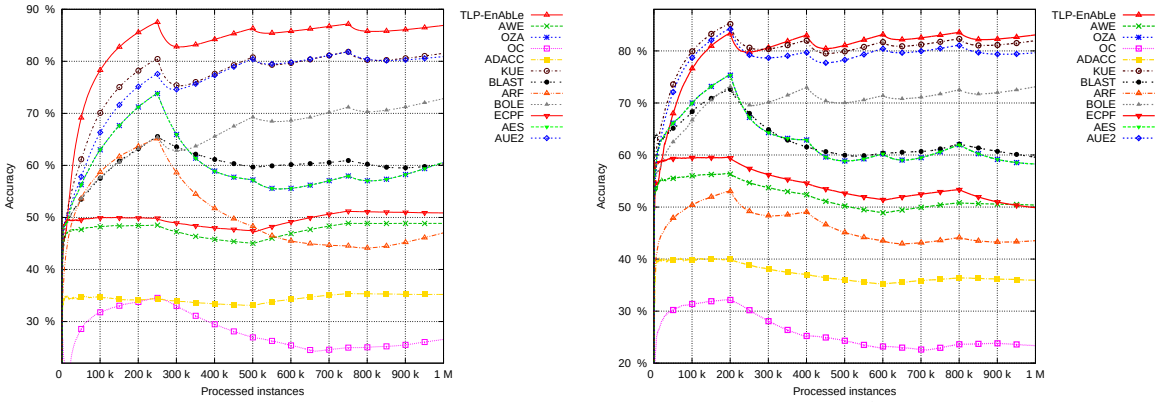
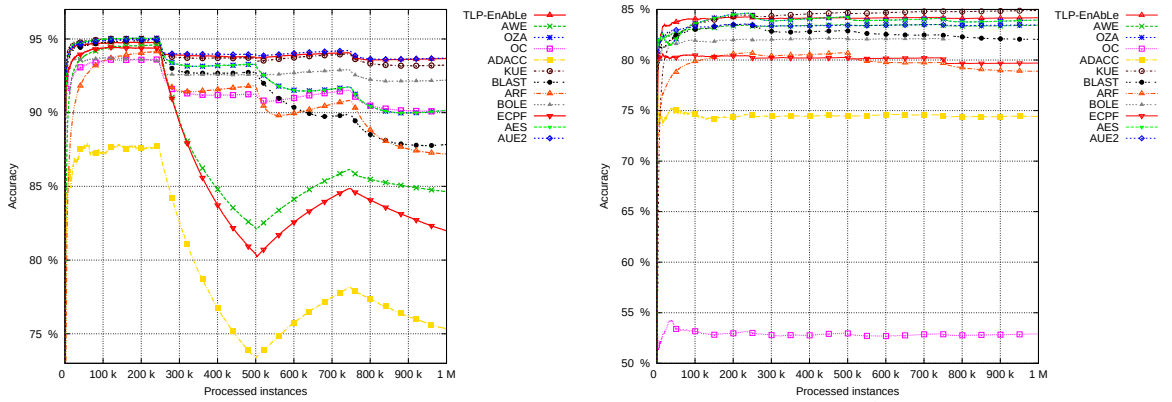


Figure 4.4: Predictive accuracy on recurring drifts a) Random Tree (Gradual) b) Random Tree (Abrupt)

Combination of Drifts

While considering gradual and abrupt drift patterns, there is a clear indication that the strategy using diversity based pruning along with accuracy is more reasonable than the ones based on accuracy like AUE2 and AWE. It is important to evaluate the approaches on combination of both types of drifts. In Figure 4.5(a) *Agrawal* generator depicts two gradual and one abrupt drift at alternating sequence of 250k instances each. Performance of TLP-EnAbLe, KUE and AUE2 is similar to the accuracy of KUE, dropping at the last drift point. ADACC and ECPF have sharp drops at all three drift points whereas AWE and BLAST try to adapt after second drift point. In *Wave* stream, Figure 4.5(b), TLP-EnAbLe tends to give stable accuracy throughout. AWE shows quite a stable performance; far better than ARF. Moreover, not much drop is witnessed at drift points. A faster recovery is noticed by selecting appropriate learners. At 500k instance, at the point of abrupt drift, OC's performance gives

least accuracy.



(a) (b)

Figure 4.5: Predictive accuracy on combination of drifts a) Agrawal b) Wave.

4.3.2 Experiments on Real Datasets

Figure 4.6 depicts the performance of various algorithms on four major real datasets. In *Poker* (Figure 4.6(a)) ARF outperforms all others. Though TLP-EnAbLe, KUE and AUE2 are on the similar level, AWE is the lesser performing one. Resuse of similar learners in ECPF does not show much advantage. Anticipating future concepts has not helped ADACC much in recovering from unknown drifts in this dataset. In *Electricity* and *Weather* datasets depicted in Figure 4.6(b) and Figure 4.6(c) respectively, ARF is the best performer. The diverse diversification due to random sampling is one of the major reasons behind this pattern. However, in both cases the proposed approach shows quite stable predictive accuracy levels. There is a significant improvement as compared to other ensemble approaches like AWE, BOLE and ECPF in all the four datasets discussed. In case of *Electrcity* ADACC shows sharp decrease in performance at 20k instances. On Sensor dataset, depicted in Figure 4.6(d), BOLE has shown best performance. Since BOLE permits only selected learners which exceed the threshold level to participate in voting, this leaves only best performing ones for decision making. Diversification offered by KUE does not prove to be that useful

in managing drifts in this dataset. However, TLP-EnAbLe and AUE2 are just behind BOLE in achieving stability. Thus, on other real datasets used in experimentation, the proposed approach has good adaptability leading to higher accuracy levels.

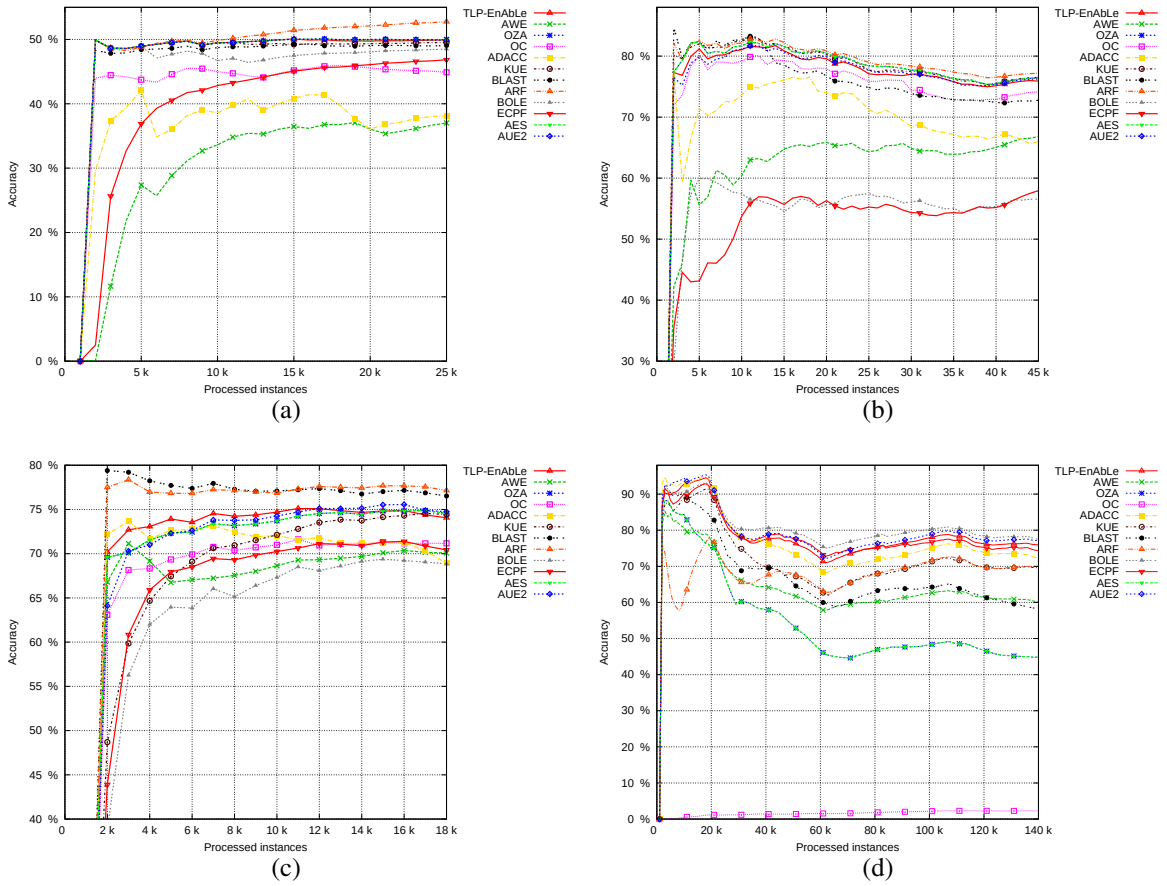


Figure 4.6: Predictive accuracy on real datasets a) Poker b) Electricity c) Weather d) Sensor

4.3.3 Statistical Evaluation

Statistical tests using non-parametric Friedman test are performed to rank the performance of all the algorithms. The null hypothesis states that there exists no significant difference between the compared algorithms. The F_f static value is calculated separately for classification accuracy and kappa statistic. In this work, the significance value considered is 0.05 (N=29, k=12). For accuracy, the F_f value is 22.9 which is higher than the critical

value 1.82, thus rejecting the null hypothesis. The Nemenyi test [154] post-hoc analysis computes the critical difference value = 3.06. It is used for pair-wise comparison of the algorithms. Table 4.5 depicts the statistical ranks obtained over the datasets. It is evident that TLP-EnAbLe is significantly better than AWE (8.6-2.8 = 5.8 $\geq$ 3.06), OZA(6.2-2.8 = 3.4 $\geq$ 3.06), OC(9.5-2.8 = 6.7 $\geq$ 3.06), ADACC(10.1-2.8 = 7.2 $\geq$ 3.06), BOLE(6.3-2.8 = 3.5 $\geq$ 3.06), AES(6.3-2.8 = 3.4 $\geq$ 3.06) and ECPF (8.9-2.8 = 6.1 $\geq$ 3.5) in terms of classification accuracy.

For kappa statistic, Table 4.6 provides the statistical ranks and TLP-EnAbLe has lowest rank which indicates that it provides more generalized learners as compared to others. Statistical tests applied on kappa with F_f static = 18.8 showed significant difference at α_{sig}= 0.05 (18.8 $\geq$ 1.8). Applying the post-hoc analysis shows that TLP-EnAbLe is significantly better than AWE (8.4-3.1 = 5.4 $\geq$ 3.06), OZA(6.5-3.1 = 3.4 $\geq$ 3.06), OC(9.5-3.1 = 6.5$\geq$ 3.06), ADACC(9.9-3.1 = 6.8 $\geq$ 3.06), BOLE(6.2-3.1 = 3.2 $\geq$ 3.06), AES(6.5-3.1 = 3.4 $\geq$ 3.06) and ECPF (8.6-3.1 = 5.6 $\geq$ 3.06).

4.3.4 Analysis of Diversity Measures

The objective of this analysis is to study the influence of diversity measure on the overall selection of learners for pruning. To accomplish this, the proposed approach is tested with three diversity measures Q-static *(Q)*, double fault *(DF)* and disagreement *(DS)*. The results obtained from the three are compared to identify which of them fits the best for a particular dataset and overall. This helps to analyse the extent to which the proposed approach is stable under heterogeneous settings. Table 4.9 presents the variation in results with all three measures with ranks and Disagreement *(DS)* gives best accuracy for majority of the datasets.

Table 4.9: Predictive accuracy (%) under various diversity measures

Datsets	Q-static (Q)	Double fault (DF)	Disagreement (DS)
Hyper_(Slow)	92.22	92.37	91.88
Wave_(Grdl)	82.99	82.85	82.75
Agrawal_(Grdl)	93.79	94.08	94.28
Gaussian_(Grdl)	100.0	100.0	100.0
RTree_(Grdl+Slow)	82.53	82.55	83.03
Hyper_(Fast)	92.21	92.36	91.87
Wave_(Abrpt)	83.66	83.67	83.63
Agrawal_(Abrpt)	93.85	94.02	94.22
Gaussian_(Fast)	100.0	100.0	100.0
Mixed_(Fast)	98.00	97.86	98.00
Stagger_(Abrpt)	99.83	99.83	99.83
Sine_(Abrpt_Fast)	98.71	98.66	98.71
RTree_(Abrpt+Fast)	89.70	89.40	89.71
Agrawal_(Mix)	93.56	93.73	93.94
Wave_(Mix)	83.98	83.96	83.89
RBF_(Grdl+Recc)	95.47	95.49	95.41
RTree_(Grdl+Recc)	83.08	82.65	83.25
RTree_(Abrpt+Recc)	79.31	79.77	81.39
Mixed_(Fast+Recc)	92.41	92.26	92.28
Poker	49.71	49.26	49.69
Electricity	76.31	76.14	76.18
Weather	73.93	74.20	73.59
Shuttle	98.78	98.76	98.79
Powersupply	13.97	14.13	13.64
BNG_lymph	86.20	86.09	86.73
BNG_wine	93.44	93.14	93.25
BNG_zoo	93.84	94.00	94.12
Covertype	84.06	84.18	84.05
Sensor	79.16	79.29	79.67
Ranks	2.00	2.10	1.80

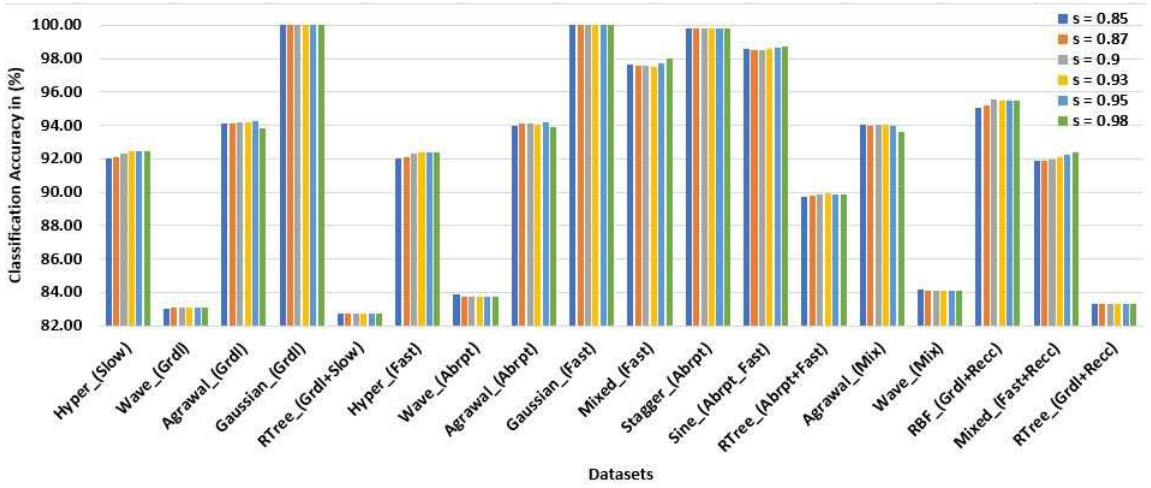
4.3.5 Parametric Analysis

In the proposed approach two parameters are considered. First is the similarity index of the learners s and second one is the abstain confidence factor α_c. As both of them play a significant role in the performance of the approach, it is important to analyze the variation in the performance of proposed approach under their different settings.

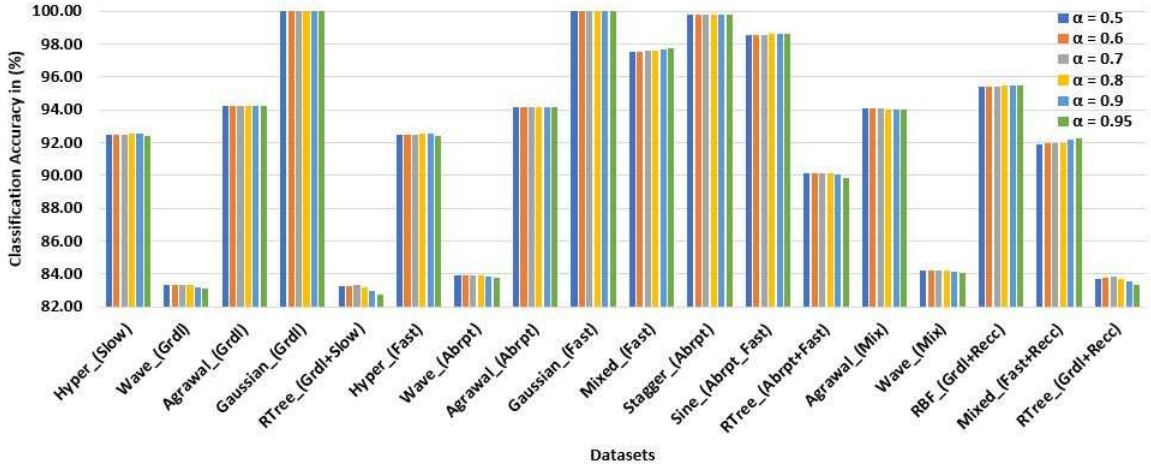
In case of real world problems, the difficult part may be to set their optimum value as it may change with the underlying dataset or time. However, s and α_c are analyzed by varying their values and observing their effect on prediction accuracy, as seen in Figure 4.7(a) and Figure 4.7(b) respectively. The values of s and α_c are randomly chosen in the range [0,1]. The values of s are varied keeping α_c fixed at 0.95 and the results are shown in Table 4.10

Table 4.10: Effect of s on predictive accuracy (%) $\alpha_c = 0.95$ (Fixed)

Dataset	s = 0.85	s = 0.87	s = 0.90	s = 0.93	s = 0.95	s = 0.98
Hyper_(Slow)	92.03	92.08	92.34	92.43	92.44	92.43
Wave_(Grdl)	83.00	83.12	83.12	83.12	83.13	83.12
Agrawal_(Grdl)	94.09	94.09	94.16	94.17	94.24	93.83
Gaussian_(Grdl)	100.0	100.0	100.0	100.0	100.0	100.0
RTree_(Grdl+Slow)	82.75	82.75	82.75	82.75	82.77	82.75
Hyper_(Fast)	92.02	92.07	92.33	92.42	92.42	92.42
Wave_(Abrpt)	83.86	83.76	83.76	83.76	83.76	83.76
Agrawal_(Abrpt)	93.98	94.09	94.11	94.02	94.19	93.87
Gaussian_(Fast)	100.0	100.0	100.0	100.0	100.0	100.0
Mixed_(Fast)	97.62	97.60	97.59	97.50	97.73	98.00
Stagger_(Abrpt)	99.83	99.83	99.83	99.83	99.85	99.83
Sine_(Abrpt_Fast)	98.58	98.51	98.51	98.56	98.65	98.69
RTree_(Abrpt+Fast)	89.71	89.77	89.84	89.91	89.88	89.88
Agrawal_(Mix)	94.02	93.99	94.02	94.01	94.00	93.63
Wave_(Mix)	84.14	84.07	84.07	84.07	84.07	84.07
RBF_(Grdl+Recc)	95.04	95.21	95.57	95.49	95.49	95.49
RTree_(Grdl+Recc)	83.31	83.31	83.31	83.32	83.31	83.31
RTree_(Abrpt+Recc)	79.87	80.30	80.30	80.30	80.30	80.31
Mixed_(Fast+Recc)	91.91	91.91	91.98	92.08	92.24	92.37
Poker	49.35	49.35	49.34	49.34	49.62	49.72
Electricity	76.05	76.21	76.65	76.35	76.35	76.35
Weather	74.07	74.09	74.07	74.07	74.07	74.07
Shuttle	98.80	98.80	98.80	98.76	98.78	98.78
Powersupply	14.42	14.42	14.42	14.08	14.23	13.98
BNG_lymph	85.09	86.55	86.35	86.32	86.32	86.32
BNG_wine	93.28	93.12	93.19	93.45	93.51	93.51
BNG_zoo	93.57	93.57	93.74	93.64	93.82	93.80
Covertype	83.69	83.91	84.05	84.05	84.06	84.05
Sensor	78.19	78.23	78.49	78.24	78.77	78.83
Ranks	4.2	4.1	3.3	3.5	2.8	3.1



(a)



(b)

Figure 4.7: Effect of parameters a) similarity index s b) abstain confidence factor α_c over predictive accuracy on artificial datasets.

which clearly indicate that with the increase in the value of s, accuracy tends to increase but it tends to converge for values $s \geq 0.9$. By increasing value of s the ranks decreased from 4.2 to 3.1. Lower rank depicts higher classification performance.

In another set of experiments, α_c is analyzed under different random values keeping s static at 0.95. This value is chosen for subsequent experiments since it achieved highest accuracy for proposed approach. The results of predictive accuracy are shown in Table 4.11. Statistical tests are performed to depict the overall results over all the datasets for both of the

above two experiments. Though average ranks proved to be best at $\alpha_c = 0.8$, there is no fixed pattern of results with increase or decrease in value of α_c.

Table 4.11: Effect of α_c on predictive accuracy (%) $s = 0.95$ (Fixed)

Datsets	$\alpha_c = 0.5$	$\alpha_c = 0.6$	$\alpha_c = 0.7$	$\alpha_c = 0.8$	$\alpha_c = 0.9$	$\alpha_c = 0.95$
Hyper_(Slow)	92.46	92.46	92.50	92.59	92.54	92.43
Wave_(Grdl)	83.32	83.32	83.33	83.33	83.22	83.12
Agrawal_(Grdl)	94.25	94.24	94.24	94.23	94.24	94.24
Gaussian_(Grdl)	100.0	100.0	100.0	100.0	100.0	100.0
RTree_(Grdl+Slow)	83.23	83.29	83.34	83.21	82.97	82.75
Hyper_(Fast)	92.45	92.45	92.49	92.58	92.53	92.42
Wave_(Abrpt)	83.89	83.90	83.89	83.89	83.83	83.76
Agrawal_(Abrpt)	94.20	94.19	94.19	94.18	94.19	94.19
Gaussian_(Fast)	100.0	100.0	100.0	100.0	100.0	100.0
Mixed_(Fast)	97.54	97.55	97.58	97.64	97.69	97.73
Stagger_(Abrpt)	99.83	99.83	99.83	99.84	99.83	99.83
Sine_(Abrpt_Fast)	98.58	98.59	98.59	98.61	98.65	98.65
RTree_(Abrpt+Fast)	90.15	90.15	90.13	90.11	90.07	89.88
Agrawal_(Mix)	94.10	94.10	94.08	94.05	94.02	94.00
Wave_(Mix)	84.23	84.24	84.24	84.23	84.15	84.07
RBF_(Grdl+Recc)	95.42	95.43	95.44	95.46	95.48	95.49
RTree_(Grdl+Recc)	83.71	83.77	83.83	83.73	83.51	83.31
RTree_(Abrpt+Recc)	82.88	82.85	82.69	82.27	81.32	80.14
Mixed_(Fast+Recc)	91.93	91.95	91.98	92.00	92.16	92.24
Poker	49.62	49.59	49.61	49.62	49.63	49.62
Electricity	76.71	76.72	76.70	76.71	76.60	76.35
Weather	74.46	74.47	74.53	74.60	74.34	74.07
Shuttle	98.78	98.78	98.78	98.80	98.78	98.78
Powersupply	14.21	14.21	14.22	14.27	14.25	14.23
BNG_lymph	86.36	86.39	86.43	86.55	86.44	86.32
BNG_wine	93.60	93.60	93.60	93.59	93.56	93.51
BNG_zoo	93.45	93.50	93.54	93.58	93.63	93.82
Covertype	84.05	84.04	84.02	84.03	84.07	84.05
Sensor	77.52	77.68	77.85	78.07	78.43	78.76
Ranks	3.7	3.4	3.3	3.1	3.4	4.1

4.4 Discussion

TLP-EnAble has the ability to adapt to different types of drifts, such as abrupt, gradual, recurring and even their complex combination.

- By adding diversity based pruning to accuracy based one, the proposed approach provides reliable selection of best suitable learners for classification system.

- Experimental evaluation demonstrates that TLP-EnAbLe is significantly better than other state-of-the-art concept drift algorithms in terms of classification accuracy. The results which have been validated under various parametric settings show the robustness of the proposed approach.

- The novel two-level abstaining is quite efficient while selecting ensemble learners in decision making.

- Use of two-level pruning has proved to be more advantageous as compared to only accuracy based pruning for both artificial and real data streams.

Coming chapter (Chapter 5) discusses a dual ensemble based approach to deal with drifting distributions of various patterns with comprehensive experiment analysis.

Chapter 5

Dynamically Adaptive and Diverse Dual Ensemble (DA-DDE) Learning Approach for Concept Drift

In this chapter a Dynamically Adaptive and Diverse Dual Ensemble (DA-DDE) learning approach has been proposed for drifting data streams. It uses a hybrid framework of two types of ensembles which have specific concept drift handling mechanism for abrupt and gradual type of drifts. In this chapter apart from the artificially generated drifting streams and real datasets, two potential applications namely Electricity pricing prediction and spam filtering are taken up to demonstrate the effectiveness of the proposed approach.

5.1 Introduction

The changes in data distribution can be gradual, abrupt, reoccurring or a combination of these. In many streaming applications various types of drifts co-exist in the incoming data. Hence, it is challenging to deal with different types of drifts together. Many algorithms have

been proposed in literature to deal with a specific type of drift [139, 155, 156]. The proposed approach focusses on providing a common platform to handle multiple types of drifts.

As discussed in Section 2.2 diversity among the constituent learners of ensemble approaches is crucial while solving different types of concept drifts. Algorithms which work only with explicit drift detection capability lack in leveraging the significance of diverse ability of ensembles [65]. There is a need to build integrated approaches which can utilize the diversity of ensembles for improving their prediction capability.

Further, in ensembles, there is a scope of improving the traditional weighing mechanisms of component learners which consider their performance on the current block of instances only [84, 85]. These approaches tend to ignore the historic predictive performance of the learners.

DA-DDE uses a hybrid model of dual ensembles namely active and passive, where each ensemble is specifically trained to handle a particular type of data pattern. DA-DDE has following characteristics:

1. The proposed approach introduces a dual diversified update ensemble framework which leverages the characteristics of both online and block-based ensembles. By combining active and passive ensemble models, the technique offers an advantage over existing approaches by equipping with a specific strategy to handle both abrupt and gradual drifts respectively.

2. The technique proposes a novel Dynamic Dual Selective Voting Mechanism (DDSVM) for hypothesis generation which considers the prediction capability of both the underlying ensembles. Subsequently, it dynamically selects the best ensemble based on the correctness of previous chunk in real time.

3. The proposed novel weight setting is a proportionate exponential function which is

Table 5.1: Summary of main notations

Symbol	Description	Symbol	Description
D_i	Old distribution	D_{i+t}	New distribution
E_{psv}	Passively Updated Ensemble	E_{actv}	Actively Updated Ensemble
n_{actv}	Number of experts in active ensemble	n_{psv}	Number of experts in passive ensemble
λ_{actv}	Diversity level of active ensemble	λ_{psv}	Diversity level of passive ensemble
$\hat{S}$	Data stream of instances	β	decaying factor
ξ_{boost}	Boosting factor	t_0	Old timestamp
t_1	Current timestamp	$\hat{x_i}$	Current instance
$P_t(X, y_i)$	Joint probability at time t	$[Current_c]$	Current chunk
$[Warning_c]$	Warning chunk	n_c	size of current chunk
n_w	size of warning chunk	p_{del}	delaying factor
α^{thresh}	Absolute credibility threshold	Ψ^{thresh}	Relative credibility threshold
f_k	Output classifying function	$\hat{H}(x_t^i)$	Final hypothesis function
MSE_i^j	Mean square error of j^{th} expert	MSE_r	Mean square error of randomly predicting expert
λ	Poisson distribution value	$p(y)$	Distribution of class y
P_{psv}	Prediction count of passive ensemble	P_{actv}	Prediction count of active ensemble

based on the learner's performance on the latest data chunk along with its historic accuracy. This improves the ensemble adaptability to both gradual and recurrent drifts.

4. Compared to existing approaches, DA-DDE provides better generalization while handling various kinds of drifts. This is done by the use of diversity control mechanism and combination of mutually complementary classifiers that cover the decision space more efficiently.

5.2 Proposed Approach: DA-DDE

In this section, a dual diversified update ensemble framework has been proposed which has the capability to handle various types of data patterns. The approach, called DA-DDE, includes various components viz. an active ensemble E_{actv}, a passive ensemble E_{psv}, a drift detector, adaptive mechanism to handle drifts, error-based weighting mechanism and a novel dynamic voting hypothesis generation strategy. In the forthcoming sections, all the major components of DA-DDE are described. Later Section 5.2.8 explains the steps followed in the proposed approach.

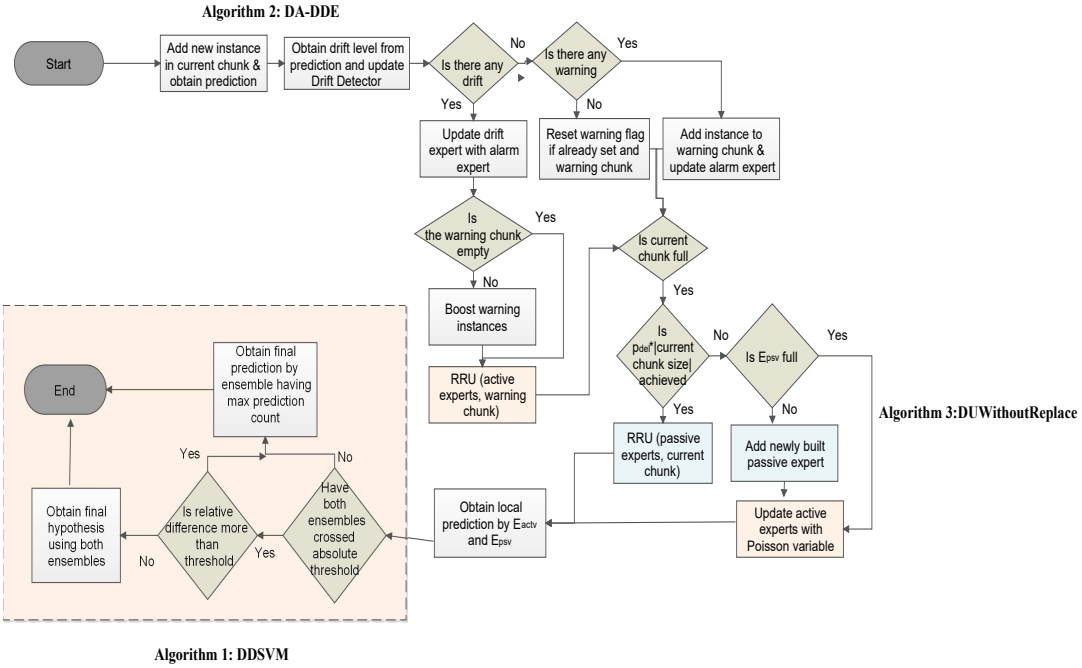


Figure 5.1: The flowchart for the proposed approach DA-DDE.

RRU: Reweight experts (of E_{actv}/E_{psv}) with specified chunk (warning/current), replace poorest expert with newly built one and update existing experts.

All the notations and symbols used in the proposed work with their descriptions are provided in Table 5.1. Figure.5.1 provides a pictorial view of the flow of the proposed approach DA-DDE.

5.2.1 Construction of Dual Ensemble and Training

Two ensembles namely actively updated ensemble (E_{actv}) and passively updated ensemble (E_{psv}) are constructed which vary in their diversity levels and adaptive mechanism towards concept drift. Initially, the data instances are processed one by one and collected to form the chunk of predefined size to construct both ensembles. While training, the chunk instances are induced with the specified diversity levels for both ensembles. The newly built

expert is added to the ensemble setting with an initial weight. As the size of the ensemble grows, experts are added with weights based on their accuracy of prediction on current and previous chunk. These experts are updated by incrementally training them on the new chunk of latest instances. To keep the ensembles consistent with the current distribution, the poorly performing expert is periodically substituted with a new one. There is a drift detection mechanism that explicitly keeps monitoring the upcoming drift, if any (Explained in the Section 5.2.2).

5.2.2 Detecting Concept Drift via Drift Detector

DA-DDE handles the concept drift by means of both implicit adaptive strategy and explicit drift detector. Drift Detection Method (DDM) proposed by Gama *et al.*[65] is used as the explicit drift detector in DA-DDE. This detector uses an underlying learner to classify the incoming instances in the distribution and the classification result obtained is used to compute the error-rate of the learner. This result indicates whether the classifier has correctly classified the incoming instance or not. If there is a change in the concept, the learner will incorrectly classify the instances. An increase in the error-rate beyond the defined thresholds signifies a concept drift. The two thresholds namely warning and drift are defined by Eq. (5.1a) and Eq. (5.1b) respectively. The error rate (p_i) and standard deviation (s_i) are computed after every instance and stored if $s_i + p_i$ reaches it minimum (obtaining s_{min} and p_{min}). According to Eq. (5.1a), when the warning level is reached, the instances are stored, anticipating a possible drift situation. A concept drift is indicated in case the drift condition is satisfied as stated by Eq. (5.1b). Thereafter the learner and the values of s_i, p_i are reset.

$$s_i + p_i >= 2 * s_{min} + p_{min} \tag{5.1a}$$

$$s_i + p_i >= 3 * s_{min} + p_{min} \tag{5.1b}$$

5.2.3 Adaptive Mechanism of E_{actv} and E_{psv}

In DA-DDE a specific process is followed by both ensembles to react to different types of concept drifts. E_{actv} follows a pro-active mechanism using an explicit drift detector to respond to abrupt drifts whereas E_{psv} deals with gradual and recurring drifts using a deferred update technique. It is important to keep the ensemble structure up-to-date with latest distribution which is achieved by E_{actv} by immediate updating as soon as the drift is signaled. It becomes crucial to retain the concepts for a certain period, to prevent loss of information while handling gradual and recurring drifts. E_{psv} delays updating for a certain amount of time to provide better adaptability to slowly changing concepts.

A novel concept of upgrading the instances, which are vulnerable in causing drift (the risky instances), has been proposed in DA-DDE. All the instances coming after the warning level are boosted by a factor ξ_{boost} to ensure that special weight is given to such instances. They are collected in a separate warning chunk. The E_{psv} holds on its current distribution and defers the training process of its component experts till a predefined chunk period, $p_{del} * n_c$. Here p_{del} is empirically chosen such that the previous concept is retrained and at the same time it does not get obsolete. This setup preserves the historic knowledge of the previous distribution for a longer time as compared to the former ensemble, thus ensuring efficient response to the recurrent drifts. The periodic updating and re-weighting mechanism of E_{psv} gradually accommodates the stationary concepts for longer period. Since updating the underlying experts is a costly computation process, delaying this phase for comparatively longer time helps to maintain the overall performance. The training instances are randomized using the optimum diversity level λ_{psv} with online bagging approach while tuning the ensembles.

5.2.4 Weighting Function of DA-DDE

DA-DDE achieves high predictive performance and knowledge transfer between the current data and previous distribution by utilizing the expert's information on historic data while calculating the current weights of the underlying experts of the dual ensembles.A proportionate exponential function which combines past and current performances of the experts is used in weighing function. For deriving the weight of experts Xe_i, (i = 1, 2,...k), mean square error (MSE) of a randomly predicting expert and current one is calculated.

$$MSE_r = \sum_y p(y)(1-p(y))^2 \tag{5.2}$$

$$w_{newExpert} = e^{-(MSE_r)} \tag{5.3}$$

Weight of new expert: The weight of the newly added expert $\hat{X}'_p$ is calculated through Eq. 5.3,which depends upon MSE_r only. As the new expert ($\hat{X}'_p$) is not assigned weight based on its testing performance on the current chunk, it is assigned a weight which is dependent on the current class distribution of the current chunk and hence treated as the perfect learner. $p(y)$ in Eq. 5.2, is the distribution of class y which is the reference point to the current class distribution. This approach assumes that the latest chunk provides best representation of current and future data distribution of the concept [85]. However, from the forthcoming chunks after training, its weight is updated based upon its predictive performance as in case of other experts.

$$MSE_i^j = \frac{1}{|C_j|} \sum_{\{x,y\} \in C_j} (1 - p_y^i(x))^2 \tag{5.4}$$

MSE_i^j of the expert Xe_i is calculated while considering the probabilities of all the classes in the distribution. MSE_i^j estimates the accuracy error of the supervised data on the chunk

C_j of the latest concept using the Eq. (5.4) .

In Eq. (5.4), $p_y^i(x)$ denotes the probability by which the expert Xe_i classifies the data instance x as class $y(x)$ over the current distribution. The final accuracy weight function of the expert is calculated using Eq. (5.5) which depends on both the errors MSE_r and MSE_i^j.

$$w_i^j = \begin{cases} e^{-(MSE_r + MSE_i^j + \beta * MSE_r^{j-1})}, \\ \qquad \text{if } FLAG_{replace} = TRUE \parallel [E_{psv/actv}] < n_{psv/actv} \\ e^{-(MSE_r + MSE_i^j + \beta * MSE_i^{j-1})}, \text{otherwise} \end{cases} \tag{5.5}$$

Given MSE_r and MSE_i^j, w_i^j denotes the weight of expert i over the data chunk C_j and β represents the decaying function. MSE_i^{j-1} gives the mean square error of the current expert on the previous chunk C_{j-1}. $[E_{psv/actv}]$ refers to the selection of current size of either of two ensembles, E_{actv} or E_{psv}, and correspondingly their maximum size n_{actv} and n_{psv}. The $FLAG_{replace}$ is a binary variable which is set TRUE in case the least performing expert, in any of the two mentioned ensembles, needs to be replaced by a new one. The utilization of the historic error in calculating the current weight depends on the assumption that recent past performance directly impacts the latest prediction capability of an expert.

Since two adjacent concepts are closely related, the performance of expert on previous data chunk should not be completely ignored. It has been analysed through empirical calculations that considering the more consistent expert over a period of two chunks is better than the one performing efficiently in the recent one. Thus, the knowledge transfer of the classification credibility between the consecutive chunks of data leads to a better adaptability to the evolving trends.

5.2.5 Hypothesis Generation via Dynamic Ensemble Selection

Dynamic Dual Selective Voting Mechanism (DDSVM), proposed in DA-DDE ensures that the most relevant ensemble participates in the voting process. In some scenarios, one ensemble may outperform the others and, in such situations, it is not wise to use the outdated ensemble for prediction of incoming instances. With the onset of concept drift, the historical information and parameters like accuracy and errors become less related; so, relying on only single ensemble is not preferable. Dynamic abstaining principle is considered, where the ensemble which performs well and is more suited for the current drift, is considered, and others are abstained from voting. A voting count probability corresponding to both the ensembles is obtained and the one with maximum value is used for periodic evaluation. It is directly proportional to the number of correct predictions given for the previous chunk.

Algorithm 7 Dynamic Dual Selective Voting Mechanism (DDSVM) (E_{psv}, E_{actv}, [$Current_c$])

for each instance $\hat{x_i}$ in the [$Current_c$] **do**
  $\vartheta_{psv} \Leftarrow$ global prediction by E_{psv} using Eq. (5.6b);
  $\vartheta_{actv} \Leftarrow$ global prediction by E_{actv} using Eq. (5.6a);
  if ϑ_{psv}==1 **then**
    cnt_{psv}++;
  end if
  if ϑ_{actv}==1 **then**
    cnt_{actv}++;
  end if
end for
$P_{psv} = cnt_{psv}/||Current_c||$;
$P_{actv} = cnt_{actv}/||Current_c||$;
if $P_{psv} > \alpha^{thresh}$ *and* $P_{actv} > \alpha^{thresh}$ **then**
  if $|P_{psv} - P_{actv}| \leq \Psi^{thresh}$ **then**
    $E_{final} \Leftarrow E_{psv} + E_{actv}$;
  end if
else
  $E_{final} \Leftarrow E(argmax(cnt_{actv}, cnt_{psv}))$;
end if

Algorithm 7 explains DDSVM which defines two important threshold parameters, α^{thresh} and Ψ^{thresh}. α^{thresh} is the minimum accuracy (minimum number of correct predictions) that

an ensemble must have for effective participation in the proposed voting scheme. If an ensemble outperforms the other by Ψ^{thresh}, it is considered for final hypothesis generation. Every input instance is tested on the previous chunk and dynamic selection of ensemble is done based upon following cases:

Case 1: If both P_{actv} and P_{psv} cross the α^{thresh} level and their difference is less than Ψ^{thresh}, then global prediction of the both the ensembles is considered.

Case 2: In all other cases, the ensemble with maximum global prediction count *i.e* the one performing better on the given chunk, is considered for hypothesis generation.

$f(E_j(\hat{x}_i), \hat{y}_i)$ and $f(E_k(\hat{x}_i), \hat{y}_i)$ are the output classifying functions of experts of the ensembles E_{actv} and E_{psv} given by Eq. 5.6a and Eq. 5.6b respectively.

$$f(E_j(\hat{x}_i), \hat{y}_i) = \begin{cases} 1, & \text{if } E_j(\hat{x}_i) = \hat{y}_i \\ 0, & \text{otherwise} \end{cases} \tag{5.6a}$$

$$f(E_k(\hat{x}_i), \hat{y}_i) = \begin{cases} 1, & \text{if } E_k(\hat{x}_i) = \hat{y}_i \\ 0, & \text{otherwise} \end{cases} \tag{5.6b}$$

Here $E_j(\hat{x}_i)$ and $E_k(\hat{x}_i)$ are prediction results obtained from the ensemble expert E_j, (j = 1, 2...n_{actv}) and E_k, (k = 1, 2...n_{psv}) respectively. $\hat{x}_i$ denotes the testing instance and $\hat{y}_i$ is the true label for all the possible class values of $\hat{x}_i$. We achieve best performance over the current chunk with selective decision control, allowing the component experts to contribute their useful information about the recent concept.

5.2.6 Diversity Generation in Ensembles

In ensemble learning, having a diverse set of experts trained over varied sample input space ensures that some of them will have a boundary of decision that can be adapted to new concepts quickly leading to correct anticipation of an incoming drift. DA-DDE uses online bagging (*explained in Algorithm 8*) with dual diversity levels in both the ensembles trained with the diverse incoming instances controlled by λ_{actv} and λ_{psv}. These parameters obtain a random factor ξ_k from $Poisson(\lambda)$ distribution and generate randomized subspace of input distribution.

Algorithm 8 OnlineBagging($[Buffer_{chk}]$, EB)

function OnlineBagging($[Buffer_{chk}]$, EB) **Symbol:** λ: Parameter for Poisson distribution

1: **for** each instance $(\hat{x_i}, \hat{y_i})$ in $[Buffer_{chk}]$ **do**
2: **for** each $\hat{X_p}$ in EB **do**
3: Obtain randomized input ξ_k from $Poisson(\lambda)$ distribution;
4: **if** $\xi_k > 0$ **then**
5: $weighted_{inst}\,\omega_b$ = weight(b^i) * ξ_k
6: train the p^{th} expert $\hat{X_p}$ with ω_b
7: **end if**
8: **end for**
9: **end for**

5.2.7 Adaptability to Noisy Streams

If the data instances are affected by noise, they will shift their relative positions with respect to a specific decision boundary. Decrease in classification accuracy can arise due to such unwanted setting. The diverse ensemble experts help to overcome such adverse situations with their varying decision boundaries. The distance between decision boundary and the noisy instance directly impacts the certainty of base experts. If all the experts are allowed to be trained with same subset of subspace, overlapping would lead to shifting of the instance and high uncertainty. The selective aggregated result of the diverse experts of both

Algorithm 9 Dynamically Adaptive and Diverse Dual Ensemble (DA-DDE)

Input: Data stream $\hat{S}$ with incoming instances $\hat{x_i}$, Dual Ensembles E_{actv}, E_{actv}
Output: Predictive ensemble E_{final}, Hypotheis : $\hat{H}$

Obtain next input instance $\hat{x_i}$ from $\hat{S}$;
add $\hat{x_i}$ to $[Current_c]$;
pred $\Longleftarrow$ driftExpert.classify($\hat{x_i}$);
driftLevel $\Longleftarrow$ DriftDetector(pred);
train DriftDetector with current data instance $\hat{x_i}$;
swtich (driftLevel)
case Warning:
add $\hat{x_i}$ to $[Warning_c]$;
$FLAG_{warn}$ = true;
train the alarmExpert with $\hat{x_i}$;
case Drift:
$FLAG_{drift}$ = true;
transfer the alarmExpert model to driftExpert;
if $[Warning_c]$!= $\emptyset$
for each instance x in $[Warning_c]$
$\hat{x_{wt}} = x_{wt} * \xi_{boost}$
end for
end if
E_{actv}.DiversifiedUpdateWithReplace($FLAG_{drift}$, actv);
reset the $[Warning_c]$
$FLAG_{drift}$ = false, $FLAG_{warn}$ = false;
default:
if $FLAG_{warn}$ = true
$FLAG_{warn}$ = false; case of false alarm
reset the $[Warning_c]$
end if
if $[Current_c]$ mod $n_c = 0$ **then**
  if $[Current_c]$ mod $p_{del} n_c = 0$ **then**
    E_{psv}.DiversifiedUpdateWithReplace($FLAG_{drift}$, psv);
  else
    E_{psv}.DiversifiedUpdateWithoutReplace($FLAG_{drift}$, psv);
  end if
  E_{actv}.DiversifiedUpdateWithReplace($FLAG_{drift}$, actv) ;
  Obtain predictive ensemble $E_{final} \Leftarrow$ DDSVM(E_{psv}, E_{actv}, $[Current_c]$)
  Output final hypothesis $\hat{H}$: $\mathbf{X} \rightarrow \mathbf{Y}$
  Predict with $\hat{h}_k(x_t^i) : \delta\ E_{final}[h_k(x_t^i) = y_t^i]$
  return $\hat{H}(x_t^i)$: arg $\max_{y_t^i} \sum_{k=1}^{L} \hat{w}_k \hat{h}_k(x_t^i)$
  $cnt_{actv} = 0, cnt_{psv} = 0$;
end if

the ensembles makes this approach robust against varying noise levels.

5.2.8 DA-DDE- Algorithm

In this section steps followed in the proposed approach are presented in detail as per Algorithm 9.Initially a new data instance $\hat{x_i}$ is obtained from the current data stream $\hat{S}$ and added to the current chunk (lines 1-2). The drift detection method recognizes the appropriate drift level (warning, drift or default) based upon the statistics obtained by the underlying detector (lines 3-4); DDM is the drift detector used in DA-DDE. The drift detector model is trained with the current instance. If the model is at warning level (line 5), the vulnerable warning instance is stored in a separate warning chunk which is boosted later when the drift occurs and meanwhile *alarmExpert* is updated with the new instance. Once the drift is signalled (line 11-13), the *alarmExpert* trained over the warning instances is transferred to the *driftExpert*.

The warning buffer chunk, $WARNING_c$, is checked for emptiness and all the warning instances are boosted with a factor ξ_{boost} to highlight the impact of such instances (lines 14-17). Later Algorithm 10 is executed for the E_{actv} over the warning chunk. All the existing active experts are reconfigured by calculating new weights using Eq. (5.5) over the boosted warning chunk. Consequently, re-weighted setting is updated with drifting concept. A new expert $\hat{X}'_p$ is constructed and substituted with the old active expert with minimum accuracy weight.

To make the existing active experts consistent with the latest concepts, they are trained in diverse setting using the Online Bagging (Algorithm 8). Instead of training the experts incrementally by direct original instance, they are updated with re-sampled weighted instances which later improves their predictive performance (line 19). The warning chunk and flags are reset later (lines 20-21). If there is no-drift scenario, a false alarm is detected if original

warning flag was set TRUE (lines 22-24).

Algorithm 10 function DiversifiedUpdateWithoutReplace ($FLAG_{drift}$, type)

Input: Dual Ensembles E_{actv}, E_{actv}, Warning chunk $[Warning_c]$, Current chunk $[Current_c]$

1: **if** type = actv **then**
2: $EB \Leftarrow E_{actv}$;
3: **else**
4: $EB \Leftarrow E_{psv}$;
5: **end if**
6: construct perfect $\hat{X}'_p$ using Eq. (5.3)
7: **if** $FLAG_{drift}$ = TRUE **then**
8: **for** all experts $\hat{X}_p \in EB$ **do**
9: re-weight $\hat{X}_p$ on $[Warning_c]$ using Eq. (5.5)
10: OnlineBagging($[Warning_c]$, E_{actv})
11: **end for**
12: locate the expert with min. wt. and substitute it with $\hat{X}'_p$
13: **else**
14: **if** $\|EB\| < n_{psv}$ **then**
15: add $\hat{X}'_p$ to the E_{psv}
16: **end if**
17: OnlineBagging($[Current_c]$, EB)
18: **end if**

Once the chunk size is achieved, deferred update of passive ensemble is checked after the drift detection module (lines 26-27). In case the condition of reaching defined chunk cycle is satisfied (line 27), passive ensemble is updated and it replaces the weakest expert in the underlying structure, else it re-trains its existing experts with latest chunk (line 28). The active ensemble necessarily reconfigures (retrained and replaced) itself this time with $[Current_c]$ (line 32). In the E_{psv}, the removal of outdated expert is postponed until three chunk cycles to store the historic distribution for a longer time, making it suitable for adaptation to gradual and recurring drifts. In Algorithm 10 and Algorithm 11, weights of component experts are calculated using Eq. (5.5). Finally, to obtain the final hypothesis, ensemble to be used for prediction is decided by executing DDSVM (line 33) described by Algorithm 7. Next, current hypothesis is drawn using this final ensemble obtained from previous training

chunk (lines 35-37).

Algorithm 11 function DiversifiedUpdateWithReplace ($FLAG_{drift}$, type)

Input: Dual Ensembles E_{actv}, E_{actv}, Warning chunk $[Warning_c]$, Current chunk $[Current_c]$

1: **if** type = actv **then**
2: $EB \Leftarrow E_{actv}$;
3: **else**
4: $EB \Leftarrow E_{psv}$;
5: **end if**
6: construct perfect $\hat{X}'_p$ using Eq. (5.3)
7: **if** $FLAG_{drift}$ = TRUE **then**
8: **for** all experts $\hat{X}_p \in EB$ **do**
9: re-weight $\hat{X}_p$ on $[Warning_c]$ using Eq. (5.5)
10: OnlineBagging($[Warning_c]$, E_{actv})
11: **end for**
12: locate the expert with min. wt. and substitute it with $\hat{X}'_p$
13: **else**
14: **for** all experts $\hat{X}_p \in EB$ **do**
15: re-weight $\hat{X}_p$ on $[Current_c]$ using Eq. (5.5)
16: **end for**
17: locate the expert with min. wt. and substitute it with $\hat{X}'_p$
18: OnlineBagging($[Current_c]$, EB)
19: **end if**

5.3 Datasets

All the datasets, artificial and real, used to analyse the proposed approach, along with different variations of drifts simulated, are discussed briefly in this section (Table 5.2).

5.3.1 Artificial Datasets

Data stream generators which are available in the MOA framework have been used to construct 24 synthetic datasets. These datasets were created using the SEA, Hyperplane, RBF, Sine, Waveform, Agrawal, Tree, Mixed, Gaussian, LED and Stagger generators; details concerning the description of each generator have been discussed in Section 4.3.

Table 5.2: Characteristics of Datasets

Dataset	Generator	# Inst	# Attr	# Cls	Noise[%]	Type of Drift	# Drifts
MIX_{grdl}	Mixed	100k	4	2		gradual	1
SEA_{grdl}	Sea	100k	3	2	10	gradual	3
$SINE_{grdl}$	Sine	100k	2	2		gradual	1
$WAVE_{grdl}$	Wave Drift	400k	40	3		gradual	3
SEA_{abrpt}	Sea	100k	3	2	10	abrupt	9
$HYPER_{abrpt}$	Hyperplane	100k	10	2	5	abrupt	1
MIX_{abrpt}	Mixed	100k	4	2		abrupt	1
$WAVE_{abrpt}$	Wave Drift	400k	40	3		abrupt	3
AGG_{mix}	Agrawal	100k	9	10	10	mixed	3
RBF_{mix}	RandomRBF	100k	10	4		mixed	4
$WAVE_{mix}$	Wave Drift	100k	40	3		mixed	3
RBF_{grdl_rec}	RandomRBF	100k	10	4		gradual recurring	4
RBF_{abrpt_rec}	RandomRBF	100k	10	4		abrupt recurring	3
$TREE_{grdl\_rec}$	RandomTree	100k	5	4		gradual recurring	4
$TREE_{abrpt\_rec}$	RandomTree	100k	5	4		abrupt recurring	5
HYP_{slow}	Hyperplane	100k	10	2	10%	Slow moving	δ = 0.0010
$STAGGER_{grdl}$	Stagger	100k				gradual	3
$STAGGER_{abrpt}$	Stagger	100k				abrupt	3
LED_{slow}	LED Drift	100k			Alternate Noise Percentage: 20% and 30% in streams	Slow moving	3
LED_{fast}	LED Drift	100k			1st stream: 15%, 2nd stream:25%	fast moving	1
SEA_{grdl_rec}	SEA	100k	3	2	Alternate Noise Percentage: 20% and 30% in streams	gradual recurring	4
AGG_{abrpt}	Agrawal	100k	9	10	Alternate Noise fraction: 0.2 and 0.4 in streams	abrupt	9
$GAUSS_{slow}$	Gaussian	100k				Slow moving	4
$GAUSS_{fast}$	Gaussian	100k				fast moving	4
Covertype	real	581k	13	7		unknown	NA
Nursery	real	12,960	8	3		unknown	NA
Shuttle	real	58k	9	2		unknown	NA
Weather	real	58k	9	3		unknown	NA
Intel Lab Sensors	real	100k	5	58		unknown	NA
Airlines	real	5,39,383	7	2		unknown	NA
BNG_bridges	real	100k	12	6		unknown	NA
BNG_hepatitis	real	100k	18	2		unknown	NA
Sensor	real	1,40,000	5	54		unknown	NA
BNG_lymph	real	100k	19	4		unknown	NA
Connect-4	real	67,557	42	3		unknown	NA

5.3.2 Real Datasets

Real datasets which are commonly used in the concept drift domain have been considered for experimentation and evaluation purpose. *Nursery* database [157] which is derived from a hierarchical decision model, was developed originally to rank nursery school applications. It consists of 12,960 instances with 8 attributes defining the expert system shell for decision making. In *Shuttle* [158] dataset instances are in the timely order which could presumably be relevant in classification. Built on 58,000 instances with 9 features, about 80% data of this dataset belongs to class 1. Additionally *Covertype* and textitWeather datasets discussed in Section 3.3.1 have been considered. Apart from these, other datasets namely Airlines, Intel Sensors, BNG_bridges, BNG_hepatitis, BNG_lymph, Sensor and Connect-4 are also popular traditional streaming datasets which are used in the study for experimentation. These are described in Table 5.2 along with their characteristics.

5.4 Results and Discussion

In this section, experimental study is presented for evaluating the effectiveness of the proposed dual-ensemble approach under various environments. The experimental setups were designed to answer the following research questions:

- How does proposed scheme perform under different forms of drifts: gradual, abrupt, recurring, mixed, *etc*. in presence of noisy streams?
- How can variation in different parameter values of the proposed technique influence the prediction accuracy of the model, thus finding their optimized values using optimization technique for each dataset setting?
- How does decay factor which alters weightage of previous chunk's misclassification

error, influence prediction performance?

- Does the choice of base classifier has any impact on the performance of ensemble model?

5.4.1 Setup

During the experiments, we have majorly compared the prediction performance of our proposed model with two categories of algorithms: Ensemble based and single classifier based. All the experiments were performed using the Massive Online Analysis (MOA) framework [137] on the machine equipped with Intel(R) Core i7-7700 CPU @2.80 GHz Processor with 8.00 GB of RAM. The ensemble experts are trained in parallel using separate individual threads which reduced the training time considerably.

5.4.2 Evaluation Mode and Parametrization

For the evaluation purposes, each instance is first used to evaluate the existing classifier before using it for updating process. An online prediction setup with the test-then-train strategy named as *InterleavedTestThenTrain* evaluation scheme is used [137]. The proposed scheme uses the default ensemble size of 10 learners with the block size $|b|$ = 500 as this size is considered as the minimal suitable size for block-based ensembles [84][85]. Default parameter values of all the compared algorithms were used as their initial configuration. The base classifier used is Hoeffding tree with the split confidence = 0.01, grace period value n(min) = 100 and tie-threshold used to break the ties t_i = 0.05. The default values of parameters of DA-DDE for experimentation are set to λ_{psv} = 2, λ_{actv} = 2, β = 0.08, ξ_{boost} = 2, α^{thresh} = 300, $\Psi^{thresh} = 30$ obtained by averaging optimized results through Genetic Algorithm (GA) on various datasets [159]. Experiments were performed by considering various values of p_{del} and its value is taken to be 3 as it gave better results on the consid-

Table 5.3: Average classification accuracies in percentage [%]

DataSet	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
MIX_{grdl}	94.10	95.49	93.06	88.85	93.09	89.43	91.10	94.81	91.70	90.79	95.31	97.21	97.49	94.63	96.10
RBF_{mix}	75.03	81.30	76.00	72.11	82.45	71.88	70.15	78.16	72.12	73.17	82.48	88.66	88.84	59.11	88.47
$WAVE_{mix}$	80.89	82.41	80.81	76.15	82.41	78.54	81.82	79.35	78.32	79.93	83.09	78.71	76.41	78.67	83.34
RBF_{grdl_rec}	81.30	88.97	82.49	74.58	89.44	79.56	75.16	82.00	74.69	75.87	88.88	90.35	91.79	79.41	91.83
RBF_{abrpt_rec}	90.18	92.10	90.43	76.12	93.42	87.28	85.81	89.12	85.82	85.98	93.40	92.11	93.62	87.92	93.87
$TREE_{grdl\_rec}$	50.75	53.32	46.76	45.76	52.00	46.85	46.05	52.68	47.41	42.70	57.03	34.18	40.21	53.52	61.68
$TREE_{abrpt\_rec}$	54.80	57.59	53.14	47.47	57.08	50.71	50.23	57.14	52.74	48.96	60.33	34.82	39.87	57.55	64.14
SEA_{grdl}	84.26	84.47	84.07	82.98	84.33	83.54	84.07	82.98	83.25	82.15	84.42	84.69	83.86	84.71	85.08
$SINE_{grdl}$	93.74	95.75	93.81	88.57	94.46	90.58	89.80	94.17	91.17	89.36	95.73	98.32	98.02	95.69	96.37
$WAVE_{grdl}$	81.09	83.91	80.68	77.85	83.91	80.58	82.50	80.90	79.45	80.11	84.16	80.19	78.45	83.29	83.82
SEA_{abrpt}	86.39	86.12	86.18	84.86	85.87	85.13	86.09	85.51	84.98	84.43	86.49	87.06	85.48	86.34	87.54
$HYPER_{abrpt}$	85.97	84.67	86.85	82.37	83.57	84.46	85.99	84.93	83.13	85.98	85.46	80.07	78.84	84.67	86.75
MIX_{grdl}	94.14	95.37	93.14	89.41	93.34	89.99	91.23	94.74	91.63	90.96	93.74	97.04	97.63	94.22	96.18
$WAVE_{abrpt}$	81.35	84.00	81.02	76.19	84.00	80.75	82.91	81.41	79.50	80.40	84.42	80.71	78.72	83.11	84.14
AGG_{mix}	88.84	91.51	87.87	89.10	91.40	89.58	83.24	89.16	87.58	85.20	92.43	92.48	85.39	92.09	90.71
HYP_{slow}	78.71	81.91	74.80	69.24	81.45	70.05	64.18	72.72	69.09	63.69	81.45	76.67	72.18	78.92	83.22
$STAGGER_{grdl}$	97.71	97.97	97.98	90.01	95.14	92.83	97.78	97.73	97.71	64.03	97.80	97.97	97.33	100.00	97.69
$STAGGER_{abrpt}$	99.31	99.31	99.27	97.66	98.97	99.16	99.31	99.37	99.31	49.91	99.31	99.31	99.31	100.00	99.31
LED_{slow}	62.30	62.39	61.88	62.50	62.51	62.41	62.36	62.19	62.38	57.60	62.49	60.16	61.92	62.37	62.35
LED_{fast}	53.41	53.42	53.18	53.44	53.56	53.35	53.33	53.21	53.35	38.82	53.53	51.20	51.03	53.44	53.42
SEA_{grdl_rec}	70.92	70.62	69.63	70.52	71.06	70.64	69.88	70.12	70.52	67.40	71.08	69.76	68.55	70.96	70.28
AGG_{abrpt}	70.44	69.97	68.36	62.47	69.37	66.47	65.40	67.56	66.98	67.93	73.70	69.96	61.75	77.20	71.08
$GAUSS_{slow}$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100	100.0	100.0	100.0
$GAUSS_{fast}$	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100	100.0	100.0	100.0
Covertype	69.01	60.49	69.39	66.97	71.90	68.37	65.01	68.27	63.40	62.45	69.65	68.48	70.37	73.60	71.65
Nursery	81.53	84.76	84.75	81.32	82.13	81.66	75.73	82.19	73.60	75.39	81.18	80.36	71.55	80.43	81.34
Shuttle	96.87	99.10	96.75	96.06	99.10	96.14	96.82	97.24	96.09	96.29	99.09	99.45	99.38	99.35	99.48
Waether	70.54	69.12	72.34	71.61	76.13	72.25	68.87	75.17	61.72	73.20	75.84	77.24	77.33	67.86	75.34
Intel Lab Sensors	99.18	98.79	98.77	98.71	98.67	98.81	78.58	99.18	98.79	78.55	98.81	99.17	99.05	98.79	99.06
Airlines	65.29	68.28	66.94	67.76	68.96	68.38	59.76	67.48	67.36	63.44	68.59	65.9	66.93	64.81	67.74
BNG_bridges	72.08	73.79	71.53	73.89	73.79	73.58	71.67	72.58	73.52	72.64	73.77	67.48	72.34	66.13	73.64
BNG_hepatitis	88.72	90.59	89.16	89.23	90.17	89.31	87.75	89.71	89.22	87.74	90.17	86.50	89.02	89.80	90.00
Sensor	79.83	77.17	79.04	39.67	68.15	71.17	64.62	78.53	77.20	19.14	75.81	86.50	68.57	73.64	78.20
BNG_lymph	85.97	87.71	86.62	86.53	88.08	86.41	85.15	87.58	86.29	86.07	88.02	83.61	88.11	87.56	88.14
Connect-4	64.50	64.59	68.31	67.17	72.02	68.70	60.26	63.12	62.26	65.94	68.43	71.80	70.70	65.09	69.65

Table 5.4: Evaluation time (in centiseconds) for artificial and real data sets

DataSet	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
MIX_{grdl}	2.5	1.2	1.0	0.2	0.8	0.1	3.9	0.2	0.2	1.8	1.3	4.5	2.1	0.6	16.2
RBF_{mix}	19.8	8.6	9.1	1.0	7.8	0.6	14.0	1.2	1.0	18.9	6.8	16.5	3.6	4.8	15.4
$WAVE_{mix}$	33.4	27.7	19.0	0.6	19.3	1.7	31.5	1.5	1.8	84.0	10.7	80.5	7.4	6.9	23.5
RBF_{grdl_rec}	22.5	17.9	6.1	0.8	11.9	1.0	17.3	0.8	1.2	52.8	7.9	27.1	3.3	5.3	21.8
RBF_{abrpt_rec}	21.5	17.6	4.9	0.4	9.2	0.9	15.7	0.7	1.3	41.3	12.3	22.1	3.0	4.7	17.6
$TREE_{grdl\_rec}$	7.5	4.3	5.8	0.1	4.2	0.3	5.8	0.3	0.8	15.6	4.3	27.8	3.8	1.9	11.1
$TREE_{abrpt\_rec}$	7.1	4.8	6.2	0.1	4.1	0.4	6.0	0.3	1.0	16.0	2.6	22.4	3.7	2.1	10.8
SEA_{grdl}	2.6	2.6	3.0	0.0	1.2	0.3	4.3	0.2	0.2	2.2	2.6	16.7	7.6	0.8	16.4
$SINE_{grdl}$	1.9	0.9	0.6	0.1	0.7	0.1	3.6	0.1	0.1	1.5	1.7	3.1	2.2	0.4	16.2
$WAVE_{grdl}$	37.7	33.1	20.6	0.4	21.0	1.7	55.6	1.9	1.5	140.0	27.7	168.4	9.1	13.5	56.4
SEA_{abrpt}	2.5	1.8	4.0	0.0	1.1	0.2	4.3	0.1	0.4	2.2	3.0	7.9	3.4	0.7	20.7
$HYPER_{abrpt}$	6.9	5.8	3.8	0.3	4.4	0.4	12.3	0.5	0.6	6.6	7.2	9.8	5.6	2.1	32.4
MIX_{grdl}	2.5	1.1	1.0	0.1	0.8	0.1	3.8	0.1	0.1	1.7	2.4	4.9	1.1	0.5	8.8
$WAVE_{abrpt}$	37.5	37.5	20.0	0.5	29.0	1.7	51.8	1.9	2.2	156.1	14.8	173.6	7.6	7.3	26.6
AGG_{mix}	4.8	4.7	1.4	0.1	5.3	0.2	4.1	0.2	0.2	4.6	2.7	15.7	4.5	1.3	12.1
HYP_{slow}	10.5	5.5	1.8	0.2	4.0	0.6	5.4	1.2	0.3	14.7	4.8	36.2	8.8	2.0	27.7
$STAGGER_{grdl}$	1.2	0.6	0.6	0.1	0.4	0.1	1.0	0.1	0.0	1.6	0.6	2.2	1.0	1.6	6.7
$STAGGER_{abrpt}$	0.8	0.5	0.1	0.0	0.2	0.1	1.0	0.1	0.0	2.3	0.4	1.2	0.7	0.8	6.6
LED_{slow}	5.6	2.7	3.6	0.1	0.7	0.1	4.8	0.4	0.3	8.6	1.2	7.0	1.9	1.4	8.4
LED_{fast}	3.2	2.4	4.0	0.2	0.7	0.2	4.8	0.2	0.4	8.5	1.3	4.8	2.0	1.6	7.9
SEA_{grdl_rec}	1.9	2.1	2.4	0.0	1.0	0.3	3.6	0.2	0.2	5.0	1.3	19.5	8.4	0.7	8.6
AGG_{abrpt}	4.2	3.7	2.5	0.0	3.0	0.5	7.6	0.8	0.3	7.1	2.1	15.4	10.7	1.6	13.0
$GAUSS_{slow}$	4.3	1.1	0.1	0.2	0.4	0.0	7.2	0.2	0.1	17.7	1.1	1.2	1.0	0.6	6.7
$GAUSS_{fast}$	4.5	1.0	0.1	0.3	0.5	0.1	6.8	0.1	0.1	17.0	1.2	1.2	0.9	0.8	6.8
Covertype	32.0	32.5	12.8	1.4	11.4	1.5	30.9	2.3	2.1	37.8	11.0	36.8	11.9	11.1	65.8
Nursery	8.2	27.6	1.7	0.3	2.0	0.3	7.4	0.6	0.6	0.9	2.6	8.8	7.1	2.3	53.7
Shuttle	35.0	15.7	6.3	0.9	3.7	3.1	26.3	1.1	1.8	30.2	7.4	12.4	3.6	6.4	52.8
Waether	10.8	70.4	10.3	0.6	5.7	1.5	10.8	1.8	2.0	4.0	7.5	49.8	18.2	8.1	77.0
Intel Lab Sensors	5.2	3.3	0.3	0.4	1.0	0.2	6.1	0.2	0.2	12.7	2.1	3.1	1.6	1.4	24.3
Airlines	520.8	334.6	31.6	41.0	97.9	18.9	612.4	18.9	15.8	1272.1	214.6	309.3	161.0	138.9	2433.7
BNG_bridges	53.5	42.5	54.5	0.6	27.3	3.4	45.4	2.3	3.3	131.3	40.2	318.3	24.6	12.5	65.3
BNG_hepatitis	25.3	45.2	17.6	0.8	33.5	2.1	20.3	2.5	0.9	45.3	40.4	398.8	14.0	8.4	87.6
Sensor	287.1	170.5	61.5	4.5	18.5	10.5	448.6	9.6	11.2	1044.8	64.1	44.2	17.6	90.0	91.6
BNG_lymph	27.6	26.6	22.4	1.3	16.3	1.0	29.7	1.2	1.1	64.4	22.7	143.0	8.3	6.6	56.3
Connect-4	38.6	42.7	25.9	0.7	13.4	1.9	28.6	2.7	2.6	15.0	11.2	106.5	13.7	11.1	64.5

Table 5.5: Average Kappa statistic in percentage [%]

DataSet	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
MIX_{grdl}	88.21	90.98	86.12	77.70	86.19	78.86	82.20	89.61	83.41	81.58	90.61	94.42	94.97	89.27	92.20
RBF_{mix}	66.42	75.00	67.72	62.50	76.54	62.18	60.05	70.64	62.53	63.90	76.57	84.72	85.01	44.12	84.57
$WAVE_{mix}$	71.34	73.62	71.21	64.22	73.62	67.80	72.73	69.01	67.48	69.90	74.64	68.06	64.61	67.99	75.01
RBF_{grdl_rec}	74.89	85.23	76.49	65.89	85.87	72.55	66.77	75.84	66.06	67.61	85.12	87.03	88.99	72.38	89.06
RBF_{abrpt_rec}	86.52	89.19	86.88	67.30	90.98	82.63	80.61	85.16	80.66	80.84	90.95	89.18	91.26	83.34	91.57
$TREE_{grdl\_rec}$	34.20	37.67	28.91	27.56	35.93	29.01	27.95	36.79	29.79	23.59	42.62	12.34	20.18	37.93	48.85
$TREE_{abrpt\_rec}$	39.05	42.85	36.89	28.98	42.13	33.41	33.00	42.24	36.25	31.32	46.55	11.35	18.32	42.69	51.75
SEA_{grdl}	68.53	68.95	68.13	65.96	68.66	67.07	68.14	65.97	66.50	64.31	68.85	69.38	67.72	69.42	70.17
$SINE_{grdl}$	87.47	91.49	87.61	77.14	88.93	81.17	79.60	88.34	82.34	78.72	91.45	96.64	96.05	91.38	92.74
$WAVE_{grdl}$	71.64	75.87	71.03	66.78	75.87	70.87	73.75	71.35	69.18	70.17	76.25	70.29	67.68	74.93	75.73
SEA_{abrpt}	72.77	72.24	72.36	69.72	71.74	70.25	72.18	71.03	69.97	68.87	72.97	74.14	70.96	72.69	75.08
$HYPER_{abrpt}$	71.95	69.33	73.68	64.73	67.14	68.93	71.98	69.86	66.26	71.95	70.92	60.10	57.67	69.35	73.50
MIX_{grdl}	88.29	90.75	86.28	78.82	86.67	79.98	82.47	89.48	83.26	81.93	87.48	94.10	95.26	88.44	92.37
$WAVE_{abrpt}$	72.02	76.01	71.54	64.29	76.01	71.13	74.37	72.12	69.25	70.61	76.62	71.06	68.08	74.67	76.21
AGG_{mix}	77.67	83.03	75.73	78.21	82.79	79.16	66.49	78.31	75.16	70.40	84.85	84.97	70.78	84.18	81.43
HYP_{slow}	55.24	61.78	47.42	35.21	60.88	37.23	24.53	42.68	35.14	21.92	60.88	53.38	40.62	55.32	64.90
$STAGGER_{grdl}$	95.42	95.93	95.95	80.02	90.28	85.66	95.55	95.46	95.42	28.06	95.60	95.95	94.67	100.0	95.37
$STAGGER_{abrpt}$	98.63	98.63	98.54	95.33	97.95	98.32	98.63	98.74	98.63	-0.17	98.63	98.63	98.62	100.0	98.62
LED_{slow}	58.11	58.22	57.70	58.33	58.35	58.23	58.17	57.99	58.20	52.87	58.32	55.74	57.69	58.18	58.16
LED_{fast}	48.24	48.24	47.98	48.26	48.40	48.17	48.15	48.00	48.16	31.98	48.37	45.79	45.59	48.26	48.24
SEA_{grdl_rec}	41.84	41.23	39.25	41.02	42.11	41.28	39.75	40.24	41.03	34.81	42.16	39.53	37.09	41.93	40.56
AGG_{abrpt}	40.87	39.94	36.71	24.95	38.74	32.94	30.80	35.11	33.97	35.86	47.40	39.94	23.50	54.40	42.16
$GAUSS_{slow}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$GAUSS_{fast}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Covertype	53.78	39.17	54.40	49.26	58.29	52.77	48.04	51.61	42.11	41.32	54.77	53.83	55.88	60.81	58.14
Nursery	70.89	75.76	76.87	71.51	71.99	72.01	63.39	72.27	61.02	61.76	70.56	70.19	51.19	68.43	71.63
Shuttle	91.20	97.44	90.95	89.12	97.44	89.34	90.72	92.27	89.19	89.42	97.42	98.47	98.26	98.19	98.55
Waether	25.82	24.86	32.49	30.78	37.91	27.66	31.72	36.87	20.12	31.28	37.19	40.35	39.93	24.86	40.41
Intel Lab Sensors	54.01	53.03	52.99	52.87	52.81	53.06	0.06	54.01	53.03	0.00	53.06	54.00	53.77	53.03	53.78
Airlines	18.96	27.80	23.18	26.10	28.34	27.21	14.07	26.44	25.73	16.44	28.05	21.54	25.74	20.95	28.91
BNG_bridges	62.12	64.57	61.42	64.70	64.57	64.16	61.72	62.73	64.26	62.75	64.44	55.67	61.40	54.01	64.67
BNG_hepatitis	65.52	69.16	65.12	66.51	67.64	65.69	63.64	66.87	65.86	63.32	67.64	54.16	62.38	65.73	67.57
Sensor	79.39	76.68	78.59	38.29	67.47	70.56	63.87	78.07	76.71	17.78	75.30	52.95	67.89	73.07	77.73
BNG_lymph	74.11	77.08	75.21	75.07	77.74	74.81	72.64	76.85	74.56	74.27	77.63	70.84	77.40	76.78	77.96
Connect-4	24.71	24.08	28.03	26.12	30.84	29.11	21.96	25.78	22.66	24.81	26.16	30.41	18.25	22.71	33.73

Table 5.6: Memory consumed (in Kilobytes) in artificial ad real datasets

DataSet	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
MIX_{grdl}	565.6	1533.2	394.8	100.2	935.9	84.0	264.3	42.9	13.8	1240.2	594.5	3187.0	5293.0	494.8	1611.3
RBF_{mix}	657.8	2334.0	506.8	99.5	1328.1	40.6	405.2	22.9	25.1	385.4	1054.7	7773.0	2519.5	2929.7	2890.6
$WAVE_{mix}$	2334.0	8877.0	976.6	246.6	7627.0	573.4	818.9	377.7	84.8	2578.1	2304.7	56976.2	29003.9	5468.8	4150.4
RBF_{grdl_rec}	1289.1	6738.3	460.8	280.2	4882.8	363.6	559.2	94.5	36.6	1943.4	2959.0	21472.8	8173.8	4355.5	8486.3
RBF_{abrpt_rec}	1572.3	4697.3	522.7	116.7	3066.4	151.0	579.6	165.3	38.6	1533.2	1709.0	16194.4	5898.4	3867.2	4960.9
$TREE_{grdl\_rec}$	569.5	1445.3	340.8	91.7	1709.0	126.7	321.8	32.5	24.8	736.6	518.4	20827.9	14941.4	951.7	2070.3
$TREE_{abrpt\_rec}$	547.3	1210.9	419.1	63.6	1679.7	70.7	327.6	29.3	12.6	880.6	592.5	15522.6	14355.5	1015.6	1816.4
SEA_{grdl}	976.2	2539.1	1035.2	40.7	1357.4	105.5	220.9	98.3	28.0	928.5	1328.1	15169.3	19726.6	966.2	1826.2
$SINE_{grdl}$	534.7	854.8	252.4	90.4	899.5	140.1	273.3	32.5	16.9	1289.1	493.1	1787.4	5195.3	478.9	1298.8
$WAVE_{grdl}$	4209.0	31543.0	1142.6	253.1	28613.3	2382.8	852.7	924.1	703.0	15234.4	11328.1	153759.8	113281.3	13867.2	10058.6
SEA_{abrpt}	964.3	1005.9	1572.3	40.7	1328.1	102.7	229.9	27.7	10.6	1005.9	741.5	4938.6	18066.4	575.9	1044.9
$HYPER_{abrpt}$	1523.4	3115.2	690.1	272.1	4794.9	475.5	440.7	110.5	33.8	1875.0	1904.3	6586.0	29882.8	2910.2	3593.8
MIX_{grdl}	558.3	1523.4	389.7	98.2	872.6	83.0	269.6	39.2	12.7	1250.0	825.9	4078.3	4931.6	576.4	1611.3
$WAVE_{abrpt}$	3076.2	32421.9	1123.0	255.5	29394.5	2509.8	856.9	965.2	717.2	14550.8	6767.6	127309.6	112304.7	14062.5	7002.0
AGG_{mix}	996.1	1523.4	334.0	149.0	2841.8	207.5	433.3	72.9	29.3	1621.1	1699.2	8158.1	21972.7	1123.0	2910.2
HYP_{slow}	3476.6	5527.3	495.0	164.7	4775.4	303.0	322.9	393.5	111.5	1269.5	4794.9	15527.1	33593.8	3681.6	17089.8
$STAGGER_{grdl}$	301.6	532.5	218.5	62.9	307.8	60.6	225.0	9.2	4.0	930.1	175.1	306.6	704.9	89.9	366.3
$STAGGER_{abrpt}$	225.3	492.8	61.9	54.3	202.1	36.3	222.3	2.2	2.2	1123.0	146.5	169.5	366.7	90.7	289.3
LED_{slow}	325.2	712.5	371.3	62.5	561.0	32.7	301.2	23.2	5.3	879.5	357.1	2217.9	1806.6	297.4	1855.5
LED_{fast}	316.7	815.0	427.3	58.3	562.7	16.5	301.4	20.2	8.1	549.9	440.4	2904.6	1826.2	239.3	1464.8
SEA_{grdl_rec}	476.4	1816.4	362.0	40.6	1210.9	63.8	215.5	53.3	26.0	622.6	1064.5	17188.0	24902.3	636.2	1425.8
AGG_{abrpt}	774.2	1972.7	336.3	63.0	3720.7	274.6	339.3	295.4	41.5	1377.0	1162.1	8629.4	29296.9	315.3	2871.1
$GAUSS_{slow}$	368.1	401.9	26.2	55.1	137.7	7.1	362.5	7.0	7.0	1669.9	161.0	163.4	232.1	138.8	265.8
$GAUSS_{fast}$	368.9	402.5	27.1	55.5	138.6	7.1	362.9	7.0	7.0	1669.9	161.9	164.3	232.9	139.2	266.0
Covertype	1054.7	752.8	689.4	144.7	2558.6	210.0	440.8	87.6	18.0	1875.0	630.9	3956.1	10156.3	1728.5	2880.9
Nursery	168.3	547.3	104.2	51.0	173.3	21.5	144.4	4.1	3.6	150.6	144.7	469.0	1230.5	141.9	293.3
Shuttle	539.3	846.8	247.5	70.2	505.1	53.9	360.9	39.4	15.7	849.1	498.1	1897.6	963.6	452.9	1035.2
Waether	329.1	563.5	300.2	57.7	650.7	35.5	204.9	46.5	4.3	201.8	517.4	6012.8	4853.5	874.1	1748.0
Intel Lab Sensors	241.5	408.0	29.7	50.3	146.0	21.1	237.3	2.6	5.0	1230.5	137.6	127.8	311.6	120.2	195.5
Airlines	600.6	27641.6	1026.3	980.1	24682.5	3202.3	565.8	356.8	535.0	16822.3	1383.7	26269.5	181740.2	982.5	23980.4
BNG_bridges	1279.3	2900.4	1025.4	124.7	2168.0	183.7	569.0	173.8	68.8	2402.3	1992.2	42639.9	9666.3	395.4	4912.1
BNG_hepatitis	2285.2	5214.8	1103.5	200.5	4267.6	490.6	350.2	389.1	126.6	1630.9	4296.9	42552.8	15625.0	2587.9	9033.2
Sensor	720.2	1123.0	241.1	116.2	1991.0	372.1	680.2	18.2	18.3	4248.0	279.5	8501.4	14257.8	665.5	572.3
BNG_lymph	751.0	2548.8	1132.8	137.3	1855.5	138.4	380.2	154.4	46.3	1738.3	1855.5	28735.9	6845.7	1084.0	5068.4
Connect-4	781.8	1132.8	926.7	207.6	3496.1	369.8	568.7	35.6	14.0	1357.4	574.9	11557.3	10546.9	499.0	3076.2

ered datasets. We analyse the algorithms by comparing their prediction accuracy and other performance metrics like evaluation time, memory, kappa statistic value.

5.4.3 Comparative Models and Algorithms

For the purpose of comparative evaluation, DA-DDE has been compared with ten state-of-the-art algorithms popular in the domain of classification. Methods considered are:

1. *Accuracy Updated Ensemble (AUE)* [106] is representative block-based approach with a fixed block size. The final hypothesis is drawn from the weighted pool of classifiers forming the ensemble. It also dynamically updates the internal classifiers with every set of new data. Block size is set to 500 with 10 experts being preserved in ensemble as suggested by original paper.

2. *Diversity for dealing with Drifts (DDD)* [112] is a popular diversity based approach with EDDM incorporated as the drift detector to trigger the updating of existing ensemble. $\lambda_l = 1$ and $\lambda_h = 0.1$ are the values for low and high diversity ensembles.

3. *Dynamic Weighted Majority (DWM)* [113] introduces concept of age-based and accuracy-based pruning of experts. To penalize the experts, β is set to 0.5, minimum fraction weight θ is set as 0.01 and value for period between removal of expert p is set to 50.

4. *OzaBag (Oza)* [97] is the instance based Online Bagging scheme which promotes sampling of input data defined by Poisson distribution. Ensemble size of 10 and $\lambda = 1$ is used in the experimentation.

5. *Hoeffding Adaptive Tree(HAT)* [160] is an online adaptive approach for classifying the input instances based upon the Hoeffding bounds.

6. *Accuracy Weighted (AWE)* [84] is another block based approach which introduced the concept of mean square error for obtaining the weights for the classifiers for sequential chunks of data stream. Block size is set to 500.

7. *Drift Detection Method (DDM)* [65] is a single classifier based online method for drift detection implied via statistical threshold for warning and drift levels. The default values of the DDM parameters window instances n = 30, warning level $\alpha_w = 2$ and drift level α_d = 3 are used.

8. *Early Drift Detection Method (EDDM)* [66] is another drift detector which is an extension to DDM and works well for gradual drifts by defining new parameters using misclassification rates. Default values as given in MOA framework: window instances n = 30, e = 15, drift level d = 0.9, and warning level w = 0.95 are used here.

9. *LearnNSE (NSE)* [140] handles the evolving streams of data with all types of concept drifts. The voting-based mechanism which does not involve pruning works by keeping check on the accuracy of members on past and current predictive performance.

10. *Oza Bag Adwin (OzaAdw)*[161] extends the Online Bagging approach by adding the drift detector ADWIN. Ensemble size of 10 and λ = 1 is used in the experimentation.

11. *Kappa Updated Ensemble (KUE)*[94] is an ensemble based approach which combines block based and online methods. Block size = 500 is used here for experimentation.

12. *Adaptive Random Forest (ARF)*[91] includes resampling mechansim and adaptive operators for handling concept drift. λ = 6 and default values are used.

13. *Leveraging Bag (Lev)*[98] is ensemble based online approach which extends bagging. Default values as given in MOA implementation: ensemble size = 10, weight Shrink = 6, δ_{Adwin} = 0.002 are used.

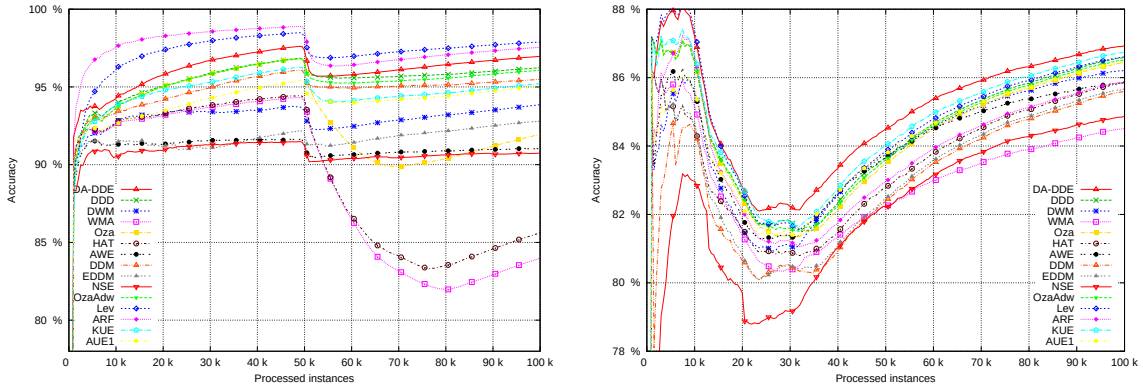


Figure 5.2: Classification accuracy on datasets with gradual drifts a) MIX_{grdl} b) SEA_{grdl}

We evaluate the behaviour of data stream algorithms for concept drift handling in comparison with the proposed approach. Four drifting data streams have been considered: artificial gradually drifting concepts; artificial abruptly drifting concepts; artificial recurring drifting concepts and the popular real-world streams.

In Table 5.3, the average classification accuracy results of all techniques have been listed for artificial and real datasets. It is observed that DA-DDE obtains the highest average classification accuracy amongst all the techniques considered. In comparison to the ensemble-based approaches like DWM, WMA, NSE, AWE, DA-DDE achieves a considerable increase in the accuracy. DDD is another diversity-based approach which performs well for all datasets but DA-DDE gives better results in all the data streams. The reason behind that is DA-DDE considers an extra boosting option for the vulnerable instances near the drift zone and takes into account the performance of the experts on previous chunks even more efficiently. The dual ensembles used in this algorithm apart from just being distinct in terms of diversity are also better trained by utilizing active and passive modes of learning.

5.4.4 Results on Artificial Gradually Drifting Concepts

Table 5.3 represents the average accuracy results obtained on all the drifting streams. In the Figure.5.2(a), results for the MIX_{incr} are depicted for a gradually drifting scenario with one drift point and two concepts alternating at 50k instances. WMA and HAT are least performing algorithms in terms of accuracy. It is evident that compared to AUE1, DWM, and DDM two algorithms namely AWE and NSE have performed slightly less. This can be attributed to the slow replacement of outdated sub classifiers and lack of any drift detection mechanism. However, on the other hand DDM and EDDM which are better drift detectors, do not handle the drifts well in this case. Oza, which only manipulated the input instances to construct the final hypothesis showed a severe drop in accuracy at the drift point. ARF, Lev and DA-DDE are the three top performers which inspite of drop at drift point, later adapted to the change quickly.

Thus, it can be concluded that amongst the techniques considered for comparison, DA-DDE and KUEhave the best adaptability to the drifts with DA-DDE outperforming all others. Figure. 5.2(b) depicts the plot between accuracy and number of instances for the SEA_{grdl} dataset. Most of the algorithms such as HAT, WMA and NSE show a significant drop at the third drift point at 30k instances. Four major concepts with drift points at 10k, 20k and 30k are depicted. While rest of the methods such as AUE1, AWE, DWM show close results, proposed technique shows best recovery to the drift and consequently a consistent increase in the accuracy later. Moreover, other diversity-based techniques such as Lev, DDD and Oza adapt to the gradually drifting scenario efficiently.

For $SINE_{grdl}$ dataset, there are two concept drifts at 50k instances whereas in the $WAVE_{grdl}$, there are two drift points at 100k and 200k instances. In case of $WAVE_{grdl}$ where a total of 400k instances are generated, except for DA-DDE, DDD and Oza all others have poorly performed to the change. Although EDDM is usually well suited for gradual drifts, it

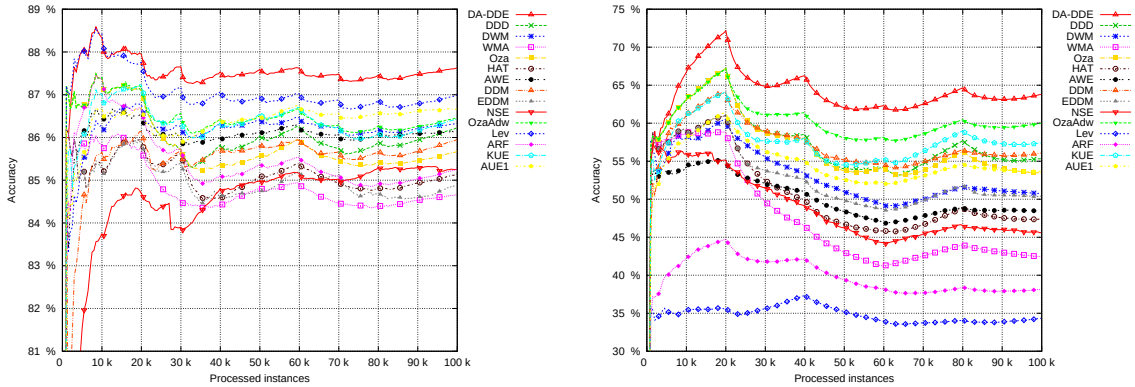


Figure 5.3: Classification accuracy on datasets with a) abrupt drift SEA_{abrpt} b) recurring drift $TREE_{abrpt\_rec}$

is the worst performer here. This could be due to the lack of adaptive mechanism involved in tuning mechanism. In case of $SINE_{grdl}$, DA-DDE achieves high accuracy by rebuilding the ensemble quickly and giving special treatment to the vulnerable instances at 500k.

5.4.5 Results on Artificial Abruptly Drifting Concepts

In this work, we have generated four streams which have abrupt drifts in the concepts. According to the experimental results depicted in Table 5.3, DA-DDE achieves highest accuracy in all four streams and LevBag is the second best. Accuracies of tested datasets SEA_{abrpt} is illustrated in Figure.5.3(a). Unlike other datasets, here WMA and HAT have shown better performance. NSE is the least performer amongst all others in case of SEA_{abrpt} dataset. This dataset is an interesting use case of multiple drifts after every 10k instances. This dynamically moving conceptual pattern is adapted well by DA-DDE which shows a significant improvement over its counterparts. Choosing the best ensemble after every chunk and boosting the warning instances is the reason behind accurate performance. Good performers like Oza and DDD fail to achieve consistent accuracy.

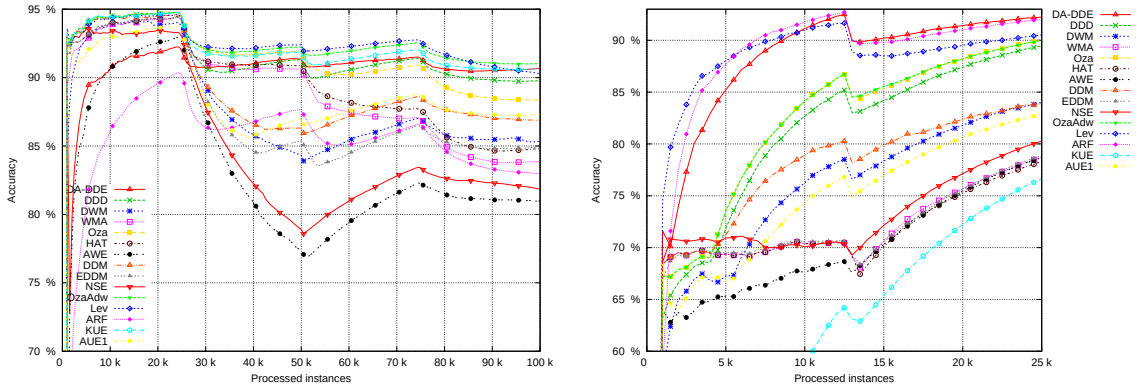


Figure 5.4: Classification accuracy on datasets with mixed drifts a) AGG_{mix} b) RBF_{mix}

5.4.6 Results on Artificial Recurring Drifting Concepts

Artificially built recurring data pattern is formed by Tree generator, depicted in Figure.5.3(b). It is quite evident that DA-DDE has shown best accuracy results that too significantly better than others while handling the recurring drift situations. The capability to sustain accuracy in such complex case is attributed to the timely updating, pruning of outdated experts and diversified dynamic ensemble selection. Best ensemble is selected at every point which makes DA-DDE adaptable to the current concept for which it was trained in the past. Six concepts have been induced alternating at 20k instances. Starting from the first drift point to the end, DA-DDE gives a significant increase in accuracy for gradual and abrupt drift forms. Even though AUE1 prunes the least performing classifier from the ensemble, lack of diversity in it makes it less accurate than DA-DDE. Oza again proved to be a non-performer. In case of sudden recurring drift in $TREE_{abrpt\_rec}$ LevBag, ARF and WMA are worst performing algorithms. DDM which is equipped with a drift detection mechanism stays accurate after 50k and DA-DDE is best performer.

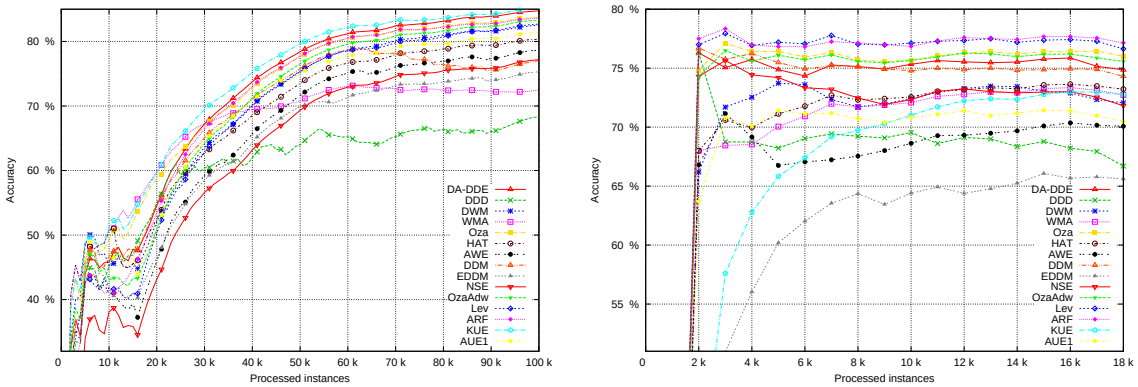


Figure 5.5: Classification accuracy on real datasets a) Covertype b) Weather

5.4.7 Results on Artificial Mixed Drift Concepts

A practical and complex scenario wherein multiple types of drifting concepts exists within a single data stream is shown in Figure.5.4(a) and Figure.5.4(b). First dataset is the AGG_{mix} which has one sudden and two gradual drifts switching at 25k. A significant difference between DA-DDE and other algorithms can be observed. The rebuilding of passive ensemble which contains diverse experts can be attributed to the performance of DA-DDE. LevBag and OzaAdw are good performers whereas ARF gives sharp drops at drift points. The previously learned components which are not dynamically selected for making the final hypothesis can be one of the reasons behind this. Even in RBF_{mix} and $WAVE_{mix}$ scenario, DA-DDE renders consistent performance. In case of the latter, Oza, OzaAdw and DDD show nearly overlapping plots, all three being diverse in nature but giving lower accuracy than the proposed scheme. There is a lack of any drift detection mechanism in WMA, HAT, AWE and KUE thereby causing poor reaction to the abrupt drifts. Moreover, the dynamic updating does not prune outdated classifiers, making them incapable to gradually drifting concepts.

5.4.8 Results on Real Datasets

In this subsection, we compare the performance of proposed techniques with other algorithms over the real data streams. From the Table 5.3 it is clear that there is variation in the best performing algorithm in the datasets considered for evaluation. Figure.5.5(a) gives the plot over Covertype dataset which clearly indicates that the plot of DA-DDE and KUE has more accuracy than that of others. It is noticed that EDDM and DDM which are non-ensemble approaches have given considerably lower results than the ensemble-based approaches, whereas Oza, DDD, DWM, AUE1, AWE give a consistent increase in accuracy over others. The process of rebuilding the model on occurrence of a drift proves beneficial in case of DA-DDE.

In case of Weather dataset, DA-DDE is much more efficient than its counterparts. Figure.5.5(b) shows that DDD, which usually is a top performer in most of the artificial datasets is incompetent as it has poor performance. A fixed diverse mechanism is insufficient to handle real drifts. As constant training for the base experts is needed, it can be concluded that DA-DDE with its dynamically chosen ensemble for decision making and boosting the vulnerable instances provides best solution to handle drifting scenarios.

5.4.9 Experiments using Different Parametric Settings

To access the significance of base learner we have tested our proposed technique on three different setting: Multilayer Perceptron (MLP), Naive Bayes (NB), Hoeffding Tree (HT). The results of classification accuracy on all the datasets considered are given in Table 5.7. The parameter setting of all the three learners is discussed below: Multilayer Perceptron: learning rate = 1.5, momentum = 0.3, activation Function = sigmoid and Hoeffding tree: split confidence = 0.01, grace period value n(min) = 100 and tie-threshold used to break the ties t_i = 0.05. As seen in Table 5.7, HT gives best results for all the datasets while MLP gives

Table 5.7: Average classification accuracies in percentage [%] on different base learner setting

DataSet	MLP	NB	HT
MIX_{grdl}	83.74	91.20	95.92
RBF_{mix}	70.40	80.61	87.89
$WAVE_{mix}$	81.59	81.43	84.00
RBF_{grdl_rec}	72.97	88.98	91.85
RBF_{abrpt_rec}	83.87	91.96	94.16
$TREE_{grdl\_rec}$	46.64	34.78	59.72
$TREE_{abrpt\_rec}$	50.78	40.36	63.42
SEA_{grdl}	82.55	68.68	85.09
$SINE_{grdl}$	84.15	83.83	96.27
$WAVE_{grdl}$	80.94	83.02	84.34
SEA_{abrpt}	84.89	72.70	87.30
$HYPER_{abrpt}$	84.24	87.57	86.26
MIX_{grdl}	83.37	91.24	96.06
$WAVE_{abrpt}$	82.49	83.14	84.64
AGG_{mix}	76.82	51.73	92.92
HYP_{slow}	64.63	64.22	82.60
$STAGGER_{grdl}$	97.01	96.00	97.75
$STAGGER_{abrpt}$	98.08	99.00	99.31
LED_{slow}	62.39	58.87	62.35
LED_{fast}	53.27	49.45	53.37
SEA_{grdl_rec}	70.84	56.34	71.07
AGG_{abrpt}	65.47	50.84	74.26
$GAUSS_{slow}$	100.0	100.0	100.0
$GAUSS_{fast}$	100.0	100.0	100.0
Covertype	69.12	47.12	71.71
Nursery	78.71	79.28	81.64
Shuttle	90.49	78.34	99.48
Waether	70.99	55.29	76.39
Intel Lab Sensors	99.05	99.07	99.09
Airlines	67.64	60.28	68.40
BNG_bridges	71.58	34.72	73.43
BNG_hepatitis	86.83	77.78	89.74
Sensor	71.58	4.03	78.20
BNG_lymph	83.52	79.73	88.10
Connect-4	67.49	65.06	71.15

the least.

Table 5.8 displays the prediction accuracy results for different values of β. With the help of Friedman test [142] on β = [0.05, 0.1, 0.5, 0.6, 0.8], statistical ranks are obtained. This clearly indicates that DA-DDE with $\beta = 0.8$ has best accuracy results. Having a larger decaying factor indicates an increased weightage of the expert on previous chunk performance. It can enhance the overall predictive performance of the classifier by not just relying on current one but taking estimation of previous predictions on historic data as well.

5.4.10 Application of Proposed Approach in Large-Scale Data Streams

As the dynamic nature of the data streams needs periodic updates, it is crucial that it includes knowledge from the latest batch. There are various domains where data stream classification is used which include weather forecasting, fraud detection in case of credit card and debit card transactions, classification of news feed from RSS readers, network monitoring *etc*. Apart from the twenty-four artificially generated drifting streams and eleven real datasets that are considered, two potential applications are taken up to demonstrate the effectiveness of proposed approach. Two applications which utilize large scale data streams, considered for experimental purpose are: change in electricity prices with demands and spam filtering.

Predicting electricity pricing trends: Electricity dataset, collected from Australian New South Wales Electricity Market [162] displays an evolving relationship between the market parameters and prices. This time series data is a popular application of large-scale drifting data streams as the electricity prices are not constant and are set every five minutes. Various factors such as market demand, season, time of the day, supply keep changing seasonally and affect the price of electricity. Temporal dependency is exhibited in this dataset comprising of 45,312 instances and attributes. Two possible class labels namely up and down, indicate the

Table 5.8: Average classification accuracies (%) of DA-DDE using different values of β

DataSet	0.05	0.10	0.50	0.60	0.80
MIX_{grdl}	95.86	95.92	96.15	96.12	96.08
RBF_{mix}	87.88	87.90	88.48	88.48	88.41
$WAVE_{mix}$	84.00	83.84	83.88	83.93	83.91
RBF_{grdl_rec}	91.88	91.92	92.46	92.44	92.43
RBF_{abrpt_rec}	94.37	94.61	94.80	94.82	94.85
$TREE_{grdl\_rec}$	59.71	59.88	61.70	61.70	62.44
$TREE_{abrpt\_rec}$	63.09	63.39	63.26	63.01	63.22
SEA_{grdl}	85.08	85.11	84.85	84.85	84.84
$SINE_{grdl}$	96.24	96.27	96.28	96.28	96.31
$WAVE_{grdl}$	84.39	84.42	84.71	84.76	84.71
SEA_{abrpt}	87.28	87.20	86.75	86.63	86.88
$HYPER_{abrpt}$	86.25	86.21	86.03	85.98	85.91
MIX_{grdl}	95.98	96.06	96.10	96.04	96.04
$WAVE_{abrpt}$	84.62	84.64	84.94	84.95	84.93
AGG_{mix}	92.94	92.94	92.80	92.80	92.89
HYP_{slow}	82.24	82.58	82.31	82.31	82.66
$STAGGER_{grdl}$	97.84	97.71	97.64	97.59	97.62
$STAGGER_{abrpt}$	99.31	99.31	99.31	99.31	99.31
LED_{slow}	62.37	62.38	62.43	62.44	62.43
LED_{fast}	53.32	53.43	53.38	53.39	53.41
SEA_{grdl_rec}	71.05	71.07	71.01	71.01	71.04
AGG_{abrpt}	74.25	74.24	73.73	73.74	73.99
$GAUSS_{slow}$	100.00	100.00	100.00	100.00	100.00
$GAUSS_{fast}$	100.00	100.00	100.00	100.00	100.00
Covertype	71.10	71.41	72.05	72.49	72.08
Nursery	81.50	81.63	81.87	81.87	81.90
Shuttle	99.47	99.49	99.48	99.48	99.57
Waether	76.33	76.32	75.94	75.92	75.92
Intel Lab Sensors	99.09	99.09	99.09	99.09	99.09
Airlines	68.39	68.41	68.49	68.48	68.60
BNG_bridges	73.30	73.43	74.46	74.46	74.51
BNG_hepatitis	89.72	89.68	89.86	89.84	89.90
Sensor	77.97	78.01	77.33	77.31	77.38
BNG_lymph	88.00	87.98	88.57	88.56	88.60
Connect-4	71.44	70.75	71.44	71.39	71.64
Ranks	3.6	3.0	2.8	3.1	2.5

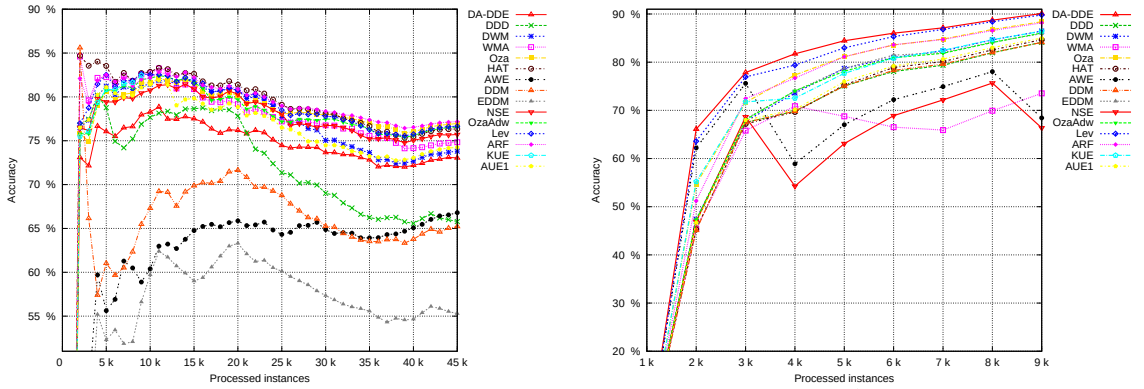


Figure 5.6: Classification accuracy on large-scale data streams a) Electricity Pricing b) Spam filtering

change in the prices. They are calculated by taking a moving average in 24 hour duration. The proposed approach has performed well on this dataset and achieved high accuracy by adapting itself to the changing trends (Figure . 5.6(a)).

Spam filtering: Another important application is spam content filtering. Here, a real-worl dataset of spam and legitimate emails comprising of 9,324 instances and 500 attributes has been used [163]. The emails having information of exact date and time, when they were sent or received was extracted and chronologically ordered according to the time interval. This dataset represents the case of gradual concept drift due to its time-evolving nature. The proposed approach obtained high prediction accuracy while classfying the emails as spam or legitimate as seen in Figure. 5.6(b). As compared to other algorithms it has achieved continuous increase in the accuracy and no drop in performance throughout the dataset. The classification accuracy results on these two applications are shown in Table. 5.9.

5.5 Statistical Analysis

To complement the analysis of experimental results, we performed statistical tests using the non-parametric Friedman test [141] as described in Section 3.4.3. F_f is used to obtain the

ranks for all the algorithms considered for comparison with the proposed approach as stated in Eq. 3.5. We computed the Friedman value $F_{(14,476)}$ value = 1.71 as there are 15 algorithms (k) tested over 35 datasets (N) taking 0.05 as significance level (α_{sig}). Table 5.10 shows the average ranks for accuracy obtained with the Friedman Test.

Post-hoc analysis

Statistical ranks obtained as per Table 5.10 are further compared using Bonferroni-Dunn [141] post-hoc analysis test. It is used for determining which algorithms are significantly better than DA-DDE. Value of critical difference, CD = 3.62 corresponding to $\alpha_{sig} = 0.05$, is obtained. For classification accuracy F_f value obtained from the average ranks is 15.9, leading to rejection of the hypothesis. In terms of classification accuracy, DA-DDE outperformed AUE1, DWM, WMA, HAT, AWE, DDM, EDDM, NSE, Lev and ARF since the difference in their ranks w.r.t. DA-DDE is more than the critical difference as per Table 5.10. Similarly, for the Kappa statistic, F_f = 16.4, leading to rejection of the hypothesis. The proposed approach has given better results than AUE1, DWM, WMA, HAT, AWE, DDM, EDDM, NSE, Lev and ARF. In terms of memory, DA-DDE consumes less memory than the algorithms Lev and ARF. However, DDD consumes almost same memory as our proposed approach. Though, other algorithms have less memory consumption (F_f = 180.5, rejecting null hypothesis), but its worth mentioning that their classification acuracy achieved is also quite less than DA-DDE. In terms of evaluation time (F_f = 167.1, rejecting null hypothesis), DA-DDE being a dual ensemble approach takes more time than other approaches, primarily because all other approaches except DDD use only a single ensemble for training and prediction. Approaches like DDM, EDDM, HAT, WMA which take quite less time than DA-DDE, do not handle concept drifts well, giving very low prediction performance.

The proposed approach DA-DDE which combines the block-based and online ensemble approaches handles multiple drift forms. A novel dual ensemble diversity-based approach

Table 5.9: Average classification accuracys large-scale data streams in percentage (%)

	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
Electricity	76.89	77.97	74.84	79.62	79.78	78.43	79.06	72.53	77.58	78.39	66.55	56.65	79.74	63.02	78.70
Spam filtering	73.59	64.50	82.76	78.07	76.45	75.06	81.66	72.87	75.23	65.84	72.69	75.22	72.96	69.67	78.55

Table 5.10: Average algorithm ranks of different algorithms for all evaluation parameters in the Friedman test

	AUE1	DDD	DWM	WMA	Oza	HAT	AWE	DDM	EDDM	NSE	OzaAdw	Lev	ARF	KUE	Proposed
Accuracy	7.7	5.5	8.3	11.2	5.5	9.4	10.8	7.5	10.6	12.2	4.3	7.7	8.6	6.7	3.9
Memory	8.3	11.8	6.3	3.1	10.8	3.8	6.1	2.4	1.2	10.2	8.4	13.6	14.0	7.9	12.1
Kappa	7.6	5.5	8.2	11.1	5.6	9.5	10.6	7.3	10.5	12.2	4.4	7.7	9.3	6.8	3.6
Evaluation Time	11.4	10.1	7.7	1.9	7.1	2.5	11.5	2.9	2.9	12.3	8.2	13.4	8.2	6.3	13.7

addresses drift scenarios in an adaptive environment. Additionally, the dynamic hypothesis generation strategy enables the algorithm to boost the predictive performance by entirely utilizing the underlying diverse ensembles built over non-overlapping input spaces. Experimental evaluations through 25 benchmark artificial and 11 real datasets with noisy streams proves the efficiency of proposed dynamic hybrid approach. As a hybrid ensemble, the novel weighting mechanism takes into account the historic performance of underlying experts which deal simultaneously with recurring and other types of drifts. Moreover, the introduction of parallel computation by multi-threading has helped in improving the efficiency of the streaming algorithms.

Chapter 6 concludes the entire thesis and provide the major contributions along with future directions for this research work.

Chapter 6

Conclusion and Future Scope

Concept drift detection and handling is a challenging task where complexity increases manifold especially, when dealing with various types of drift patterns. As underlying data distributions often tend to change with time, changing concepts in the incoming streams of data need to be detected. This thesis work provides comprehensive discussions on adaptive learning in case of concept-drifting data streams and proposed techniques for the same. Section 6.1 concludes the thesis work and section 6.2 discusses the contribution of proposed techniques. Finally, section 6.3 provides the future scope of extension in the proposed solutions.

6.1 Conclusion

The focus of this thesis has been on detection and handling of various types of concept drifts. This involves analyzing existing techniques and properties of learning systems and handling the deteriorating performance of predictive ensembles.

A novel hybrid diversity based approach, En-ODDD, is proposed which is in line with the characteristic features of both the explicit drift detection and adaptive techniques. To

accommodate the drifting distributions, online bagging has been employed at the time of creation as well as updation of ensemble experts. This led to high predictive performance even in case of multiple drift forms such as abrupt, gradual, recurring *etc.*

As pruning strategies and diversity in ensembles play a significant role in determining the predictive capability of ensembles, another technique named TLP-EnAbLe is proposed to handle drifting streams. TLP-EnAbLe uses similarity based pruning along with diversity to enhance overall performance of classifications systems under varied concept drift patterns.

Further, DA-DDE is proposed which combines online and block-based ensemble techniques and is equipped with a specific strategy to handle both abrupt and gradual drifts respectively. In this technique a novel voting mechanism named DDSVM is proposed for hypothesis generation which dynamically selects the best ensemble based on the correctness of previous chunk in real time.

6.2 Thesis Contributions

Major contributions of this work are:

- Three techniques: En-ODDD, TLP-EnAble, DA-DDE have been proposed for detection and handling of concept drifts and results achieved validate the effectiveness of the proposed work.

- All techniques use variants of ensemble based learning which improve classification performance of drifting data streams. They have proved their ability to adapt to different types of concept drifts, such as abrupt, gradual, recurring and even their complex combinations. As compared to the existing state-of-the-art online and block-based techniques, proposed approaches have performed better in terms of classification accuracy.

- The techniques proposed in this thesis can be successfully applied in various real time applications which include weather forecasting, spam filtering, sensor networks, predictive data streaming *etc.*

6.3 Future Scope

In this thesis work, performance of concept drifting data streams has been improved using ensemble based detection and handling approaches. The above stated contributions open different directions for future work. In Chapter 3, in the proposed approach En-ODDD, bagging is used to induce diversity in the ensemble system. In future, other techniques like various types of boosting and randomization can be explored to incorporate diversity. Currently only ADWIN detector has been used for drift detection. However, many other drift detection techniques and their combinations might be worth analyzing.

In TLP-EnAble three diversity measures have been used to calculate diversity of underlying learners. Various other measures like Kohavi-Wolpert, interrater agreement *etc.* can also be explored to calculate diversity to improve accuracy of pruning strategy. Other levels of pruning involving better similarity measures may also be investigated.

The third approach DA-DDE, can be extended to cater the unlabelled streams under drifting scenarios. Other hypothesis generation methods to determine which ensemble should participate in decision making can be explored to improve overall accuracy. Moreover, providing faster adaptation under combination of varied drift patterns and weighting of learners in ensemble schemes are future lines of research.

Bibliography

[1] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[2] Y. Bai, F. Wang, and P. Liu, "Efficiently filtering rfid data streams.," in *CleanDB*, Citeseer, 2006.

[3] J. Gama and M. M. Gaber, *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[4] C. C. Aggarwal, "Data streams: An overview and scientific applications," *Scientific data mining and knowledge discovery*, pp. 377–397, 2009.

[5] S.-H. Liao, P.-H. Chu, and P.-Y. Hsiao, "Data mining techniques and applications–a decade review from 2000 to 2011," *Expert systems with applications*, vol. 39, no. 12, pp. 11303–11311, 2012.

[6] M. Bharati and M. Ramageri, "Data mining techniques and applications," 2010.

[7] B. Agarwal and N. Mittal, "Hybrid approach for detection of anomaly network traffic using data mining techniques," *Procedia Technology*, vol. 6, pp. 996–1003, 2012.

[8] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[9] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80, ACM, 2000.

[10] M. P. Singh, M. A. Hoque, and S. Tarkoma, "A survey of systems for massive stream analytics," *arXiv preprint arXiv:1605.09021*, 2016.

[11] T. Bleifuß, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, and D. Srivastava, "Exploring change: a new dimension of data analytics," *Proceedings of the VLDB Endowment*, vol. 12, no. 2, pp. 85–98, 2018.

[12] S. Ben-David, E. Kushilevitz, and Y. Mansour, "Online learning versus offline learning," *Machine Learning*, vol. 29, no. 1, pp. 45–63, 1997.

[13] T. I. Dhamecha, R. Singh, and M. Vatsa, "On incremental semi-supervised discriminant analysis," *Pattern Recognition*, vol. 52, pp. 135–147, 2016.

[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[15] Ł. Korycki and B. Krawczyk, "Adversarial concept drift detection under poisoning attacks for robust data stream mining," *arXiv preprint arXiv:2009.09497*, 2020.

[16] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, no. 1, pp. 1–23, 2018.

[17] S. Fukushima, A. Nitanda, and K. Yamanishi, "Online robust and adaptive learning from data streams," *arXiv preprint arXiv:2007.12160*, 2020.

[18] J. L. Lobo, J. Del Ser, E. Osaba, A. Bifet, and F. Herrera, "Curie: A cellular automaton for concept drift detection," *arXiv preprint arXiv:2009.09677*, 2020.

[19] M. M. YACOUB, A. REZK, and M. SENOUSY, "Adaptive classification in data stream mining," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 13, 2020.

[20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification, 2nd Edition*. Wiley, 2001.

[21] T. Escovedo, A. S. Koshiyama, A. V. A. da Cruz, and M. M. B. R. Vellasco, "Detecta: abrupt concept drift detection in non-stationary environments," *Appl. Soft Comput.*, vol. 62, pp. 119–133, 2018.

[22] M. Manoj Kumar, L. Thomas, and B. Annappa, "Capturing the sudden concept drift in process mining," *Algorithms & Theories for the Analysis of Event Data (ATAED'15, Brussels, Belgium, June 22-23, 2015)*, p. 132, 2015.

[23] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *International Conference on Advanced Information Systems Engineering*, pp. 391–405, Springer, 2011.

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

[25] S. Ren, B. Liao, W. Zhu, Z. Li, W. Liu, and K. Li, "The gradual resampling ensemble for mining imbalanced data streams with concept drift," *Neurocomputing*, vol. 286, pp. 150–166, 2018.

[26] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Self-adaptive windowing approach for handling complex concept drift," *Cognitive Computation*, vol. 7, no. 6, pp. 772–790, 2015.

[27] P. Duda, M. Jaworski, and L. Rutkowski, "On ensemble components selection in data streams scenario with reoccurring concept-drift," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.

[28] P. Dhaliwal and M. Bhatia, "Effective handling of recurring concept drifts in data streams," *Indian J. Sci. Technol*, vol. 10, no. 30, pp. 1–6, 2017.

[29] S. Ramamurthy and R. Bhatnagar, "Tracking recurrent concept drift in streaming data using ensemble classifiers," in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pp. 404–409, IEEE, 2007.

[30] R. Elwell and R. Polikar, "Incremental learning of variable rate concept drift," in *International workshop on multiple classifier systems*, pp. 142–151, Springer, 2009.

[31] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, and I. P. Vlahavas, "An adaptive personalized news dissemination system," *J. Intell. Inf. Syst.*, vol. 32, no. 2, pp. 191–212, 2009.

[32] S. Mittal and S. Tyagi, "Computational techniques for real-time credit card fraud detection," in *Handbook of Computer Networks and Cyber Security*, pp. 653–681, Springer, 2020.

[33] H. Huggard, Y. S. Koh, G. Dobbie, and E. Zhang, "Detecting concept drift in medical triage," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1733–1736, 2020.

[34] D. Pradheep, R. Gokul, V. Naveen, and J. Vijayarani, "Anomaly intrusion detection based on concept drift," *Global Journal of Computer Science and Technology*, 2020.

[35] T. Lane and C. E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 295–331, 1999.

[36] M. M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams," in *Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, April 27-30, 2009, Proceedings*, pp. 363–375, 2009.

[37] A. Patcha and J. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[38] O. Mazhelis and S. Puuronen, "Comparing classifier combining techniques for mobile-masquerader detection," in *Proceedings of the The Second International Conference on Availability, Reliability and Security, ARES 2007, The International Dependability Conference - Bridging Theory and Practice, April 10-13 2007, Vienna, Austria*, pp. 465–472, 2007.

[39] C. S. Hilas, "Designing an expert system for fraud detection in private telecommunications networks," *Expert Syst. Appl.*, vol. 36, no. 9, pp. 11559–11569, 2009.

[40] S. Donoho, "Early detection of insider trading in option markets," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pp. 420–429, 2004.

[41] D. Soemers, T. Brys, K. Driessens, M. Winands, and A. Nowé, "Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[42] A. Somasundaram and S. Reddy, "Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance," *Neural Computing and Applications*, vol. 31, no. 1, pp. 3–14, 2019.

[43] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection and concept-drift adaptation with delayed supervised information," in *2015 international joint conference on Neural networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[44] V. Middha, S. K. Sonbhadra, and S. Agarwal, "Concept drift detection in email dataset through intention based segmentation," in *2019 IEEE Students Conference on Engineering and Systems (SCES)*, pp. 1–5, IEEE, 2019.

[45] S. J. Delany, P. Cunningham, and A. Tsymbal, "A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering.," in *FLAIRS Conference*, pp. 340–345, 2006.

[46] J.-J. Sheu, K.-T. Chu, N.-F. Li, and C.-C. Lee, "An efficient incremental learning mechanism for tracking concept drift in spam filtering," *PloS one*, vol. 12, no. 2, p. e0171518, 2017.

[47] V. Iosifidis, A. Oelschlager, and E. Ntoutsi, "Sentiment classification over opinionated data streams through informed model adaptation," in *International conference on theory and practice of digital libraries*, pp. 369–381, Springer, 2017.

[48] A. Jalilvand and N. Salim, "Sentiment classification with concept drift and imbalanced class distributions," *Jurnal Teknologi*, vol. 78, no. 12-2, 2016.

[49] M. Althabiti and M. Abdullah, "Classification of concept drift in evolving data stream," *Emerging Extended Reality Technologies for Industry 4.0: Early Experiences with Conception, Design, Implementation, Evaluation and Deployment*, p. 189, 2020.

[50] F. Dong, G. Zhang, J. Lu, and K. Li, "Fuzzy competence model drift detection for data-driven decision support systems," *Knowledge-Based Systems*, vol. 143, pp. 284–294, 2018.

[51] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 157–163, 2020.

[52] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections," in *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pp. 679–684, IEEE, 2006.

[53] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, no. 1, pp. 56–68, 2008.

[54] R. Bellazzi, "Big data and biomedical informatics: a challenging opportunity," *Yearbook of medical informatics*, vol. 9, no. 1, p. 8, 2014.

[55] C. Alippi, G. Boracchi, M. Roveri, G. Ditzler, and R. Polikar, "Adaptive classifiers for nonstationary environment," *Contemporary Issues in Systems Science and Engineering*, pp. 265–288, 2015.

[56] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[57] T. S. Sethi and M. Kantardzic, "Handling adversarial concept drift in streaming data," *Expert systems with applications*, vol. 97, pp. 18–40, 2018.

[58] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," *Knowl.-Based Syst.*, vol. 18, no. 4-5, pp. 187–195, 2005.

[59] A. A. Beyene, T. Welemariam, M. Persson, and N. Lavesson, "Improved concept drift handling in surgery prediction and other applications," *Knowl. Inf. Syst.*, vol. 44, no. 1, pp. 177–196, 2015.

[60] M. Pechenizkiy, J. Bakker, I. Zliobaite, A. Ivannikov, and T. Kärkkäinen, "Online mass flow prediction in CFB boilers with explicit detection of sudden concept drift," *SIGKDD Explorations*, vol. 11, no. 2, pp. 109–116, 2009.

[61] R. Xu, Y. Cheng, Z. Liu, Y. Xie, and Y. Yang, "Improved long short-term memory based anomaly detection with concept drift adaptive method for supporting iot services," *Future Generation Computer Systems*, vol. 112, pp. 228–242, 2020.

[62] J. Zenisek, F. Holzinger, and M. Affenzeller, "Machine learning based concept drift detection for predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106031, 2019.

[63] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," *Big data analysis: new algorithms for a new society*, pp. 91–114, 2016.

[64] M. M. Kumar, L. Thomas, and B. Annappa, "Phenomenon of concept drift from process mining insight," in *2014 IEEE International Advance Computing Conference (IACC)*, pp. 517–522, IEEE, 2014.

[65] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian Symposium on Artificial Intelligence*, pp. 286–295, Springer, 2004.

[66] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, pp. 77–86, 2006.

[67] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern recognition letters*, vol. 33, no. 2, pp. 191–198, 2012.

[68] A. Bifet and R. Kirkby, "Data stream mining a practical approach," 2009.

[69] R. S. M. de Barros, D. R. de Lima Cabral, P. M. G. Jr., and S. G. T. de Carvalho Santos, "RDDM: reactive drift detection method," *Expert Syst. Appl.*, vol. 90, pp. 344–355, 2017.

[70] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[71] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi, "Test of page-hinckley, an approach for fault detection in an agro-alimentary production system," in *2004 5th Asian Control Conference (IEEE Cat. No. 04EX904)*, vol. 2, pp. 815–818, IEEE, 2004.

[72] I. Žliobaitė, "Combining time and space similarity for small size learning under concept drift," in *International Symposium on Methodologies for Intelligent Systems*, pp. 412–421, Springer, 2009.

[73] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, SIAM, 2007.

[74] A. Bifet and R. Gavalda, "Kalman filters and adaptive windows for learning in data streams," in *International conference on discovery science*, pp. 29–40, Springer, 2006.

[75] B. Krawczyk and M. Woźniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Computing*, vol. 19, no. 12, pp. 3387–3400, 2015.

[76] G. Hulten, L. Spencer, and P. M. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pp. 97–106, 2001.

[77] E. Cohen and M. Strauss, "Maintaining time-decaying stream aggregates," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 223–233, 2003.

[78] N. S. Punn and S. Agarwal, "Testing concept drift detection technique on data stream," in *International Conference on Big Data Analytics*, pp. 89–99, Springer, 2018.

[79] B. Zhang and Y. Chen, "Research on detection and integration classification based on concept drift of data stream," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 86, 2019.

[80] A. Pesaranghader, H. Viktor, and E. Paquet, "Mcdiarmid drift detection methods for evolving data streams," *CoRR*, vol. abs/1710.02030, 2017.

[81] A. Pesaranghader and H. L. Viktor, "Fast hoeffding drift detection method for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 96–111, Springer, 2016.

[82] R. S. M. de Barros and S. G. T. de Carvalho Santos, "A large-scale comparison of concept drift detectors," *Inf. Sci.*, vol. 451-452, pp. 348–370, 2018.

[83] L. I. Kuncheva, "Classifier ensembles for changing environments," in *International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, 2004.

[84] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 226–235, AcM, 2003.

[85] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.

[86] P. Kosina and J. Gama, "Very fast decision rules for multi-class problems," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 795–800, 2012.

[87] B. Silva, N. Marques, and G. Panosso, "Applying neural networks for concept drift detection in financial markets," in *Workshop on Ubiquitous Data Mining*, p. 43, 2012.

[88] J. L. Lobo, I. Laña, J. Del Ser, M. N. Bilbao, and N. Kasabov, "Evolving spiking neural networks for online learning over drifting data streams," *Neural Networks*, vol. 108, pp. 1–19, 2018.

[89] H. Tian, L. Wang, H. Shen, and A. Liew, "Improved ensemble classification for evolving data streams," *IEEE Intelligent Systems*, 2020.

[90] 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.

[91] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

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

[93] P. Zhao, L.-W. Cai, and Z.-H. Zhou, "Handling concept drift via model reuse," *Machine Learning*, vol. 109, no. 3, pp. 533–568, 2020.

[94] A. Cano and B. Krawczyk, "Kappa updated ensemble for drifting data stream mining," *Machine Learning*, vol. 109, no. 1, pp. 175–218, 2020.

[95] P. Sidhu and M. Bhatia, "An online ensembles approach for handling concept drift in data streams: diversified online ensembles detection," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 6, pp. 883–909, 2015.

[96] Y. Sun, Z. Wang, H. Liu, C. Du, and J. Yuan, "Online ensemble using adaptive windowing for data streams with concept drift," *International Journal of Distributed Sensor Networks*, vol. 12, no. 5, p. 4218973, 2016.

[97] N. C. Oza, "Online bagging and boosting," in *Systems, man and cybernetics, 2005 IEEE international conference on*, vol. 3, pp. 2340–2345, IEEE, 2005.

[98] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," *Machine Learning and Knowledge Discovery in Databases*, pp. 135–150, 2010.

[99] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda, "Improving adaptive bagging methods for evolving data streams," in *Asian conference on machine learning*, pp. 23–37, Springer, 2009.

[100] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin, "Online coordinate boosting," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 1354–1361, IEEE, 2009.

[101] R. E. Schapire, "Explaining adaboost," in *Empirical inference*, pp. 37–52, Springer, 2013.

[102] M. Deckert and J. Stefanowski, "Comparing block ensembles for data streams with concept drift," in *New Trends in Databases and Information Systems*, pp. 69–78, Springer, 2013.

[103] Z. Li, W. Huang, Y. Xiong, S. Ren, and T. Zhu, "Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm," *Knowledge-Based Systems*, vol. 195, p. 105694, 2020.

[104] Y. Lu, Y.-M. Cheung, and Y. Y. Tang, "Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 2764–2778, 2019.

[105] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382, 2001.

[106] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," *Hybrid artificial intelligent systems*, pp. 155–163, 2011.

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

[108] D. Brzezinski and J. Stefanowski, "Combining block-based and online methods in learning ensembles from concept drifting data streams," *Information Sciences*, vol. 265, pp. 50–67, 2014.

[109] M. S. Althabiti and M. Abdullah, "Cddm: Concept drift detection model for data stream.," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 10, 2020.

[110] E. S. Babüroğlu, A. Durmuşoğlu, and T. Dereli, "Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection," *Expert Systems with Applications*, vol. 163, p. 113786, 2021.

[111] K. Nishida, K. Yamauchi, and T. Omori, "Ace: Adaptive classifiers-ensemble system for concept-drifting environments," in *International Workshop on Multiple Classifier Systems*, pp. 176–185, Springer, 2005.

[112] 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, pp. 619–633, 2012.

[113] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790, 2007.

[114] R. S. M. de Barros, S. G. T. de Carvalho Santos, and P. M. G. Júnior, "A boosting-like online learning ensemble," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1871–1878, IEEE, 2016.

[115] R. Sebastião, J. Gama, and T. Mendonça, "Fading histograms in detecting distribution and concept changes," *International Journal of Data Science and Analytics*, pp. 1–30.

[116] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, "Having a blast: Meta-learning and heterogeneous ensembles for data streams," in *2015 ieee international conference on data mining*, pp. 1003–1008, IEEE, 2015.

[117] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Systems with Applications*, vol. 138, p. 112832, 2019.

[118] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.

[119] M. M. Idrees, L. L. Minku, F. Stahl, and A. Badii, "A heterogeneous online learning ensemble for non-stationary environments," *Knowledge-Based Systems*, vol. 188, p. 104983, 2020.

[120] P. Sidhu and M. Bhatia, "A novel online ensemble approach to handle concept drifting data streams: diversified dynamic weighted majority," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 1, pp. 37–61, 2018.

[121] R. A. S. Albuquerque, A. F. J. Costa, E. M. dos Santos, R. Sabourin, and R. Giusti, "A decision-based dynamic ensemble selection method for concept drift," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1132–1139, IEEE, 2019.

[122] J. L. Lobo, J. Del Ser, M. N. Bilbao, C. Perfecto, and S. Salcedo-Sanz, "Dred: An evolutionary diversity generation method for concept drift adaptation in online learning environments," *Applied Soft Computing*, vol. 68, pp. 693–709, 2018.

[123] G. Jaber, A. Cornuéjols, and P. Tarroux, "Anticipative and dynamic adaptation to concept changes," *Real-World Challenges for Data Stream Mining*, p. 22, 2013.

[124] Q. L. Zhao, Y. H. Jiang, and M. Xu, "Incremental learning by heterogeneous bagging ensemble," in *International Conference on Advanced Data Mining and Applications*, pp. 1–12, Springer, 2010.

[125] M. K. Olorunnimbe, H. L. Viktor, and E. Paquet, "Dynamic adaptation of online ensembles for drifting data streams," *Journal of Intelligent Information Systems*, vol. 50, no. 2, pp. 291–313, 2018.

[126] R. A. S. Albuquerque, A. F. J. Costa, E. M. dos Santos, R. Sabourin, and R. Giusti, "A decision-based dynamic ensemble selection method for concept drift," *CoRR*, vol. abs/1909.12185, 2019.

[127] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[128] R. Anderson, Y. S. Koh, and G. Dobbie, "Cpf: Concept profiling framework for recurring drifts in data streams," in *Australasian Joint Conference on Artificial Intelligence*, pp. 203–214, Springer, 2016.

[129] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Syst. Appl.*, vol. 138, 2019.

[130] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *International conference on machine learning*, pp. 1009–1017, PMLR, 2014.

[131] F. L. Minku and X. Yao, "Using diversity to handle concept drift in on-line learning," in *2009 International Joint Conference on Neural Networks*, pp. 2125–2132, IEEE, 2009.

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

[133] M. Krysmann and M. Kurzynski, "Methods of learning classifier competence applied to the dynamic ensemble selection," in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, pp. 151–160, Springer, 2013.

[134] P. P. Chan, Q.-Q. Zhang, W. W. Ng, and D. S. Yeung, "Dynamic base classifier pool for classifier selection in multiple classifier systems," in *2011 International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1093–1096, IEEE, 2011.

[135] B. Krawczyk and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams," *Applied Soft Computing*, vol. 68, pp. 677–692, 2018.

[136] K. Jackowski, "Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers," *Pattern Analysis and Applications*, vol. 17, no. 4, pp. 709–724, 2014.

[137] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[138] A. Bifet, J. Read, G. Holmes, and B. Pfahringer, "Streaming data mining with massive online analytics (moa)," *Data Mining in Time Series and Streaming Databases*, pp. 1–25, 2018.

[139] P. M. Gonçalves Jr and R. S. M. De Barros, "Rcd: A recurring concept drift framework," *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1018–1025, 2013.

[140] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[141] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[142] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Communications in Statistics-Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.

[143] R. Malhotra and M. Khanna, "An exploratory study for software change prediction in object-oriented systems using hybridized techniques," *Automated Software Engineering*, vol. 24, no. 3, pp. 673–717, 2017.

[144] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: a survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[145] M. Wozniak, "Application of combined classifiers to data stream classification," in *Computer Information Systems and Industrial Management - 12th IFIP TC8 International Conference, CISIM 2013, Krakow, Poland, September 25-27, 2013. Proceedings*, pp. 13–23, 2013.

[146] Y. Yang, X. Wu, and X. Zhu, "Mining in anticipation for concept change: Proactive-reactive prediction in data streams," *Data mining and knowledge discovery*, vol. 13, no. 3, pp. 261–289, 2006.

[147] E. K. Tang, P. N. Suganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65, no. 1, pp. 247–271, 2006.

[148] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: a real-time analytics open source framework," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 617–620, Springer, 2011.

[149] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE transactions on knowledge and data engineering*, vol. 5, no. 6, pp. 914–925, 1993.

[150] H. Yang and S. Fong, "Countering the concept-drift problems in big data by an incrementally optimized stream mining model," *Journal of Systems and Software*, vol. 102, pp. 158–166, 2015.

[151] S. JARAMILLO-VALBUENA, J. M. LONDOÑO-PELÁEZ, and S. A. CARDONA, "Performance evaluation of concept drift detection techniques in the presence of noise," *Performance evaluation*, vol. 38, no. 39, 2017.

[152] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Generation Computer Systems*, vol. 75, pp. 187–199, 2017.

[153] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing," in *International conference on discovery science*, pp. 264–269, Springer, 2007.

[154] P. Nemenyi, *Distribution-free multiple comparisons, unpublished Ph. D.* PhD thesis, Ph. D. Dissertation, thesis, Princeton University, Princeton, New Jersey, 1963.

[155] T. Escovedo, A. Koshiyama, A. A. da Cruz, and M. Vellasco, "Detecta: Abrupt concept drift detection in non-stationary environments," *Applied Soft Computing*, vol. 62, pp. 119–133, 2018.

[156] A. Cano, M. Gómez-Olmedo, and S. Moral, "A bayesian approach to abrupt concept drift," *Knowledge-Based Systems*, vol. 185, p. 104909, 2019.

[157] D. J. Lizotte, O. Madani, and R. Greiner, "Budgeted learning of naive-bayes classifiers," *CoRR*, vol. abs/1212.2472, 2012.

[158] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 504–509, 2006.

[159] S. G. T. de Carvalho Santos, R. S. M. de Barros, and P. M. G. Júnior, "Optimizing the parameters of drift detection methods using a genetic algorithm," in *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Italy, November 9-11, 2015*, pp. 1077–1084, 2015.

[160] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, vol. 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[161] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 139–148, ACM, 2009.

[162] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[163] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, and I. Vlahavas, "An adaptive personalized news dissemination system," *Journal of Intelligent Information Systems*, vol. 32, no. 2, pp. 191–212, 2009.

List of Publications

1) Kanu Goel and Shalini Batra, "Dynamically updated diversified ensemble based approach for handling concept drift," *Turkish Journal of Electrical Engineering and Computer Sciences, The Scientific and Technological Research Council of Turkey*, vol. 28, no. 1, pp. 556–574, 2020. -SCIE Indexed (IF: 0.682)

2) Kanu Goel and Shalini Batra, "Two-Level Pruning based Ensemble with Abstained Learners for Concept Drift in Data Streams," *Expert Systems, Wiley*, vol. 38, no. 3, p. e12661, 2021. -SCIE Indexed (IF: 2.58)

3) Kanu Goel and Shalini Batra, "Dynamically adaptive and diverse dual ensemble learning approach for handling concept drift in data streams," *Computational Intelligence, Wiley*, 2021. -SCIE Indexed (IF: 2.33)

4) Kanu Goel and Shalini Batra, "Adaptive Online Learning for Classification under Concept Drift," *International Journal of Computational Science and Engineering, Inderscience*, vol. 24, no. 2, pp. 128–135, 2021. -SCOPUS Indexed