Towards handling temporal dependence in concept drift streams.

WARES, S.B.

2023

Towards Handling Temporal Dependence in Concept Drift Streams

Scott Brian Wares

A report submitted as part of the requirements for the degree
of PhD Computing
at the School of Computing
Robert Gordon University
Aberdeen, Scotland

May 2023

Supervisor Dr. John Isaacs

Abstract

Modern technological advancements have led to the production of an incomprehensible amount of data from a wide array of devices. A constant supply of new data provides an invaluable opportunity for the access to qualitative and quantitative insights. Organisations recognise that in today's modern era, data provides a means of mitigating risk and loss whilst maximising efficiency and profit. However, processing this data is not without its challenges. Much of this data is produced in an online environment. Real-time stream data is unbound in size, variety and velocity. Data may arrive complete or with missing attributes, and data availability and persistence is limited to a small window of time. Classification methods and techniques that process offline data are not applicable to online data streams. Instead, new online classification methods have been developed. Research concerning the problematic and prevalent issue of concept drift has produced a considerable number of methods which allow online classifiers to adapt to changes in the stream distribution. However, recent research suggests that the presence of temporal dependence can causing misleading evaluation when accuracy is used as the core metric. This thesis investigates temporal dependence and its negative effects upon the classification of concept drift data. First, this thesis proposes a novel method for coping with temporal dependence during the classification of real-time data streams where concept drift is present. Results indicate that a statistical based, selective resetting approach can reduce the impact of temporal dependence in concept drift streams without significant loss in predictive accuracy. Secondly a new ensemble based method, KTUE, that adopts the Kappa-Temporal statistic for vote weighting is suggested. Results show that this method is capable of outperforming some state-of-the-art ensemble methods in both temporally dependent and non-temporally dependent environments. Finally this research proposes a novel algorithm for the simulation of temporally dependent concept drift data, which aims to help address the lack of established datasets available for evaluation. Experimental results show that temporal dependence can be injected into fabricated data streams using existing generation methods.

keywords: data streaming, concept drift, temporal dependence

Acknowledgements

I would like to take this opportunity to thank those of you who provided your unrelenting and unnerving support over the years. Without you this research would not have been possible.

First and foremost I would like thank my wife, Claire. You have offered unwavering support for me since the very beginning. If it wasn't for you, this would never of happened.

To Dr John Isaacs, I offer my greatest and most sincere gratitude. You have worked with me for many years now, since back during my undergraduate days. You offered me this PhD opportunity and I will be forever indebted for it. Your help, advice and support, both academic and personal, has been invaluable.

Dr Eyad Elyan, you have been an incredible supervisor throughout this journey. Thank you for all the motivation, and for the focus and drive you instilled in me. Your knowledge of machine learning has proven priceless time and again.

Finally, my daughter Ava. You arrived in the closing stages of this journey, and provided a continuous, evolving stream of noise and distraction.

Declaration

I confirm that the work contained in this PhD project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed ..... Date .....

Scott Brian Wares

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

ADWIN Adaptive Sliding Window

AWE Accuracy Weighted Ensemble

AUE Accuracy Updated Ensemble

BD-SCR Burst Detection-based Selective Classifier Resetting

CUSUM Cumulative Sum

CVFDT Concept-adapting Very Fast Decision Tree

DDM Drift Detection Method

DW-CAV Dynamically Weighted Consult And Vote

DWM Dynamic Weighted Majority

ECHO Efficient Concept Drift and Concept Evolution Handling over Stream Data

E-CVFDT Efficient Concept-adapting Very Fast Decision Tree

EDDM Early Drift Detection Method

FPDD Fisher Proportions Drift Detector

FSDD Fisher-based Statistical Drift Detector

FTDD Fisher Test Drift Detector

GUI Graphical User Interface

HLFR Heirarchical Linear Four Rates

LFR Linear Four Rates

MDDM McDiarmid Drift Detection Methods

MOA Massive Online Analysis

MTD Mean Time to Detection

MTFA Mean Time to False Alarm

MTR Mean Time Ratio

OMM Open Mobile Miner

PH Page-Hinckley Test

RAM Random Access Memory

RDDM Reactive Drift Detection Method

RMOA R- Massive Online Analysis

SAND Semi-supervised Adaptive Novel Class Detection and Classification over Data Stream

SEA Streaming Ensemble Algorithm

SMOTE Synthetic Minority class Oversampling Technique

SPRT Sequence Probability Ratio Test

STEPD Statistical Test of Equal Proportions

TDI-CDS Temporal Dependence Inclusion for Concept Drift Simulation

TSI Temporal Stability Index

VEDAS Vehicle Data Stream

VFDT Very Fast Decision Tree

List of Publications

The following publications have been achieved as part of the research project contained within this thesis.

- Wares, S., Isaacs, J. and Elyan, E., 2019. Data stream mining: methods and challenges for handling concept drift. SN Applied Sciences, 1(11), pp.1-19.
- Wares, S., Isaacs, J. and Elyan, E., 2021. Burst Detection-Based Selective Classifier Resetting. Journal of Information & Knowledge Management, 20(02), p.2150027.

Chapter 1

Introduction

1.1 Introduction

The global datasphere is a notional environment in which data produced worldwide is contained. According to Reinsel et al. (2017), the global datasphere contained over 20 zettabytes of data (20 billion terabytes) in 2017. Its growth is expected to continue rapidly, and by 2025 it is hypothesised to contain 160 zettabytes of data. Such exponential growth is due, in part, to hardware developments and an increase in user availability and accessibility.

Data is now generated at a near constant and limitless rate, resulting from an array of devices, networks and everyday tasks such as credit card transactions and mobile phones (Aggarwal 2007, Mohammadi et al. 2018). Data arriving online and in a sequential, continuous fashion is known as a data stream. These data streams provide a potential source of valuable quantitative and qualitative data, providing it can be extracted in a timely manner. A multitude of machine learning techniques can be used to harvest interesting information from the stream in a process known as stream mining. However, the volume, velocity and temporal nature of the arriving data can cause complex challenges for stream mining.

The domain of stream mining has evolved significantly over the years since its inception. With traditional offline machine learning techniques such as a classification well established, researchers published methods for adapting these techniques to apply to online stream learning (Domingos & Hulten 2000). For example, in offline classification the datasets used are available in their entirety at all times whereas in an online streaming scenario data arrives sequentially and continuously. Since the size and velocity of such data streams are typically unknown and unbound it is infeasible to simply opt to

store all of the arriving data. Instead, algorithms for stream mining must be capable of processing streaming data using a one-pass methodology.

The evaluation of these developed online streaming algorithms also posed an original challenge for early research. In traditional offline scenarios where datasets are available on demand it is often common practice to isolate a portion of the data as a designated "test set" which is used to evaluate classifiers after they have been trained on the remaining data. In a streaming scenario this practically impossible since arriving examples from a stream are continuous, with the potential volume of arriving data being completely unknown. The most common way for evaluating online classifiers is through the "test then train" approach otherwise known as prequential evaluation (Ramírez-Gallego et al. 2017). This involves first testing base classifiers with arriving instances from a stream before training.

Stream mining's most prolific problem lies within the sub-domain of concept drift, with a large volume of research being dedicated to developing various algorithmic solutions over the past several years (Gama et al. 2014, Wares et al. 2019, Lu et al. 2019). Not only do data streams produce data in a sequential fashion at a potentially limitless rate which provides its own set of innate challenges, but the distribution of the streams themselves are also subject to change. Consider a situation of modelling customer purchasing behaviour as an example. A model trained to predict weekly sales for a clothing store might use attributes such as advertising costs, promotions and customer footfall. However, this model may become less accurate over time due to seasonality. In the summer months the model may function correctly. However, in the winter months the model is likely to classify sales as low since it has no understanding of the shift in concept due to the change in seasonality; winter months typically see less customers.

From the perspective of stream classification, concept drift introduces a severe problem. Algorithms must be capable of monitoring the distribution of a stream and have the ability to adapt, otherwise underlying base classifiers will become outdated, inaccurate and unreliable. This can render models used in real world scenarios virtually useless, just like the clothing store example. Researchers have proposed numerous and varied algorithms that provide a statistical means for monitoring and detecting concept drift. Established methods include ADWIN (Bifet & Gavalda 2007), CUSUM (Page 1954), DDM (Gama et al. 2004), EDDM (Baena-Garcıa et al. 2006) and the Page Hinckley test (Page 1954). More modern methods include ECHO (Haque, Khan, Baron, Thuraisingham & Aggarwal 2016), RDDM (Barros et al. 2017) and the collection of MDDM algorithms (Pesaranghader et al. 2018). Whilst the implementation of these techniques varies wildly, the overarching thematic architecture remains the same; once a drift

has been detected, the base classifier is reset or forced to "forget" the now outdated instances it was trained on.

While concept drift methods have attracted considerable research, a lack of established datasets has proved troublesome for the evaluation of such methods. Two popular datasets for concept drift evaluation are the Electricity (Harries & Wales 1999) and Forest Covertype datasets (Blackard et al. 1998). Alternative methods for evaluation include synthesised datasets through the use of tools such as MOA (Bifet, Holmes, Pfahringer, Kranen, Kremer, Jansen & Seidl 2010).

Bifet (2017) notes that in the presence of temporal dependence, that is in scenarios where arriving instances of a stream are not independent of their time of arrival, concept drift detection algorithms appear to achieve high levels of classification accuracy, even when the drift detector itself performs no statistical drift detection. The reason for this phenomenon is caused by the resetting of base classifiers when temporal dependence is present. This is problematic for the development and progression of the field of concept drift detection on two fronts. Firstly, it directly challenges the architecture and assumptions behind resetting base classifiers when drifts are detected. This is the primary way in which concept drift detection algorithms facilitate the "forget and adapt" behaviour of classifiers and Bifet (2017) suggests that this causes misleading classifier accuracy when temporal dependence is present in the data. Secondly, the lack of established datasets becomes a more pressing problem for the evaluation of new techniques that can handle temporal dependence and concept drift. Whilst this is already an issue in the domain of drift detection it is possible to use established data synthesisers for evaluation. However, no data synthesisers currently exist for simulating temporally dependent concept drift data.

1.1.1 Core Concepts and Themes

The work contained within this thesis discusses multiple different themes and concepts related to stream mining. This includes the different types of data involved such as static, streaming, evolving and temporal, as well as various datasets that contain different combinations of such types of data. Whilst the field of stream mining is not solely focused on classification tasks, this thesis does focus on classification and is therefore not concerned with other types of stream mining tasks such as regression. This section provides a high level illustration of the different themes and concepts related to stream mining that are discussed throughout this thesis.

Classification tasks are concerned with the prediction of class labels for some set of input data. In traditional machine learning scenarios the data is held offline and is available via some storage medium. The data itself is available in its entirety; the full

scale of the data is observable. Such data is referred to as batch or static data. In classification tasks for this type of data, a portion is usually reserved as testing data whilst the remainder is used as training data. The classifier is then typically initially trained using the training data before being evaluated using the testing data split.

In contrast to batch data, streaming data contains characteristics that are directly opposing. Where batch data is available entirely with known data dimensions and is fully traversable, streaming data is not. Streaming data arrives sequentially, resulting in its availability being limited at any given time. In addition, data dimensions and size are unknown. This causes implications in processing and storage capabilities. Since streaming data may potentially be unlimited in size, storing the data can prove expensive. Similarly the computational resources required for applying offline machine learning methods to streaming data can become exponentially vast. To combat this, specialised streaming algorithms have been developed specifically for applying machine learning techniques to streaming data. These algorithms are designed to computationally inexpensive, lightweight and have no requirement for storing the data. Chapter 2 provides more in-depth discussion.

Complications with streaming data and machine learning have become more apparent over recent years. In the early stages of research the focus surrounded the need for lightweight and adaptive classifiers that could cope with the challenges of streaming data in comparisons to that of batch data. However, as research progressed it was discovered that streaming data can also be evolving in some circumstances. When streaming data is evolving, the underlying features that inform the class label shift over time. The result of which is that classifiers can be become outdated if they do not keep up with changes in the distribution. This is what is referred to as concept drift.

Concept drift can manifest in various forms, such as gradual, sudden, incremental or recurring. Gradual drift is where a new concept gradually replaces an old concept over some extended time period. A good example of gradual drift is changes in fashion trends, where one emerging trend slowly erodes an older trend. Sudden drift is the stark opposite of gradual drift where a new concept very quickly replaces a old one. An example of sudden drift may include changes in stock prices due to global events like COVID-19. Incremental drift occurs when a new concept incrementally changes to a new concept over time. The stock exchange is a again good example of this, but instead of reacting suddenly to some event the stock price slowly increases (or decreases) over time. Finally recurring drift is where an old concept recurs after some time period, for example footfall in retail stores might be higher during weekends than during the week.

To combat concept drift various detection algorithms have been developed and proposed in research. These can be effectively categorised into statistical, window-based and ensemble approaches. At a fundamental level, concept drift detectors reset the base classifiers whenever a drift is detected. This means that historical information is wiped from the classifier and it is able to stay up-to-date with any shifts in the distribution caused by concept drift. However recent advances in research have shown that the existence of temporal dependence within streaming data can have a negative effect upon the evaluation of such methods.

Temporal dependence exists when the arriving class labels of a stream are not independent of the time of their arrival. This means that at different time periods across the stream, identical class labels arrive sequentially for some unknown amount of time. If during such temporal events the base classifier is reset, it will be trained on potentially only a single class arriving from the stream and as such when evaluated will appear to have achieved a high classification accuracy. In reality, while the classification accuracy may appear to be high, this is misleading since the classifier is oblivious to other instances that may arrive after the temporal dependence subsides. The current state-of-the-art research is only beginning to address this issue. New metrics for evaluating classifier performance in temporal environments have been proposed, however advances in algorithmic solutions for coping with temporal dependence in evolving data streams are yet to be developed.

A major challenge in furthering research in this field is the availability of datasets that encapsulate the various themes highlighted. Datasets that are not evolving and are not temporally dependent are in abundance; any offline dataset can effectively be streamed and used for evaluating research that is concerned with developing new streaming classifiers. Datasets become less available when concept drift is required. For research surrounding the development of new concept drift detection methods, there are very few established datasets that contain concept drift. As a solution, various synthesiser methods have been developed that simulate concept drift data. Finally datasets that contain both concept drift and temporal dependence are even scarcer, and at the time of writing this thesis there exist no simulation methods for producing evolving data streams that also contain temporal dependence.

The contributing chapters of this thesis propose novel solutions to the gaps in current research highlighted above. Chapters 4 and 5 propose original methods for handling temporal dependence in evolving data streams. Chapter 6 offers a unique data synthesiser that is capable of producing evolving data streams using any existing concept drift simulation method, but also injecting temporal dependence into the produced instances.

1.2 Research Question and Hypothesis

This project forms two research questions:

1. How can temporal dependence be accounted for during the classification of concept drift data streams?

The motivation behind this research question originates from the stagnation in development of novel methods for handling temporal dependence in concept drift data streams. The prevalence of temporal dependence in concept drift streams is a recently pressing issue, directly challenging both the reliability of existing concept drift methods and hindering the development of new methods. When temporal dependence is present in concept drift streams it can cause base classifiers suited for streaming data to produce misleading accuracy results when evaluated. A full critical, in depth explanation is provided throughout Chapter 2.

2. Can temporally dependent concept drift data streams be simulated in order to improve the means for evaluation?

This research question was constructed through the observation that evaluating methods designed for handling temporal dependence in concept drift streams is particularly difficult due to a lack of established benchmark datasets. Datasets used for the evaluation of concept drift detectors are limited, and the problem only compounds itself when extended to temporally dependent concept drift streams. For evaluating drift detectors this is often circumvented by using published data simulators for evaluating new techniques, however, at the time of writing, no data simulation methods exist for including temporal dependence in concept drift streams. Existing datasets and simulators are fully described in Chapter 2.

This research hypothesises that the development of an original algorithm that challenges the existing architectural relationship between concept drift detection and base classifier will provide positive, novel improvement for performing classification on temporally dependent concept drift data streams. It is also hypothesises that statistical adaptations can be made to existing data synthesisers that would facilitate the simulation of temporally dependent concept drift streams for evaluation purposes.

1.3 Aims and Objectives

The overarching aim of this research is to provide original contributing work towards coping with temporal dependence in concept drift streams and simulating temporal data to aid in evaluation. To achieve this, the project is split into two independent

aims with a corresponding set of objectives for each:

1. Design, implement and evaluate novel approaches to handling temporal dependence in concept drift data streams

   (a) Through an in-depth critical analysis of existing literature, review both established and recently developed methods of concept drift and temporal dependence.

   (b) By addressing the findings and shortcomings of existing literature identified in the objective above, design an original solution that contributes toward handling temporal dependence in data streams.

   (c) The produced solution should be statistically evaluated using established datasets and metrics as used in existing published research. This will also inform a critical discussion surrounding the performance of the developed solution; focusing on its strengths, weaknesses and potential future improvements

2. Design, implement and evaluate algorithmic data simulator capable of synthesizing temporally dependent concept drift data

   (a) Through an in-depth critical analysis of existing literature, review the existing popular datasets and simulators used in published literature.

   (b) With reference to the identified relevant gaps in literature design and implement a novel data simulator that is capable of simulating temporally dependent concept drift data streams.

   (c) The produced solution should be statistically evaluated using published metrics for monitoring levels of temporal dependence. This should also inform a critical discussion surrounding the performance of the developed solution; focusing on its strengths, weaknesses and potential future improvements.

1.4 Original Contributions

The contributions contained in this thesis are outlined as follows:

1. A novel algorithm for handling temporal dependence in concept drift streams is created. The proposed algorithm in this project challenges the historic assumption that classifiers should be reset each time a drift detector signals an occurring drift in a stream (Gama et al. 2014, Wares et al. 2019). Instead, this project suggests an algorithm that monitors the levels of temporal dependence to make

informed decisions surrounding classifier resetting. The proposed algorithm has been published as a journal paper (Wares et al. 2021).

2. An original ensemble method for performing stream classification in temporally dependent evolving data streams is presented. This technique uses the Kappa-Temporal statistic as a weighting mechanism for ensemble component replacement. This allows base classifiers within the ensemble to take temporal dependence into consideration when deciding which components to replace.

3. Introduce an original data simulation algorithm that is capable of introducing temporal dependence into concept drift data streams. This data simulation method offers various existing concept drift data generators the ability to also generate temporally dependent data. The severity of the temporal dependence is controllable through a series of defined parameters. Evaluating temporal methods is currently challenging due to a lack of datasets; this proposed simulation method offers a novel solution to this problem.

1.5 Project Methodology

A critical in depth review of published literature is conducted in order to identify existing gaps in research. This review primarily includes research in the fields of stream mining, classification, evaluation, concept drift and temporal dependence, but also extends briefly to domains such as time series analysis. The gaps identified in this review form the inspiration and concrete basis for the original work contained in this project.

Areas for further research identified from the reviewed literature inform the decision to develop original solutions that contribute towards solving these identified problems. The design and development process of this project utilises an experimental methodology. Statistical evaluation using appropriate metrics is conducted on the developed methods in order to compare their performance against existing published methods.

1.6 Thesis Outline

This thesis is divided into eight chapters. Chapter 1 contains an introduction to the thesis itself before providing some background to the research, in particular highlighting key gaps in existing published research that this thesis aims to address. This first chapter also states the two research questions involved in this thesis, the aims and objectives of the project, the research methodology used throughout and the original

contributions offered by this research.

The Chapter 2 offers an extensive literature review covering stream mining and its subdomains relevant to this research including concept drift, temporal dependence and evaluation. This literature review chapter provides background to support the aims and objectives of this thesis, particularly focusing on the lack of established datasets for evaluation and the need for a temporal simulation tool, as well as the problems caused by temporal dependence in the presence of concept drift. The chapter itself is based on the published literature review produced as part of this research (Wares et al. 2019), but with some adaptations.

In Chapter 3 an in-depth explanation of temporal dependence is provided. This includes statistical descriptions, examples of real world scenarios, an analysis of temporally dependent datasets and a discussion on the similarities and differences between class imbalance and temporal dependence.

Chapter 4 contains the methodology, design, implementation, experimental setup, evaluation and discussion of the developed method for handling temporal dependence. The proposed algorithm in this chapter is compared statistically in terms of its classifier accuracy to existing state-of-the-art drift detectors.

Chapter 5 suggests an original, ensemble based technique for performing stream classification in temporally dependent evolving data streams. This includes the methodology, design, implementation and experimentation process adopted for developing the proposed algorithm. The discussion and conclusions are drawn from an analysis of the performance of the method in comparison to other state-of-the-art ensemble methods.

In Chapter 6 the philosophy behind the temporal simulator is fully described alongside its implementation, experimental setup, evaluation and discussion. The developed simulator is capable of using existing concept drift data generators to produce data but allows for a customisable level of temporal dependence to be included. The performance of this developed simulator is compared by a statistical analysis of the temporal dependence levels at different parameter settings using various concept drift data generators.

Penultimately, Chapter 7 provides a reflection on the project as a whole. This includes project setbacks, successes and a general reflection.

Finally Chapter 8 discusses the conclusions of the project by referring to the research questions and project aims of Chapter 1. Areas for future research based on the results of this thesis are also suggested.

Chapter 2

Background Research

The work contained in this chapter has been published as a review paper in Springer SN Applied Sciences 2019 (Wares et al. 2019). This chapter provides a critical review of stream mining with a focus on stream mining challenges, concept drift detection algorithms and problems with their evaluative process such as a lack of established datasets. This research is primarily concerned with supervised data stream mining and drift detection methods. Literature and techniques involving unsupervised approaches, such as Sethi & Kantardzic (2017) and de Mello et al. (2019), are not reviewed. Unlike existing reviews such as Gama et al. (2014), modern, recent approaches to handling concept drift, as well as established methods, are discussed. Ditzler et al. (2015) provides a summary of the challenges and approaches for learning in both static and evolving data streams. Similarly, the recent review by Krawczyk et al. (2017) primarily focuses on ensemble methods, and only briefly highlights the most popular non-ensemble based techniques. Successful mining of data streams can potentially provide rich quantitative and qualitative information. Such information could have a tremendous impact on business practices across various industries, such as oil and gas where streaming data is abundant. However, because of the challenges posed by stream mining it is relatively unharnessed. This research suggests that the reason for stream mining not being fully harnessed is that the domain of concept drift detection is not progressing at a quick enough pace. For example, the domain of imbalanced data, in the context of stream mining, has produced various adaptations for algorithms to handle class imbalance, such as DDM-OCI (Wang et al. 2013), LFR (Wang & Abraham 2015), Learn++.NIE (Ditzler & Polikar 2010) and Learn++.CDS (Ditzler & Polikar 2013). The field of class imbalance in stream mining has also published recent reviews that provide timely suggestions for future research (Krawczyk 2016), whereas in contrast the reviews for concept drift are several years old (Gama et al. 2014) and do not encapsulate recent

advancements.

2.1 Introduction

Data streams possess specific and unique characteristics that differentiate them from other forms of data. In traditional machine learning contexts, the data is often referred to as "batch" data. That is, all of the data is immediately available in its entirety and is stored in memory. This is of stark contrast to stream mining, where data streams produce elements in a sequential, continuous fashion, and may also be impermanent, or transient, in nature (Babcock et al. 2002, Gama 2010). This means stream data may only be available for a short time. The difference between traditional methods and data streams is described by Babcock et al. (2002) in the following ways:

1. Data elements in the stream arrive in real-time.
2. The system has no control over the order in which data elements arrive to be processed, either within a data stream or across data streams.
3. Data streams are potentially unbound in size.
4. Once an element from a data stream has been processed, it is discarded or archived. It cannot be retrieved easily unless it is explicitly stored in memory, which is small relative to the size of data streams.

The unique characteristics of a data stream contribute to the challenges in processing its arriving elements. Since batch data is persistent, it can be queried once in its entirety and individual data elements can be accessed at random. Data streams however, since transient, must be queried continuously by the algorithm. The data elements in the stream cannot be accessed at random; they can only be accessed in the sequence in which they arrive from the stream. The key differences between processing traditional batch data and stream data are shown in Table 2.1.

Data streams are either *static* (sometimes referred to as stationary) or *evolving*, and are classified depending on the condition of their core distribution.

Static data has an underlying distribution that does not shift over time. That is, the features that define the target label for learning remain constant and consistent. Static datasets are frequent in traditional machine learning contexts where features defining ground truth labels do not change (Chu & Zaniolo 2004).

Unlike static data, the distribution for an evolving data stream may change over time. Feature vectors may change over a time period t such that mapping from the feature vector to the class label becomes obsolete. Aggarwal (2007) describes how data streams

Table 2.1 Batch Data vs Streaming Data

Batch Data	Stream Data
Offline	Real-time
Persistent data	Transient data
Process entire data	Process samples of data
Constant availability	Limited availability
Complex techniques used if required	Linear techniques widely used
Fixed size	Unbound in size
Random access	Sequential access
Known data characteristics	Unpredictable data characteristics

possess an inherent temporal component that causes them to be time dependent by nature. This shift in distribution over time is known as *concept drift*.

In the domain of machine learning, traditional applications implement batch learning techniques on static datasets. Batch learning approaches involve having the entirety of the training data available at any given time. The data can be processed once or multiple times before an algorithm produces an output decision.

Data streams by nature are incompatible to batch learning for a number of reasons. Most obviously where traditional applications have all of the data available immediately, data streams must be mined in a distributed fashion since examples arrive continuously and in a sequential manner. In contrast to batch and multi-pass learning, stream mining algorithms must be designed to work with one pass of the data only (Aggarwal et al. 2003, Aggarwal 2007).

A solution to the ineffectiveness of batch learning may at first seem obvious; translate batch learning algorithms into one-pass variants. However, the innate temporal nature of data streams may render this solution redundant as a one-pass conversion approach may not consider the evolution of the underlying distribution. Concept drift is something stream mining algorithms simply must take into consideration.

Since data streams are unbound in size, the volume and velocity (Tran et al. 2014) can result in the imposition of hardware limitations. The most obvious of which is memory; it is not feasible to continuously explicitly store stream elements since the instance limit is almost always unknown. A second hardware limitation is processing resources. The speed at which elements arrive from a stream, as well as the stream size, can quickly consume available resources. As such, stream mining algorithms should be both computationally fast and lightweight (Chu & Zaniolo 2004). A concise description of stream mining and its popular subdomains is provided by (De Francisci Morales et al. 2016).

2.2 Stream Mining Applications

Stream mining is an attractive domain for businesses since it offers unparalleled access to a near limitless supply of quantitative and qualitative data. This data can be leveraged by both researchers who aim to improve stream mining algorithms and further the research field, as well as companies who may look to this data to increase profit, efficiency, customer satisfaction or gather diagnostic data. These use cases are just some examples of where stream mining can have an impact on industry, and how organisations can adopt stream mining methods for their own gain and advantage.Some examples of real world scenarios where stream mining might have an influential impact are itemised below. Whilst this research focuses on the scientific aspect of stream mining and proposes novel statistical methods, it is important to address that these developments are important to real world applications and have potential industry use, and are not just for academic purposes. The work of (Krempl et al. 2014) provides numerous real world applications of stream mining, as well as outlining the problems and considerations associated with them.

1. Stream mining offers the potential to be adopted as an automated method for aiding threat detection in various forms, such as malicious network traffic or URL detection (Khan & Fan 2012). By deploying online classifiers to observe network traffic in a real-time scenario, it becomes possible for malicious packets to be detected quickly and to mitigate both attacks and the corresponding damage. Utilising online stream classifiers in this manner offers real world potential for organisations to improve and increase the security and monitoring of their resources.

2. Road traffic management, as suggested by (MOHAMED et al. 2010), is another real world scenario where online classifiers and streaming data can prove invaluable. Sensor networks which monitor traffic flow along major routes can produce real-time data describing attributes of traffic flow such as average speed, density and vehicle type. Such data can be used to monitor and control traffic delays by dynamically controlling road speeds or opening and closing lanes. Examples of this do exist in parts of society, such as smart motorways in the UK.

3. Gaber (2012) states that the demand for the ability to perform stream mining on mobile platforms is ever increasing. The applications for mobile stream mining may initially appear elusive initially, but real world mobile solutions to problems have been developed. MobiMine (Kargupta et al. 2002) is a mobile stream mining system which performs stock market monitoring. VEDAS (**VE**hicle **DA**ta **S**tream (Kargupta et al. 2004) provides a solution for monitoring the health and

wellbeing of drivers and vehicles in a fleet. OMM (Open Mobile Miner) offers a generic toolkit for broad spectrum mobile mining (Krishnaswamy et al. 2009). Since mobile platforms typically possess less processing power than that of a desktop or server device, employing load shedding techniques can aid in reducing the processing demand (Babcock et al. 2003, Tatbul et al. 2003).

4. Privacy preservation is an niche yet interesting topic in the domain of stream mining. Since data is produced online and often originates from personal devices such as mobile phones, it is important to ensure that data being mined does not breach or compromise personal privacy (Aggarwal & Philip 2008, Al-Hussaeni et al. 2014). In traditional scenarios with offline or batch data, privacy concerns can be alleviated by removing identifiable information from the data before publishing.

2.3 Stream Mining Toolkits

Developing algorithms and applications that perform stream mining is a challenging and complicated process, however there are publicly available packages for popular programming languages. For example, the popular statistical programming language R has packages such as Stream (Hahsler et al. 2017), RMOA (Wijffels 2014) and StreamMOA(Hahsler et al. 2015) available, and Python offers its ever popular statistical package SciPy (Virtanen et al. 2020). However with the constant evolution of techniques available for stream mining, additional software packages became inadequate. Instead, independent stream mining frameworks were developed that were updated frequently to keep up with developments and cutting edge research in the context of stream mining. These frameworks offer a means for developing, testing and evaluating methods for stream mining across various programming languages. An outline of these is given in Table 2.2. The selection of the most appropriate framework to be used for development lies predominantly with the software language individual researchers are most comfortable with. The research contained within this thesis opts to use MOA as a framework for the development of the novel methods proposed herein. It is worth noting that of the frameworks given in Table 2.2, all of them are open source apart from RapidMiner. RapidMiner has adopted a semi open source model where the core of the application is open source, but it still maintains some proprietary aspects.

2.4 Concept Drift

Traditional machine learning algorithms operate with the assumption that the data distribution is static. For data streams, the distribution of arriving examples may change

Table 2.2 Stream Mining Frameworks

Framework	Language/Platform	Reference
MOA	Java	(Bifet, Holmes, Pfahringer, Kranen, Kremer, Jansen & Seidl 2010)
Scikit-Multiflow	Python	(Montiel et al. 2018)
StreamDM	Spark	(Bifet, Maniu, Qian, Tian, He & Fan 2015)
RapidMiner	Java	(Kotu & Deshpande 2014)
S4	Java	(Neumeyer et al. 2010)

over time due to the stream's innate temporal nature. This renders traditional batch learning algorithms unsuitable for applications that learn from data streams.

In reality, data streams produce copious amounts of data at a near constant rate. Gama (2010) provides examples of such streams, including surveillance systems, sensor networks and telecommunication systems. However, with recent modern hardware developments these streams are produced by new devices such as smart household appliances and car navigation systems. Algorithms that seek to learn from data streams must be able to accurately model the underlying distribution. The ability to detect and continuously adapt (Aggarwal et al. 2004) to changes in the distribution of examples is paramount for data stream mining algorithms.

The shift in the underlying distribution of examples arriving from a data stream is referred to as concept drift. Concept drift occurs over time and the rate at which the drifts occur varies. It can be responsible for various symptoms, including previous examples to become irrelevant; their distribution no longer accurately reflects their corresponding class label. This is reflected in the clothing store sales prediction example previously mentioned. If seasonality causes clothing sales to be higher in the summer months, then examples from winter months may not accurately reflect class labels. It may be the case that the model predicts low sales, but due to seasonality there are less people shopping and the sales are in fact high for a winter season. As such, models must be capable of forgetting previous examples once concept drift has occurred.

2.4.1 Definition

A learning algorithm observing examples with a stationary distribution would observe training examples in the form $(\vec{x}_i, y_i)$, where $\vec{x}_i$ is the feature vector and $y_i \epsilon \{c_1, c_2 ... c_n\}$ for the ith example. A class prediction at a particular time stamp t would be given as $\hat{y}_t$ based on the feature vector $\vec{x}_t$.

In contrast, a data stream may produce examples with a non-stationary distribution.

The stream may initially consist of examples $e_i = (\vec{x}_i, y_i)$, similar to that of a static distribution. However, if the distribution should shift at some point in time then this may no longer hold true. According to Gama et al. (2014), concept drift between time t_0 and t_1 can be defined as:

$$p_{t_0}(\vec{x}_i, y_i) \neq p_{t_1}(\vec{x}_i, y_i), \tag{2.1}$$

where p_t is the joint distribution at time t between the feature vector $\vec{x}_i$ and the target class label y_i.

Kelly et al. (1999) states that concept drift may occur in three distinct ways. Firstly, the class prior probabilities $p(y)$ may change over time. Secondly, the class distributions $p(\vec{x} \mid y)$ may change over time. And thirdly the class posterior distribution $p(y \mid \vec{x})$ may change. From the point of view of classification, only $p(\vec{x} \mid y)$ changes would affect the prediction and therefore require an algorithmic response.

The rate at which concept drift occurs can be categorised as one of three distinct forms; sudden drifts, gradual drifts and recurring drifts. Brzezinski & Stefanowski (2014) concisely describes these three types of concept drift; sudden drifts occur when the source distribution is suddenly replaced by another distribution entirely, gradual drifts occur at a much slower rate, and recurring drifts occur when older concepts reappear after some time period. Drifts can also be described as incremental where the drift consists of many intermediate changes, for example a network sensor deteriorates and becomes less accurate (Gama et al. 2014).

Similarly to categorising the rate of change, concept drift itself can be defined as either *real* or *virtual* (Gama et al. 2014, Zliobaite 2010, Widmer & Kubat 1993). Real concept drift refers to changes in $p(y \mid \vec{x})$, a change in the probability of a class label y given feature vector $\vec{x}$. This can result in classifier decision boundaries becoming affected. Alternatively, virtual drift refers to changes in $p(\vec{x})$ but not in $p(y \mid \vec{x})$. The result in this case is that the distribution has changed but the decision boundaries of the classifier are unaffected. Figure 2.1 illustrates the differences between real and virtual drift, where the directional arrow portrays the difference between changes in $p(\vec{x})$ and $p(y \mid \vec{x})$.

As is described by Zliobaite (2010), it is not important if the drift is real or virtual since $p(y \mid \vec{x})$ is dependant on $p(\vec{x} \mid y)$. Explicit characterisation of various types of concept drift is effectively illustrated by Webb et al. (2016).

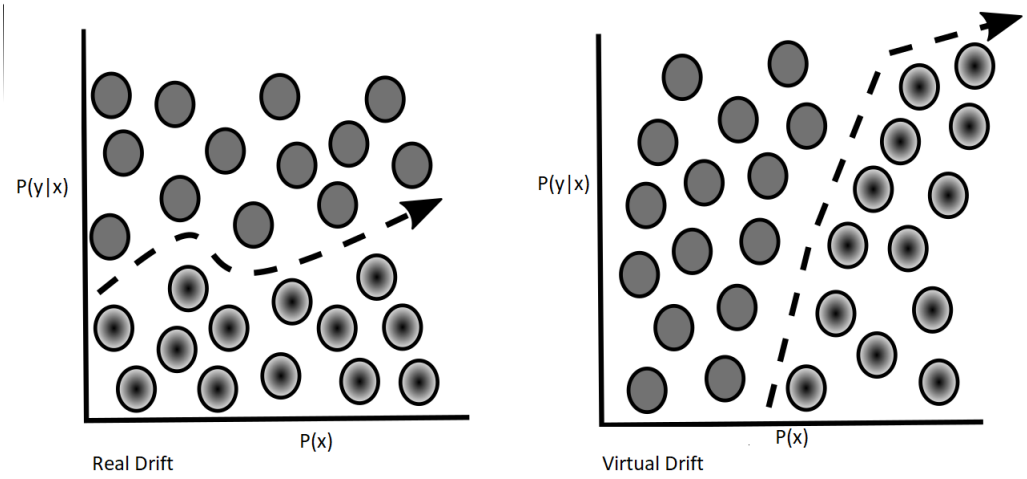

Figure 2.1: Difference between real and virtual concept drift

Where stream mining algorithms bring with them their own set of challenging algorithmic requirements, algorithms for concept drift detection must also meet certain demands. The following points are considered to be critical challenges that concept drift detection algorithms should overcome (Gama et al. 2013, 2014):

1. Detect as soon as possible the point at which the distribution has changed.
2. The crossover period during a shift in concept can produce noise. For example as distribution D_0 shifts to D_1, examples produced by D_0 will act as noise for D_1.
3. Algorithms should be computationally faster than the arrival time of examples from the stream. They should also be lightweight enough to not consume more than some fixed amount of memory for storage. Note that this is true for streaming algorithms in general and not specifically concept drift detection methods.

Any classifier operating with stream data must contain mechanisms to meet these requirements, otherwise their predictive performance will diminish over time. The predictive model will likely have to be capable of updating with new data as it arrives, or even replacing itself entirely.

2.5 Drift Detectors

Methods that focus on the detection of concept drift during online classification can be broadly categorised into four main categories. These are statistical based, window based, block based ensemble and incremental based ensemble. This categorisation can be seen in Table 2.3, which provides a full illustration of the categorisation and

techniques covered by this chapter.

Table 2.3 Summary of Concept Drift Detection Techniques

Type	Algorithm	Reference
Statistical based	CUSUM	Page (1954)
	PH	Page (1954)
	DDM	Gama et al. (2004)
	EDDM	Baena-Garcıa et al. (2006)
	RDDM	Barros et al. (2017)
	LFR	Wang & Abraham (2015)
	HLFR	Yu & Abraham (2017)
	STEPD	Nishida & Yamauchi (2007)
	FPDD	de Lima Cabral & de Barros (2018)
	FSDD	de Lima Cabral & de Barros (2018)
	FTDD	de Lima Cabral & de Barros (2018)
	MDDMs	Pesaranghader et al. (2018)
	MDDM-A	Pesaranghader et al. (2018)
	MDDM-G	Pesaranghader et al. (2018)
	MDDM-E	Pesaranghader et al. (2018)
Window based	CVFDT	Domingos & Hulten (2000)
	E-CVFDT	Liu et al. (2013)
	ADWIN	Bifet & Gavalda (2007)
Block-based Ensembles	SEA	Street & Kim (2001)
	AWE	Wang et al. (2003)
	AUE	Brzeziński & Stefanowski (2011)
	SAND	Haque, Khan & Baron (2016)
	ECHO	Haque, Khan, Baron, Thuraisingham & Aggarwal (2016)
Incremental-based Ensembles	DWM	Kolter & Maloof (2003)
	Learn++	Polikar et al. (2001)
	Learn++.MT	Muhlbaier et al. (2004)
	Learn++.NC	Muhlbaier et al. (2009)
	Learn++.NSE	Muhlbaier & Polikar (2007)
	Learn++.NIE	Ditzler & Polikar (2010)
	Learn++.CDS	Ditzler & Polikar (2013)

2.5.1 Statistical Methods

The Sequential Probability Ratio Test (SPRT) (Wald 1973) is the backbone to a number of algorithms for concept drift detection. Given two distributions P_0 and P_1 for time period w, should the underlying distribution shift from P_0 to P_1 then the probability of observing elements from P_1 should be higher than the probability of observing elements from P_0. The statistical test is given as:

$$T_w^n = log\frac{P(x_w...x_n \mid P_1)}{P(x_w...x_n \mid P_0)} = \sum_{i=w}^{n} log\frac{P_1[x_i]}{P_0[x_i]} = T_w^{n-1} + log\frac{P_1[x_n]}{P_0[x_n]} \tag{2.2}$$

Introduced by Page (1954), the Cumulative Sum (CUSUM) is a statistical technique based on the SPRT and is commonly adopted for concept drift detection. It receives as an input the residual of any filter, for example a Kalman filter, and outputs an alarm when the mean of the input data differs greatly from zero. Bifet (2017) gives CUSUM as

$$\begin{aligned} &g_0 = 0 \\ &g_t = max(0, g_{t-1} + \epsilon_t - v) \\ &\text{if } g_t > h \text{ then alarm and } g_t = 0, \end{aligned} \tag{2.3}$$

where ϵ_t is the current observed value, v is the allowed magnitude of change, t is the current time and h is a parameter defined threshold. This expressions functions for detecting changes that occur in a positive direction. If changes in a negative direction are required for detection then the $min()$ function should be used in place of $max()$.

CUSUM is memoryless in the sense that the probability of a drift being detected is not related to a drift having already been detected. It is also worth noting that the accuracy of CUSUM is dependent on the parameters v and h. Low values of v enable faster detection rates but at the cost of an increased rate in false positives.

The Page-Hinckley (PH) test, also proposed by Page (1954), is a variant of the previously mentioned CUSUM test. The PH test can detect changes in the average behaviour of a process. The PH test for an increasing signal can be given as (Bifet 2017):

$$\begin{aligned} &g_0 = 0 \\ &g_t = g_{t-1} + (\epsilon_t - v) \\ &G_t = min(g_t, G_{t-1}) \\ &\text{if } g_t - G_t > h \text{ then alarm and } g_t = 0, \end{aligned} \tag{2.4}$$

where ϵ_t is the current observed value, v is the allowed magnitude of change, t is the current time and h is a parameter defined threshold. If the signal is decreasing then $G_t = max(g_t, G_{t-1})$ and $G_t - g_t > h$ should be utilised as the stopping rule instead. Similarly to CUSUM, the PH test is memoryless but its accuracy is again parameter dependent on the values of v and h. Larger values of h will result in a lower false alarm rate, but some changes may also be missed altogether.

While the two algorithms are similar, they do offer solutions for different data streaming scenarios. Since CUSUM uses the residual from any predictor as an input, it is well suited for various applications of stream mining. CUSUM has been recently used for anomaly detection in video streams by Yang et al. (2018). PH is instead ideally suited for detecting abrupt changes in signal processing environments, but has also been adopted recently for the development of a rule learning algorithm for regression (Duarte et al. 2016).

The performance of drift detection algorithms based on SPRT is often reliant on their false alarm and missed detection rates, as noted by Gama et al. (2014). However, during evaluative procedures such measures are usually overlooked as metrics to evaluate drift detector performance. This is explained fully in Section 2.5.4 of this research. The primary drawback and impact on performance is both algorithms' reliance on the parameters v and h. Both CUSUM and Page-Hinckley are considered state-of-the-art drift detectors in this category.

Another statistical method for detecting concept drift is based around monitoring the class distribution's constancy over time, as described by Brzezinski & Stefanowski (2014). This is undertaken by adopting various statistical techniques that produce "alarms" when the class distribution begins to change as time passes.

Drift Detection Method (DDM) (Gama et al. 2004) is a method that statistically compares two windows and controls the errors produced by a learning model during prediction. One window contains all of the data, and the second window consists of only data from the start of the stream to the point at which the error rate of the prediction model increases. The windows are not kept in memory, only statistical information and recent errors are stored.

It is assumed that the error rate will decrease as the number of examples for observation increases, so long as the distribution is stationary. It is therefore suggested that a significant increase in the error rate of some learning model would indicate a change in the class distribution. DDM involves two principle variables in the form of p_t and s_t, where p is the probability of misclassification, s is the standard deviation and t is time of arrival. The standard deviation s_t is given as $s_t = \sqrt{p_t(1-p_t)/t}$. When $p_t + s_t$

reaches its minimum value the following conditions are checked:

- $p_t + s_t \geq p_{min} + 2 \cdot s_{min}$ as a warning level. Examples are stored in preparation of contextual drift.
- $p_t + s_t \geq p_{min} + 3 \cdot s_{min}$ as drift level. Concept drift is assumed to exist, the learning model is reset and a new model is trained using examples stored since the warning level was triggered. p_{min} and s_{min} are also reset.

DDM is almost memoryless, only statistics p_t and s_t are stored alongside the necessity of some available memory to store examples for retraining. A major flaw with DDM is that is is only suitable for the detection of abrupt drifts. Gradual drifts can cause examples to be stored in memory for lengthy time periods which has the potential to cause catastrophic memory overflows.

A number of algorithms have been built upon DDM and have aimed to improve its performance. The most famous of these is the Early Drift Detection Method (EDDM) (Baena-Garcıa et al. 2006). EDDM uses the same approach and heuristics as DDM, however, rather than monitoring error rates, the distances between errors is measured. As predictions improve, the distance between two misclassification errors should increase. The window resizing follows the same procedure as DDM. The fundamental drawback to EDDM is that a fixed number of at least 30 errors are required for calculation which causes issues when applying this to imbalanced datasets. A more modern proposal to improve DDM is that of Barros et al. (2017), who suggest the Reactive Drift Detection Method (RDDM) which discards older instances of particularly gradually occurring drifts in order to overcome potential memory overflows.

The DDM-OCI algorithm (Wang et al. 2013) was developed to solve the problem with using EDDM and imbalanced datasets. DDM-OCI makes the assumption that for imbalanced datasets, drifts only occur when there is a change in the minority class recall during classification. However it is entirely possible that a drift can occur without affecting the minority class recall, for example a drift from data that has an unbalanced class distribution to one which is balanced. DDM-OCI also suffers from an issue where a number of false positives can be triggered due to a weakness in its test statistic $\hat{R}^t_{TPR}$.

With DDM-OCI falling short of overcoming the class imbalance problems present in EDDM, the Linear Four Rates framework (LFR), was proposed by Wang & Abraham (2015). It is designed as a direct improvement over the DDM-OCI algorithm. LFR monitors the four values, or rates, given by a typical confusion matrix; precision and recall for both minority and majority classes. Statistical bounds are set as thresholds

and should any of the four rates exceed the threshold then a drift is assumed to have occurred.

The current, most recent advancement in these algorithms is the Hierarchical Linear Four Rates method (HLFR) proposed by Yu & Abraham (2017), given in Algorithm 1. HLFR operates using a two layer, hierarchical structure wherein the first layer is responsible for detecting potential drifts, and the second layer validates said drift and communicates this information back to the first layer. Layer one monitors the same four rates of the confusion matrix as was introduced by LFR. When a drift is detected, layer two applies a permutation test to confirm if the detected drift is true or a false positive. In the occurrence of a false positive the testing process restarts.

Algorithm 1 HLFR Algorithm

Require: Data $\{X_t, y_t\}_{t=0}^{\infty}$ where $X_t \in R^d, y_t \in \{0,1\}$
Ensure: Concept drift time points $\{T_{cd}\}$

for each $t = 1$ to ∞ **do**
  Perform Layer-I hypothesis testing
  if Layer-I detects potential drift point T_{pot} **then**
    Perform Layer-II hypothesis testing on T_{pot}
    if Layer-II confirms the potentiality of T_{pot} **then**
      $\{T_{cd}\} \leftarrow T_{pot}$
    else
      Discard T_{pot}; Reconfigure and restart Layer-I
    end if
  end if
end for

Experimental results using real datasets indicate that HLFR outperforms DDM, EDDM and LFR in terms of not only accuracy but also in terms of its time to detection (detection delay). The framework presented with HLFR symbolises a move away from the traditional "concept drift detector plus classifier" approach. This current method suffers from various evaluation problems, discussed in depth in Section 2.5.4. The proposed framework of HLFR is a concrete starting point for future research that aims to address such issues.

DDM and EDDM are considered state-of-the-art statistical based detectors (Bifet 2017), even though they have drawbacks in relation to imbalanced data. Algorithms that have aimed to address this, such as DDM-OCI and LFR have fallen short. The result of which is that two, aged drift detectors that are sub-par in terms of performance are still considered state-of-the-art. A second reason is due to the datasets used in concept drift experimentation. This is explained fully in Section 2.5.4, but a lack of benchmark datasets means that many experiments used simulated data in

which parameters can be defined to avoid class imbalance, thus artificially avoiding the drawbacks of DDM and EDDM.

STEPD (Nishida & Yamauchi 2007), or Statistical Test of Equal Proportions, monitors the predictions of a base classifier for drift detection in a similar manner to that of DDM and EDDM. However, STEPD also uses two parameters as significance levels to distinguish between detected drifts and warnings. These are $\alpha_d = 0.003$ and $\alpha_w = 0.05$ respectively. STEPD uses two windows to compare the result of the classifier. The first window is a "recent" window, of which its size is defined by a parameter with a default value of 30 instances. The second window is the "older" window which contains all instances observed since the last detected drift. STEPD compares the accuracies of these windows through a hypothesis test of equal proportions, given as

$$T(r_o, r_r, n_o, n_r) = \frac{|r_o/n_o - r_r/nr| - 0.5 \times (1/n_o + 1/n_r)}{\sqrt{\hat{P} \times (1 - \hat{P}) \times (1/n_o + 1/n_r)}} \tag{2.5}$$

where r_o is the number of correct predictions from examples within the n_o "older window", r_r is the correct predictions from examples within the n_r "recent" window and $\hat{P} = (r_o + r_r)/(n_o + n_r)$. The result is used to calculate the p-value from the standard normal distribution table and is then compared against α_d and α_w to determine if a drift or warning alarm must be issued. If p-value $< \alpha_d$ then a drift is detected, similarly if p-value $< \alpha_w$ then a warning is signalled.

The authors recognised that for small sample sizes the statistical test of equal proportions was ineffective, admitting that Fisher's Exact test (Fisher 1992) should have been used but was ignored due to its high computational cost. Recent work by de Lima Cabral & de Barros (2018) proposes three new methods using the Fisher's Exact test, Fisher Proportions Drift Detector (FPDD), Fisher-based Statistical Drift Detectors (FSDD) and Fisher Test Drift Detector (FTDD). All three methods use the same approach as STEPD in regards to the two windows and the significance thresholds. However, these approaches use different statistical tests and measure the difference in errors rather than correct predictions.

FPDD uses the Fisher's Exact test when the number of errors or correct predictions in either window is smaller than five, otherwise it operates exactly like STEPD. FSDD extends FPDD such that instead of using the test of equal proportions in situations where the number of errors or correct predictions in either window is smaller than five, the chi-square test for homogeneity of proportions is applied (Chernoff & Lehmann 1954) FTDD explicitly uses the Fisher's Exact test for drift detection. Experimental results showed that all three proposed methods outperformed STEPD, with very little difference between themselves.

The McDiarmid Drift Detection Methods (MDDMs) proposed by Pesaranghader et al. (2018) uses a weighting scheme to give substance to elements of a sliding window to enable faster concept drift detection. MDDM applied McDiarmid's inequality (McDiarmid 1989) to detect drifts in evolving streams. The algorithm uses a sliding window of size n which stores prediction results. If the prediction is correct, a 1 is inserted into the window, otherwise a 0 is inserted. Each element in the window is weighted such that $w_i < w_{i+1}$. This means that elements at the head of the window have larger weights than that of the tail. Three methods based on different weighting schemes are presented. MDDM-A which uses the arithmetic weighting scheme, MDDM-G which uses the geometric weighting scheme and MDDM-E which uses the Euler weighting scheme. The arithmetic weighting scheme for MDDM-A is given as

$$w_i = 1 + (i - 1) \times d \tag{2.6}$$

where $d \geq 0$ is the difference between two consecutive weights. The geometric scheme used by MDDM-G is given as

$$w_i = r^{(i-1)} \tag{2.7}$$

where $r \geq 1$ is the ratio of two consecutive weights. Finally, the Euler scheme adopted for MDDM-E is given by the authors as

$$r = e^{\lambda} \tag{2.8}$$

where $\lambda \geq 0$. Prediction results are processed sequentially and the weighted average of all elements within the window is calculated alongside each arriving prediction result. This is used to update two variables, μ_w^t the current weighted average and μ_w^m the maximum weighted average observed thus far. A drift is detected if there is a significant difference between μ_w^m and μ_w^t. The significance is determined by McDiarmid's inequality. The authors selected popular concept drift datasets for experimental testing, including Elec2, Forest Covertype and Poker Hand. These are explained in Section 2.5.4 of this chapter. Their results found that MDDMs outperformed existing methods including EDDM, CUSUM and Page-Hinckley in terms of drift detection delay and classification accuracy.

2.5.2 Window-based Detectors

Concept drift detection algorithms that are window-based operate by analysing instances arriving at different intervals. Rather than monitoring arriving instances from a stream individually, sliding windows of varying sizes, also known as widths, are instead statistically monitored (Widmer & Kubat 1996). Larger windows correlate with higher

performance accuracy, however they may also contain concept drift within themselves that could escape unnoticed. Smaller windows tend to facilitate better concept drift detection. The sizing of a window is a fundamental problem when designing any stream mining algorithm that utilises sliding windows. Figure 2.2 provides an illustration of how a sliding window operates.

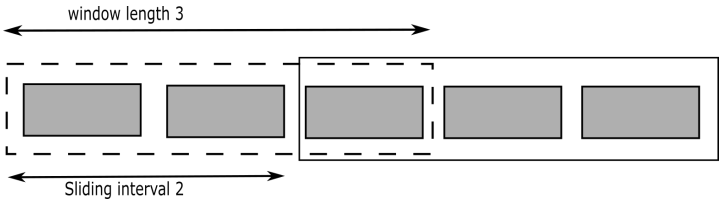

Figure 2.2: Sliding window. Borders identify two different windows.

The Hoeffding Tree is a mathematically justified algorithm used to construct decision trees. An in-depth, statistical coverage of decision tree based classification is given by Breiman (1984). The Very Fast Decision Tree (VFDT) is a heuristic algorithm proposed by Domingos & Hulten (2000) that is based on the Hoeffding Tree. The VFDT is an algorithm which incrementally constructs a decision tree of incoming examples from a data stream, without the need to store examples in memory. A critical difference between the Hoeffding tree and traditional classification trees lies in the selection of which node is used for splitting. Where traditional trees adopt techniques such as Information Gain (Quinlan 1986) to determine the best node for splitting, more recent proposed methods such as (Rutkowski et al. 2015) have suggested new splitting criteria for the construction of decision trees. VFDTs use the Hoeffding bound to determine how many examples are necessary to identify the best splitting node based on a user defined confidence threshold. The Hoeffding bound states that, for a random variable r in range R and where $\bar{r}$ is the mean of n observations, with probability $1-\delta$ the true mean is at least $\bar{r}-\epsilon$ where ϵ is given as

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{2.9}$$

It is also worth noting that the Hoeffding bound has been suggested to be statistically inappropriate for constructing decision trees for data stream mining (Rutkowski et al. 2012), (De Rosa & Cesa-Bianchi 2015), (Jaworski et al. 2017). Hoeffding bound has also been used as the core basis for drift detection methods (Frías-Blanco et al. 2015). VFDTs are, however, only suitable for static streams and include no method for forgetting or restarting learning in the presence of concept drift. In order to enable VFDTs to account for concept drift, Hulten et al. (2001) introduced Concept-adapting Very Fast Decision Tree (CVFDT). CVFDT monitors a sliding window of examples. As new examples arrive, CVFDT updates its node statistics by incrementing counts of

the new, arriving, examples. Counts that relate to the oldest example in the window, which must be forgotten, are decremented. If necessary, older examples are removed from the window. Hulten et al. (2001) states that if the concept is changing then nodes that previously passed the Hoeffding test may no longer pass. In this case, CVFDT grows a second sub-tree with the new best attribute, according to the Hoeffding bound, at its root. If the new sub-tree's accuracy outperforms that of the older tree then it replaces it completely.

Criticising CVFDT for not offering mechanisms which handle specific types of drift, such as gradual or abrupt, Liu et al. (2013) proposed the E-CVFDT algorithm which utilises a caching mechanism. Their results show that E-CVFDT yields a higher classification accuracy for gradual concept drifts, but does not make any improvements during sudden drifts.

The Adaptive Sliding Window (ADWIN) algorithm (Bifet & Gavalda 2007) is another extensively used, window-based detector for coping with concept drift. Assuming a stream of examples $x_1, x_2..., x_n$, produced by some distribution at time t, these serve as inputs to ADWIN to produce sliding window W. Let $\hat{\mu_w}$ denote the average of examples contained within W and μ_w represent the unknown average of μ_t such that $t \in W$. Let n be the length of W and n_0 and n_1 be the lengths of W_0 and W_1 respectively, such that $n = n_0 + n_1$. Algorithm 2 provides the algorithm for ADWIN.

Algorithm 2 ADWIN Algorithm

```
1: Initialise window W
2: for each t > 0 do
3:    W ← ∪W{x_t} (add x_t to head of W)
4:    repeat
5:       Drop elements from the tail of W
6:    until |μ̂_{W_0} − μ̂_{W_1}| > ε_cut holds for every split of W into W = W_0 · W_1
7:    output μ_W
8: end for
```

Whenever two "large enough" subwindows of W display "distinct enough" averages, a drift is assumed to have occurred, and the older of the two subwindows is dropped. The terms "large enough" and "distinct enough" are defined by the Hoeffding bound statistic. The average of the two subwindows are tested to determine if they are larger

than ϵ_{cut}, given as $|\mu w_0 - \mu w_1| > 2\epsilon_{cut}$ where

$$\begin{aligned} m &= \frac{1}{1/n_0 + 1/n_1} \\ \delta' &= \frac{\delta}{n} \\ \epsilon_{cut} &= \sqrt{\frac{1}{2m} \cdot ln\frac{4}{\delta'}} \end{aligned} \tag{2.10}$$

This was considered computationally expensive since all "large enough" subwindows are checked for potential cuts. The window contents are also explicitly stored and memory requirements scale linearly with the window size. This has the obvious drawback of potentially large memory and processing requirements.

A solution to the resource demands of ADWIN is proposed in the form of ADWIN2, provided in Algorithm 3.

Algorithm 3 ADWIN2 Algorithm

```
1: for each time t do
2:   for all partitions of W into W_1 + W_2 do
3:     if W_2 compromises exactly a number of buckets then
4:       if test (W_1, W_2, δ) ≥ ε(|W_1|, |W_2|, δ) then
5:         declare change and drop the older bucket
6:       end if
7:     end if
8:   end for
9: end for
```

In order to reduce the computational time when determining the best cutting point in the window, buckets are adopted as a means for grouping data within the window. Such buckets have two core elements; capacity and content. Each time a new example arrives from the stream, if the element is "1" then a new bucket is created of *content 1* and *capacity* equal to the number of elements arrived since the last observed "1". The remaining buckets are then compressed.

In order to eliminate the problems of high memory demands caused by the explicit storing of all window contents, a variation of the exponential histogram (Datar et al. 2002) is used. The performance of ADWIN2 is given by the authors in Big O notation as $O(logW)$ memory and cutpoints.

ADWIN2 is usually referred to directly as ADWIN in published work and is considered one of the state-of-the-art concept drift detectors. A common problem with sliding windows is that the width usually needs to be predefined. Varying the width of the window has an impact on performance, thus applying the correct width value is important.

Typically this is done by means of some user-defined parameter. ADWIN, however, is parameterless and the sliding window is sized dynamically by the algorithm itself. It also provides excellent performance due to the use of buckets and the adaptation of the exponential histogram for compression.

2.5.3 Block-based Ensemble Detectors

In machine learning, an ensemble refers to a group or collection of classifiers that work together to achieve greater predictive performance. Block-based ensembles process data in blocks, or chunks, of some specified size. The performance of block-based ensemble methods is based heavily on the chunk size. Similarly to that of sliding windows, larger chunks tend to produce more accurate classifiers but may contain concept drift within themselves. Alternatively, smaller based chunk sizes are typically more effective at drift detection but produce inferior performing classifiers. Ensembles can incur high performance costs due to the sheer number of base classifiers used in any ensemble, and this has the potential for weakening their effectiveness in a real world scenario (Zhang et al. 2015).

The Streaming Ensemble Algorithm (SEA) was first proposed by Street & Kim (2001) and is a block-based ensemble learning algorithm. Individual classifiers are constructed from examples read in sequential blocks (chunks), which are then added to a fixed size ensemble. If the ensemble is full, then the worst performing classifier is removed from the ensemble entirely.

In their experiments, C4.5 (Quinlan 1993) classifiers are used for building the ensemble. The output prediction is given as the simple majority voting of the entire ensemble. Results for testing with concept drift showed that the algorithm was capable of recovering quickly by discarding classifiers trained on the outdated data.

A notable drawback to SEA is the way in which classifiers are replaced. Merely replacing the worst performing classifier with the most recently trained has the potential to still leave several, outdated and poorly performing classifiers in the ensemble, depending on the predetermined ensemble size.

This is improved upon by Wang et al. (2003)'s Accuracy Weighted Ensemble (AWE), a block-based algorithm which trains a new learning model with each new chunk of data in a similar fashion to that of SEA. Where AWE improves upon SEA is in the model replacement. AWE uses a version of the mean square error to select n best classifiers to construct an entirely new ensemble, thus removing all other outdated and poorly performing classifiers. The algorithm for AWE is given in Algorithm 4.

Let S denote a data stream in chunks $S_1, S_2, ..., S_n$ where each chunk is of equal size

Algorithm 4 AWE Algorithm

Input: S a dataset of $ChunkSize$ from the stream
K the total number of classifiers
C a set of previously trained classifiers
Output: C a set of K classifiers with updated weights

1: Train classifier C' from S
2: Computer error rate of C' via cross validation on S
3: Derive weight w' for C'
4: **for** each classifier $C_i \in C$ **do**
5: apply C_i on S to derive MSE_i
6: Compute w_i
7: **end for**
8: $C \leftarrow K$ of the top weighted classifiers in $C \cup \{C'\}$
9: return C

and C_i represent some classifier for S_i. The weight of classifier C_i is the estimated prediction error using the most recent data S_n. Since S_n is a data stream and will produce examples in the form $(\vec{x}, c)$ where $\vec{x}$ is the feature vector and c is the class label, the classification error of C_i is $1 - f_c^i(x)$ where $f_c^i(x)$ is the probability that x is an example of class c. As such, the mean square error of C_i is given by

$$MSE_i = \frac{1}{S_n} \sum_{(x,c) \in S_n} (1 - f_c^i(x))^2 \tag{2.11}$$

Should a classifier predict randomly then the mean square error can be given as:

$$MSE_r = \sum_c p(c)(1 - p(c))^2 \tag{2.12}$$

A random classifier contains no meaningful knowledge of the data, it makes predictions simply at random. Therefore MSE_r is used as a threshold for weighting and classifiers whose error rate is at least equal to MSE_r are discarded. The weight of a classifier C_i is given as:

$$w_i = MSE_r - MSE_i \tag{2.13}$$

One drawback to AWE is the issue of chunk-size optimisation, however it should be noted that this is common in all block-based ensemble methods. A second drawback is the weighting function of AWE, in particular the MSE_r threshold. In environments with sudden concept drift it can have a silencing effect on the entire ensemble resulting in no class prediction (Brzeziński & Stefanowski 2011).

Brzeziński & Stefanowski (2011) proposed the Accuracy Updated Ensemble (AUE) algorithm as an improvement to that of AWE. The algorithm for AUE is given in Algorithm 5.

Algorithm 5 AUE Algorithm

Input: S a data stream, k the number of ensemble classifiers
Output: $\mathcal{E}$ an ensemble of k online classifiers with updated weights

$C \leftarrow 0$
for (all data chunks $x_i \in S$) **do**
  train classifier C' on x_i
  compute error rate MSE of C' on x_i
  derive weight w' for C' using (3)
  for (all classifiers $C_i \in C$) **do**
    apply C_i on x_i to derive MSE_i
    compute weight w_i based on (3)
  end for
  $\mathcal{E} \leftarrow k$ of the top weighted classifiers in $C \cup \{C'\}$
  $C \leftarrow C \cup \{C'\}$
  for (all classifiers $C_e \in \mathcal{E}$) **do**
    if $w_e > \frac{1}{MSE_r}$ and $C_e \neq C'$ **then**
      update C_e with x_i
    end if
  end for
end for

AUE implements online classifiers enabling the individual learning models to be updated directly rather than only adjusting weights as per AWE. If no concept drift were to occur between a series of chunks then the classifiers would improve as if they were trained on one large chunk. This means the block size can be reduced without risking the performance accuracy of the ensemble classifiers. The weighting function in AUE is a simplified version of that used in AWE, and is given as

$$w_i = \frac{1}{MSE_i + \epsilon}, \tag{2.14}$$

where MSE_i is calculated identically as it is in AWE and ϵ is a small constant value to allow weighting calculations when MSE_i is equal to 0.

Experimental results showed that AUE performed more accurately than AWE on all similar datasets apart from one where the performance accuracy was equal. A second implementation of this algorithm, AUE2, was suggested by Brzezinski & Stefanowski (2014) which improves on the memory usage and accuracy of AUE by implementing a new weighting function and pruning base learners.

AUE overcomes the problems present in AWE. The weighting function is redesigned to

cope with sudden concept drifts, and the use of online classifiers allows smaller chunk sizes to be used without a severe reduction in classifier accuracy. However when no concept drift is occurring, all classifiers are updated with arriving chunks. Should this continue over multiple chunks then the outcome is that ensemble classifiers lose their uniqueness.

Recent advances in concept drift detection using block-based ensembles have introduced new algorithms entirely. The Semi-supervised Adaptive Novel Class Detection and Classification over Data Stream (SAND) framework is proposed by Haque, Khan & Baron (2016). SAND consists of four independent modules; Classification, Novel Class Detection, Change Detection and Update.

The framework maintains an ensemble of classifiers based on k-nearest neighbour, using algorithms such as k-means. The ensemble is initially trained on some training data. When an instance arrives from some stream it is classified using majority voting. It also produces a confidence value which indicates the ensemble's confidence in the prediction. These confidence values are stored in a sliding window.

The Change Detection module monitors the distribution of confidence values within the sliding window. Any significant change in the distribution is assumed to be caused by the existence of concept drift. Once change has been detected, a new chunk of data is used to update the ensemble and the chunk boundaries are determined dynamically. Updating is undertaken by requesting only class labels for instances in the current chunk where the confidence values were weak. The ensemble is then updated with the new model and the sliding window is reset.

Experimental results showed that SAND was capable of achieving good prediction accuracy using limited labelled data, however its execution time was inefficient due to the high resource cost of the Change Detection module being executed after the calculation of individual confidence scores.

In order to attempt to remedy the poor execution time of SAND, the Efficient Concept Drift and Concept Evolution Handling over Stream Data (ECHO) was proposed by Haque, Khan, Baron, Thuraisingham & Aggarwal (2016). ECHO operates in the same manner as SAND, however, the execution of the Change Detection module is selective rather than at each calculation of the confidence threshold. Two methods of selectively invoking the Change Detection module. The first is to use a fixed threshold γ, such as the classifier confidence threshold. If the confidence of a test instance C^x is less than γ, the Change Detection module is invoked. The second proposed approach is to calculate the probability of invocation based on the C^x. A high confidence value would result in a low probability of invocation.

ECHO performs competitively and is suitable for use in environments which generate a low level of labelled data. However, in stream mining there is an innate assumption that class labels are always available with arriving instances. Whilst this may not be the case in real world scenarios, since concept drift detection is still in its infancy it follows that this assumption can continue to be made. Classifiers are only updated by labels for which there was a low confidence value. While this does aid in situations where labels are missing by lowering the demand for labels, there is no existing mechanism for actually determining if a label is available or not. The result is that classifiers may not be updated with information when there is an available class label, which will negatively impact the potential performance of the ensemble.

2.5.4 Incremental Ensemble Detectors

Incremental, or online, ensembles are another method of ensemble based learning. In contrast to block-based ensembles, incremental ensembles process elements sequentially rather than in chunks.

Dynamic Weighted Majority (DWM) was first proposed by Kolter & Maloof (2003). DWM is an ensemble of classifiers referred to as 'experts', where each is given an associated weight. For some test example, the experts each provide a prediction. This is then used in combination with their weights to output the overall prediction in the form of the class which has the largest accumulated weight total.

Should an individual expert provide an incorrect prediction, then its corresponding weight is reduced. If the output prediction of DWM is incorrect then a new expert is created and is assigned a weighting of 1. Experts are normalised by uniform scaling such that the highest weighting possible is 1. Experts with a weight lower than a user-defined threshold value are removed. Through the use of uniform weights and incremental learning, the authors state that the DWM algorithm is capable of handling concept drift.

DWM is considered one of the state-of-the-art concept drift detection methods, and has been a benchmark algorithm in recent studies reviewing concept drift (Gama et al. 2014, Zliobaite et al. 2015, Bifet 2017). However, one particular problem with DWM is the way in which experts are added. Rather than adding a new expert when the ensemble prediction is incorrect, the age of experts and historical prediction accuracy could be taken into consideration. The base learner also explicitly maintains examples in memory which has the potential to consume large amounts of resources, depending on the stream size. Other algorithms such as CVFDT and ADWIN have already solved this issue so it follows that similar implementations could be made to DWM.

Learn++ Algorithms

The Learn++ algorithm family is a set of algorithms consisting of an ensemble of incrementally trained classifiers using batches of data and weighted majority voting. According to Liao et al. (2016) existing algorithms in the Learn++ family include Learn++, Learn++.NC, Learn++.MT, Learn++.NSE, Learn++.NIE and Learn++.CDS. Elwell & Polikar (2011) also mention the Learn++.MF algorithm as part of the family.

The original Learn++ algorithm, suggested by Polikar et al. (2001), constructs k classifiers for a single batch of incoming data. Examples from this batch are used to train a single, first classifier. Prediction errors are used to produce a weighted distribution of all examples, with misclassified examples possessing a higher probability of being sampled. Training the second classifier through to the k classifier for the ensemble, training examples are selected based on the weighted distribution of all examples. Classification errors are then used to update the weighted distribution.

One problem with Learn++ is that all base classifiers are persisted over time T, resulting in old data never being forgotten by the ensemble. This can cause a problem known as 'outvoting'. Older classifiers in the ensemble may produce incorrect predictions due to the aged examples they are trained on. If the outdated classifiers make up the majority of the ensemble, even if their weights are small, they can 'outvote' the classifiers trained on newer data. Thus through the majority weighted voting procedure, produce incorrect predictions. Another problem is that without forgetting old information, concept drift cannot be accounted for.

Older algorithms in the Learn++ family sought to provide solutions to the concept drift problem. Learn++.MT (Muhlbaier et al. 2004) solves the outvoting problem by using a dynamic weighted voting technqiue. Learn++.NC (New Class), proposed by Muhlbaier et al. (2009), furthers this concept and introduces a Dynamically Weighted Consult and Vote (DW-CAV) mechanism which enables incremental learning of new classes. Learn++.NC enabled base classifiers within the ensemble to consult among themselves when classifying a given example, and weights the decision of each base classifier. Classifiers check their predictions against classes which they are trained and, based on the decision of other classifiers, check if their prediction is in-line with others. If a classifier's decision is an outlier compared to the majority, the classifier may either reduce it's voting weighting or withdraw from predicting altogether. These algorithms, however, are only suitable for static, stationary, environments where the distribution does not change.

Learn++.NSE, proposed by Muhlbaier & Polikar (2007), aims to account for various

forms of concept drift. An ensemble of classifiers is trained on the current data distribution D at time t. Change is monitored by examining the performance of the ensemble over time. Learn++.NSE will generate a new classifier and combine it with the ensemble when the prediction error of the current ensemble falls below some threshold. Classifiers are weighted according to the time t they were instantiated, such that newer classifiers have a larger weighting than older classifiers during prediction.

Whilst Learn++.NSE aimed to address the issue of evolving data, none of the existing Learn++ algorithms accounted for class imbalance at this stage. Learn++.NIE (Ditzler & Polikar 2010) extends Learn++.NSE, but incorporates evaluation measures for data class imbalance, such as f-measure. Learn++.NIE also implements sub-ensembles in place of individual classifiers in order to reduce stochastic errors. An alternative approach was proposed by Ditzler & Polikar (2013), whose Learn++.CDS algorithm instead uses preprocessing with SMOTE, an oversampling method, rather than changing the evaluation metric to account for class imbalance. SMOTE adds instances to the minority class in order to create a more balanced dataset.

The comparative performance of the Learn++ algorithms was reviewed by Liao et al. (2016). The performance of each algorithm is dependent heavily on the base classifier used. This was especially apparent in environments with imbalanced data and incremental learning. The current state of the Learn++ algorithm family requires considerable work to produce solutions that can cope with concept drift and imbalanced data, although discussion of the latter is out of scope for this research. Learn++.NSE provides a starting point for using the Learn++ family for concept drift detection. However it is outdated and its approach weighting favours newly created classifiers during prediction which is a flawed approach; it is entirely feasible that older classifiers in an ensemble may be better equipped to make predictions than newer classifiers.

2.6 Datasets and Evaluation

In traditional machine learning scenarios, the typical evaluation procedure is to train a model, cross-validate and then test using metrics such as prediction accuracy or f-score. For stream mining this approach is ineffective. Since stream data arrives online in a continuous and sequential fashion, it is not possible to first train the model and then test. Instead one of two methods can be used; prequential evaluation, sometimes referred to as interleaved-test-then-train, or holdout evaluation.

Prequential evaluation is implemented using the following procedure. For each arriving element from a stream the model is first tested by predicting the class label, after which the same element is used to train the model. Prequential evaluation can be used in

conjunction with sliding windows and decaying factors to improve classification results in evolving data streams. A full comparative assessment is given by Gama et al. (2013) and (Bifet, de Francisci Morales, Read, Holmes & Pfahringer 2015) also provides further analysis of the prequential method.

The holdout evaluation procedure offers an alternative approach. This involves withholding a subset of data examples from the classifier to be used as a training set at specific time intervals, for example every one hundred thousand instances. Algorithm 6, adapted from Bifet & Kirkby (2009), shows an example algorithm for holdout evaluation. The most obvious problem for holdout is the acquisition of examples for use as training data. A solution to this is to systematically store arriving samples from the stream at varying intervals. Similarly, ascertaining the adequate number of examples to provide accurate evaluation measurements also poses a challenge. It is suggested by Bifet & Kirkby (2009) that a test set in the region of tens of thousands of examples is sufficient, however, this is an enormous potential range and doesn't provide a concise estimate.

Algorithm 6 Holdout Evaluation Framework

m_{bound}: the maximum memory allocation for the model
n_{test}: Holdout examples as a test set
n_{train}: Training examples arriving from the stream
while Evaluation is required **do**
  for $i = 1$ to n_{train} **do**
    Get next example e_{train} from the stream
    Train and update the model, ensuring m_{bound} is valid
  end for
  for $i = 1$ to n_{test} **do**
    Get next example e_{test} from the test set
    Test the model using e_{test} and update model accuracy
  end for
end while

Krempl et al. (2014) states that a problem with evaluating stream mining classifiers in general is a lack of benchmark datasets for cross comparison. Instead, datasets are often synthesised using tools such as MOA (Bifet, Holmes, Pfahringer, Kranen, Kremer, Jansen & Seidl 2010). This is also a problem for evaluating concept drift detection algorithms. Few benchmark datasets exist for testing concept drift detectors. The most popular dataset used for evaluating concept drift detection algorithms is the Electricity Dataset (Harries & Wales 1999). This dataset is used in various concept drift related research publications (Kolter & Maloof 2003, Gama et al. 2004, Zliobaite 2013, Bifet 2017). This dataset was taken from the Australian New South Wales electricity market, in which electricity prices are not statically set. Instead, the price fluctuates

according to demand. The dataset is constructed of 45312 electricity prices which were taken at 30 minute intervals. Examples are labelled as either "UP" or "DOWN" which reflect their current price in comparison to the last 24 hours. Another popular dataset is the Forest Covertype (Blackard et al. 1998) dataset. This consists of 581012 instances of 54 attributes that describe various types of forest cover of the Roosevelt National Forest in northern Colorado. The KDD'99 dataset is well known and subject to somewhat extreme temporal dependence. This dataset contains information pertaining to simulated intrusions in a military network environment. It contains 23 class labels representing either normal traffic or some form of intrusion. The dataset contains over 494,000 records of 41 features each. This dataset is of considerable age, but due to the high level of temporal dependence has been used in recent studies such as Zliobaite et al. (2015). The Poker Hand dataset from UCI Machine Learning Repository (Dua & Graff 2017) contains one million records of 11 attributes that represent a poker hand of five cards from a standard 52 card deck. Each card has two corresponding attributes, the suit and the rank, and there is one additional attribute that describes the hand, e.g. royal flush or full house. A dataset similar to Poker Hand that contained more class labels was used in the work of Cattral et al. (2002). The Airlines dataset from Data Expo 2009 (Ikonomovska 2008) contains 120 million records of 13 attributes relating to flight departure and arrival information from internal commercial flights in the USA between October 1987 and April 2008. The target class label is the arrival delay given in seconds, and the classification goal is to determine the flight delay time given the arrival and departure information. This dataset has been used in the work of Ikonomovska et al. (2011).

Another fundamental problem in evaluating drift detectors stems from the use of classifier accuracy as an evaluation metric. This has been criticised in recent literature (Bifet, Read, Zliobaite, Pfahringer & Holmes 2013, Zliobaite et al. 2015, Bifet 2017) where it has been suggested that classifier accuracy doesn't reflect the performance of the concept drift detector. Bifet (2017) explains that drift detectors should be evaluated in terms of their ability to handle false alarms, their true detection rate and the time taken to correctly identify an occuring drift. This is reflected in evaluation criteria proposed by Basseville et al. (1993) and Gustafsson & Gustafsson (2000), which are given in Table 2.4. These are existing, historic metrics which capture properties of concept drift detectors, however, at the time of writing there appears only the work of Bifet, Read, Pfahringer, Holmes & Zliobaite (2013) utilises these metrics for evaluation.

Table 2.4 Drift Detector Evaluation Metrics

Metric	Explanation	Formula
MTFA	Mean Time to False Alarms. Frequency of false alarm triggers.	$E_{\theta_0}(t_a)$
FAR	False Alarm Rate.	$1/MTFA$
MTD	Mean Time to Detection. How quickly occurring drift is identified.	$E(t_a - t_0 + 1 \mid t_a \geq t_0)$
ARL	Average run length. Time to alarm after change of size θ.	$E(t_a - k \mid$change of size θ at time $k)$

These metrics have existed for over a decade, yet are not used in published work. One possible reason for this is a lack of frameworks which support these metrics. Each must be independently calculated when implementing models, which is time consuming and can increase complexity. Since prediction accuracy is readily available in virtually all machine learning frameworks, it's of no surprise that this metric is used to evaluate the impact of change detectors. This is only bolstered by the idea that in most instances the performance of the change detector itself may be viewed as unimportant; only the performance of the classifier truly matters.

The lack of an existing evaluation framework was an issue that was addressed by Bifet, Read, Pfahringer, Holmes & Zliobaite (2013) where the authors propose CD-MOA, a GUI extension to MOA (Massive Online Analysis). CD-MOA offers an interface for evaluating change detectors. However, the evaluation measures provided are again different. CD-MOA provides information on time and memory resources, as well as a metric called RAM-Hours which merges both time and memory together. The most recent version of CD-MOA also provides a measure based on Cohen's Kappa statistic (Cohen 1960), which compares observed accuracy with some expected accuracy.

A further issue with the metrics of Basseville et al. (1993) and Gustafsson & Gustafsson (2000), as given in Table 2.4, is their numerousness. Having five independent statistical measures to evaluate a drift detector obfuscates a true representation of performance. In order to tackle this, Bifet (2017) proposed a new single metric, the Mean Time Ratio (MTR), which encapsulates the ratio between MTFA and MTD metrics. The motivation of this was that the MTFA and MTD metrics are arguably the two most important, thus providing a single metric representing these eliminates the confusion of having five metrics. The MTR metric is given as follows.

$$MTR(\theta) = \frac{MTFA}{MTD} \times (1 - MDR) = \frac{ARL(0)}{ARL()} \times (1 - MDR) \tag{2.15}$$

Ultimately, the problem with current evaluation metrics is a lack of agreement on the most effective and suitable metrics, and the absence of gold-standard techniques. The metrics presented in Table 2.4 represent performance characteristics of a change detector, but numerousness and a lack of frameworks has left them virtually unused. The MTR metric aims to eliminate the problem of numerousness by providing a single metric which combines both the mean time to false alarms and the mean time to detection. CD-MOA provides a framework for evaluation, but provides a different set of metrics within its GUI for evaluating drift detectors. The result is that drift detectors are evaluated based on their impact to classifier accuracy, however this doesn't truly evaluate the concept drift detector itself. Further research should aim to establish a standardised set of statistical evaluation metrics that encapsulate the various performance characteristics of a drift detector, as well as its impact on classifier accuracy.

Existing statistical measures focus on the evaluation of baseline classifiers. The Kappa statistic proposed by Cohen (1960) is a statistical measure for evaluating classification results when using imbalanced data , both in the context of streaming data and in traditional batch learning. The Kappa statistic is defined as

$$k = \frac{P - P_{ran}}{1 - P_{ran}}, \tag{2.16}$$

where P is the accuracy of some base classifier and P_{ran} is the accuracy of a classifier that predicts labels at random.

A possible direction for future research is to design and produce new statistical evaluation criteria. Krawczyk et al. (2017) suggests that metrics such as memory consumption, update time and decision time of drift detectors should be taken into account for evaluation. New metrics should not focus solely on the predictive accuracy of the base classifier but incorporate performance factors of the drift detector. A potential start would be to find a suitable statistical combination to incorporate classifier accuracy with the metrics given in Table 2.4. This would provide a new, harmonious statistical measure that represents both the drift detector and classifier performance.

2.7 Temporal Dependence

The work of Bifet (2017) not only suggests that accuracy is a poor metric for evaluating drift detectors, but also that the existence of temporal dependence within datasets that contain concept drift is the cause of the false alarm phenomenon found with the superior

performance of the No Change detector. In this thesis this is referred to as the "over-resetting problem".

Temporal dependence is defined by Zliobaite et al. (2015) as "observations that are not independent from each other with respect to time of arrival" (Zliobaite et al. 2015, p. 459). In other words, an arriving stream element is not independent from the preceding element in regards to its time of arrival. Temporal dependence itself is not a new issue, and is a known problem in the field of time-series analysis - as is concept drift (Beck et al. 1998, Box et al. 2015, Cavalcante et al. 2016). However, its effects upon stream classification and concept drift are relatively unexplored. At the time of writing there exists very little work in the context of handling temporal dependence during stream mining and concept drift detection. The emergence of temporal dependence in streaming data has started to spawn new research, such as using temporal dependence in streaming data to assist in change detection using a Candidate Change Point model (Duong et al. 2018).

In a typical streaming scenario, arriving elements are assumed to be independent such that the class labels y_t are dependent on the features vectors x_t. When temporal dependence exists, the class labels are not independent and therefore are likely to be dependent on the previously seen labels. In other words arriving instances are dependent on their time of arrival. Temporal dependence is given mathematically by Zliobaite et al. (2015) as:

$$P(y_t, y_{t-1}) \neq P(y_t)P(y_{t-1}), \tag{2.17}$$

where t is some timestamp and y are class labels. The authors note that this is known as *first order* temporal dependence, since only the immediately previous label is used for observation. Temporal dependence of the lth order observes the previous l labels. Temporal dependence for a class label is positive if

$$P(y_t, y_{t-1}) < P(y_t)P(y_{t-1}) \tag{2.18}$$

or negative if the inverse is true.

One metric for monitoring temporal dependence presence in a data stream is the Kappa-Temporal statistic (Zliobaite et al. 2015) , which is defined as

$$k_{per} = \frac{P - P_{per}}{1 - P_{per}}, \tag{2.19}$$

where P_{per} is the probability of a Persistent classifier, a classifier that simply predicts

that the next class label will be the same as the immediately previously known class label. Using this measure, trained classifiers performing correctly will achieve a k_{per} score of 1, or if performing worse than the Persistent classifier P_{per}, a score of 0. The substantial drawback to the Kappa-Temporal measure is the direct inverse to the Cohen's Kappa statistic described above. Kappa-Temporal is ineffective for imbalanced datasets since a Majority class classifier, a classifier that simply predicts the class with the largest prior probabilities, will outperform that of a Persistent classifier.

Zliobaite et al. (2015) offer a solution to this problem by combining both Cohen's Kappa statistic and the Kappa-Temporal statistic together, forming the Combined Measure. This is given as

$$K^{+} = \sqrt{max(0, k)max(o, k_{per})} \tag{2.20}$$

This Combined Measure will provide a statistical evaluation score of 0 if either the Kappa or the Kappa-Temporal metrics fail. This provides a single evaluation metric that encapsulates a classifier's ability to cope with both temporal dependence and imbalanced data. However, it is only a statistical metric and does not offer any mechanism for base classifiers to handle temporal dependence during the classification process.

Zliobaite et al. (2015) propose two approaches to account for temporal dependence in the context of stream classification. The first approach, labelled as the Temporal Correction classifier, assumes a model of temporal dependence which is used to formulate an expression for estimating the posterior probabilities. The authors consider only first order dependence in this proposal, and give the expression for estimating the maximum posterior probability as:

$$\frac{P(y_t = i \mid y_{t-1})}{P(y_t = i)} P(y_t = i \mid X_t) \tag{2.21}$$

While this approach is simplistic, it only accounts for first order temporal dependence. While the assumption that the previous label will be known is commonly made in stream classification, any delay or error in the arrival of labels will negatively impact the performance of this method.

The second method proposed by the authors is described as the Temporally Augmented classifier. In contrast to the Temporal Correction classifier, this method relies solely on preprocessing techniques. The approach involves augmenting the observation feature vector X with previously seen labels. A classification model is then trained using these augmented vectors. The prediction $\hat{y}_t$ is then given as:

$$\hat{y}_t = h_t(X_t, y_{t-1}, ..., y_{t-l}), \tag{2.22}$$

Table 2.5 Persistent Classifier Performance

Dataset	Persistent Classifier Accuracy
Electricity	85.33%
Forest Covertype	95.06%

where h_t is a classification model that estimates the posterior probabilities and l is the length of temporal dependence orders. This approach is not limited to the assumption of first order dependence, as with the first proposed model. However there still exists the assumption that the previous labels will always be known.

As noted by Zliobaite et al. (2015), their approach to handling temporal dependence with the Temporally Augmented classifier is simplistic, and that it is often outperformed by a Persistent classifier; a classifier that predicts that the next arriving class label is the same as the previously seen class label. This finding was also stated by Bifet (2017).

Table 2.5 shows the results of a trained Persistent classifier on both the Electricity and Forest Covertype datasets. In both cases, particularly in the Forest Covertype dataset, the Persistent classifier outperforms the state-of-the-art classifiers using Temporally Augmented classifier to account for temporal dependence, according to the results of Bifet (2017). Simply predicting the next label will be the same as the last seen label produces higher predictive accuracy than handling temporal dependence using the Temporally Augmented approach.

The proposed Temporally Augmented approach for handling temporal dependence also only aids the baseline classifier. It does nothing to allow the drift detector itself to account for temporal dependence. The false alarm phenomenon occurs when drift detectors are subject to temporal dependence within the data, as described by Bifet (2017). As such, it should be accounted for at the drift detection level. The current state-of-the-art technique for handling temporal dependence provides no mechanisms for coping at drift detector level, it only offers a basic wrapper for baseline classifiers. Further research is required to investigate the development of new drift detection techniques, or augmentations to existing drift detection solutions, that can account for temporal dependence in streaming data.

Further evidence of the need for additional evaluation criteria out with of classifier performance is shown by Bifet (2017). A "No Change" detector is compared to state-of-the-art drift detectors using a Naive Bayes classifier with both the Electricity and Forest Covertype datasets. The No Change detector is a drift detector that performs no statistical monitoring of the stream data but instead outputs a false positive change

every 60 instances. The results of this show that the No Change detector outperforms state-of-the-art detectors in terms of accuracy on both datasets. This reinforces the concept that the use of accuracy as a metric for the performance of concept drift detectors is insufficient.

2.8 Summary

Stream mining is a challenging problem but has valuable potential yields, especially in industry and commercial applications. Data streams offer an untapped source of qualitative and quantitative information that could be used in a multitude of different ways to boost businesses in terms of profit and efficiency. However the unbound size, unknown speed and varying characteristics of data streams make applying machine learning techniques a complex task. Whilst online classifiers capable of processing streaming data have been proposed, the task is further obfuscated by concept drift. Evolving data streams with concept drift have a distribution that shifts over time, and at varying rates of severity. Classifiers must be capable of handling concept drift by forgetting outdated information when a shift in distribution occurs.

The in-depth literature review provided in this research shows that multiple approaches for handling concept drift are available. Statistical methods monitor the underlying distribution over time, signalling alarms when a drift has been detected. Window based methods use sliding windows to detect occurring drifts rather than monitoring the whole distribution. Ensemble based detectors handle concept drift by replacing outdated classifiers within ensembles through some form of a weighted voting mechanism.

Through an in-depth, critical review of existing literature, the background research of this thesis has identified the following shortcomings:

1. State-of-the-art drift detection require improvement.
2. A lack of benchmark datasets for evaluation.
3. Evaluation metrics assess the classifier rather than the drift detector, with a general over reliance on classifier accuracy.
4. The inability of drift detectors to cope with additional data anomalies, such as temporal dependence.

For the purposes of this research, these highlighted shortcomings have served as primary motivators for the definition of this project's aims and objectives, as stated in Chapter 1. In order to address these shortcomings, the following points address each of the identified shortfalls above in turn, providing direction for future research within the

domain of stream mining and concept drift detection.

The current state-of-the-art consists of algorithms that are now somewhat dated, such as ADWIN (Bifet & Gavalda 2007), and have known flaws in them. While there have been a number of attempts to further the state-of-the-art, many methods still suffer from substantial drawbacks. Statistical based methods such as DDM and EDDM have known issues in terms of their ability to handle varying types of drifts and in producing high rates of false alarms. Recent approaches such as DDM-OCI and LFR have aimed to solve these problems, but have fallen short. Window-based approaches like E-CVFDT still falter under sudden concept drifts. Block-based ensemble methods such as AWE and AUE are subject to dependency on the chunk size and weighting mechanisms. Incremental-based algorithms such as DWM could be improved by not explicitly storing instances and changing the statistical requirements for adding new classifiers to also consider classifier age and performance history.

A lack of benchmark datasets for evaluating is another crucial shortfall. As a result of a lack of benchmark datasets, it is common for datasets to be simulated using generators to account for the lack of benchmark datasets. While simulating data does work, the number of available generators is substantial and each often relies on various user specified parameters. The selection of the most suitable generator and corresponding parameters for a particular problem is open to interpretation. Future research should aim to produce gold-standard datasets, which would provide a collection of agreeable, accepted datasets to be used for experimentation.

Existing literature has exposed drawbacks in the metrics commonly used for evaluation drift detectors (Bifet, Read, Zliobaite, Pfahringer & Holmes 2013), (Zliobaite et al. 2015), (Bifet 2017). Some existing proposed metrics are given in Table 2.4, but these are particularly historic and are virtually unused in published work. This study suggests the reason for this is an issue of numerousness coupled with a lack of existing frameworks which incorporate these metrics. This research proposes that future research should aim to develop new statistical measures that capture performance properties of the drift detector and also potentially combine these with performance attributes of the baseline classifier to provide harmonious, statistically relevant metrics. An example of this is the Mean Time Ratio metric proposed by Bifet, Read, Pfahringer, Holmes & Zliobaite (2013), however this only represents the trade off between the average time to false alarms and true change detection.

The final suggestion for future research is concerned with enabling drift detection algorithms to cope with other data anomalies such as temporal dependence. This chapter has discussed and portrayed the problem of temporal dependence and its impact of

drift detectors. The current-state-of-the-art for handling this is merely a wrapper for the baseline classifier that augments the feature vector of arriving instances. However, temporal dependence should be handled at the drift detector level. The role of the classifier is well established in machine learning contexts; it is not classifier's responsibility to handle anomalies in the data stream. Concept drift is an anomaly that occurs in real-time data streams, and as such concept drift detection algorithms have been developed to work in conjunction with classifiers. It follows that since temporal dependence is also a data anomaly, this should be handled by specifically designed algorithmic solutions that can cope with both temporal dependence and concept drift. The structural framework proposed by Yu & Abraham (2017) in HLFR is of particular interest in this context and forms a good starting point for future research in this field.

Chapter 3

Exploring Temporal Dependence

3.1 Introduction

Chapter 2 gave an insight into temporal dependence and its current state within the existing literature. In the previous chapter the statistical definition, available metrics for evaluation , the "over-resetting problem", and the need for the availability of more datasets for evaluation were all clearly outlined.

In this chapter, further context and explanation is provided in relation to each of the aforementioned aspects of temporal dependence. The aim of this chapter is to provide a discussion and a statistical in-depth analysis of the key fundamentals of temporal dependence.

An overview of temporal dependence alongside examples of real-world scenarios is provided foremost. A statistical analysis of known temporally dependent datasets its provided, clearly highlighting the correlation between class label and its time of arrival from the stream. This also highlights clear parallels with the problem of a lack of established datasets for evaluation, as discussed in Chapter 2. Finally this chapter offers a comparative discussion regarding the similarities and differences between imbalanced data and temporally dependent data.

3.2 Overview and Examples

As mentioned, temporal dependence is considered to be occurring as of when arriving instances from some data stream are not independent of the time of arrival. In stream

Table 3.1 State-of-the-art drift detectors with Naive-bayes

Drift Detector	Elec2	Forest Covertype
CUSUM	79.21	81.55
Page-Hinckley	78.04	80.06
DDM	81.18	88.03
EDDM	84.83	86.08
No-Change	**86.16**	**88.79**

Table 3.2 State-of-the-art drift detectors with Hoeffding Tree

Drift Detector	Elec2	Forest Covertype
CUSUM	81.71	83.01
Page-Hinckley	81.95	81.65
DDM	85.41	87.35
EDDM	84.91	86.00
No-Change	**85.54**	**88.04**

mining there is an innate assumption that class labels are always independent of the time of arrival, and therefore research has historically ignored any temporal dependence in the development of stream mining algorithms.

Instead, temporal dependence has been a popular topic and research field within the domain of time series analysis over the years (Beck 2001, Cheng et al. 2014, Cai et al. 2018). It has only been more recently that temporal dependence has been identified as a potential problem within the field of stream mining. Bifet (2017) describes how the undetected existence of temporal dependence in evolving stream data is the cause of an "illusion" of progress within the research field. The performance of state-of-the-art concept drift detection methods were compared to a simple, non-statistical method called the "No Change" detector that simply resets the base classifier every 60 arriving instances. Results showed that due to temporal dependence in the data, the "No Change" detector outperformed the most state-of-the-art methods. Tables 3.1 and 3.2 show the comparative results of the "No Change" detector using both a Naive-bayes and Hoeffding Tree classifier. Using different classifiers demonstrates that this anomaly is not caused by a specific classifier but lies within the data itself.

As is portrayed in the results shown in Tables 3.1 and 3.2, resetting the classifier as frequently as possible yields higher performance accuracy even when no statistical drift detection is performed. This is obviously misleading since the classifier is trained on the same repeated label and will naturally report high classification accuracy during this time. Since the classifier, in normal circumstances, will not be reset again until the next detected drift, this may lead to the base classifier becoming outdated and

unreliable. Thus this is known as the "over-resetting problem".

To provide a real-world example of where temporal dependence might exist in an evolving data stream, it is paramount to understand the features of the data that must be present for both to exist. An evolving data stream, or a concept drift stream, must contain some form of shift in the relationship between the attributes and the class label such that the attributes no longer accurately describe the label. Temporal dependence exists if the arriving labels are dependent on the specific time of arrival within the stream.

With that in mind, let us consider a smart motorway in the UK. A smart motorway is a section of road, usually at least three lanes wide, where the speed limit is controlled remotely and displayed on overhead gantries. Consider we have a classifier which aims to predict changes in the speed limit. The features include the length of the road section the speed limit applies, the density of traffic, a timestamp and a flag indicating whether is daytime or night. The class label represents the change in the speed limit; higher, lower or remain the same. An example of where concept drift might occur in this scenario is during a road traffic collision or a lane closure due to maintenance. In this case, the speed limit for the road section is likely to either remain the same or lower for safety reasons, however this isn't represented in the data. As such, this will cause a shift in the relationship between the attributes and the class label.

The example can be taken further by demonstrating where temporal dependence may also be present in the data. Note that for arguments sake there is a timestamp included in the data, but temporal dependence may be present in data where no timestamps are present. In this example, temporal dependence may occur in a number of instances. During rush hour, the density of traffic is likely to be higher and therefore the speed limit is likely to be lowered. The inverse is also true where in the early evening the speed limit is likely to continue to rise back to the limit when traffic is particularly light.

The above example gives an indication of where temporal dependence might exist in an evolving data stream. In reality it is much harder to obtain such data. There are few datasets that are used for evaluating concept drift data and precious fewer that also contain temporal dependence. This is due to the real-world difficulty in acquiring the data, and instead synthesised methods are often used to generate "fake" data streams that can be used in research evaluation.

3.3 Analysis of Datasets

While very few datasets containing temporally dependent evolving data exist, there are some that have been used in the literature. These are the Electricity (Harries & Wales 1999), Forest Covertype (Blackard et al. 1998) and KDD '99 Cup datasets (Dua & Graff 2017). These datasets have been identified as temporally dependent, evolving data streams through relevant literature (Zliobaite et al. 2015, Bifet 2017). This section explores these datasets, discussing how they are temporally dependent with both qualitative discussion and statisitical analysis. The results of the autocorrelation function for each of the datasets is provided alongside the discussion. The autocorrelation function provides statistical representation between a class label and the associated time period. This provides an illustration of how "correlated" the data is, with the higher the correlation the more temporally dependent the data. Autocorrelation data was gathered using Python's Statsmodel package. The time period offset for each dataset were automatically determined by the Statsmodel algorithm. For each provided autocorrelation graph the Y-axis represents the autocorrelation value and the X-axis represents the "lags", or time period offsets.

The Electricity dataset contains 45,312 instances each containing eight attributes and a has two possible class labels; "UP" or "DOWN". Each label represents that change in electricity price relative to a moving average in the last 24 hours. The dataset itself originates from collected data from the Australian New South Wales electricity market. In this market electricity unit prices are not fixed and instead fluctuate throughout the day every five minutes. Each instance in the dataset represents a period of 30 minutes resulting in 48 instances for each 24 hour period. The attributes include the date, day of the week, time period of the measurement, the electricity price in New South Wales, the electricity demand in New South Wales, the electricity price in Victoria, the electricity demand in Victoria and the scheduled transfer of electricity between the two Australian states. Figure 3.1 shows the autocorrelation function results for this dataset.

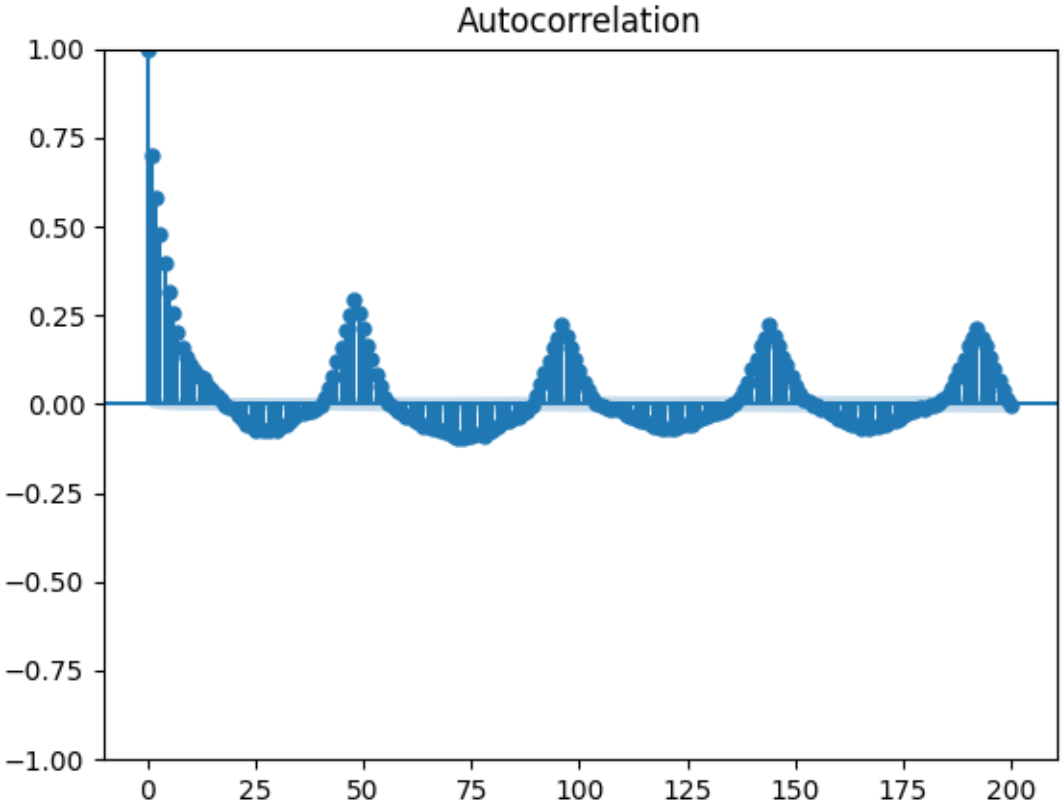

Figure 3.1: Autocorrelation function for Electricity dataset

As can be observed from Figure 3.1, the dataset is temporally dependent at recurrent intervals. The cylindrical peaks portrayed represent the autocorrelation every 48 instances, or every 24 hours in the data. The electricity price every 24 hours is heavily autocorrelated and this is likely due to consumer habits. For example in the UK the electricity demand surges in the evening due to consumer demand for cooking, relaxation and leisure. The findings here are closely mirrored by Bifet (2017).

The Forest Covertype dataset is also temporally dependent, but much more severely that than the Electricity dataset above. Forest Covertype contains 581,012 instances, each with 54 attributes and a possible 7 class labels. The dataset describes cartographic information over four wilderness areas in Northern Colardo. Each class label represents one of seven possible types of forest cover. Figure 3.2 provides the autocorrelation data for the Forest Covertype dataset. In this case the data is significantly correlated at all times, not just every n instances as in the case of the Electricity dataset.

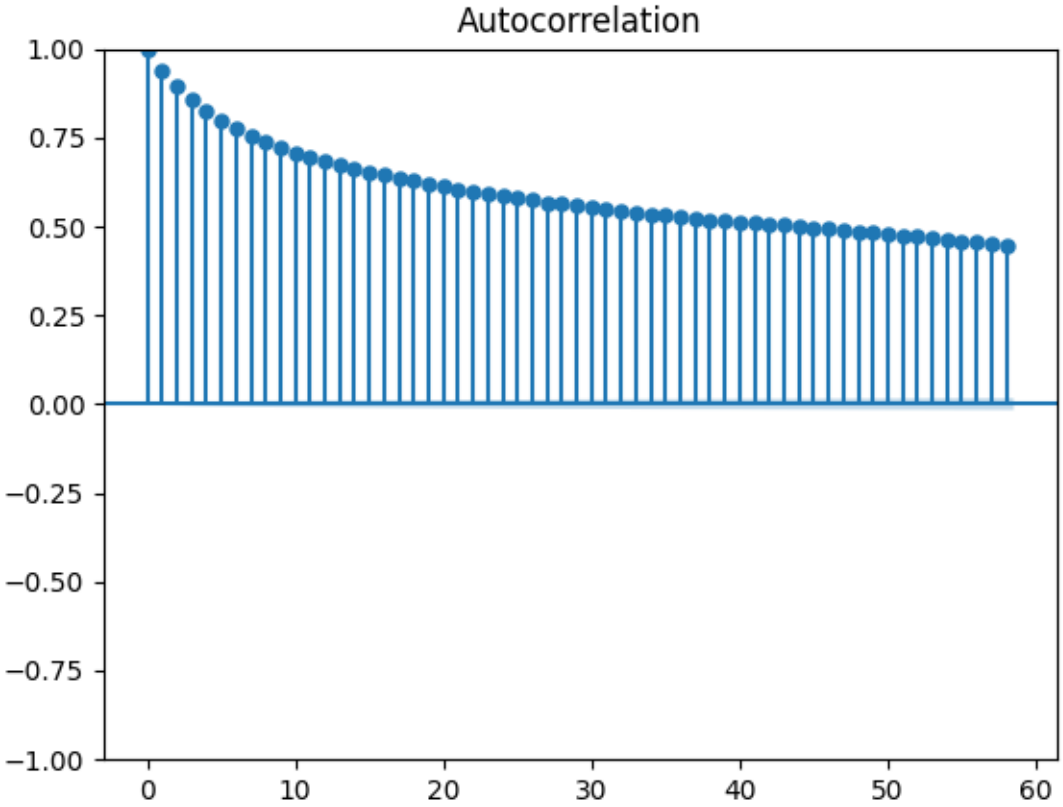

Figure 3.2: Autocorrelation function for Forest Covertype dataset

The temporal dependence in Forest Covertype is different in origin to that of Electricity. Where the Electricity dataset has clear timestamps indicating the electricity price at any particular interval, Forest Covertype contains no timestamp in its attributes since a timestamp has no bearing on the class label which describes the type of forest cover. Semantically speaking, a timestamp in this context is irrelevant. However, the data instances are not randomised but are originally presented sequentially in geographical order. This means that there is indeed temporal dependence in the data; arriving class labels are dependent on the progress of the stream in relation to the position of instances in the data. This is represented in the autocorrelation data shown in Figure 3.2 which clearly shows how continuously heavily correlated the data is for the entirety of the stream.

And containing yet further increased levels of temporally dependency is the KDD '99 Cup dataset. This dataset contains 494,020 instances with each containing 41 attributes and a possible 23 class labels. The dataset itself contains instances that simulate

intrusions on a military network, where intrusion is considered one of 23 types, for example "rootkit", and one label is reserved for "normal" indicating that the traffic is non-threatening. Figure 3.3 highlights the autocorrelation data for the KDD '99 Cup dataset.

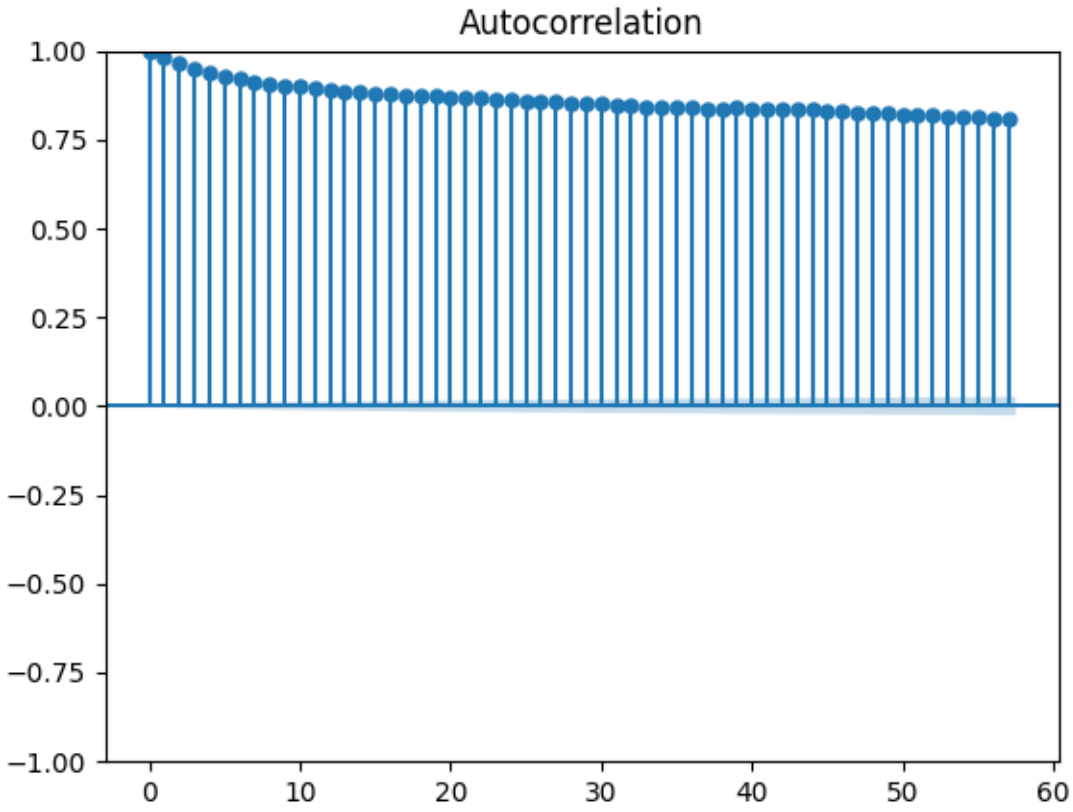

Figure 3.3: Autocorrelation function for KDD '99 Cup dataset

The autocorrelation for the KDD '99 Cup dataset shows that this is the most autocorrelated, temporally dependent data yet. The underpinning explanation is similar in context to that of Forest Covertype whereby in the data, intrusions occur in isolated time periods rather than in single instances. The result of which is that, again, the arriving class label is directly dependent on the time during which a concept drift detection algorithm processes the instance.

An emerging theme from this analysis is that often the cause is due to the order of the instances within the data. For example, in both Forest Covertype and KDD '99 Cup the instances are somewhat grouped whereby all instances of label y arrive during a particular time window T_w. One suggestion might be to randomise the instaces within

a dataset but this creates two problems. Firstly, in order to justify using an offline, static dataset to simulate the functionality of a data stream by "streaming" instances sequentially then the original order of the data must be preserved. Randomising the order of instances within the dataset would result in the creation of a fundamentally different dataset. Secondly, in an environment where a real-time data stream is used it would be impossible to randomise the arrival of instances. The instances will arrive in a sequential fashion and as such research which streams static datasets for evaluation should also seek to mirror this functionality. Arriving instances from a real-time data stream may also arrive in temporally dependent groups as is the case with the Forest Covertype and KDD '99 Cup datasets. In the context of stream mining, any static datasets that are streamed for evaluation purposes should be done so in a manner representative of a true, real-time data stream.

3.4 Comparison with Imbalanced Data

As discussed, the research tradition of utilising offline, static datasets for the evaluation of stream mining algorithms is not without its inherent problems. Class imbalance within the datasets used for streaming is also of important when considering temporal dependence in the data.

In the context of temporal dependence, it has been established that arriving class labels are dependent on their time of arrival, and this occurs over some time period. In the Electricity dataset the window of time between periods of temporal dependence was longer at some 24 hours. In contract, the KDD '99 Cup and Forest Covertype have overlapping periods of temporal dependence that occur back to back throughout the entirety of the data stream. Considering that these datasets are not real-time data streams but offline datasets that are streamed to simulate a data stream, looking at the class imbalance of these data streams can also be of value. The class imbalance for all three datasets is provided in Figure 3.4.

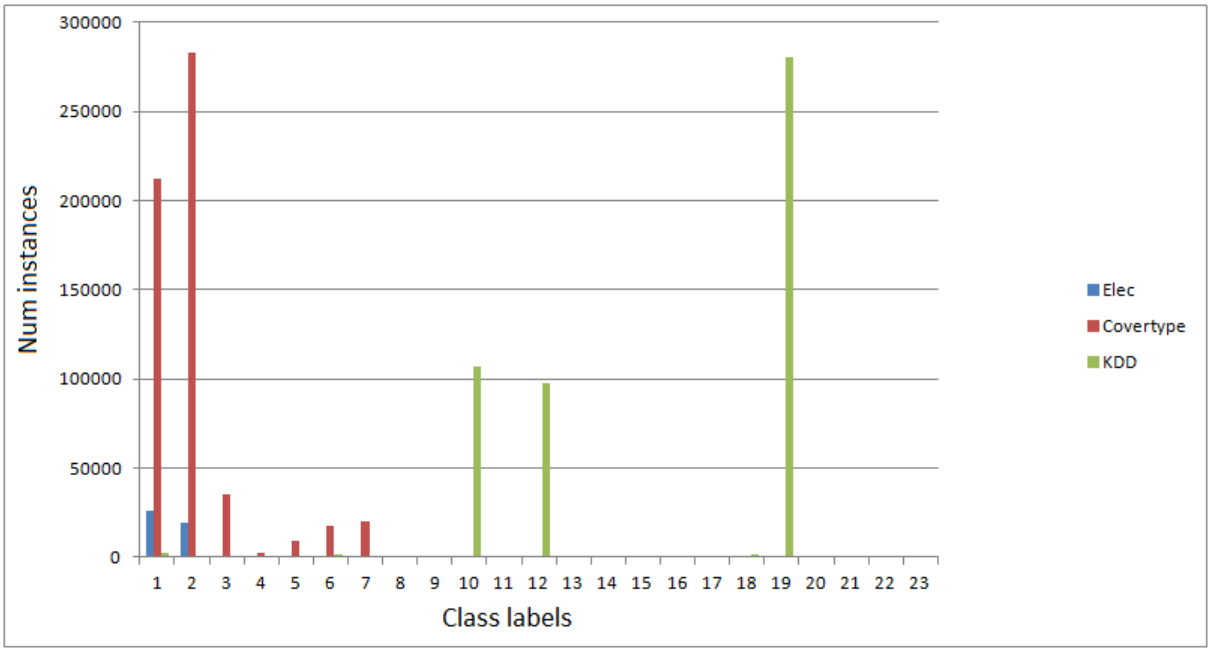

Figure 3.4: Class Imbalance for Temporal Datasets

From Figure 3.4 two main observations can be derived. Firstly, the Electricity is not only the least temporally dependent of the three but is also the most balanced in terms of class labels. Secondly, both Forest Covertype and KDD '99 Cup have an clear majority class that significantly imbalances the data. There are similarities between temporal dependence and class imbalance since during a window of temporal dependence, the arriving class labels will likely be identical and therefore created a class imbalance during the temporal dependence window. This is why the over-resetting problem is critical. Resetting a base classifier during this time causes the classifier to be trained on potentially only a single class label. This may cause misleading classification performance since the arriving instances are identical. Since the classifier is trained and evaluated during a temporal event and is being trained on a single class label, the resulting performance accuracy naturally appears attractive. However, the resulting accuracy doesn't reflect a successful predictive model. Rather the apparently effective classifier accuracy is a result of temporal dependence in the data. This can have a direct negative effect on the stability and reliability of the model in real scenarios. To improve upon this, temporal dependencies in the data should be accounted for during classifier evaluation.

However it is important to establish that temporal dependence and class imbalance are fundamentally different, and over its lifespan a real-time data stream may not be considered imbalanced whatsoever. There is a key distinction that must be made to

fully illustrate that point, and that is the fundamental difference between a real data stream and simulating an offline dataset for research purposes. Simulating an offline dataset inherently introduces any class imbalance present in the dataset to the stream. There is no opportunity for a simulated dataset to rectify its class imbalance over time. In contrast to this, a real data stream may contain class imbalance at any given point in time, perhaps during a temporal dependence window. However since real data streams are unbound in size and veracity, it is equally viable that classes within the stream may balance out over time.

For research purposes, using real-time data streams is largely impossible due to the access permissions, sourcing and their volatile nature. If research relied on real data streams for evaluation, validating or replicating studies would prove, potentially, extremely troublesome. Streaming offline datasets will remain the norm for the foreseeable future. Temporal dependence and class imbalance are also closely related concepts but are not necessarily mutually exclusive. A stream may be temporally dependent but maintain relative class balance. Imbalanced datasets may contain temporal dependence, as is the case with the Forest Covertype and KDD '99 Cup datasets, or they be absent from it entirely.

Chapter 4

Addressing the Over-Resetting Problem

This chapter contains an original contribution of this research in the form of a proposed algorithmic solution which suggests a novel approach to coping with temporal dependence in concept drift data streams. The proposed algorithm, Burst Detection-based Selective Classifier Resetting (BD-SCR), is an architectural solution which can be adopted in harmony with existing state of the art drift detectors to aid in the overcoming of the over-resetting problem present in concept drift streams that suffer from temporal dependence. The work contained in this chapter and the BD-SCR method proposed appeared in the Journal of Information and Knowledge Management (Wares et al. 2021).

This chapter is subdivided as follows: first, the over-resetting problem is defined with reference to published research that has highlighted the problem or offered some novel contribution to its cause; secondly the adopted methodology and method are explained; thirdly the experimental setup is outlined; penultimately the results of experimentation are provided alongside an in-depth, critical interpretation; finally conclusions are drawn and suggestions for future work are provided.

4.1 Novel Contributions

This chapter constitutes an original contribution to research. At the time of writing, research surrounding temporal dependence in concept drift environments is very much in its infancy. The state of the current literature is highlighting the problem and suggesting some initial avenues for future development, but there is little work proposing methods for coping with temporal dependence in concept drift data.

A new metric based on empirical reasoning for evaluating the performance of a classifier in temporal environments is proposed. The Temporal Stability Index (TSI) is a metric that compares the classifier accuracy of the entire stream to that of incrementing sliding windows. This allows a comparison to be made between the overall performance, and the performance over the most recently arriving instances. The metric provides values between -1 and 1, where a value of 1 indicates the classifier is stable and adaptable to temporal dependencies in the data.

This chapter also proposes the BD-SCR method, which offers an innovative method for controlling the resetting of base classifiers used in conjunction with concept drift detectors. BD-SCR is also unique in that is interoperable with any existing concept drift detection method.

Currently, this is one of the earliest methods proposed in research for coping with temporal dependence in concept drift streams. It is hoped that it will provide the groundwork for future methods and adaptations for undertaking stream mining in evolving, temporal environments.

4.2 Problem Definition

As discussed in Chapter 2 and 3, the existence of temporal dependence in streams subject to concept drift can cause misleading performance accuracy when base classifiers are evaluated. This problem stems from an aged underlying architecture surrounding concept drift detection algorithms and baseline classifiers. Concept drift detection algorithms have historically resorted to forcefully resetting classifiers for each detected drift. Various example of this are provided throughout Chapter 2. The justification for this is to prevent underlying classifiers becoming outdated as the distribution of a data stream shifts from one concept to another. However, in scenarios where data streams are afflicted by both concept drift and temporal dependence, this common practice is the root cause for what is referred to in this thesis as the "over-resetting problem". In situations where both concept drift and temporal dependence are present within a data stream consistent, forced resetting when drift detection algorithms signal drift alarms causes misleading classifier performance metrics during evaluation.

For any given data stream, arriving instances are said to be temporally dependent if they are not independent of their time of arrival. In such cases they are also highly likely to contain the same class label. This is given in Equation 4.1.

$$P(y_t, y_{t-1}) \neq P(y_t)P(y_{t-1}) \tag{4.1}$$

In situations where arriving stream instances are temporally dependent and drift detectors signal an alarm indicating a detected drift, resetting the base classifier will cause it to be retrained on instances where the observed class labels are the same. This results in classifier accuracy appearing to perform well, when in reality it is an illusion. Bifet (2017) demonstrates the problem of over-resetting through experimentation with a "No-Change" detector in comparison with state of the art drift detectors. The No-Change detector is a drift detector that performs no statistical evaluation on arriving stream instances whatsoever, but instead simply signals an alarm every 60 instances that resets the base classifier. When compared to several state of the art drift detectors using two popular concept drift datasets, Electricity (Harries & Wales 1999) and Forest Covertype (Blackard et al. 1998), it was shown that the No-Change detector outperforms its counterparts across the board. To prove that this anomaly is not down to the base classifiers itself, both a Naive-Bayes and Hoeffding Tree classifier were used to achieve the same result (Tables 3.1 and 3.2). Bifet (2017) states the reason for this anomaly is due to the existence of temporal dependence within the datasets.

A recent review by Wares et al. (2019) critically discusses and evaluates the issues surrounding temporal dependence and concept drift, echoing the ever emerging need for existing methods of drift detection to be able to cope with temporal dependence. As explained in Chapter 2, this is a relatively new problem and there exists very little published research which contributes novel solutions. The work of Zliobaite et al. (2015) suggests two classifier based augmentations that allow temporal dependence to be accounted for during classification. The first method is the Temporal Correction classifier, given as

$$\frac{P(y_t = i \mid y_{t-1})}{P(y_t = i)} P(y_t = i \mid X_t), \tag{4.2}$$

where y_t is a class label at time t, X_t is the corresponding feature vector and i is some class from the set $i \in \{1, ..., k\}$. The Temporal Correction classifier provides a single score which indicating its ability to handle not only temporal dependence but also imbalanced data. The key drawback to this method is the assumption surrounding previous labels being known. If there is any delay in the arrival or retention of a class label then the performance and effectiveness of this approach is negatively impacted.

The second method proposed by Zliobaite et al. (2015) is the Temporally Augmented classifier. This is a more elegant method which involves augmenting the features vector of an arriving instance with the last x previously observed class labels. However, as with the Temporal Correction classifier, class labels are assumed to be known. The underlying base classifier is then trained using the augmented feature vectors, with the

prediction $\hat{y}_t$ given as

$$\hat{y}_t = h_t(X_t, y_{t-1}, ..., y_{t-l}) \tag{4.3}$$

where h_t is the classification model and l is the length of the temporal dependence order. This proposed method, however, still suffers from the reliance of class labels arriving on time in the same manner the Temporal Correction classifier does. The authors also note that the Persistent classifier already performs a similar job in predicting that the next arriving label will be the same as the last, and sometimes outperforms the Temporally Augmented classifier approach. It is worth additionally noting that neither these methods do not aid in the issue of concept drift detectors over-resetting base classifiers in the presence of temporal dependence.

In addition to base classifier augmentations for coping with temporal dependence, Zliobaite et al. (2015) also suggest two metrics for evaluating the level of temporal dependence within a stream: the Kappa-Temporal Statistic (4.4) which indicates the severity of temporal dependence by comparing the performance of the base classifier and that of the Persistent classifier,

$$k_{per} = \frac{P - P_{per}}{1 - P_{per}}, \tag{4.4}$$

and the Combined Measure which provides coping mechanisms for imbalanced datasets (4.5).

$$K^{+} = \sqrt{max(0, k)max(o, k_{per})} \tag{4.5}$$

While the Kappa-Temporal statistic is a recent and, at the time of writing, one of the only proposed advancements for including temporal dependence during evaluation of classifiers in evolving data streams, it is not without its limitations. The Kappa-Temporal statistic is calculated using the probability of the next arriving class label being the same as the previous observed label. While this does provide some method of measuring a base classifier's ability to cope in temporal environments, it could be improved. One such facet of improvement is to statistically analyse the classification performance of the base classifier upon the entire stream, versus various different sliding windows that encapsulate the temporal dependencies at different time periods.

This chapter defines such a metric, the Temporal Stability Index (TSI), which is used for experimental evaluation of the BD-SCR method described below. Equation 4.6

Table 4.1 Accuracy, KT and TSI evaluation on Electricity

Drift Detector	Accuracy	KT	TSI
CUSUM	79.21	-0.44	0.34
Page-Hinckley	78.04	-0.57	0.3
DDM	81.18	-0.28	0.35
EDDM	84.83	-0.07	0.37
No-Change	86.16	-0.18	0.34

denotes how the metric is defined,

$$TSI = \frac{P_o - P_{\bar{w}}}{P_o} \tag{4.6}$$

where $-\infty \leq TSI \leq 1$, P_o is the overall predictive accuracy of the base classifier and $P_{\bar{w}}$ is the average accuracy of all computed sliding windows. The window size can be defined as any positive integer value.

In order to encapsulate how well the base classifier is capturing the temporal dependencies in the data, the classifier is evaluated on both on the entire stream and on a sliding window w containing the most recent labels of which the classifier has not yet observed. This allows the difference between the classifier's overall performance and its performance on the current window to be statistically observed.

TSI values trending towards 1 indicate that the overall performance of the base classifier is significantly higher than that of the individual sliding windows, thus the base classifier is coping with the temporal dependencies in the data and is considered more stable and adaptable. In contrast, negative TSI values occur when the classifier accuracy on individual windows greatly outperforms that of the classifier's overall performance. In such cases this can be illustrative of a classifier that is over-resetting, unstable and not adapting to temporal dependencies in the data. Tables 4.1 and 4.2 provides an overview of the accuracy, Kappa-Temporal and TSI performance using a naive-bayes classifier and state-of-the-art drift detection methods on both the Electricity and Forest Covertype datasets.

An interesting recent proposition which has inspired the architecture for the original work contained in this chapter is Hierarchical Hypothesis Testing (HHT) (Yu et al. 2019) which offers a unique framework for drift detection. HHT proposes a two layer architecture for drift detection where the first layer is responsible for detecting a drift whilst the second layer performs validation. This framework is shown in Figure 4.1. The Hierarchical Linear Four Rates method (HLFR) (Yu et al. 2019) is a modern drift

Table 4.2 Accuracy, KT and TSI evaluation on Forest Covertype

Drift Detector	Accuracy	KT	TSI
CUSUM	81.55	-2.84	0.4
Page-Hinckley	80.06	-3.23	0.39
DDM	88.03	-1.86	0.41
EDDM	86.08	-2.1	0.41
No-Change	88.79	-1.45	0.42

detection method developed under the HHT framework.

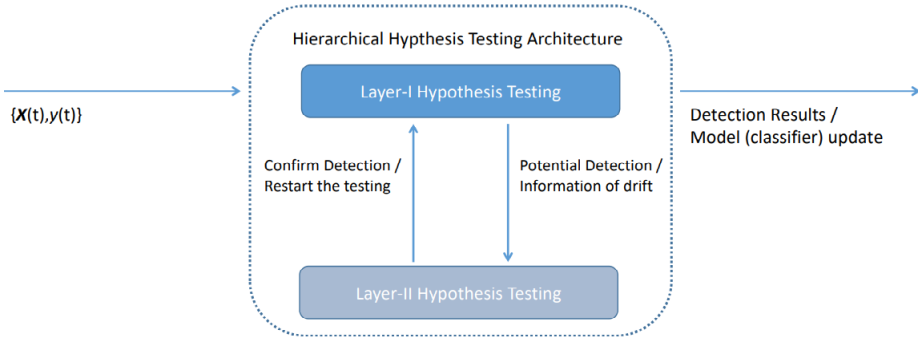

Figure 4.1: HLFR Framework (Yu et al. 2019)

4.3 Method

The methodology adopted for design and implementation of this algorithm is a incremental, experimental approach. The final solution was built through an iterative process of design, implement and evaluate through all stages of the experiment. The results of Bifet (2017), Tables 3.1 and 3.2, provide a set of baseline published results for evaluation of this algorithm. Given that the No-Change detector outperforms the state of the art drift detectors, the aim of this algorithm is to reduce the effectiveness of the No-Change detector whilst maintaining, as close as possible, the predictive accuracy of the state of the art drift detectors. Since the No-Change detector does not perform any statistical or otherwise drift detection of its own, it should not be outperforming state of the art methods. Significantly reducing its effectiveness through this proposed method would indicate the experiment has been successful.

This research proposes a novel approach which directly challenges the existing architecture for resetting base classifiers during detected drifts when temporal dependence is present. Traditionally, concept drift detection algorithms will force reset the underlying base classifiers each time a drift is detected. In doing so the base classifier

is kept relevant and up-to-date with the core distribution of the stream. However, in the presence of temporal dependence this becomes the root cause of the over-resetting problem, as discussed above. The BD-SCR method proposed in this chapter instead challenges the assumption that classifiers must be reset for every detected drift. Instead, BD-SCR statistically monitors changes in the levels of temporal dependence over the entire stream to make informed decisions about classifier resetting. Statistical monitoring of the temporal dependence levels within a stream is achieved using a double sliding window approach. A first window monitors the Kappa-Temporal values for the instances that have arrived most recently, whilst a second window observes the Kappa-Temporal values for the entire stream. Adopting a statistical burst detection mechanism, BD-SCR is then capable of determining when spikes, or increases, in the temporal dependence levels of the stream are occurring. During these spikes, BD-SCR will prevent the base classifier from resetting if a drift is concurrently detected. The sensitivity of the burst detection and the threshold at which base classifier resetting prevention will occur is controlled through user defined parameters. A full algorithmic overview of the complete method is provided in the following sections.

4.3.1 Burst Detection

Burst detection is the process of detecting abnormalities or outliers within data streams (Zhu & Shasha 2003). In essence, burst detection involves the monitoring of events over specific time period and signals an alarm when an anomaly is detected. This is of paramount importance for BD-SCR since the detection of sudden increases, or anomalies, in the temporal dependence severity of a stream is what drives and informs the decision to prevent the resetting of an underlying base classifier. Opting to not reset a base classifier during a detected drift has the potential to cause the base classifier to become obsolete and outdated, and as such become inaccurate and under perform.

In this research, BD-SCR utilises burst detection in order to determine when there has been a significant increase in the amount of temporal dependence within a data stream. The burst detection method adopted by BD-SCR is a short-term average (STA) and long-term average (LTA) analysis, which is used to determine the burst value B. This is given in equation 4.7. This method of burst detection has been used extensively in stream based event detection research, particularly in the field of earthquake detection (Sakaki et al. 2010, Earle et al. 2012, Ross & Ben-Zion 2014, Kong et al. 2015). This is an effective yet lightweight method for identifying sudden changes in normality for any given data stream. It provides a statistical means for performing burst detection but is computationally inexpensive.

$$B = STA/LTA \tag{4.7}$$

BD-SCR adopts a sliding window based approach to temporal dependence monitoring. The short term average, STA, is the average of a sliding window containing the Kappa-Temporal value of the base classifier for the most recently seen instances since the last detected drift. This window is reset each time a drift is detected. The long term average, LTA, is the average of a second sliding window which contains the Kappa-Temporal values for predictions made over the entire stream. Values of B exceeding 1.0 indicate a burst. B is recalculated each time a drift is detected by the associated drift detector. However, if $STA > 0$ at the point of drift detection then the classifier is allowed to reset as per traditional drift detection since a positive STA value indicates that the temporal dependence presence level in the current window is low.

4.3.2 Selective Resetting

The decision of when to reset the base classifier is of a critical and sensitive nature. Resetting too frequently results in the same core issue presented by the results of Bifet (2017) and the No-Change detector, where temporal dependence will cause misleading performance evaluation. However resetting too infrequently will risk the base classifier becoming outdated and obsolete as more drifts are detected and the core distribution of the stream shifts over time.

To overcome this problem, BD-SCR does not reset the base classifier in every instance where $B > 1.0$, which is the commonly accepted threshold for a burst using the short term versus long term average method discussed (Zhu & Shasha 2003). Instead, burst values B are compared against a user specified parameter T, which defines a threshold indicating the maximum amount of increase, or maximum burst, of temporal dependence that is permitted between detected drifts. For any detected drift, the following conditions are evaluated to determine whether or not the base classifier is reset:

$$\begin{aligned} &\text{If } B < T \text{ reset base classifier} \\ &\text{If } B \geq T \text{ do not reset base classifier,} \end{aligned} \tag{4.8}$$

The justification and motivation for including T in the decision making process is non trivial. The frequency of bursts in temporal dependence in a data stream is opaque, as is its severity. A stream may suffer from several or few detected bursts, and the severity may range from B being only marginally greater than 1 to considerably more.

Considering that the decision to refrain from resetting the base classifier contains the risk of the classifier becoming outdated, it follows that the severity of the burst should be included in the resetting decision. To that end BD-SCR exposes T as a maximum tolerance threshold for severity of detected bursts.

Algorithm 7 below provides an overview of the entire BD-SCR method. Note that the algorithm covers the burst detection and selective resetting process, base classifiers are trained using arriving instances as normal.

Algorithm 7 BD-SCR Algorithm

$wSTA$: Sliding window of Kappa-Temporal values since last detected drift
$wLTA$: Kappa-Temporal values for the whole stream
STA: The average of $wSTA$
LTA: The average of $wLTA$
B: Burst value
T: Burst threshold
KT_i: Kappa-Temporal value for arriving instance i
C: The base classifier

```
1: Add KT_i to wSTA and wLTA
2: if drift detected then
3:    if STA > 0 then
4:       Reset C
5:    else
6:       calculate STA from wSTA
7:       calculate LTA from wLTA
8:       calculate B
9:       if B < T then
10:         reset C
11:      end if
12:      reset wSTA
13:   end if
14: end if
```

4.4 Experimental Setup

Experimentation was implemented and evaluated using the popular streaming tool kit MOA. In order to evaluate BD-SCR and determine its effectiveness in overcoming the temporal issue presented by Bifet (2017), this experiment mirrors the experimental setup in terms of drift detectors, base classifiers and datasets. BD-SCR is tested using a Naive Bayes classifier in conjunction with the Drift Detection Method (DDM), Early Drift Detection Method (EDDM), Page Hinckley (PH) and Cumulative Sum (CUSUM) drift detectors. Since it is already proven that the anomaly is not caused by the base classifier, this experiment tests only with a Naive-bayes classifier and not also

with a Hoeffding Tree classifier. T values for experimentation range from 1 to 3 with increments of 0.1 in order to provide maximum statistical coverage. The justification for maximising T at a value of 3 is the logical observation that it is unlikely detected bursts with a Kappa-Temporal value triple the average of the stream would be accepted in any real situation. As such, experimenting up to this value provides a clear portrayal of a wide statistical coverage.

Datasets used for evaluation are Electricity (Harries & Wales 1999) and Forest Covertype (Blackard et al. 1998). The selection of these datasets for experimentation is twofold. Firstly they are the datasets used in published work which has identified this over-resetting problem (Bifet 2017), inspiring this BD-SCR method. Secondly these datasets are well established in the domain of concept drift detection and have been used in various published research both, both recent and historic (Baena-Garcıa et al. 2006, Bouguelia et al. 2018). The Electricity dataset contains some 45312 instances each containing five fields. The dataset itself represents fluctuating electricity prices in Australia's New South Wales. The Forest Covertype dataset contains 581012 instances each containing 54 attributes. This dataset reflects the type of forest cover within four wilderness areas of the Roosevelt National Forest in Northern Colorado.

In this experimentation it is fundamental to note that classifier accuracy is not considered a key performance metric used for evaluation. As discussed previously, in temporally dependent environments classifier accuracy can provide misleading values due to the time dependent arrival of class labels. Instead, this experimentation focuses on the Kappa-Temporal statistic and the Temporal Stability Index for statistical evaluation of the proposed BD-SCR method. Rather than omit classifier accuracy from the results entirely it is provided in the experimental results for clarity and continuity purposes. However, it is not included in the discussion for the previously mention reasons. Whilst other metrics have become popular for the evaluation of statistical drift detectors, for example False Alarm Rate (FAR), Mean Time between False Alarms (MTFA) and Mean Time to Detection (MTD), these are not used used in this experimentation. The aforementioned metrics are used for evaluating the statistical performance of an individual concept drift detector, whereas BD-SCR is a novel algorithm for handling and coping with temporal dependence during concept drift detection. It is important to BD-SCR is not an independent drift detector, nor does it alter or amend the original statistical methods of any existing drift detectors used in conjunction with it.

4.5 Results and Discussion

The discussion of the experimental results is provided in this section through three separate subsections. The first provides analysis of the results for the Electricity dataset,

secondly the results for the Forest Covertype are discussed and finally suggestions surrounding setting the optimal value for T are provided. Tables 4.3 and 4.4 portray the experimentation results. In all cases of discussion, the results of BD-SCR are evaluated through the Kappa-Temporal and TSI metrics, and compared against the baseline results provided in Tables 4.3 and 4.4.

One important observation to note that is most apparent in the Electricity dataset is the T value at which evaluation metrics start to plateau or repeat. This is due to no further increases in temporal dependence in the data which surpass the value of T. For example, in the Electricity dataset performance values will typically start to repeat on average around $T = 2$. This is because there are no burst values surpassing $T = 2$, or similar, in the data, and therefore the classifier achieves the same performance as it would for any value of T which encapsulates the maximum burst amount in the data. Since various drift detection methods perform drift detection with vastly different mechanisms, this results in variations in the T values where performance beings to plateau. More in-depth discussion around optimising T is provided further on in this chapter.

A second observation is that it can be observed that the Kappa-Temporal and TSI metrics improve at a similar rate and for near identical values of T. The mirrored improvement is to be expected considering both metrics statistically evaluate classifier capability in temporal environments. This trend in the evaluation is positive for BD-SCR since it highlights the performance improvements that can be benefited by adopting BD-SCR in temporal environments.

Finally, the performance results of the No-Change detector are included for clarity purposes. However since the No-Change detector performs no statistical drift detection and is not a published, state-of-the-art drift detection method, its results are not as of the same importance as that of the other drift detection methods. The No-Change detector exists only to prove that over-resetting the classifier in temporal dependent environments can cause misleading performance accuracy, as previously discussed. The No-Change detector is included in the experimentation results below to illustrate how other temporal evaluation metrics indicate that the No-Change method does not perform adequately, especially compared to other state-of-the-art drift detection methods.

4.5.1 Results of the Electricity Dataset

For the Electricity dataset results provided in Table 4.3, an immediate observation is that there is improvement in the Kappa-Temporal and TSI scores for each drift detection method with the exception of No-Change. Additionally results show that for

Table 4.3 BD-SCR Results for Electricity Dataset

	DDM			EDDM			CUSUM			PH			NO-CHANGE		
T	Acc	KT	TSI	Acc	KT	TSI	Acc	KT	TSI	Acc	KT	TSI	Acc	KT	TSI
1	80.88	-0.3	0.33	80.93	-0.3	0.34	77.51	-0.53	0.32	75.36	-0.68	0.28	76.12	-0.62	0.29
1.1	76.54	-0.6	0.3	80.61	-0.32	0.35	77.86	-0.51	0.33	76.47	-0.53	0.31	76.11	-0.62	0.29
1.2	76.56	-0.6	0.3	81.27	-0.28	0.34	77.98	-0.5	0.33	76.47	-0.6	0.31	76.17	-0.62	0.29
1.3	81	-0.3	0.33	83.75	-0.11	0.37	78.01	-0.5	0.33	76.47	-0.6	0.31	76.23	-0.62	0.29
1.4	81.01	-0.29	0.33	84.74	-0.04	0.38	77.7	-0.52	0.33	76.75	-0.58	0.32	76.15	-0.63	0.29
1.5	81.28	-0.28	0.34	84.74	-0.04	0.38	77.7	-0.52	0.33	76.58	-0.6	0.32	76.17	-0.62	0.29
1.6	81.48	-0.26	0.34	84.74	-0.04	0.38	77.89	-0.51	0.33	76.5	-0.54	0.33	76.14	-0.63	0.29
1.7	82.09	-0.22	0.36	84.74	-0.04	0.38	77.89	-0.51	0.33	77.36	-0.54	0.33	76.29	-0.61	0.29
1.8	82.13	-0.22	0.36	84.76	-0.04	0.38	78.77	-0.45	0.34	77.39	-0.54	0.33	75.91	-0.64	0.29
1.9	82.9	-0.17	0.37	84.76	-0.04	0.38	79.08	-0.43	0.36	77.39	-0.54	0.33	76.07	-0.63	0.29
2	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.1	-0.63	0.29
2.1	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.08	-0.63	0.29
2.2	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.13	-0.63	0.29
2.3	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.46	-0.6	0.29
2.4	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.21	-0.62	0.29
2.5	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.22	-0.62	0.29
2.6	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.32	-0.61	0.29
2.7	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.37	-0.61	0.29
2.8	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.39	-0.61	0.29
2.9	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.71	-0.58	0.29
3	83.88	-0.1	0.37	84.76	-0.04	0.38	79.3	-0.41	0.36	77.39	-0.54	0.33	76.84	-0.57	0.3

the Electricity dataset it is empirically possible to identify the optimal value for T for all drift detection methods.

Starting with DDM this method achieves its maximum Kappa-Temporal performance with $T = 2$, achieving an improved score of 0.1, up from -0.28. This is a significant improvement and is the largest margin of increase out of all methods, across both datasets for all metrics. This improvement brings the Kappa-Temporal performance of DDM on the Electricity dataset very close to be out of the negative Kappa-Temporal range, clearly outlining the benefit of controlling the classifier resetting in temporally dependent environments. TSI values follow a similar pattern, albeit without the same margin of improvement, where its maximum value of 0.37 is achieved at $T = 1.9$, an improvement over the TSI value of 0.35 without BD-SCR.

Following from DDM, its successor EDDM obtains significantly improved Kappa-Temporal results even without BD-SCR, as shown in Table 4.1. These results are even further improved when EDDM is used alongside the proposed BD-SCR method. Results in Table 4.3 highlight two key areas where EDDM outperforms other methods. Firstly its Kappa-Temporal scores not only improve with BD-SCR, but EDDM is the top performing method across all experimental results for the Electricity dataset with regards to the Kappa-Temporal metric. Using BD-SCR, the Kappa-Temporal score for EDDM improves from -0.07 to -0.04. Secondly, it can be observed that EDDM achieves its maximum Kappa-Temporal score with the lowest value of T of all other methods. EDDM is capable of obtaining its peak Kappa-Temporal performance at a $T = 1.4$, up

Table 4.4 BD-SCR Results for Forest Covertype Dataset

	DDM			EDDM			CUSUM			PH			NO-CHANGE		
Burst	ACC	KT	TSI	ACC	KT	TSI	ACC	KT	TSI	ACC	KT	TSI	ACC	KT	TSI
1	65.17	-6.05	0.3	65.21	-6.05	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.28	-6.03	0.3
1.1	65.21	-6.04	0.3	65.21	-6.04	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.31	-6.02	0.3
1.2	65.22	-6.04	0.3	65.22	-6.04	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.32	-6.02	0.3
1.3	65.22	-6.04	0.3	65.24	-6.04	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.34	-6.02	0.31
1.4	65.2	-6.04	0.3	65.26	-6.03	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.39	-6	0.31
1.5	65.2	-6.04	0.3	65.29	-6.03	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.43	-6	0.3
1.6	65.2	-6.04	0.3	65.3	-6.03	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.51	-5.98	0.31
1.7	65.22	-6.03	0.3	65.36	-6.01	0.3	64.77	-6.13	0.31	64.07	-6.27	0.32	65.6	-5.96	0.31
1.8	65.24	-6.03	0.3	65.44	-6	0.3	64.83	-6.12	0.31	68.08	-5.46	0.34	65.63	-5.95	0.31
1.9	68.89	-5.29	0.32	67.31	-5.62	0.31	69.58	-5.16	0.34	71.15	-4.84	0.36	66.06	-5.87	0.31
2	71.67	-4.73	0.35	71.23	-4.82	0.34	70.98	-4.88	0.35	77.09	-3.64	0.4	71.69	-4.73	0.34
2.1	75.58	-3.94	0.37	72.9	-4.49	0.35	73.2	-4.43	0.37	77.09	-3.64	0.4	73.44	-4.37	0.35
2.2	78.54	-3.34	0.39	75.51	-3.96	0.37	77.26	-3.61	0.39	77.17	-3.62	0.4	76.36	-3.78	0.37
2.3	81.02	-2.84	0.4	79.54	-3.14	0.4	79.33	-3.19	0.4	77.35	-3.59	0.4	80.44	-2.95	0.38
2.4	81.46	-2.75	0.41	80.58	-2.29	0.41	79.35	-3.18	0.4	77.35	-3.59	0.4	81.88	-2.66	0.38
2.5	81.01	-2.85	0.4	80.74	-2.9	0.41	79.35	-3.18	0.4	77.35	-3.59	0.4	81.99	-2.64	0.4
2.6	82.05	-2.39	0.42	82.51	-2.54	0.41	80.76	-2.9	0.41	79.36	-3.18	0.41	83.55	-2.33	0.4
2.7	83.68	-2.3	0.42	82.74	-2.49	0.41	81.15	-2.82	0.41	79.43	-3.17	0.41	84.16	-2.2	0.4
2.8	83.67	-2.3	0.42	82.85	-2.47	0.41	81.15	-2.82	0.41	79.43	-3.17	0.41	84.24	-2.19	0.4
2.9	84	-2.23	0.43	82.87	-2.47	0.41	81.17	-2.82	0.42	79.43	-3.17	0.41	84.33	-2.17	0.4
3	84.23	-2.16	0.43	84.47	-2.14	0.42	81.17	-2.82	0.42	79.43	-3.17	0.41	84.42	-2.15	0.41

to 60% sooner than methods such as DDM or Page-Hinckley. The TSI values of EDDM imitate the pattern of improvement shown in the Kappa-Temporal results. There is a marginal improvement in TSI from 0.37 to 0.38, occurring at $T = 1.4$.

Thirdly CUSUM also reports improvements in the Kappa-Temporal statistic when used alongside BD-SCR. A Kappa-Temporal improvement from 0.44 to 0.41 is achieved, and similarly to DDM the optimal T value for CUSUM is $T = 2$, where its maximum Kappa-Temporal score is first achieved. As with the previous method the TSI value also improves in line with the Kappa-Temporal values, in this case from 0.34 to 0.36 at $T = 1.9$. This increase in TSI is the joint largest all methods, shared with Page-Hinckley.

The final state-of-the-art method evaluated is Page-Hinckley, which predictably achieves Kappa-Temporal performance improvements using BD-SCR algorithm. Page-Hinckley reports Kappa-Temporal results improving from -0.57 to -0.54, however it achieves this at the second lowest T value of all methods at $T = 1.6$. As mentioned previously, Page-Hinckley also jointly holds the largest increase in TSI values, from 0.3 to 0.33 achieved at $T = 1.6$, as with its Kappa-Temporal evaluation.

The No-Change detector does not improve its Kappa-Temporal performance using the BD-SCR method whatsoever. In fact it actively performs worse when used with BD-SCR, with its Kappa-Temporal score degrading from -0.18 to a maximum of -0.57. The reason for this is the underlying drift detection mechanism used in the method. The No-Change detector performs no statistical drift detection and instead simply

signals an alarm every 60 instances. The BD-SCR burst detection algorithm is also then triggered every sixty instances since it correlates with detected drifts. Where the No-Change detector was previously free to reset the classifier every 60 instances, BD-SCR restricts that ability based on the temporal dependence severity of the stream at the time the drift is detected. The frequency of resetting then becomes unreliable, and since no real concept drift detection takes place the base classifier becomes outdated. This is reflected in the drop in classifier accuracy for the No-Change detector. The decrease in Kappa-Temporal and TSI performance is due to a similar reason. When the classifier was reset every 60 instances, it was not necessarily being training on temporally dependent data for very long. This is especially true in the Electricity dataset where the temporal dependence occurs in clear cylindrical peaks, as shown in Figure 3.1. Preventing the frequent resetting exposes the classifier to longer windows of temporally dependent data, thus impacting its Kappa-Temporal and TSI scores.

4.5.2 Results of the Forest Covertype Dataset

The Forest Covertype results provided in Table 4.4 illustrate a different picture than those of the Electricity dataset. It is apparent from the offset that optimal values for T are fuzzy, certainly less immediately clear when compared to the Electricity dataset results. Performance results are positive and follow a similar trend to the results of the Electricity datasets discussed above. Evaluation scores for TSI are improved for all drift detection methods other than the No-Change detector. Kappa-Temporal values are improved for some methods; others achieve similar performance to the results in 4.2 where the BD-SCR method is not used. This is due to the experimental values of T stopping at 3. The Forest Covertype dataset contains severe temporal dependencies in its data, as shown in Figure **??**. The T value at which Kappa-Temporal results begin to improve further reinforce the issue with temporal dependence in the Forest Covertype dataset. The increases in temporal dependence are so severe that there is virtually no significant change in the Kappa-Temporal scores for state-of-the-art methods before $T = 1.9$. Some methods would benefit with T values beyond 3, however for consistency in the experimental process T values above 3 are not used.

DDM is the first method where its Kappa-Temporal statistic does not improve over its baseline score of -1.86, instead achieving a maximum Kappa-Temporal score of -2.16. However from analysing the linear increase of Kappa-Temporal scores for DDM over values of T above 1.9, it is predictable that the Kappa-Temporal values will continue to increase beyond the experimental T limit of 3. The TSI metric highlights a similar trend in the T values and therefore the drastic increases in temporal dependence in the data. TSI values also do not begin to improve until $T = 1.9$, however improvement is reached at $T = 2.6$, and the maximum TSI value of 0.43 is achieved at $T = 2.9$.

EDDM performs slightly better than DDM in terms of its Kappa-Temporal statistic. Whilst it also does not improve over its baseline score, achieving 2.14 at $T = 3$, it is essentially achieving the same performance since the difference is only 0.04. In a similar fashion to DDM, it is clearly predictable that the Kappa-Temporal scores for EDDM will continue to improve above $T = 3$. The evaluation shows linear improvement above $T = 1.9$ and the values have not begun to plateau as is evident in the result for the Electricity dataset. With regards to the TSI metric, the story is again similar to DDM. EDDM reports a base TSI score of 0.41 with the Forest Covertype dataset, and achieves a score of 0.42 at $T = 3$. Again, this is likely to improve with T values beyond 3.

CUSUM reports improved performance with regards to both Kappa-Temporal and TSI statistics, and does so well before T reaches the experimental limit. This is in stark contrast to DDM and EDDM, where both methods indicate that they would benefit from a T value that is higher. CUSUM improves its Kappa-Temporal value from -2.84 to -2.82 at $T = 2.7$. The TSI metric for CUSUM reports an improved score of 0.42 at $T = 2.9$ over its baseline score of 0.4.

The most improved method with the Forest Covertype dataset is the Page-Hinckley test. This is quite interesting considering that Page-Hinckley is the worst performing method without BD-SCR, achieving the worst Kappa-Temporal and TSI scores; a clear indication that this method is poor in temporal environments. However, when combined with BD-SCR, Page-Hinckley reports clear performance improvements. The most significant is the Kappa-Temporal score where Page-Hinckley achieves a score of -3.17, an improvement of 0.06 over its baseline score of -3.23. This is largest improvement margin in Kappa-Temporal across all methods using the Forest Covertype dataset.

Finally the No-Change detector makes no improvements on either Kappa-Temporal or TSI statistics, however its TSI value of 0.41 is not significantly worse than its base score of 0.42. The same cannot be said for Kappa-Temporal, however, where No-Change reports a score of -2.15 at $T = 3$ whereas its base score without BD-SCR is -1.45. The reasons for the lack of improvement for No-Change have been explain in-depth above during the discussion of the results for the Electricity dataset.

4.5.3 Optimal Values for Parameter T

The T parameter describes the maximum amount of burst to be tolerated when using the BD-SCR method to manage classifier resetting. Effectively, this represents the amount of sudden change in the temporal dependence of the data that is to be considered acceptable as defined by the user. Higher values of T allow for larger increases in temporal dependence, while the base classifier is reset as normal during detected

concept drifts. Lower values result in more frequent preventions of resetting the base classifier since the method is less amenable to changes in the temporal nature of the data.

Deciding the optimal value for T is a non-trivial task. Lower values which restrict resetting frequently may result in the classifier becoming outdated due to concept drift in the data. This can result in reduced performance, as illustrated by the comparatively lower classifier accuracy for low values of T in Tables 4.3 and 4.4. On the contrary, higher values representing more tolerant approaches may not fully encapsulate the temporal dependence severity during the resetting decision and can result in the over-resetting problem where classifier accuracy can become misleading, as discussed above. Instead, when temporal dependence is present metrics such as the Kappa-Temporal statistic or TSI should be used over classifier accuracy. To optimise T the value which maximises metrics such as the Kappa-Temporal statistic or TSI should be chosen. This allows for the classifier to be as stable and adaptable to temporal dependencies in the data as possible, whilst still allowing the classifier to be reset for detected drifts. Identifying the maximum T value for these metrics however is non-trivial. Depending on the dataset and the temporal dependencies in the data, T may well trend well beyond the experimental value of 3 used in this experimentation. This is evident in the datasets used in this evaluation. The Electricity dataset contains lagged periods of temporal dependence as shown in Figure 3.1 and represented by the relatively low average optimal T value of 2. In contract, the Forest Covertype dataset is much more heavily temporally dependent, and the optimal T value for this dataset clearly trends beyond 3. Future improvements to this method could focus on improving the optimisation of T through dynamic allocation of its optimal value for any given dataset.

4.6 Summary

This research presents BD-SCR as a novel approach to classifier resetting during concept drift detection in the presence of temporal dependence. Using a short term and long term average statistical event detection method, BD-SCR analyses the severity of temporal dependence to make an informed decision to reset the base classifier. Experimental results show that BD-SCR is effective at overcoming the problem of over-resetting during periods of temporal dependence by selectively resetting the classifier.

The sensitivity of the threshold parameter T means finding the optimal balance between statistical detectors and No-Change is difficult and unclear. In some datasets, such as Electricity, it is fairly trivial to analyse the results and determine the optimal values for T. However in datasets with more complex temporal dependencies, such as the Forest Covertype dataset, the optimal values for T are less obvious and unclear. The

complexity would broaden further if real-time data was used.

BD-SCR challenges the current architectural approach to drift detection to accommodate for other stream related anomalies. Future work in this field should look to improve BD-SCR by expanding the algorithm to automatically determine the optimal value for T. Additionally the proposed structure of BD-SCR could be used to develop more sophisticated ways of handling temporal dependence in concept drift streams.

Chapter 5

Accounting for Temporal Dependence with Classifier Ensembles

This chapter proposes an ensemble based method, Kappa-Temporal Updated Ensemble (KTUE) for data stream mining that is capable of coping with concept drift and temporal dependence through the use of the Kappa-Temporal statistic proposed by Zliobaite et al. (2015). This method is based upon the Kappa Updated Ensemble method proposed by Cano & Krawczyk (2020), which uses the Kappa statistic as the basis for its voting system among internal classifiers. Whilst KUE has shown to be an effective ensemble method for classification of evolving data streams, it does not offer any mechanism for including temporal dependence in the decision process for resetting base classifiers. Since the Kappa-Temporal statistic is an immediate progressive evolution of the Kappa statistic, it follows that KUE should serve as a suitable basis for developing an ensemble based method for evolving stream mining in temporally dependent environments. The proposed method contained in this chapter is an original contribution of this thesis.

The format of this chapter is as follows: initially some background into ensemble methods provided; secondly the proposed KTUE method is explained; thirdly the experimental design and setup is described; fourthly the results of experimentation are provided alongside an in-depth, critical discussion; and finally conclusions are provided alongside suggestions for improvement.

5.1 Problem Definition

Adopting ensembles for concept drift detection can be an effective solution, but it can also incur high costs. In comparison to statistical and window-based methods where a single base classifier is trained and tasked to perform predictions, ensemble methods instead deploy a number of classifiers. Since ensembles opt to use a larger number of classifiers, it follows that is less likely for the entire ensemble to become outdated or irrelevant during concept drift periods since it becomes possible to only reset the most poorly performing classifiers within the ensemble. However, this also results in potentially higher performance costs than using a single classifier.

The performance of a ensemble-based concept drift detection method is ultimately concerned with the voting and resetting mechanisms involved. There have been many suggested methods for ensemble-based drift detection proposed in the literature that have portrayed varying implementations for voting and resetting classifiers within an ensemble. For example, the Streaming Ensemble Algorithm (SEA) (Street & Kim 2001) is a simple approach whereby individual classifiers give their predictions using simple majority voting and then classifier which is performing most poorly is automatically removed from the ensemble and a new classifier is introduced to replace it. The fundamental idea behind SEA is that by removing the most poorly performing classifier, the ensemble is kept as up-to-date and relevant as possible. However, simple removing only the most poorly performing classifier can result in several other under-performing classifiers remaining in the ensemble. This issue is only exacerbated if the margin of difference in performance between outdated classifiers is minimal.

To address the shortcomings of SEA, the Accuracy Weighting Ensemble (AWE) was introduced by Wang et al. (2003). AWE calculates the mean square error of all classifiers in the ensemble to replace n poorly performing classifiers. AWE was found to be an effective improvement over SEA. However it suffered in instances of sudden concept drift where often the majority of the ensemble would be replaced, essentially resulting in a silencing effect on the entire ensemble resulting in no class prediction.

A further improvement, Accuracy Updated Ensemble (AUE), was proposed by Brzeziński & Stefanowski (2011) to improve upon the drawbacks of AWE. The core optimisation proposed by AUE surrounded the weighting mechanisms for classifiers within the ensembles. AUE opted to introduce online classifiers that could be updated directly rather than relying on adjusting weights, as in AWE. Essentially AUE allowed for a reduction in block size without a detrimental effect on the ensemble classification performance. This weighting function was further improved with the proposal of AUE2 (Brzezinski & Stefanowski 2014).

While research continues in the field of concept drift detection for data stream mining, only the most recent research has started to identify some underlying problems (Zliobaite et al. 2015, Bifet 2017, Wares et al. 2019, 2021). Recently published work in the field of concept drift detection has suggested that in situations where a data stream is subject to an anomaly known as temporal dependence, resetting the base classifier during a detected concept drift can lead to misleading and inaccurate performance evaluation. In fact Bifet (2017) found that in the presence of temporal dependence, resetting the base classifier periodically without performing any statically relevant concept drift detection actually outperformed state-of-the-art drift detectors in terms of predictive accuracy. This suggests that without accounting for temporal dependence in the data, methods for concept drift detection cannot be reliably evaluated.

At the time of writing, no ensemble-based methods have been suggested for adapting to temporal dependence within evolving stream data. The Kappa Updated Ensemble (KUE) (Cano & Krawczyk 2020) is an ensemble method that uses the Kappa statistic for performing classification in evolving data streams. This forms a concrete basis for constructing a Kappa-Temporal-based ensemble for classification tasks in temporally dependent evolving data streams.

5.2 Novel Contributions

This chapter proposes an original contribution to research in the form of a novel ensemble method for performing classification in evolving data streams containing temporal dependence in the data. Current literature contains various suggestions for ensemble methods that can be adopted for concept drift detection, but at the time of writing no such methods exist for also coping with temporal dependence in the stream data.

It is anticipated that the KTUE ensemble method proposed herein will not only offer an ensemble algorithm for stream mining in evolving, temporal environments, but also provide the basis for further research in the future.

5.3 Kappa-Temporal Updated Ensemble

This subsection presents the Kappa-Temporal Updated Ensemble (KTUE) algorithm in terms of its design and motivation. As aforementioned, the Kappa-Temporal statistic proposed by Zliobaite et al. (2015) is an evaluation metric providing a means for considering the performance of classifiers in temporally dependent environments. The Kappa-Temporal statistic is an advancement of the Kappa statistic that aims to address its shortcomings in temporal environments. Similarly, the Kappa Updated Ensemble (KUE) Cano & Krawczyk (2020) is a recent proposed ensemble method for machine

learning in evolving data streams. Since KUE used the Kappa statistic as its fundamental evaluation mechanism for assessing ensemble components during the learning process, it follows that KTUE can adopt a similar platform but instead opt for the Kappa-Temporal statistic for evaluation in temporally dependent environments.

KTUE is based upon the KUE ensemble platform, and follows similar structure and algorithmic design. The ensemble is constructed of k components, base classifiers, such that $y_j \in \varepsilon(j = 1, 2, ...k)$ where y_j is a base classifier and ε is the ensemble. Random dimensional subspacing is used for varying the dimensionality of new ensemble members whilst the Poisson distribution is used for weighting replacement components within the generated subspaces.

The concept of feature subspace dimensionality is a research domain within itself (Fan & Lv 2010, Faust et al. 2018) and is considered out of scope for the purposes of this research, however the fundamental outline is provided as follows. For each arriving chunk of data, the instances are portrayed through existing random subspaces belonging to each classifier within the ensemble. Each arriving instance is weighted using the Poisson distribution (Bifet, Holmes & Pfahringer 2010). This approach has been previously successful in popular ensemble based stream learning methods, such as OzaBag (Oza & Russell 2001) and Adaptive Random Forest(Gomes et al. 2017). The full algorithm for KTUE is provided in Algorithm 8

Algorithm 8 KTUE Algorithm

S: A data stream
C: A single chunk
KT_c: The Kappa-Temporal statistic for a single classifier
m: Maximum chunk size
p: Number of chunks processed
i: A single instance
k: The maximum number of classifiers in the ensemble
c: A single classifier in the ensemble
q: The number of new classifiers to train

for $S_i \in S$ **do**
  while $C_n < m$ **do**
    Add i to C
  end while
  for $C_i \in \{C_1, ...C_m\}$ **do**
    Add i to C
    if $p = 0$ **then**
      for $c \in \{1, ..., k\}$ **do**
        Weight instances in C using Poisson
        Calculate r-dimensional random subspace φ
        Train classifer c on $\varphi(C)$
        Compute KT_c
      end for
    else
      for $c \in \{1, ..., k\}$ **do**
        Maintain established r-dimensional random subspace φ
        Train classifier c on $\varphi(C)$
        Compute KT_c
      end for
      for $\{1, ..., q\}$ **do**
        Weight instances in C using Poisson
        Calculate r-dimensional random subspace φ'
        Train classifier c' on $\varphi'(C)$
        Compute $KT_{c'}$
        if $KT_{c'} > KT_c$ **then**
          Replace classifier c with c'
        end if
      end for
    end if
    Increment chunks processed $p = p + 1$
  end for
end for

KUE adopts the Kappa statistic as a weighting mechanism for voting predictions and has proven to be an effective ensemble method for learning in evolving data streams, particularly where the arriving data is subject to class imbalance. However, it has been shown by Zliobaite et al. (2015) that the Kappa statistic can be misleading in temporal environments where a positive value of k can be achieved in scenarios where a basic Persistent classifier will be performing poorly. As such, KTUE proposes the use of the Kappa-Temporal statistic to investigate the performance of a ensemble method in evolving data streams that also contain temporal dependence.

Classifiers in the ensemble vote on arriving instances to give their prediction. The ensemble prediction of a class label for a given instance is achieved through weighted majority voting of each classifier contained within the ensemble. This is given in Equation (5.1).

$$\hat{y} = \arg\max_{i} \sum_{j=1}^{k} KT_j \; p(i|y_j(x)), \tag{5.1}$$

Note that ensemble classifiers are not inherently authorised to vote on all instances. Only classifiers whose Kappa-Temporal statistics are equal to or greater than the ensemble average are allowed to vote. This is a considerable variation on the KUE method where classifiers are indeed restricted based on their Kappa statistic values, however KUE restricts voting to classifiers where the Kappa statistic in greater than zero. This is impractical in temporal environments, where Kappa-Temporal scores are often considerably below zero. Restricting classifiers' ability to vote to only those with a Kappa-Temporal score above zero will result in the vast majority, if not all, of the ensemble classifier abstaining.

The authors of KUE note the same problem; "...in the unlikely case of all classifiers having a Kappa value < 0 means that no classifier was able to model the data..." (Cano & Krawczyk 2020, p.185). In the context of temporal dependence and the Kappa-Temporal statistic, this is not a trivial issue. Kappa-Temporal values in datasets that contain even moderate temporal dependence frequently trend to -10 or even below. The most popular datasets containing temporally dependent evolving data are described in chapter 3. In an effort to avert this, KTUE instead monitors the average Kappa-Temporal statistic of the ensemble as a whole provides a balance between allowing classifiers to make predictions whilst still replacing the weakest performing components. Classifier components are replaced based on their Kappa-Temporal statistic. When a new classifier is trained at the end of processing a chunk, if its Kappa-Temporal score exceeds that of the weakest classifier in the ensemble (that which has the lowest

Kappa-Temporal score) then it replaces said classifier.

At the time of writing, no ensemble based method for machine learning in both temporally dependent and evolving environments exists. KTUE expands on the established method of KUE, but instead uses the Kappa-Temporal statistic as a weighting mechanism for making predictions in temporal environments where the Kappa statistic has been shown to be ineffective.

5.4 Experiment

KTUE was implemented using Massive Online Analysis (MOA) (Bifet, Holmes, Pfahringer, Kranen, Kremer, Jansen & Seidl 2010), a popular open source Java framework for stream mining. KTUE is compared to state-of-the-art ensemble methods for drift detection, including KUE. Ensemble methods used for comparison are state-of-the-art methods including block-based, bagging and boosting-based. Table 5.1 provides a list of all ensemble methods used including their base classifier type. MOA configuration details for all methods are default values.

Datasets used for experimentation are split into categories; temporally and non-temporally dependent. KTUE aims to provide an ensemble based method for classification in temporal environments and as such use the Electricity, Forest Covertype and KDD '99 Cup datasets are discussed in 3. However, non-temporally dependent datasets are also used for benchmarking. Synthesised methods are omitted as no concept drift data simulation methods are capable of including temporal dependence into the data. Table 5.2 shows the datasets used, provides an outline of their features, class labels and total number of instances and also indicates if they are temporally dependent in nature.

Table 5.1 Experimentation Ensemble Methods

Acronym	Algorithm	Classifier	Citation
LNSE	Learn++.NSE	Naive-Bayes	Elwell & Polikar (2011)
DWM	Dynamic Weighted Majority	Naive-Bayes	Kolter & Maloof (2003)
DACC	Dynamic Adaptation to Concept Changes	Naive-Bayes	Jaber et al. (2013)
ADACC	Anticipative Dynamic Adaption to Concept Changes	Naive-Bayes	Jaber et al. (2013)
OCB	Online Coordinate Boosting	Hoeffding Tree	Pelossof et al. (2009)
LB	Leveraging Bagging with ADWIN	Hoeffding Tree	Bifet, Holmes & Pfahringer (2010)
KUE	Kappa Updated Ensemble	Hoeffding Tree	Cano & Krawczyk (2020)
AUE	Accuracy Updated Ensemble	Hoeffding Tree	Brzeziński & Stefanowski (2011)
AWE	Accuracy Weighted Ensemble	Hoeffding Tree	Wang et al. (2003)
OBA	Oza Boost ADWIN	Hoeffding Tree	Oza & Russell (2001)
OBASHT	Oza Bag Adaptive Size Hoeffding Tree	Hoeffding Tree	Bifet et al. (2009)
OBAD	Oza Bag Adwin	Hoeffding Tree	Bifet & Gavalda (2007)
ARF	Adaptive Random Forest	Hoeffding Tree	Gomes et al. (2017)
HEB	Heterogeneous Ensemble Blast	Hoeffding Tree	van Rijn et al. (2018)

Table 5.2 Experimentation Datasets Summary

Dataset	Instances	Class labels	Attributes	Temp. Dep.	Drift Type
Electricity	45,312	2	8	True	Gradual
Covertype	581,012	7	54	True	Sudden
KDD '99 Cup	494,020	23	41	True	Recurrent
BNG Bridges	299,284	2	42	False	Sudden
BNG Hepatitis	45,312	2	8	False	Gradual
BNG Zoo	581,012	7	54	False	Gradual
BNG Wine	2,313,153	58	6	False	Gradual

Table 5.3 provides the Kappa-Temporal statistic results for each of the learners across all datasets. The results shown are the average Kappa-Temporal statistic values across all instances of each dataset.

5.4.1 Performance analysis in non-temporally dependent evolving environments

The datasets which are not temporally dependent but contain some form of concept drift are BNG Bridges, BNG Hepatitis, BNG Zoo and BNG Wine. The foremost point of discussion is that KUE, OBA and ARF outperforms KTUE across all four datasets. This is somewhat expected since KTUE is designed primarily to operate in temporal environments, therefore state-of-the-art ensemble methods for evolving data streams are expected to outperform KTUE. Surprisingly, however, KTUE performs rather well in comparison to other methods. Indeed KTUE is found to outperform over 50% of the other ensemble methods used for evaluation in three out of the four datasets in this area.

The outlier to this is BNG Bridges, where KTUE is only the twelfth best performing method out of fifteen. OCB performs the worst here with a Kappa-Temporal score of -8.04 and is the only method to achieve a negative Kappa-Temporal score for this dataset. AWE performs relatively poorly in comparison to the other methods with a positive Kappa-Temporal score of 8.89. This is still considerably lower that all other methods however. KTUE edges LB and falls narrowly short of besting OBAD by a Kappa-Temporal score of 3. AUE is the clear winner for BNG Bridges with a high Kappa-Temporal score of 74.44, outperforming all other methods by a considerable margin.

BNG Zoo boasts a more successful story for KTUE where it sits comfortably as the ninth best performing method. The margin in the top 10 performers for this dataset is small with a margin of 3.8 between LNSE in tenth and KUE in first place. KTUE achieves a higher Kappa-Temporal statistic than LNSE, AWE and OCB once again, but also now performing better than LB, and both DACC and ADACC.

For BNG Wine and BNG Hepatitis, KTUE performs better than half of all other ensemble methods in both scenarios. Results show that KTUE is the seventh best performing algorithm for BNG Wine, and the sixth best performing for BNG Hepatitis. Again, KTUE achieves a higher Kappa-Temporal score than AWE, LNSE and OCB, as has been the case across all non-temporal datasets. Interestingly AUE is the worst performing method for BNG Hepatitis when it has been one of the best performing methods across the other datasets.

Table 5.3: Experimentation Results (Kappa-Temporal)

Learner	BNG Bridges			Covertype			Electricity			BNG Hepatitis			KDD '99 Cup			BNG Wine			*BNG Zoo*		
	Acc	*KT*	TSI	Acc	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DACC	59.69	46.44	0.57	90.75	-55.26	0.43	83.43	27.94	0.43	84.27	52.21	0.18	99.74	-83.57	0.06	87.36	80.8	0.61	88.03	84.36	0.74
ARF	58.41	44.75	0.5	94.27	4.52	0.44	90.62	36.09	0.46	92.5	77.22	0.25	99.8	-40.73	0.06	94.5	91.65	0.64	91.83	89.33	0.74
LB	45.63	27.77	0.37	91.76	-82.2	0.42	89.8	30.46	0.4	92.28	76.54	0.25	99.77	43.42	0.06	94.36	84.74	0.64	88.32	84.74	0.73
HSB	74.28	65.83	0.66	91.54	-40.92	0.43	81.77	-24.27	0.34	91.14	73.09	0.24	99.67	-61.86	0.05	92.86	89.15	0.64	92.72	90.49	0.75
OBAD	58.87	45.36	0.5	85.22	-151.7	0.38	84.35	-6.69	0.38	91.64	74.59	0.24	99.7	-105.85	0.05	93.67	90.38	0.64	91.99	89.54	0.75
DWM	72.45	63.4	0.65	82.89	-189.06	0.39	79.7	-38.68	0.35	87.67	62.55	0.24	99.52	-230.97	0.04	91.98	87.82	0.63	92.6	90.34	0.75
OBASHT	67.92	57.39	0.58	83.92	-172.34	0.36	83.25	-14.2	0.35	90.17	70.13	0.24	99.46	-268.13	0.05	93.16	89.6	0.64	92.05	89.62	0.75
ADACC	59.69	46.45	0.57	90.32	-62.38	0.43	89.62	29.22	0.43	84.45	52.75	0.18	99.64	-146.27	0.06	87.39	80.85	0.61	88.04	84.37	0.74
KUE	60.6	51.49	0.52	86.6	-126.32	0.38	76.4	-56.39	0.29	91.3	73.98	0.23	94.09	-3704.17	0.02	93.8	90.43	0.64	92.7	91.07	0.75
KTUE	54.15	36.07	0.51	85.53	-144.32	0.38	72.82	-81.3	0.28	90.53	70.99	0.24	93.54	-4692.03	0.02	93.1	89.57	0.64	91.09	88.21	0.74
OBA	54.34	39.35	0.51	92.19	-30.4	0.44	88.76	23.36	0.4	87.99	63.52	0.22	85.84	-9851.45	0.06	92.67	88.87	0.64	90.98	88.22	0.74
OCB	18.67	-8.05	0.38	77.76	-279.28	0.33	89.75	30.15	0.41	90.16	70.1	0.25	88.41	-72424.02	0.05	63.36	51.94	0.45	58.58	45.88	0.53
AWE	31.42	8.89	0.29	80.48	-229.5	0.34	70.92	-98.21	0.27	88.33	64.56	0.24	81.23	-66011.8	0.11	92.09	87.99	0.63	56.67	43.38	0.52
AUE	61.21	74.44	0.53	87.21	-116.14	0.39	77.44	-53.8	0.31	91.59	48.48	0.24	85.38	-107.81	0.15	93.18	89.65	0.64	91.24	88.56	0.75
LNSE	68.47	58.11	0.62	68.94	-439.67	0.26	71.06	-97.26	0.29	86.99	60.47	0.24	79.19	-72152.2	-0.13	91.26	86.73	0.63	90.26	87.28	0.74

In terms of classifier stability represented by the TSI metric, KTUE performs well across these datasets in comparison to other methods. For BNG Bridges, KTUE outperforms AWE, OCB and LB in terms of TSI score. In addition to this it performs closely to its immediate neighbours KUE, ARF and OBA. BNG Hepatitis shows KTUE outperforming ADACC, DACC and KUE in terms of TSI. However the TSI scores for the remaining methods in this data set are very close and only vary by minor margins. In this respect KTUE is a joint top performer for BNG Hepatitis, but as mentioned the separating margin between methods is minimal. A similar scenario exists for BNG Wine, where KTUE is again a joint top performer in terms of TSI, however the margins are again slim. In this dataset KTUE outperforms LNSE, AWE, OCB, DWM and DACC. Finally BNG Zoo also portrays KTUE as a joint top performing method in terms of TSI, with it clearly outperforming LNSE, AWE, OCB and LB.

5.4.2 Performance analysis in temporally dependent environments

KTUE is expected to perform well in temporally dependent evolving data streams since its vote weighting mechanism, the Kappa-Temporal statistic, is a metric designed to evaluate base classifiers in temporal environments. The results show that across three well-known temporally dependent data streams in literature; Electricity, Forest Covertype and KDD '99 Cup.

The worst performing temporal dataset for KTUE is Electricity where it finishes as the thirteenth best performing method with a Kappa-Temporal score of -81.3. There is a considerable and notable difference between the next best performing method, KUE, which achieves -56.4. This is also the start of a observable trend where KTUE is almost always outperformed by KTUE, however not always by such a meaningful margin. KTUE does considerably outperform both LNSE and AWE as mentioned above.

KDD '99 Cup sees KTUE perform eleventh best, but it misses out on the top ten by a considerable margin. KDD '99 Cup sees all ensemble methods perform poorly, with the best performing method ARF achieve a Kappa-Temporal score of -40.73, and the worst method OCB achieve a woeful -72424.01. KTUE again outperforms AWE, LNSE and OCB by considerable margins, but loses significantly to KUE by a Kappa-Temporal margin of nearly 1000. KDD '99 Cup is a dataset that is known to contain difficult periods of recurring concept drift and significant periods of high levels of temporal dependence (Zliobaite 2010). This goes some way to explaining the poor performance of all methods across the board as this could perhaps be considered the most challenging dataset of the three.

Finally, Covertype houses KTUE's best performance where it achieves ninth place, outperforming LNSE, OCB and AWE as expected, but also DWM, OBAD and OBASHT. KTUE sits comfortably ahead of the ensemble methods below it, in fact there is a margin of almost 300 between itself and worst performing LNSE. KUE again is the method that comes in just ahead of KTUE by a margin of 22. ARF is the top performing method, as it has been for 50% of the temporally dependent datasets, with the only positive Kappa-Temporal score of 4.52.

With regards to the TSI metric it can be observed that in datasets that are temporally dependent there is more variation and spread in the TSI scores for all methods. That is, the TSI values are not so tightly compacted as they are in the non-temporally dependent datasets. Beginning with Covertype, KTUE outperforms LNSE, AWE, OCB and OBASHT in terms of its TSI score. It performs in the middle of the pack when compared to the other state-of-the-art methods, and is jointly performable with OBAD and KUE with a TSI value of 0.38. The Electricity dataset projects a different story for KTUE, where its TSI performance is closer to the bottom in comparison to other methods. KTUE outperforms only AWE, and only by a very trivial margin. It is worth noting, however, that the KUE method outperforms KTUE by the same slim margin that itself outperforms AWE. Whilst it is positive that KTUE outperforms a state-of-the-art method, in terms of classifier stability other methods offer more competitive solutions. Finally the KDD '99 Cup dataset provides an interesting picture of classifier stability for all methods. TSI values across the board are at their lowest values for this dataset, which is to be expect considering how temporally dependent the dataset is. AUE clearly leads in TSI performance here, with a value of 0.15 which clearly surpasses the average. In contrast to this, LNSE takes up last place by a clear margin with a TSI value of -0.13. KTUE comes in joint second last alongside KUE with a TSI value of 0.02. However, the other methods are not significantly outperforming KTUE with other methods averaging a TSI score of 0.05. Whilst KTUE isn't the most stable classifier in this dataset, it isn't considerably under performing in comparison to other methods. In addition, the KDD '99 Cup is a challenging dataset with a heavily temporally dependent disposition.

Overall we can observe that KTUE is shown to be an effective method at stream classification in evolving temporal environments. In particular, the results indicate that as the temporal dependence severity in the data increases, so does the performance of KTUE. This is evident by observing the datasets where KTUE is most and least effective in comparison to other methods. The Electricity dataset contains the lowest levels of temporal dependence out of the three datasets and is the least effective dataset for KTUE. However, Forest Covertype and KDD '99 Cup contain much more severe levels

of temporal dependence and KTUE's performance improves for these datasets.

5.4.3 Temporal Dependence and Class Imbalance

While KTUE does outperform some state-of-the-art methods in temporal environments, it is still outperformed by ensemble methods that are not explicitly designed for coping with temporal dependence. In turn, other state-of-the-art methods, whilst performing largely better than KTUE, still return hugely negative Kappa-Temporal results across the temporally dependent datasets, indicating that they are performing inadequately in these scenarios.

It is surprising that KTUE does not outperform more of the existing state-of-the-art methods, perhaps most importantly KUE which in many ways may be considered its predecessor considering the Kappa-Temporal statistic is, essentially, an advancement on the Kappa statistic. The explanation for KTUE's inability to best KUE may be due to class imbalance in the datasets.

Temporal dependent data streams do not have to contain class imbalance. While a period of time may exist where arriving class labels are the same, this does not necessarily result in an imbalance in the data. Over time the stream may naturally correct itself as temporal dependence windows diminish or expire, allowing time agnostic instances to arrive. Since data streams are unbound in size, volume and veracity, it remains possible for the class balance of a stream to self-correct over time.

In research scenarios, however, this is not the case. Rather than use real, active data streams for developing new methods for stream mining, streaming static datasets has become the norm. It has already been discussed that this has led to difficulty in acquiring datasets that contain concept drift, and is only more difficult to acquire datasets that also contain temporal dependence. For those datasets that are used, such as those used in this experiment, the more severe the temporal dependence in the data then the larger the class imbalance. Since these datasets are static, offline data that is streamed to online methods in a simulated fashion, there exists no natural mechanism for the stream to balance itself out over time, as may be possible with a real active stream.

The Kappa-Temporal statistic does not take class imbalance into account. In stark contrast to this, the Kappa statistic, as used in KUE, is a particularly effective measure for accounting for class imbalance. This means that whilst KTUE should theoretically offer a more suitable solution for stream mining in temporal environments than KUE, it may not truly be the case since the datasets used for evaluation contain, sometimes vast, class imbalance. The result is that ensemble methods more suitable for coping with

class imbalance in evolving data end up outperforming KTUE, even though the Kappa-Temporal statistic is a more suitable weighting mechanism for such environments.

5.5 Summary

This chapter discusses how the Kappa-Temporal statistic provides a more appropriate metric for evaluating classifier performance in temporal environments, highlighting how one downfall of the Kappa statistic is its inability to operate effectively in such scenarios. A new ensemble method, Kappa-Temporal Updated Ensemble (KTUE) is proposed as a method for machine learning in evolving temporal environments. KTUE is based on the Kappa Updated Ensemble (KUE) but instead uses the Kappa-Temporal statistic as a vote weighting mechanism. Results indicated that while KTUE is capable of outperforming some state-of-the-art methods, it is still behind in comparison to others, including KUE.

Class imbalance in the datasets used for evaluation in one suggestion for KTUE's poorer performance compared to KUE in temporal scenarios. Real life data streams may not necessarily be imbalanced even if they are temporally dependent since class labels may balance out over time. In research environments, the norm is to stream offline, static datasets to simulate data streams which removes the possibility and potentiality for class imbalance to naturally equalise over time. Instead this may result in severe class imbalance in the data. Since the Kappa-Temporal statistic is not capable of coping with class imbalance, but the Kappa statistic is, this may provide some justification to KUE's superior performance.

Chapter 6

Simulating Concept Drift Data with Temporal Dependence

This chapter focuses on the design, implementation and evaluation of the novel algorithm **T**emporal **D**ependence **I**nclusion for **C**oncept **D**rift **S**imulation (TDI-CDS). TDI-CDS is a novel algorithm for concept drift data simulation. This proposed method differs from existing, published techniques by providing a mechanism for including temporal dependence in the generated instances.

This chapter is constructed in the following manner: first the problem surrounding a lack of datasets for the evaluation of new methods in the domain of concept drift detection is explained; second the method and contributions of TDI-CDS are described; thirdly the experimental setup is defined; fourth the experimental results are critically discussed; finally conclusive remarks and suggestions for future improvements are provided.

6.1 Problem Definition

One of the key existing research gaps in the domain concept drift detection is a lack of established datasets that can be used for the evaluation of newly developed methods. This problem has been identified in various published research, particularly in more recent works Krempl et al. (2014), Wares et al. (2019), Mehmood et al. (2021). Whilst very few benchmark datasets exist, there have emerged a small number of popular datasets that have been widely adopted. These include the Forest Covertype (Blackard et al. 1998), Electricity (Harries & Wales 1999), KDD '99 Cup (Olusola et al. 2010) and Poker Hand (Dua & Graff 2017). Table 6.1 presents an overview of these datasets, including their year of publication, number of instances and number of attributes.

Table 6.1 Concept Drift/Temporal Dependence datasets

Dataset	Instances	Attributes	Year
Poker Hand	1025010	11	2007
KDD '99	4000000	42	1999
Forest Covertype	581012	54	1998
Electricity	45312	5	1999

A lack of benchmark datasets presents a unique problem for the publication and proposal of new methods. Without a variety of established, benchmark datasets, statistical evaluation of new methods becomes particularly challenging. In an effort to overcome this problem, various simulation, or synthesiser, algorithms have been proposed. These methods are capable of producing data streams consisting of statistically generated instances. Continued research into this solution also saw the development of data simulation methods capable of also containing concept drift. These methods have allowed for robust evaluation of concept drift detection methods without the reliance on only a few select datasets. Published methods often choose to conduct their evaluation on a combination of both synthesised and real datasets, and surveys may include an analysis of both in their review (Iwashita & Papa 2018).

However, with the recent discoveries surrounding the negative impact of temporal dependence upon the evaluation of concept drift detectors (Zliobaite et al. 2015, Bifet 2017, Wares et al. 2019, 2021) this problem has resurfaced. Whilst the over-resetting problem has been acknowledged in literature, the development of novel methods is stunted by a lack of established datasets that contain temporal dependence within the concept drift data. The current evaluative process is to use existing popular concept drift datasets since these have been to shown to contain some levels of temporal dependence within them (Bifet 2017). However, this falls victim to the same drawback as traditional concept drift evaluation methods; a lack of datasets. Table 6.2 shows the maximum temporal lengths across four popular concept drift datasets. Note that the maximum repetitions is the maximum number of times an arriving class label repeats itself in succession.

Table 6.2 Maximum Temporal Window Lengths

Dataset	Maximum Repetitions
Electricity	82
Covertype	563
KDD 99' Cup	193127
Poker Hand	152

Where concept drift methods can now be evaluated using many of the published statistical data generators, the same does not hold true for the evaluation of methods that aim to address temporal dependence within concept drift data. The existing, published methods for data simulation simply do not offer a means for including temporal dependence within the generated instances. Additionally, Zliobaite (2010) suggests that a lack of real data is slowing broader research into the domain concept drift. Taking temporal dependence and the over-resetting problem as a single example of a subdomain within concept drift, synthesising the data required is an ideal solution in contexts where real data is absent.

6.1.1 Data Simulation Methods

Various data simulation methods have been developed over the years to provide alternative measures for evaluating concept drift detection algorithms in the face of a lack of established datasets. MOA, for example, offers a multitude of algorithms for data simulation. These include the STAGGER, Agrawal and SEA methods explained in this section and used in the experimentation contained within this chapter. However, MOA also exposes additional method such as LED, Waveform and RandomRBF (Bifet, Holmes, Pfahringer, Kranen, Kremer, Jansen & Seidl 2010). In this section the STAGGER, Agrawal and SEA methods for data simulation are fully explained. These are popular synthesisers used in literature for the evaluation of proposed methods, and as such these are used in the experimentation process of the TDI-CDS method outlined in this chapter.

According to Narasimhamurthy & Kuncheva (2007) the most popular of data simulation technique is the STAGGER (Schlimmer & Granger 1986) method. STAGGER concepts are boolean functions which contain three distinctive features; shape, size and colour. Three functions are offered by STAGGER for data generation, and STAGGER also supports class imbalance through an optional boolean parameter that force balances the number of generated instances for each concept. The support for class imbalance is common across various data simulation methods. STAGGER generation has been used in various published research in the domain of stream mining (Bifet et al. 2009, Srimani

& Patil 2016, Zou et al. 2021) and remains a popular method for data simulation.

Another popular method for data generation that continues to be utilised in recent literature (Pietruczuk et al. 2017) is the Streaming Ensemble Algorithm (SEA) concepts generator (Street & Kim 2001). SEA generates instances consisting of three attributes in the numerical range 0-10, however only two of these attributes are actually relevant to the classification process. Four functions for classification are available, where each function compares the two attributes that are relevant for classification against some threshold value. The threshold value varies across functions. Class labels assigned are binary and are based on the result of the threshold evaluation. Table 6.3 shows the thresholds across the four classification functions. SEA also supports class imbalance.

Table 6.3 SEA Function Classification

Function	Threshold	Label
1	≤ 8	0 if true, else 1
2	≤ 9	0 if true, else 1
3	≤ 7	0 if true, else 1
4	≤ 9.5	0 if true, else 1

The Agrawal generator (Agrawal et al. 1993) is another popular data generation method that, despite its age, continues to be used in recent literature for evaluating concept drift detectors (Barros & Santos 2018, Yan 2020). The method produces a stream of instances that each contain nine features. Of these nine features, six are numeric and three are categorical. Each instance represents a "loan application" and the class label represents a binary decision of said loan approval. Table 6.4 shows the features, description and value range for each generated instance.

Table 6.4 Agrawal Features

Feature Name	Description	Value
Salary	Salary	20k to 150k
Commission	Commission	If salary $\leq 75k$ then 0, 10k to 75k
Age	Age	20 to 80
eLevel	Education level	0 to 4
Car	Car manufacturer	1 to 20
Zipcode	Zipcode	0 to 8
hValue	House value	50k X Zipcode to 100k X Zipcode
hYears	Years house owned	1 to 30
Loan	Loan amount requested	0 to 500k

6.2 Method

TDI-CDS is designed to facilitate the extension of existing concept drift data simulation algorithms such that they can also produce temporally dependent data. TDI-CDS

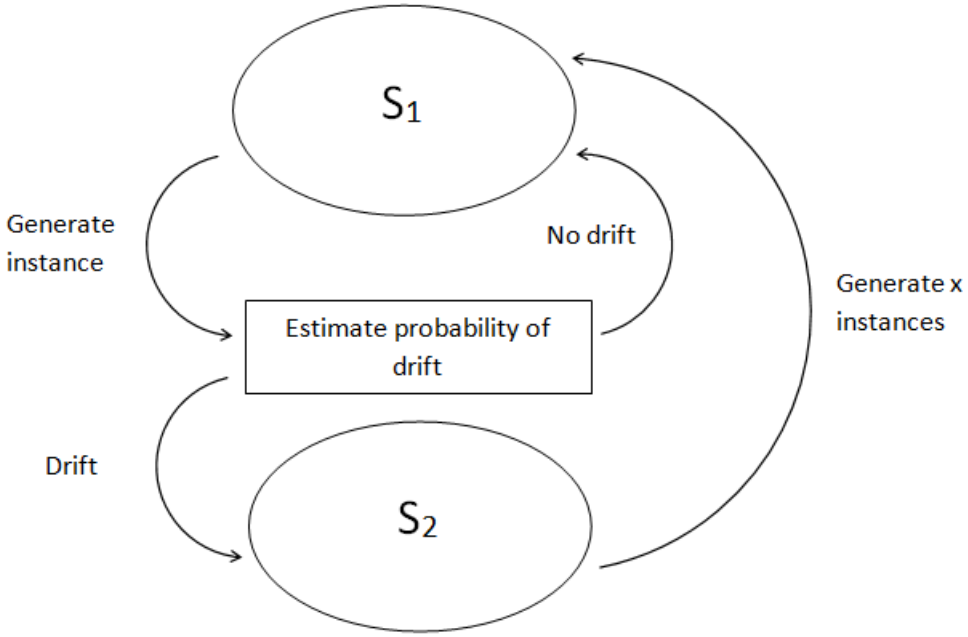

Figure 6.1: MOA Concept Drift Simulation Framework

allows any existing concept drift simulation methods to be used in conjunction with each other to produce synthesised concept drift data that may also contain temporal dependence. The temporal dependence within the data is controlled through the exposure and setting of additional user parameters. A sliding-window based approach is adopted for predicting the occurrence and length of periods of temporal dependence at any given point during the current distribution of the stream.

Experimental work in this thesis used MOA as a utility for stream mining tasks. The initial steps of designing and implementing this algorithm involved understanding and analysing the existing framework used by MOA for simulating concept drift. The native method used by MOA allows for two separate data streams to be defined; the initial distribution and the post drift distribution. The initial, or starting, distribution is the data stream that will commence the production of instances, whilst the post drift distribution is the stream responsible for generating instances after a drift has occurred. The post drift distribution must either utilise a different generator from the initial distribution entirely, or alternatively if the generator is shared between both distributions then the generator must be capable of producing data using two different functions. Various published methods for simulating concept drift data are provided by MOA for generating stream data. The overall architecture of simulating concept drift data in MOA is given in Figure 6.1.

In terms of simulating concept drift MOA achieves this through the transition between one distribution to another. Figure 6.1 portrays this through the transition from S_1 to S_2, so long as a drift has occurred. MOA defines the probability of a drift P_d occurring as

$$\begin{aligned} &P_d = 1/(1+e^x), \text{ where} \\ &x = -4 * (n-p)/w, \text{ and} \\ &n = \text{number of observed instances in the stream} \\ &p = \text{central position of the concept drift} \\ &w = \text{width of concept drift} \end{aligned} \tag{6.1}$$

The following explanation provides the process of simulating concept drift data in MOA, derived from Figure 6.1. First S_1 is defined as a user selection of one of the existing concept drift data simulation methods available within MOA. This denotes the core distribution used for generating instances when a drift is not occurring. Secondly S_2 must be configured as a second instance of a concept drift data simulation method which is used for the generation of stream instances during a simulated concept drift event. Note that S_1 and S_2 need not be the same concept drift data simulator, although they may be if desired. It is entirely possible to have two unique simulators used. In addition to the configuration of both S_1 and S_2 , additional user parameters include the options for stating the width, or duration, of a simulated concept drift event and a random seed for the inclusion of noise. Adjusting the width of the concept drift event itself denotes how many instances will be generated during a simulated concept drift event. Changing the given seed for the random generation of noise will provide some alterations to the noise values applied to the attributes of instances generated either by S_1 or S_2.

As can be observed from the above explanation of how MOA simulates concept drift, there are no mechanisms in place for the inclusion of temporal dependence in the generated data. However, the underlying framework is particularly abstract in the sense that concept drift data simulation methods are not tightly coupled with the statistical decisions that inform the probability of an occurring drift event. It is, therefore, possible to extend this framework to include temporal dependence within the generated data allowing for any existing concept drift data simulators to deployed without inherent changes. Algorithm 9 provides a complete, in depth algorithmic explanation of the functionality of TDI-CDS.

Algorithm 9 Algorithm for TDI-CDS

P_t: The probability of a temporal event occurring - user defined

T_w: The maximum width of the temporal sliding window - user defined

T_n: Random number of temporal instances to generate

T_{event}: Boolean denoting if a temporal event is occurring

T_o: The number of observed temporal instances

I_{temp}: Instance to be generated during a temporal event

P_{cd}: Probability of a concept drift event to occur

R: A random double between 0 and 1

I: Instance to be generated

I_n: Number of instances to be generated - user defined

I_o: Number of observed instances

S_1: Initial and main selected method for simulating concept drift data - user defined

S_2: The selected method for simulating concept drift data during a concept drift event - user defined

S: The currently active simulator (S_1 or S_2)

1: **while** $I_o < I_n$ **do**
2: Increment I_o
3: Calculate P_{cd}
4: **if** $R > P_{cd}$ **then**
5: $S = S_1$
6: **else**
7: $S = S_2$
8: **end if**
9: Calculate T_n such that $0 < T_n < T_w$
10: **if** $R <= P_t$ and $T_{event} = false$ and $T_o < T_n$ **then**
11: $T_{event} = true$
12: **else**
13: $T_{event} = false$
14: **end if**
15: **if** $T_{event} = true$ **then**
16: **if** I_{temp} is null **then**
17: Set $I_{temp} = I$, the current instance generated by S
18: **end if**
19: Increment T_o
20: return instance I_{temp}
21: **else**
22: Set $T_o = 0$
23: Set $T_n = 0$
24: Set T_{temp}= null
25: return instance I
26: **end if**
27: **end while**

TDI-CDS exposes two key parameters that are responsible for the configuration and inclusion of temporal dependence in the generated concept drift data. P_t is the probability that a period of temporal dependence will occur, and this parameter is user-defined as a value between 0 and 1. Since temporal dependence data is concerned with the time of arriving instances, it follows that there should exist a parameter which denotes the periodicity of which temporal dependence should occur. To this end, P_t provides a controllable solution to the problem. In addition to the frequency of temporal events, the amount of time that they occur for must also be accounted for. To achieve this, TDI-CDS adopts a sliding window T_w of a user specified length. The length of this window denotes the maximum number of temporal dependences that will be generated, should P_t define that a temporal event is to occur. Table 6.2 provides the maximum number of sequential instances within four popular temporal dependence datasets; Forest Covertype (Blackard et al. 1998), Electricity (Harries & Wales 1999), Poker Hand (Dua & Graff 2017) and KDD '99 Cup (Dua & Graff 2017). Note that the maximum window length for KDD '99 Cup is significantly larger than its peers. This is because the KDD '99 Cup dataset represents packets on a computer network during a denial-of-service (DoS) attack, and since this involves sending a very large number of packets to over a network its corresponding size is reflective. To this end, for experimentation purposes, we ignore this outlier when configuring the length of T_w.

6.3 Experimental Setup

As mentioned in Chapter 2, and similarly to Chapter 4, experimentation for TDI-CDS was undertaken using the MOA toolkit. Through evaluating the results of this experiment, the effectiveness of TDI-CDS in introducing temporal dependence to simulated concept drift data streams can be determined and discussed. The evaluation process involved in this experiment includes the use of a Naive Bayes base classifier in coordination with four state-of-the-art drift detectors; DDM, EDDM, PH and CUSUM. An in depth explanation of these drift detection methods can be found in Chapter 2. The statistical performance of TDI-CDS is evaluated through the observation of the Kappa-Temporal and Temporal Stability Index statistics, which provide an indication of temporal dependence in the data.

The temporal dependence levels for TDI-CDS can be controlled via configuration of the P_t and T_w parameters. For this experiment T_w is set 563 to match that of the Forest Covertype dataset, and varying values for P_t are tested, such that $0 \leq P_t \leq 1.0$ in incrementing intervals of 0.25, however $P_t = 0.1$ is explicitly included to indicate the impact of the introduction of temporal dependence to the data. When P_t is equal to 0, no temporal dependence is included in the generated data; it is concept drift data

only. Note that T_w does not consider the KDD '99 Cup temporal window length for experimentation since it is such an extreme outlier in comparison to the other datasets. This experimentation adopts the SEA (Street & Kim 2001), STAGGER (Schlimmer & Granger 1986) and Agrawal (Agrawal et al. 1993) generators for concept drift data generation. The full settings for configuring this experiment can be found in Appendix 8.2.

6.4 Results and Discussion

The results and discussion for this experiment are divided into three subsections, each of which is concerned with the results of TDI-CDS in conjunction with one of the three generation methods used for testing; SEA, STAGGER and Agrawal. Each subsection that follows provides an explanation of the generation method used and a critical discussion of performance results achieved. The provided discussion involves an analysis of both the accuracy of the base classifier and the Kappa-Temporal statistic in order to evaluate how effective TDI-CDS has been at introducing temporal dependence into the data. Finally, section 6.5 presents a summary of TDI-CDS, including key trends identified from the results and suggestions for future improvements of the proposed method.

6.4.1 Agrawal Generator Results

The Agrawal generator (Agrawal et al. 1993) is an established data generation method that generates a data stream consisting of 9 features, six of which are numerical and three are categorical, that describe hypothetical loan applicants. Ten functions are provided that are capable of classifying the data in a binary fashion, where the decisions are to either approve or deny the loan. The data is generated randomly by means of a user specified seed integer. It is also possible to uniformly force balance the generated classes and to add noise to the data. The specific MOA settings for Agrawal data generation used in this research are given in Appendix. Table 6.5 provides the full experimental results for TDI-CDS using the Agrawal generation method. Additionally, Figure 6.2 provides a visual illustration of the the Kappa-Temporal statistic across all four drift detection methods and all tested values of P_t.

From the results of TDI-CDS given in Table 6.5 three clear statistical trends can be immediately identified. Firstly it can be observed that by simply introducing temporal dependence at a probability rate of 0.1 that the performance of the base classifier increases spectacularly across all four drift detectors, with an average performance increase of 32.6%. This is a significant increase in classifier performance which, as demonstrated by Bifet (2017), is due to the temporal dependence in the data. Even

Table 6.5 TDI-CDS Results: Agrawal

	0			0.1			0.25		
Detector	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	65.28	0.28	0.49	98.87	-0.31	0.48	99.37	-0.34	0.46
EDDM	65.3	0.28	0.5	98.19	-1.1	0.48	98.78	-1.57	0.46
CUSUM	65.22	0.28	0.5	97.38	-2.03	0.49	97.8	-3.65	0.45
PH	65.28	0.28	0.49	96.97	-2.85	0.47	97.44	-4.41	0.46
	0.5			0.75			1.0		
	Acc	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	99.67	-0.27	0.45	99.73	-0.38	0.42	99.76	-0.45	0.39
EDDM	98.88	-3.29	0.44	98.95	-4.32	0.42	99	-5.04	0.38
CUSUM	98.04	-6.48	0.44	98.23	-7.91	0.41	98.29	-9.36	0.36
PH	97.69	-7.82	0.44	97.93	-9.45	0.41	97.88	-11.85	0.39

when the probability that a temporal dependence window will occur is as low as 0.1, the mere inclusion of temporal data at all immediately causes the classifier accuracy to become misleading. Observing the performance accuracy values for increasing values of P_t shows immediately diminishing returns. The base classifier accuracy does trend upwards with gradual improvement as P_t reaches closer to 1.0, but the gains are very marginal; particularly in comparison to the increase when $0 \leq P_t \leq 0.1$. The escalation in classifier performance accuracy as the data becomes subjected to increasingly common temporal dependence windows shows that TDI-CDS is effectively introducing temporal dependence into the data stream.

The second statistical trend that can be observed from Table 6.5 is the instantly apparent degradation of the Kappa-Temporal statistic as P_t trends towards 1.0. For all drift detectors, the KT score without the presence of temporal dependence is positive at 0.28. However, introducing temporal dependence at a low rate of $P_t = 0.1$ causes significant and varied negative responses reflected in the KT scores for each drift detection method. CUSUM and PH are the most diversely affected with resulting KT scores of -2.03 and -2.85 respectively. In the case of CUSUM and PH, the Kappa-Temporal scores continue to diminish severely as P_t increases. EDDM is also negatively affected immediately, although not quite so momentously as CUSUM or PH. In the case of EDDM the KT value when temporal dependence is merely introduced at $P_t = 0$ is -1.1, indicative that the underlying classifier is not coping with the temporal dependence levels. Whilst this is not as severe as CUSUM and PH, it is nevertheless detrimental to the performance of the base classifier and clearly indicates that its corresponding high accuracy is due to the temporal dependence levels introduced through TDI-CDS. DDM is the least affected drift detection method using the Agrawal generator and TDI-CDS. However whilst the Kappa-Temporal values for DDM are compellingly lower than its counterparts, its Kappa-Temporal values do trend negatively as soon as temporal dependence is introduced into the data.

Finally by observing the TSI metric it can also be clearly derived that TDI-CDS is indeed injecting temporal dependencies into the data. With $P_t = 0$, that is no temporal dependence included, the TSI metric indicates a good level of stability in the base classifier with TSI values averaging 0.5. However as P_t increases, the TSI value for all methods begins to decrease steadily. Up to $P_t = 0.25$ TSI values decrease by at least 0.03 in comparison to their starting value. The highest decrease from the starting TSI value and $P_t = 0.25$ is obtained by CUSUM, which is 0.05 lower than its original value. The trend continues from $P_t = 0.5$ to $P_t = 1$, where the TSI scores are clearly observed to continue to readily decrease as P_t increases. The final results at $P_t = 1$ show a significant deterioration in TSI when compared to $P_t = 0$. DDM and Page Hinckley drops b a total of 0.1 whilst EDDM reports a higher decrease of 0.12. The largest difference of 0.14 lies with CUSUM, which has a TSI value of 0.36 at $P_t = 1$. The continued, steady decrease of TSI as P_t increases clearly portrays the success of TDI-CDS injecting temporal dependency into the Agrawal data.

In summary it is clear that using the Agrawal method as a base data generator, TDI-CDS is capable of introducing temporal dependence into the data stream at varying levels of severity. The Kappa-Temporal statistic values for all four drift detection methods uniformly deteriorates as the probability of temporal dependence introduced through TDI-CDS increases. This coupled with the observed increase in base classifier accuracy clearly portrays that temporal dependence is being included in the data.

6.4.2 SEA Generator Results

The Streaming Ensemble Algorithm (SEA) generation method (Street & Kim 2001) generates three numerical attributes but only two of which are considered relevant for classification purposes. SEA exposes four classification functions, each of which compares the sum of the two relevant attributes to a threshold value unique to that particular classification function. The result of this comparison is one of two binary labels that is then applied to the generated data. The method also allows the user to define an integer seed value for random generation, a value between 0 and 1 for the inclusion of noise in the data, and a binary option to force balance the class labels. The settings for using SEA generation with MOA utilised in this experiment are provided in Appendix. The performance results for all four drift detectors (DDM, EDDM, PH and CUSUM) are provided in Table 6.6. Figure 6.3 offers an illustration of the statistical trends of the Kappa-Temporal values for each of the drift detection methods and varying intervals of P_t.

With respect to the performance of the base classifier, the first and most obvious trend is the difference in performance accuracy between $P_t = 0$ and all subsequent

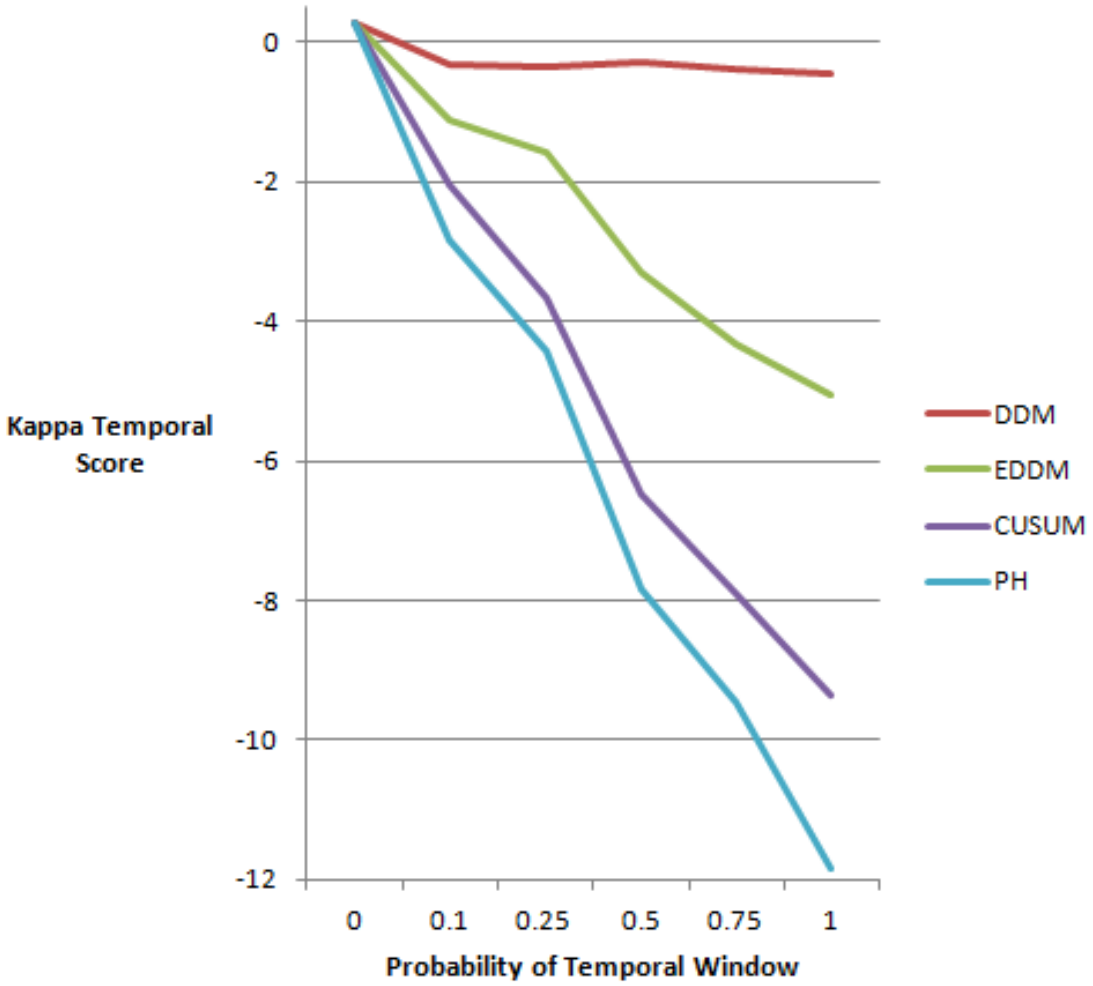

Figure 6.2: TDI-CDS Results: Agrawal

values of P_t. For all drift detection methods used in this experimentation, there is a significant increase in performance accuracy when $P_t > 0$ and temporal dependence is being included in the generated data. This increase further reinforces the over-resetting problem discussed in Chapter 4, which BD-SCR offers a potential . The existence of temporal dependence in concept drift data causes predictive accuracy to become misleading as a metric since it suggest that the base classifiers are performing at improved rates. The largest observed increase in classifier performance occurs when temporal dependence is introduced into the simulation; during the shift from $P_t = 0$ to $P_t = 0.1$. The margins of improvement in performance accuracy as P_t increases continue above 0.1 become less significant. This is a similar pattern to that observed in the results obtained from experimentation with the Agrawal generator.

Table 6.6 TDI-CDS Results: SEA

	0			0.1			0.25		
Detector	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	88.75	0.77	0.43	99.21	0.1	0.48	99.51	-0.01	0.45
EDDM	88.73	0.77	0.43	98.23	-1.01	0.47	98.79	-1.45	0.45
CUSUM	88.75	0.77	0.43	97.09	-2.31	0.46	97.86	-3.35	0.44
PH	88.75	0.77	0.43	96.79	-2.65	0.46	97.28	-4.51	0.44
	0.5			0.75			1.0		
	Acc	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	99.73	0.07	0.44	99.76	-0.12	0.43	99.78	-0.35	0.41
EDDM	98.76	-3.29	0.43	98.94	-3.83	0.41	98.98	-5.36	0.4
CUSUM	97.83	-6.52	0.43	98	-8.14	0.4	98.06	-11.08	0.4
PH	97.46	-7.8	0.42	97.61	-9.92	0.41	97.59	-13.99	0.39

Conversely, increases in the value of P_t have very significant effects on the Kappa-Temporal values. When no temporal dependence is included in the generated data, when $P_t = 0$, the Kappa-Temporal values for all drift detectors are at a respectable 0.77, indicating that temporal dependence is not affecting the performance of the classifier. These initial Kappa-Temporal values are significantly higher than those observed in the data produced by the Agrawal generator. However, even introducing temporal dependence with a low rate of 10% chance of occurrence has a severely negative effect on the Kappa-Temporal scores. It can be observed for $P_t = 0.1$, only DDM retains a positive Kappa-Temporal score at 0.1; although it is worth noting that a KT value of 0.1, whilst positive, suggests that the base classifier is only marginally outperforming the Persistent classifier. For the remaining drift detection methods used in this experimentation, EDDM, PH and CUSUM, each produce Kappa-Temporal values which drop drastically below zero. This, again, is similar to the results discussed with respect to the Agrawal generator. Further increases to the occurrence chance of a temporal event, portrayed by P_t, show continued, incremental decreases in the Kappa-Temporal scores. After configuring $P_t = 0.25$, all drift detectors are achieving a Kappa-Temporal score in the negative range, indicating that the level of temporal dependence is having adverse effects on the classifier.

TSI values follow a similar trend to those report in the Agrawal results. However, in the context of SEA there is one key difference that can be observed in the data. At $P_t = 0.1$ the TSI values for all drift detection methods increase from their starting values. That would indicate that the base classifier is more stable when a small amount of temporal dependence is injected into the data than compared to none at all. This is likely due to the small amount of temporal dependence being injected having a disproportionate affect on the accuracy of sliding windows used in the TSI metric. Where it would be expected that TSI would decrease given the inclusion of temporal dependence, the accuracy of individual windows actually decreases which leads to a higher

TSI value. This is supported by the observation that as P_t increases and the injected temporal dependencies are stronger, this anomaly disappears and the TSI values drop as expected. TSI values decrease steadily at all P_t values above 0.1, and at $P_t = 0.75$ TSI values are lower than that of their starting scores with the exception of DDM. At $P_t = 1$ all TSI values have decreased below their starting values but with varying secverity. DDM is slightly lower at 0.41 compared to 0.43, and EDDM and CUSUM both report TSI values of 0.39 compared to their starting values of 0.43. The largest decrease is shown by Page Hinckley which reports a TSI score of 0.39 at $P_t = 1$, a drop of 0.04 over its starting score. Even though TSI score initially increase, this metric still continues to highlight the success of TDI-CDS injecting temporal dependence into the generated SEA instances through the illustrated decrease in classifier stability.

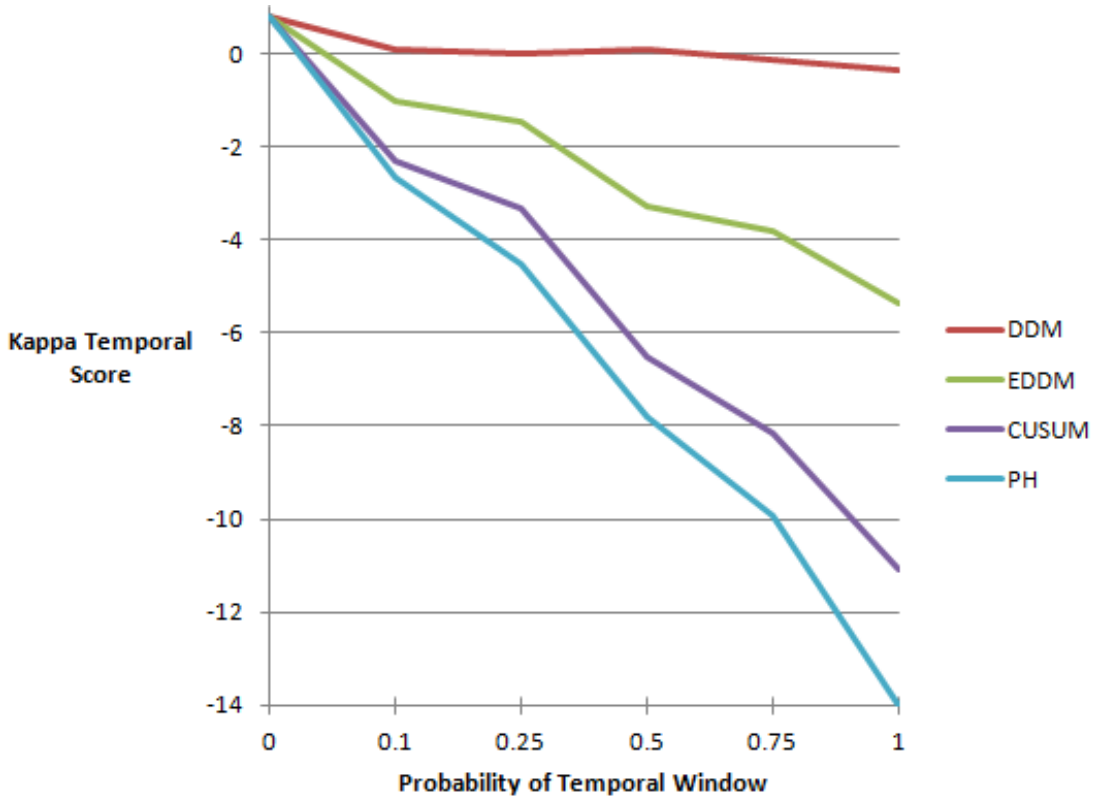

Figure 6.3: TDI-CDS Results: SEA

The results shown in Table 6.6 and Figure 6.3 convey and illustrate the continuous, negative impact that increased values of P_t have upon the Kappa-Temporal scores for each drift detection method. As can be observed from the results, TDI-CDS is effectively introducing temporal dependence into the data streams at various rates controlled through the P_t parameter. The trends observed from the results of the SEA generator follow similar statistical patterns from those observed in the results of the

Agrawal data. This acutely indicates that TDI-CDS is consistently and reliably introducing temporal dependence into data streams generated by standalone, established generation methods.

6.4.3 STAGGER Generator

The final data generation method used in the experimentation of TDI-CDS is the STAGGER (Schlimmer & Granger 1986) generator. The STAGGER generation method functions by producing a series of random data concepts consisting of three features; size, shape and colour. The "size" feature is made up of the possible values small, medium or large, "shape" is constructed of circle, square or triangle and "colour" may be either red, blue or green. Three possible functions exist for classification; function one labels concepts that are "small" and "red", function two applies to concepts that are "green" or "circle" and function three applies to concepts that are "medium" or "large". In similar fashion to both Agrawal and SEA generators, STAGGER allows the options for classes to be balanced and for random seeds to be specified by the user. The specific settings used in this experiment for the STAGGER generator are given in Appendix. Table 6.7 contains the performance results for all four concept drift detection methods using in conjunction with the STAGGER generation method, for increasing values of P_t. In addition, Figure 6.4 provides an illustration of these results that represent the statistical trends exposed by this research.

Table 6.7 TDI-CDS Results: STAGGER

	0			0.1			0.25		
Detector	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	99.99	1	0.5	99.94	0.93	0.48	99.94	0.87	0.46
EDDM	99.99	1	0.5	99.99	0.99	0.48	99.95	0.89	0.46
CUSUM	99.99	1	0.5	99.99	0.99	0.48	99.92	0.81	0.46
PH	99.99	1	0.5	99.99	0.99	0.48	99.93	0.85	0.46
	0.5			0.75			1.0		
	Acc	*KT*	*TSI*	*Acc*	*KT*	*TSI*	*Acc*	*KT*	*TSI*
DDM	99.97	0.83	0.45	99.96	0.78	0.41	99.99	0.77	0.37
EDDM	99.98	0.84	0.45	99.97	0.85	0.42	99.97	0.81	0.39
CUSUM	99.98	0.81	0.44	99.97	0.81	0.42	99.97	0.79	0.39
PH	99.98	0.85	0.45	99.97	0.82	0.42	99.97	0.81	0.38

Regarding base classifier performance first, for the STAGGER generator it can be observed that the initial classifier accuracy is considerably higher in comparison to both Agrawal and SEA generators. The inclusion of temporal dependence has no significant impact on the classifier accuracy, simply because initial classifier accuracy is already exceptionally high. There are some variations in accuracy across the various values of P_t, but classifier accuracy never drops below 99.9% for any drift detection method. In

this case it is of paramount importance to carefully consider the changes of the Kappa-Temporal statistic in order to determine if TDI-CDS is effectively introducing temporal dependence into the generated STAGGER concepts. In contrast to the previous generators, STAGGER does not suffer from an immediate degradation in Kappa-Temporal values once temporal dependence has been introduced. Rather, only DDM has any real observable impact on the Kappa-Temporal value when $P_t = 0.1$, and it is still minor. Instead, the real difference in Kappa-Temporal values occurs when $P_t = 0.25$. In this instance an average decrease of 12% occurs, with DDM the least affected with a KT decrease of 0.06, whilst CUSUM suffers worst with a decrease of 0.18. The Kappa-Temporal values for all four drift detectors when using STAGGER continues an expected negative trend as temporal dependence is injected into the data. This is reflected in Figure 6.4.

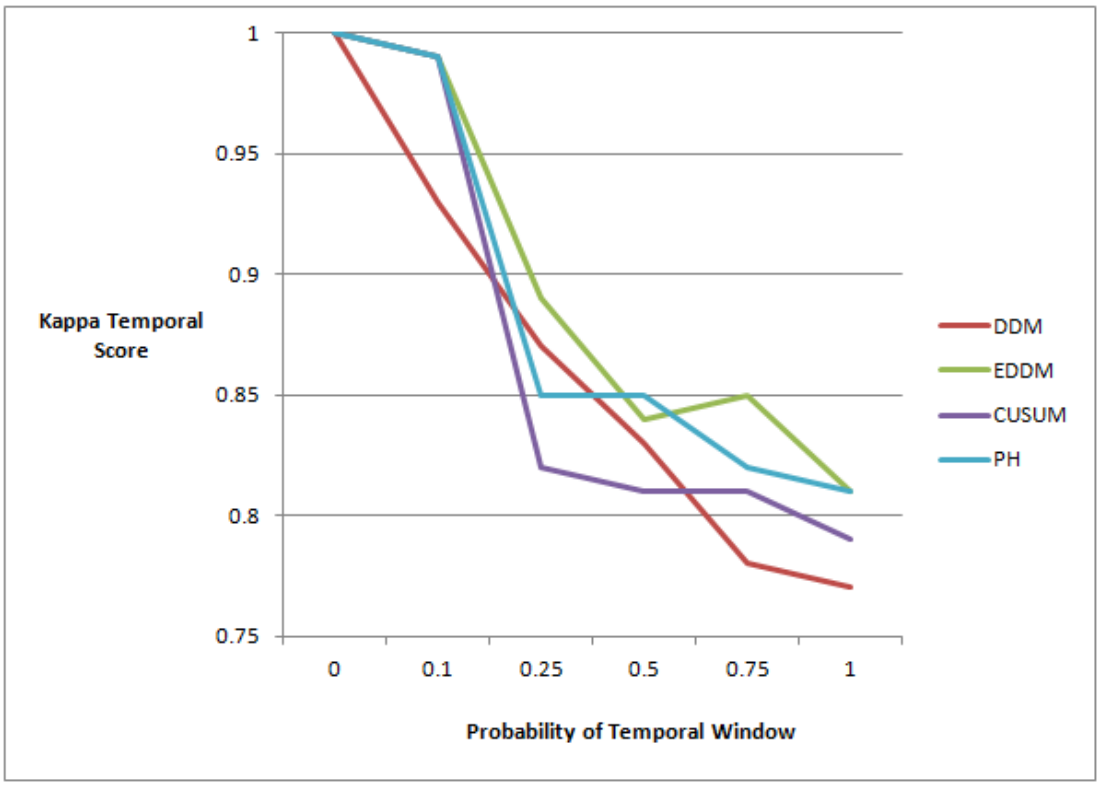

Figure 6.4: TDI-CDS Results: STAGGER

With respect to TSI values, the STAGGER results are more closely representative to that of Agrawal. Where SEA displayed an increase in TSI at very low P_t values, this is not reciprocal with STAGGER. TSI values decrease steadily and consistently over increase values for P_t. Until $P_t = 1$ there are virtually no discernible differences in the TSI values for all drift detection methods. This is similar to the discussion around classifier accuracy above. However, at $P_t = 1$ there is some noticeable difference in the

TSI values. EDDM and CUSUM perform best with TSI values of 0.39, a drop of 0.11 from their initial scores. Page Hinkcley is next best performing with a TSI score of 0.38, a decrease of 0.12. Finally DDM is worst performing with a TSI value of 0.37 representing a total TSI degradation of 0.13. In a similar pattern to both Agrawal and SEA, the TSI metric with STAGGER instances again demonstrates how TDI-CDS is capable of introducing temporal dependencies into the data. Drift detection methods with STAGGER start off with strong TSI values indicating a stable classifier. As P_t increases and the injected temporal dependencies increase, the TSI values fall almost linearly.

In conclusion, TDI-CDS is shown to be as effective at introducing temporal dependence to data streams where the concepts are generated using the STAGGER method. However, the immediate impact of temporal dependence upon STAGGER generated concepts is considerably lessened in comparison to that of Agrawal or SEA generated instances. Instead, STAGGER requires a higher severity of temporal dependence to be present within the data before the effects start to show. Even with extremely high probability rates for temporal windows to occur, STAGGER maintains positive Kappa-Temporal values across all four drift detection methods for all values of P_t using a window size of 563. Nevertheless, the deteriorating in Kappa-Temporal values caused by TDI-CDS injecting temporal dependence into the data is positive for this experimentation is it continues to demonstrate that TDI-CDS is capable of adding temporal dependence into data streams that are generated using established, statistical data generators.

6.5 Summary

This thesis chapter proposes the TDI-CDS algorithm for introducing temporal dependence into simulated concept drift data. The aim of this algorithm was to provide an original method for introducing temporal dependence into concept drift data such that existing concept drift data simulation methods, such as the Agrawal, SEA and STAGGER methods used in this experimentation, can be used for simulating data streams for the evaluation of new methods and algorithms which aim to handle and process temporally dependent data. TDI-CDS provides a flexible algorithmic solution that provides configurable user parameters that control the occurrence rate and length of temporal events within the generated concept drift data.

Experimental results using a temporal window length of 563 showed that introducing temporal dependence even at a low rate of occurrence has severe negative effects upon the Kappa-Temporal scores. This is only exasperated as the rate of occurrence is increased. Some datasets such as STAGGER proved more stubborn and were less effected

with the immediate inclusion of temporal dependence, but increasing the probability of the occurrence of a temporal window still caused a significant decrease in the Kappa-Temporal value which worsened as the probability increased. Increasing the window size would allow for a more exasperated impact on Kappa-Temporal values.

Whilst these results focus heavily on negative trends, these statistics are positive for TDI-CDS as they clearly portray how temporal dependence is being successfully included in the generated concept drift data. TDI-CDS offers a unique and novel approach to aid the problem surrounding a lack of established datasets for the evaluation of new temporal dependence coping mechanisms. Without the need to develop new generation methods, TDI-CDS offers a flexible algorithm that is interoperable with existing concept drift data generators, but offers the extensibility for the inclusion of temporal dependence in the generated data.

Chapter 7

Project Evaluation

This chapter focuses on the evaluation of the project by reflecting upon its aims and objectives, hypothesis and research question. Critically evaluating each of these aspects provides a thorough understanding of the project's overall success.

7.1 Aims and Objectives

In order to provide an answer to the research questions of this project, a series of aims and objectives were defined for the scope of this body of research. These aims are reiterated below in sequential order, where their corresponding objectives are critically discussed to determine if they have been satisfactorily achieved.

1. Design, implement and evaluate a novel approach to handling temporal dependence in concept drift data streams

   (a) Through an in-depth critical analysis of existing literature, review both established and recently developed methods of concept drift and temporal dependence.

   (b) By addressing the findings and shortcomings of existing literature identified in the objective above, design an original solution that contributes toward handling temporal dependence in data streams.

   (c) The produced solution should be statistically evaluated using established datasets and metrics as used in existing published research. This will also inform a critical discussion surrounding the performance of the developed solution; focusing on its strengths, weaknesses and potential future improvements

Aim one of this research is concerned with the design and construction of an original method for coping with temporal dependence in concept drift data streams. The purpose of this aim is to produce a novel algorithm that offers a statistical approach to combating the negative impact of temporal dependence during the evaluation of concept drift detection. To achieve this aim, three core objectives were established.

The first objective for this aim involves providing a critical analysis of existing, published literature surrounding concept drift and temporal dependence. This provides justification and motivation for the aim by exposing gaps in the literature. The literature review conducted and contained within 2 offers a thorough, in-depth discussion of various domains with stream mining; particularly focusing on temporal dependence and concept drift detection. The background and motivation surrounding the proposal of the BD-SCR method in Chapter 4 also provides specific discussion surrounding the current state-of-the-art methods for temporal dependence, such as the Temporally Augmented classifier, Temporal Correction classifier and the Kappa-Temporal statistics proposed by Zliobaite et al. (2015). To this end, the first objective of aim one is adequately satisfied. Considerable analysis and discussion of existing methods is provided, which informs and drives the design and implementation of the BD-SCR method proposed in this thesis.

The second objective of aim one involves the specific design and implementation of an original method for coping with temporal dependence within concept drift streams. The motivation behind this first aim is derived from existing literature which highlights the negative effect of temporal dependence upon the evaluation of concept drift detectors (Bifet 2017, Wares et al. 2019). The BD-SCR method proposed within this thesis is the result of work undertaken to achieve this objective. The over-resetting problem has been shown in existing literature to be severely problematic when temporal dependence is present in data streams. The BD-SCR method offers a novel solution to this problem by refraining from automatically resetting base classifiers each time a drift detector signals an occurring drift. Instead, BD-SCR statistically monitor the levels of temporal dependence within the stream in order to make informed decisions about resetting the base classifier. The sensitivity of the algorithm is controlled by user defined parameter.

The KTUE method suggested in chapter 5 also contributes towards this second objective of aim one. KTUE is an original ensemble based method for performing stream classification in evolving, temporally dependent environments. At the time of writing, no ensemble based methods for coping with or handling temporally exist. KTUE provides a unique algorithm that adopts the Kappa-Temporal statisitic as a vote weighting mechanism for replacing ensemble components. Additionally, KTUE restricts voting to

classifiers that are performing at least equal to or above the average Kappa-Temporal score across the entire ensemble. This is in contrast to popular existing methods like KUE where components must have positive values in order to vote. In temporal environments it is often the case that Kappa-Temporal scores will be exceptionally negative, therefore an alternative restriction was necessary. In combination with the BD-SCR method mentioned, this achieves the completion of the second objective of aim one for this research project.

The third and final objective of aim one is the evaluation of both the BD-SCR and KTUE methods. This objective requires the evaluation to utilise datasets and metrics used in published research, and for an analysis of the performance of the method to also be provided. The experimental setup, results, discussion and evaluation of the BD-SCR method are all provided in this thesis. The datasets that are used for experimentation are the Electricity (Harries & Wales 1999) and Forest Covertype (Blackard et al. 1998) datasets, both of which have been used extensively in the published literature. Additionally, the statistical metrics used for performance evaluation, such as the Kappa-Temporal statistic are also published (Zliobaite et al. 2015). The Kappa-Temporal statistic provides an indication of the severity of temporal dependence, whilst classification accuracy is a well known, fundamental performance metric for evaluating any classification model. Finally, an in-depth discussion of the BD-SCR performance based on the aforementioned metrics and datasets is provided alongside the experimental results. The results showed that BD-SCR was effective at reducing the impact of temporal dependence upon classification evaluation. Additionally the results have portrayed that classifier performance is not significantly impacted by the refusal to reset base classifiers during temporal peaks. As such, this adequately completes the third and final objective for the first aim of this research.

Similarly the full experimental design, construction and process for KTUE is provided in chapter 5. The datasets used for evaluation are known, established datasets used in the existing literature that contain temporal dependence and are known to be subject to some form of concept drift. Additionally for benchmarking purposes, KTUE is evaluated against evolving datasets that are not temporally dependent. Results showed that KTUE is effective in temporal environments and outperformed some existing state-of-the-art methods. In particular it was shown that KTUE was most effective in datasets with more severe levels of temporal dependence. However, KTUE was repeatedly surpassed by some methods, somewhat surprisingly. This thesis suggests that the reason for this is class imbalance in the data, and the inability for the Kappa-Temporal statistic to cope effectively with class imbalance. To further this point, parallels between

temporal dependence and class imbalance were illustrated. While temporally dependent datasets may contain class imbalance in research contexts, in reality a data stream may be capable of self-balancing over time. This is a drawback to streaming static, offline datasets to simulate data streams for research purposes.

1. Design, implement and evaluate algorithmic data simulator capable of synthesizing temporally dependent concept drift data

   (a) Through an in-depth critical analysis of existing literature, review the existing popular datasets and simulators used in published literature.

   (b) With reference to the identified relevant gaps in literature design and implement a novel data simulator that is capable of simulating temporally dependent concept drift data streams.

   (c) The produced solution should be statistically evaluated using published metrics for monitoring levels of temporal dependence. This should also inform a critical discussion surrounding the performance of the developed solution; focusing on its strengths, weaknesses and potential future improvements.

The second aim of this research, shown above, focuses on the development of a data simulation method. This method is unique and acutely different from existing concept drift data generators since it allows temporal dependence to be injected into the generated instance at varying levels of severity. The purpose of this aim is to provide a data simulator that can be used to artificially produce instances for the evaluation of new techniques to cope with temporal dependence.

The initial objective of this aim is, similar to aim one, to provide a review of the existing literature in order to motivate the problem and inform the design of a novel simulation method. The literature review contained within this thesis provides critical insight into the problem of a lack of datasets for evaluation of concept drift methods, and continues to show how this problem is exasperated when it comes to the inclusion of temporal dependence. This problem is further discussed and highlighted clearly during the proposition of TDI-CDS in Chapter 6. In proposing TDI-CDS as a simulation method, existing data generation methods are reviewed and explained, including the STAGGER (Schlimmer & Granger 1986), Agrawal(Agrawal et al. 1993) and SEA(Street & Kim 2001) methods. As such, it can be concluded that the first objective of aim two has been achieved.

The secondary objective for aim two is concerned with the production of a novel data simulation method. This thesis presents TDI-CDS as an original data simulation method capable of introducing temporal dependence to simulated data streams.

TDI-CDS achieves this by injecting temporal dependence into the generated instances through the exposure of two parameters. The first controls the probability of which a temporal event will occur and is set as a value in the range of 0-1. The second controls the size of a sliding window which represents the width of a temporal event. Should a temporal event be probabilistically triggered, generated instances within the width of the sliding window will contain the same class label. Furthermore, TDI-CDS does not need to statistically generate its own instances. Instead, existing data synthesisers can be "plugged in" and used for instance generation The criteria for this objective surrounding the design and development of a data simulation method for temporal dependence have been accomplished.

Finally, the last objective for the second aim of this research articulates that the proposed data simulation method, TDI-CDS, is evaluated by observing the temporal dependence levels, and also discussed in terms of its performance. Experimentation results of TDI-CDS provide both the Kappa-Temporal statistic and classifier predictive accuracy across all tested synthesisers and parameter values. Including the Kappa-Temporal statistic in the experimental results enables evaluative observations surrounding the temporal dependence levels to be made. Similarly, classifier predictive accuracy offers a clear indication of performance. A critical discussion of its performance based on the comparison of the results for each method is contained within Chapter 6. Additionally the limitations of TDI-CDS and suggestions for future improvements are discussed above.

7.2 Research Questions and Hypothesis

Evaluating this research in terms of its aims and objectives is of vital importance, but it is also crucial to reflect upon the research questions to determine if they have been answered. The research questions central to this thesis are restated below.

1. How can temporal dependence be accounted for during the classification of concept drift data streams?

2. Can temporally dependent concept drift data streams be simulated in order to improve the means for evaluation?

The first research question is concerned with investigating how temporal dependence can be handled during the classification process of concept drift streams. The motivation and justification behind this research question has been throughly described throughout this thesis within Chapters 2 and 4. To summarise, when concept drift data streams are also subject to temporal dependence classifiers appear to perform with high classification accuracy. However, this is in fact misleading and is caused

by over-resetting the base classifier during drift detection. To address this research question, this thesis first provides a critical, in-depth literature review which provides extensive background within the domain of concept drift and temporal dependence. Following, the method of BD-SCR is presented as a novel algorithm which controls the resetting of base classifiers in conditions where temporal dependence is high or rising. The results of the BD-SCR method indicate that by monitoring the temporal dependence levels, the over-resetting problem can be significantly reduced. This provides a clear answer to the first research question. Temporal dependence can be accounted for during the classification of concept drift data streams by statistically monitoring the Kappa-Temporal statistic at varying intervals during the stream's lifecycle. This research suggests refraining from resetting base classifiers each time a drift is detected without statistically considering the current levels of temporal dependence in the stream.

The second research question of this thesis aims to address a core issue in the domain of data streaming; a lack of established datasets for evaluating developed methods. As mentioned in the literature, for concept drift detection there are few established datasets that are used for evaluating new methods. This problem is further worsened when temporal dependence is also taken into account. Data simulation methods which produce concept data streams have been historically used for the evaluation of concept drift detection methods as a way of overcoming the lack of datasets. However, no existing data simulation methods offer the ability to include temporal dependence in the generated data. The presented TDI-CDS method of Chapter 6 presents a framework solution where temporal dependence can be injected into data streams generated by existing data simulation methods. By observing temporal dependence levels in the data generated by popular simulation methods, TDI-CDS demonstrates that it is capable of including temporal dependence in these streams without changing the statistical structure of the simulators themselves. Thus, this provides a direct answer to the second and final research question.

Answering both research questions allows the hypothesis to be directly addressed. The hypothesis of this research was that the development of an original algorithm that challenges the existing architectural relationship between concept drift detection and base classifier will provide positive, novel improvement for performing classification on temporally dependent concept drift data streams. It is also hypothesises that statistical adaptations can be made to existing data synthesisers that would facilitate the simulation of temporally dependent concept drift streams for evaluation purposes. The results of BD-SCR have shown positive improvement for the classification of temporally dependent concept drift streams. This has been portrayed in the results through the

degradation of the No-Change detector. Thus, the first part of this project's hypothesis has demonstrated to be correct. The results shown by TDI-CDS clearly illustrate that existing data simulation methods can be statistically adapted to inject temporal dependence into the produced data. This allows the facilitation of existing concept drift simulation methods to be adapted for use with the existence of temporal dependence. As such, the second and final portion of this project's hypothesis is also deemed true.

7.3 Summary

The evaluation of this research project as a whole is provided by demonstrating that the aims and objectives are satisfied, the research questions have been answered and that the project hypothesis has been reviewed. By reflecting individually on the aims and objectives of this project it has been clearly outlined that the project objectives have been met. This has been achieved through the an development, experimentation and evaluation of the BD-SCR, KTUE and TDI-CDS methods proposed. Through the satisfaction of these aims, the research questions defined in this research have also been answered in a positive context. Finally, the project's hypothesis has been revisited to determine if the research and the results contained herein reflect the initial expectations stated in the opening of this thesis. Both components of the hypothesis have been shown to have held true. It can therefore be fairly concluded that this project has been successful in achieving its research goals.

Chapter 8

Conclusions

This final chapter summarises the findings and original contributions of the work discussed throughout the preceding chapters of this thesis. This chapter also provides an analysis of the limitations of the original work developed in this thesis, as well as several suggestions for improvements that could be made by further research in this domain in the future.

The key findings portrayed with this thesis are as follows:

- A lack of established datasets makes the evaluation of new techniques for handling concept drift and temporal dependence difficult. While there are solutions available in the form of data simulators capable of synthesising concept drift data, these are currently not available for similar data that also suffers from temporal dependence.

- Current established methods for concept drift detection are do not account for the presence of temporal dependence. The No Change detector of Bifet (2017) indicates how in the presence of temporal dependence, the predictive accuracy of base classifiers, in conjunction with state-of-the-art drift detectors, becomes misleading. Current methods for concept drift detection will reset base classifiers whenever a drift is detected. This is hugely problematic when temporal dependence is present since arriving class labels are no longer independent of their time of arrival.

- Existing solutions for handling temporal dependence are very few (Zliobaite et al. 2015) and focus solely on classifier augmentations that allow base classifiers to look back on previously seen labels. Current methods do not enable drift detectors to account for temporal dependence. Since the over-resetting problem occurs

when drift detectors are subject to temporal dependence within the data, it should be monitored and handled at the drift detection level.

This research has provided a critical, in-depth analysis (Chapter 2) of concept drift detection techniques and the emerging problem of temporal dependence. Advancements in the development of algorithms that handle concept drift are being made constantly and consistently in this research domain. However, recent discoveries of the impact of temporal dependence remain largely unchallenged and instead only lead to the introduction of new problems and further complications. Concept drift detectors have long made the assumption that base classifiers should be reset when a drift is detected, however, in the presence of temporal dependence can lead to misleading performance accuracy during evaluation. Whilst new concept drift techniques are being developed, they do not challenge the archaic architecture that is the root of the over-resetting problem.

In addition to misleading predictive accuracy, temporal dependence causes further issues in the evaluation of concept drift detection methods; a lack of established datasets. This is a prevalent issue in the domain of concept drift detection already, but is only further worsened when it comes to evaluating methods for handling both concept drift and temporally dependent data. For evaluating concept drift data there has long been an acknowledgement of a lack of established datasets that can be used for evaluating new proposed methods. To aid this problem and to offer a method of circumvention, various methods for simulating, or synthesising, data have been published. Datasets that contain both concept drift and temporal dependence are even more scarce, and currently there exist no simulation methods for artificially producing data for evaluation. This makes evaluating new methods for coping with concept drift and temporal dependence hugely problematic.

This research presents a novel method, BD-SCR, for monitor temporal dependence in concept drift data streams (Chapter 4). Burst Detection-based Selective Classifier Resetting is an original, architectural method which makes statistically informed decisions on the resetting of a base classifier. A single user parameter defines the upper bound movement of temporal dependence to be tolerated. Using a sliding window approach and the Kappa-Temporal statistic (Zliobaite et al. 2015), the movement value is determined by a comparison of the temporal dependence levels in the current window and the whole stream up to the current time. The base classifier is allowed to reset as long as the movement is within the upper bound set by the user defined parameter. The results of this indicated that restricting classifier resetting during temporal peaks significantly reduced the impact of temporal dependence upon the evaluated classifier performance whilst retaining performance similar to that published in existing literature.

An original ensemble-based algorithm is presented in chapter 5. KTUE uses the Kappa-Temporal statistic as a vote weighting mechanism for replacing components in an ensemble. Only classifiers that are performing at least as well as the ensemble average are allowed to vote. Results showed that in temporal environments KTUE outperformed some state-of-the-art methods. However it failed to outperform other methods, including the KUE method upon which KTUE was designed. The reason for this is the inability of the Kappa-Temporal statistic to cope with class imbalance in the data, whereas the Kappa statistic used in KUE is much more effective in such circumstances.

To aid in the process of developing and evaluating new methods for handling temporal dependence and concept drift, this research also proposes a novel method, TDI-CDS, for simulating temporally dependence concept drift data. By adapting the existing SEA (Street & Kim 2001) simulator, a popular simulator for concept drift data simulation, new user defined parameters were incorporated to facilitate the control of temporal dependence in the simulated data. By comparing and examining the Kappa-Temporal values of SEA with various settings, results indicate that existing simulators can be adapted to include the simulation of temporal data without the need for the development of completely new simulators.

8.1 Limitations and Future Work

Suggestions for future work are given are provided below within three distinct contexts. First suggestions for improvements surrounding the proposed BD-SCR method are justified and described. Secondly, propositions concerning enhancements to the TDI-CDS method proposed are given. The original work in this research has shown that temporal dependence can be accounted for during drift detection, and that the lack of datasets for evaluation can be remedied by adapting existing concept drift simulation algorithms to facilitate the generation of temporal data. However, both of the original methods proposed in this paper have drawbacks and areas for improvements.

8.1.1 Improving BD-SCR

The motivation behind the BD-SCR was to produce a novel algorithmic solution that offered some relief to the problems caused by temporal dependence during concept drift detection. As this thesis has discussed with reference to relevant literature, temporal dependence can cause misleading accuracy during the evaluation of concept drift detection methods. As a potential solution to this, BD-SCR challenges the common process of resetting base classifiers for every detected drift. Instead, BD-SCR monitors the temporal dependence levels in data streams and makes informed resetting decisions.

Whilst the results of BD-SCR have shown to be positive and promising, it is worth noting that BD-SCR is not a definitive solution to this problem. Research in the area is young and BD-SCR is one of the first published methods that seeks to address this issue. A major drawback to BD-SCR is the requirement for the resetting acceptance threshold T to be user defined. Whilst this does offer an opportunity for fine tuning the optimal value within a testing environment, the same does not hold true for a real-time scenario. From a real world perspective, valuable time and data could potentially be lost attempting to discover the ideal T value. The sensitivity of T cannot be understated. It directly controls the degree to which base classifiers are allowed to reset and stay up to date with the underlying stream distribution, but consecutively controls how definitive an impact upon evaluation temporal dependence has.

The sensitivity of T is further aggravated by the temporal dependencies in the dataset itself. As demonstrated in Chapter 4, datasets such as Electricity which have less severe levels of temporal dependence, or where the temporal dependence is lagged throughout the data, the optimal T value can be empirically observed reasonably quickly through experimentation. In contrast datasets with more impactful levels of temporal dependence, in particular where the temporal dependence is consistent throughout the dataset, make it more ambiguous to infer optimal values of T. For datasets such as Covertype the optimal T value is several times higher than that of Electricity. This results in empirical observations becoming much less effective for establishing optimal T values.

In addition to this, assigning optimal T values via empirical observation is only possible after the execution of some experimental evaluation. For the average user, T values should not have to be acquired by experimental and statistical exploration of some dataset. As such, a fundamental area for improvement for BD-SCR is to adapt the algorithm to dynamically identify the best values for T. One way in which this could be achieved is through the development of a validated hyper-parameter tuning approach to dynamically assign the optimal values for T. An alternative brute force method may involve continued increase of the T value until performance evaluation gains begin to marginalise or plateau entirely.

A final avenue for future research involving BD-SCR may involve using the TSI metric for burst detection rather than the Kappa-Temporal statistic. Rather than detecting burst based on the short and long term average of the Kappa-Temporal values in the stream, the classifier stability reflected via the TSI metric could be used. If the classifier becomes suddenly less stable, as detected through the burst detection algorithm, then the selective resetting process can still apply. It would be of particular interest to compare the differences in performance between using the TSI and Kappa-Temporal

statistic as metrics for the burst detection module in BD-SCR.

8.1.2 Improving KTUE

KTUE's primary downfall is its inability to outperform the KUE method, of which its architectural platform is designed from. KUE uses the Kappa statistic which performs poorly in temporal environments. KTUE expands on KUE and uses the Kappa-Temporal statistic which was expected to result in superior performance in temporal environments, however KTUE failed to outperform KUE across all datasets. Class imbalance in the datasets used is likely the cause of KUE's success. The Kappa statistic is very effective at performing in scenarios with severe class imbalance. In real-world environments, temporally dependent data streams may not necessarily have class imbalance as the stream may self-correct over time as the temporal dependence windows pass. However when streaming offline data for research purposes to simulate a data stream, the class imbalance in the data cannot be corrected.

The first suggestion for improvement is therefore to improve the voting mechanism of the Kappa-Temporal statistic to account for class imbalance in the data. This may involve developing and evaluating an ensemble method using some newly developed metric that is capable of taking class imbalance into consideration whilst also providing a statistical measurement of temporal dependence. This may result in improved performance of KTUE.

The second suggestion for improvement is to further improve the voting restrictions for ensemble components. In its current iteration, KTUE restricts classifiers that have a Kappa-Temporal score lower than the ensemble average from voting on arriving instances. This may results in a significant number of classifiers abstaining from voting and therefore could hinder the performance of the ensemble in some circumstances. A weighted average or other statistical metric could be used to restrict the voting and improve performance.

Future research involving KTUE could also experiment with a TSI-centric ensemble. This could be used as a voting mechanism in place of the Kappa-Temporal statistic to evaluate classifier performance in temporal environments. Where the Kappa-Temporal statistic is useful for determining the severity of temporal dependence in data, the TSI metric provides a statistical score representing the classifier's stability. Future research which contrasts and compares the results of KTUE and an ensemble based on the TSI metric could provide valuable insight over which metric is more suitable as a voting mechanism for ensemble methods in evolving temporal environments.

8.1.3 Improving TDI-CDS

The development of concept drift detection has suffered from a lack of established datasets that can be used in evaluation. As such, various simulation methods have been proposed that generate synthetic data streams with concept drift that can be used for evaluating new methods. However, research that proposes new methods for processing temporal dependence alongside concept drift has a further compounded problem. Even fewer datasets exist that contain both concept drift and temporal dependence, and no data simulation methods exist as an alternative solution.

TDI-CDS, proposed in 6, is an algorithmic framework built using MOA that allows temporal dependence to be injected into data streams generated by existing concept drift data simulators. One suggestion for improvement is to remodel the parameter for the temporal window size. This exists to define a sliding window of a fixed size specified by the user during configuration. However, understanding the correlation between window sizes and the severity of temporal dependence is difficult. In the experimentation provided in this thesis, the window size is set to the maximum repetitions found across the datasets used for evaluation. For a real world scenario, it may be advantageous and improve user interaction if a series of levels were offered in parallel with this parameter. For example, allowing a user to select "low", "medium" or "high" for the temporal severity may lower the barrier for entry.

A second suggestion for future research in the context of TDI-CDS is to provide an additional option alongside the temporal dependence chance parameter. Currently this parameter defines the chance of temporal dependence occurring as a percentage from the values 0 - 1. More control of the injection of temporal dependence could be achieved by including parameters for the exact number of temporal occurrences to take place, even allowing their exact position in the stream to be defined. An example of similar functionality exists within concept drift data simulation methods in MOA. A parameter encapsulating the central position for a simulated drift is provided to the user. This could be implemented in conjunction with the first suggestion whereby each temporal occurrence may be correlated with a severity level and stream position, allowing for maximum customisation in the simulated stream.

8.2 Conclusive Remarks

Stream mining is an extremely wide and versatile research domain which also offers the potential for massive value in quantitative and qualitative data. Deploying machine learning methods to real-time data streams is complex and the process of understanding and handling concept drift is an important factor. While concept drift has attracted

a large amount of research, in recent years progress has stagnated. This is in part due to the discovery of temporal dependence and the negative impact it has on concept drift detection methods. Introducing new methods which incorporate temporal dependence during drift detection is an important step for the development of the research domain, but a lack of established datasets or methods for producing synthetic data only stall progress further. This research has demonstrated, through the content and contributions of this thesis, that simulating temporally dependent concept drift data is possible through the adaptation and amendment of existing concept drift simulation methods. Moreover, developing new methods for concept drift detection that involve monitoring and leveraging temporal dependence levels to make informed, statistical decisions on the resetting of base classifiers effectively reduces the negative impact that temporal dependence has upon classifier performance.

The fundamental hope of this thesis is to reinvigorates researchers; to inspire them to rethink concept drift detection, conducive to the progression of the research field as a whole.

Bibliography

Aggarwal, C. C. (2007), *Data streams: models and algorithms*, Vol. 31, Springer Science & Business Media.

Aggarwal, C. C., Han, J., Wang, J. & Yu, P. S. (2003), A framework for clustering evolving data streams, *in* 'Proceedings of the 29th international conference on Very large data bases-Volume 29', VLDB Endowment, pp. 81–92.

Aggarwal, C. C., Han, J., Wang, J. & Yu, P. S. (2004), On demand classification of data streams, *in* 'Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 503–508.

Aggarwal, C. C. & Philip, S. Y. (2008), A general survey of privacy-preserving data mining models and algorithms, *in* 'Privacy-preserving data mining', Springer, pp. 11–52.

Agrawal, R., Imielinski, T. & Swami, A. (1993), 'Database mining: A performance perspective', *IEEE transactions on knowledge and data engineering* **5**(6), 914–925.

Al-Hussaeni, K., Fung, B. C. & Cheung, W. K. (2014), 'Privacy-preserving trajectory stream publishing', *Data & Knowledge Engineering* **94**, 89–109.

Babcock, B., Babu, S., Datar, M., Motwani, R. & Widom, J. (2002), Models and issues in data stream systems, *in* 'Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', ACM, pp. 1–16.

Babcock, B., Datar, M., Motwani, R. et al. (2003), Load shedding techniques for data stream systems, *in* 'Proc. Workshop on Management and Processing of Data Streams', Citeseer.

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. & Morales-Bueno, R. (2006), Early drift detection method, *in* 'Fourth international workshop on knowledge discovery from data streams', Vol. 6, pp. 77–86.

Barros, R. S., Cabral, D. R., Gonçalves Jr, P. M. & Santos, S. G. (2017), 'Rddm: Reactive drift detection method', *Expert Systems with Applications* **90**, 344–355.

Barros, R. S. M. & Santos, S. G. T. C. (2018), 'A large-scale comparison of concept drift detectors', *Information Sciences* **451**, 348–370.

Basseville, M., Nikiforov, I. V. et al. (1993), *Detection of abrupt changes: theory and application*, Vol. 104, Prentice Hall Englewood Cliffs.

Beck, N. (2001), 'Time-series–cross-section data: What have we learned in the past few years?', *Annual review of political science* **4**(1), 271–293.

Beck, N., Katz, J. N. & Tucker, R. (1998), 'Taking time seriously: Time-series-cross-section analysis with a binary dependent variable', *American Journal of Political Science* **42**(4), 1260–1288.

Bifet, A. (2017), Classifier concept drift detection and the illusion of progress, *in* 'International Conference on Artificial Intelligence and Soft Computing', Springer, pp. 715–725.

Bifet, A., de Francisci Morales, G., Read, J., Holmes, G. & Pfahringer, B. (2015), Efficient online evaluation of big data stream classifiers, *in* 'Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining', ACM, pp. 59–68.

Bifet, A. & Gavalda, R. (2007), Learning from time-changing data with adaptive windowing, *in* 'Proceedings of the 2007 SIAM international conference on data mining', SIAM, pp. 443–448.

Bifet, A., Holmes, G. & Pfahringer, B. (2010), Leveraging bagging for evolving data streams, *in* 'Joint European conference on machine learning and knowledge discovery in databases', Springer, pp. 135–150.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R. & Gavalda, R. (2009), 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.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T. & Seidl, T. (2010), Moa: Massive online analysis, a framework for stream classification and clustering, *in* 'Proceedings of the First Workshop on Applications of Pattern Analysis', PMLR, pp. 44–50.

Bifet, A. & Kirkby, R. (2009), 'Data stream mining a practical approach'.

Bifet, A., Maniu, S., Qian, J., Tian, G., He, C. & Fan, W. (2015), Streamdm: Advanced data mining in spark streaming, *in* '2015 IEEE International Conference on Data Mining Workshop (ICDMW)', IEEE, pp. 1608–1611.

Bifet, A., Read, J., Pfahringer, B., Holmes, G. & Zliobaite, I. (2013), Cd-moa: change detection framework for massive online analysis, *in* 'International Symposium on Intelligent Data Analysis', Springer, pp. 92–103.

Bifet, A., Read, J., Zliobaite, I., Pfahringer, B. & Holmes, G. (2013), Pitfalls in benchmarking data stream classification and how to avoid them, *in* 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 465–479.

Blackard, J. A., Dean, D. J. & Anderson, C. (1998), 'The forest covertype dataset'.

Bouguelia, M.-R., Nowaczyk, S. & Payberah, A. H. (2018), 'An adaptive algorithm for anomaly and novelty detection in evolving data streams', *Data mining and knowledge discovery* **32**(6), 1597–1633.

Box, G. E., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2015), *Time series analysis: forecasting and control*, John Wiley & Sons.

Breiman, L. (1984), *Classification and regression trees*, Routledge.

Brzeziński, D. & Stefanowski, J. (2011), Accuracy updated ensemble for data streams with concept drift, *in* 'International conference on hybrid artificial intelligence systems', Springer, pp. 155–163.

Brzezinski, D. & Stefanowski, J. (2014), 'Reacting to different types of concept drift: The accuracy updated ensemble algorithm', *IEEE Transactions on Neural Networks and Learning Systems* **25**(1), 81–94.

Cai, Q., Xie, Z., Zhang, M., Chen, G., Jagadish, H. & Ooi, B. C. (2018), 'Effective temporal dependence discovery in time series data', *Proceedings of the VLDB Endowment* **11**(8), 893–905.

Cano, A. & Krawczyk, B. (2020), 'Kappa updated ensemble for drifting data stream mining', *Machine Learning* **109**(1), 175–218.

Cattral, R., Oppacher, F. & Deugo, D. (2002), 'Evolutionary data mining with automatic rule generalization', *Recent Advances in Computers, Computing and Communications* **1**(1), 296–300.

Cavalcante, R. C., Minku, L. L. & Oliveira, A. L. (2016), Fedd: Feature extraction for explicit concept drift detection in time series, *in* 'Neural Networks (IJCNN), 2016 International Joint Conference on', IEEE, pp. 740–747.

Cheng, D., Bahadori, M. T. & Liu, Y. (2014), Fblg: a simple and effective approach for temporal dependence discovery from time series data, *in* 'Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 382–391.

Chernoff, H. & Lehmann, E. L. (1954), 'The use of maximum likelihood estimates in χ^2 tests for goodness of fit', *Ann. Math. Statist.* **25**(3), 579–586.
URL: *https://doi.org/10.1214/aoms/1177728726*

Chu, F. & Zaniolo, C. (2004), Fast and light boosting for adaptive mining of data streams, *in* 'Pacific-Asia Conference on Knowledge Discovery and Data Mining', Springer, pp. 282–292.

Cohen, J. (1960), 'A coefficient of agreement for nominal scales', *Educational and Psychological Measurement* **20**(1), 37–46.
URL: *https://doi.org/10.1177/001316446002000104*

Datar, M., Gionis, A., Indyk, P. & Motwani, R. (2002), 'Maintaining stream statistics over sliding windows', *SIAM journal on computing* **31**(6), 1794–1813.

De Francisci Morales, G., Bifet, A., Khan, L., Gama, J. & Fan, W. (2016), Iot big data stream mining, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 2119–2120.

de Lima Cabral, D. R. & de Barros, R. S. M. (2018), 'Concept drift detection based on fisher's exact test', *Information Sciences* **442**, 220–234.

de Mello, R. F., Vaz, Y., Grossi, C. H. & Bifet, A. (2019), 'On learning guarantees to unsupervised concept drift detection on data streams', *Expert Systems with Applications* **117**, 90–102.

De Rosa, R. & Cesa-Bianchi, N. (2015), Splitting with confidence in decision trees with application to stream mining, *in* '2015 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–8.

Ditzler, G. & Polikar, R. (2010), An ensemble based incremental learning framework for concept drift and class imbalance, *in* 'Neural Networks (IJCNN), The 2010 International Joint Conference on', IEEE, pp. 1–8.

Ditzler, G. & Polikar, R. (2013), 'Incremental learning of concept drift from streaming imbalanced data', *ieee transactions on knowledge and data engineering* **25**(10), 2283–2301.

Ditzler, G., Roveri, M., Alippi, C. & Polikar, R. (2015), 'Learning in nonstationary environments: A survey', *IEEE Computational Intelligence Magazine* **10**(4), 12–25.

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

Dua, D. & Graff, C. (2017), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml*

Duarte, J., Gama, J. & Bifet, A. (2016), 'Adaptive model rules from high-speed data streams', *ACM Transactions on Knowledge Discovery from Data (TKDD)* **10**(3), 30.

Duong, Q.-H., Ramampiaro, H. & Nørvåg, K. (2018), 'Applying temporal dependence to detect changes in streaming data', *Applied Intelligence* pp. 1–19.

Earle, P. S., Bowden, D. C. & Guy, M. (2012), 'Twitter earthquake detection: earthquake monitoring in a social world', *Annals of Geophysics* **54**(6).

Elwell, R. & Polikar, R. (2011), 'Incremental learning of concept drift in nonstationary environments', *IEEE Transactions on Neural Networks* **22**(10), 1517–1531.

Fan, J. & Lv, J. (2010), 'A selective overview of variable selection in high dimensional feature space', *Statistica Sinica* **20**(1), 101.

Faust, K., Xie, Q., Han, D., Goyle, K., Volynskaya, Z., Djuric, U. & Diamandis, P. (2018), 'Visualizing histopathologic deep learning classification and anomaly detection using nonlinear feature space dimensionality reduction', *BMC bioinformatics* **19**(1), 1–15.

Fisher, R. A. (1992), Statistical methods for research workers, *in* 'Breakthroughs in statistics', Springer, pp. 66–70.

Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A. & Caballero-Mota, Y. (2015), 'Online and non-parametric drift detection methods based on hoeffding's bounds', *IEEE Transactions on Knowledge and Data Engineering* **27**(3), 810–823.

Gaber, M. M. (2009), Data stream mining using granularity-based approach, *in* 'Foundations of Computational, IntelligenceVolume 6', Springer, pp. 47–66.

Gaber, M. M. (2012), 'Advances in data stream mining', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2**(1), 79–85.

Gama, J. (2010), *Knowledge discovery from data streams*, CRC Press.

Gama, J., Medas, P., Castillo, G. & Rodrigues, P. (2004), Learning with drift detection, *in* 'Brazilian symposium on artificial intelligence', Springer, pp. 286–295.

Gama, J., Sebastião, R. & Rodrigues, P. P. (2013), 'On evaluating stream learning algorithms', *Machine learning* **90**(3), 317–346.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. (2014), 'A survey on concept drift adaptation', *ACM computing surveys (CSUR)* **46**(4), 44.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G. & Abdessalem, T. (2017), 'Adaptive random forests for evolving data stream classification', *Machine Learning* **106**(9), 1469–1495.

Gustafsson, F. & Gustafsson, F. (2000), *Adaptive filtering and change detection*, Vol. 1, Citeseer.

Hahsler, M., Bolanos, M. & Forrest, J. (2015), 'streammoa: Interface for moa stream clustering algorithms', *R package version* pp. 1–1.

Hahsler, M., Bolanos, M. & Forrest, J. (2017), 'Introduction to stream: An extensible framework for data stream clustering research with r', *Journal of Statistical Software* **76**(1), 1–50.

Hamilton, J. D. (1994), *Time series analysis*, Vol. 2, Princeton university press Princeton, NJ.

Haque, A., Khan, L. & Baron, M. (2016), Sand: Semi-supervised adaptive novel class detection and classification over data stream., *in* 'AAAI', pp. 1652–1658.

Haque, A., Khan, L., Baron, M., Thuraisingham, B. & Aggarwal, C. (2016), Efficient handling of concept drift and concept evolution over stream data, *in* '2016 IEEE 32nd International Conference on Data Engineering (ICDE)', IEEE, pp. 481–492.

Harries, M. & Wales, N. S. (1999), 'Splice-2 comparative evaluation: Electricity pricing'.

Hulten, G., Spencer, L. & Domingos, P. (2001), Mining time-changing data streams, *in* 'Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 97–106.

Ikonomovska, E. (2008), 'Data Expo 2009: Airline on time data'.
URL: *https://doi.org/10.7910/DVN/HG7NV7*

Ikonomovska, E., Gama, J. & Džeroski, S. (2011), 'Learning model trees from evolving data streams', *Data Mining and Knowledge Discovery* **23**, 128–168.

Iwashita, A. S. & Papa, J. P. (2018), 'An overview on concept drift learning', *Ieee Access* **7**, 1532–1547.

Jaber, G., Cornuéjols, A. & Tarroux, P. (2013), 'Anticipative and dynamic adaptation to concept changes', *Real-World Challenges for Data Stream Mining* **22**.

Jaworski, M., Duda, P. & Rutkowski, L. (2017), 'New splitting criteria for decision trees in stationary data streams', *IEEE transactions on neural networks and learning systems* **29**(6), 2516–2529.

Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M. et al. (2004), Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring, *in* 'Proceedings of the 2004 SIAM International Conference on Data Mining', SIAM, pp. 300–311.

Kargupta, H., Park, B.-H., Pittie, S., Liu, L., Kushraj, D. & Sarkar, K. (2002), 'Mobimine: monitoring the stock market from a pda', *SIGKDD Explor.* **3**, 37–46.

Kelly, M. G., Hand, D. J. & Adams, N. M. (1999), The impact of changing populations on classifier performance, *in* 'Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 367–371.

Khan, L. & Fan, W. (2012), Tutorial: Data stream mining and its applications, *in* 'International Conference on Database Systems for Advanced Applications', Springer, pp. 328–329.

Kolter, J. Z. & Maloof, M. A. (2003), Dynamic weighted majority: A new ensemble method for tracking concept drift, *in* 'Data Mining, 2003. ICDM 2003. Third IEEE International Conference on', IEEE, pp. 123–130.

Kong, Q., Kwony, Y.-W., Schreierz, L., Allen, S., Allen, R. & Strauss, J. (2015), Smartphone-based networks for earthquake detection, *in* '2015 15th International Conference on Innovations for Community Services (I4CS)', IEEE, pp. 1–8.

Kotu, V. & Deshpande, B. (2014), *Predictive analytics and data mining: concepts and practice with rapidminer*, Morgan Kaufmann.

Krawczyk, B. (2016), 'Learning from imbalanced data: open challenges and future directions', *Progress in Artificial Intelligence* **5**(4), 221–232.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J. & Woźniak, M. (2017), 'Ensemble learning for data stream analysis: A survey', *Information Fusion* **37**, 132–156.

Krempl, G., Zliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M. et al. (2014), 'Open challenges for data stream mining research', *ACM SIGKDD explorations newsletter* **16**(1), 1–10.

Krishnaswamy, S., Gaber, M., Harbach, M., Hugues, C., Sinha, A., Gillick, B., Haghighi, P. & Zaslavsky, A. (2009), Open mobile miner: a toolkit for mobile data stream mining, *in* 'ACM KDD'09'.

Liao, J., Zhang, J. & Ng, W. W. (2016), Effects of different base classifiers to learn++ family algorithms for concept drifting and imbalanced pattern classification problems, *in* 'Machine Learning and Cybernetics (ICMLC), 2016 International Conference on', Vol. 1, IEEE, pp. 99–104.

Liu, G., Cheng, H.-r., Qin, Z.-g., Liu, Q. & Liu, C.-x. (2013), E-cvfdt: An improving cvfdt method for concept drift data stream, *in* 'Communications, Circuits and Systems (ICCCAS), 2013 International Conference on', Vol. 1, IEEE, pp. 315–318.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. & Zhang, G. (2019), 'Learning under concept drift: A review', *IEEE Transactions on Knowledge and Data Engineering* **31**(12), 2346–2363.

McDiarmid, C. (1989), *On the method of bounded differences*, London Mathematical Society Lecture Note Series, Cambridge University Press, p. 148–188.

Mehmood, H., Kostakos, P., Cortes, M., Anagnostopoulos, T., Pirttikangas, S. & Gilman, E. (2021), 'Concept drift adaptation techniques in distributed environment for real-world data streams', *Smart Cities* **4**(1), 349–371.

MOHAMED, M., Arkady, Z. & Shonali, K. (2010), 'Data stream mining', *M. Oded, R. Lior. Data Mining and Knowledge Discovery Handbook. New York: Springer* p. 761.

Mohammadi, M., Al-Fuqaha, A., Sorour, S. & Guizani, M. (2018), 'Deep learning for iot big data and streaming analytics: A survey', *IEEE Communications Surveys & Tutorials* **20**(4), 2923–2960.

Montiel, J., Read, J., Bifet, A. & Abdessalem, T. (2018), 'Scikit-multiflow: A multi-output streaming framework', *Journal of Machine Learning Research* **19**(72), 1–5.
URL: *http://jmlr.org/papers/v19/18-251.html*

Muhlbaier, M. D. & Polikar, R. (2007), Multiple classifiers based incremental learning algorithm for learning in nonstationary environments, *in* 'Machine Learning and Cybernetics, 2007 International Conference on', Vol. 6, IEEE, pp. 3618–3623.

Muhlbaier, M. D., Topalis, A. & Polikar, R. (2009), 'Learn++nc: Combining ensemble of classifiers with dynamically weighted consult and vote for efficient incremental learning of new classes', *IEEE transactions on neural networks* **20**(1), 152–168.

Muhlbaier, M., Topalis, A. & Polikar, R. (2004), Learn++. mt: A new approach to incremental learning, *in* 'International Workshop on Multiple Classifier Systems', Springer, pp. 52–61.

Narasimhamurthy, A. M. & Kuncheva, L. I. (2007), A framework for generating data to simulate changing environments., *in* 'Artificial intelligence and applications', pp. 415–420.

Neumeyer, L., Robbins, B., Nair, A. & Kesari, A. (2010), S4: Distributed stream computing platform, *in* 'Data Mining Workshops (ICDMW), 2010 IEEE International Conference on', IEEE, pp. 170–177.

Nishida, K. & Yamauchi, K. (2007), Detecting concept drift using statistical testing, *in* 'International conference on discovery science', Springer, pp. 264–269.

Olusola, A. A., Oladele, A. S. & Abosede, D. O. (2010), Analysis of kdd'99 intrusion detection dataset for selection of relevance features, *in* 'Proceedings of the world congress on engineering and computer science', Vol. 1, WCECS, pp. 20–22.

Oza, N. C. & Russell, S. J. (2001), Online bagging and boosting, *in* 'International Workshop on Artificial Intelligence and Statistics', PMLR, pp. 229–236.

Page, E. S. (1954), 'Continuous inspection schemes', *Biometrika* **41**(1/2), 100–115.

Parker, B. S., Khan, L. & Bifet, A. (2014), Incremental ensemble classifier addressing non-stationary fast data streams, *in* '2014 IEEE International Conference on Data Mining Workshop', IEEE, pp. 716–723.

Pelossof, R., Jones, M., Vovsha, I. & Rudin, C. (2009), Online coordinate boosting, *in* '2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops', IEEE, pp. 1354–1361.

Pesaranghader, A., Viktor, H. L. & Paquet, E. (2018), Mcdiarmid drift detection methods for evolving data streams, *in* '2018 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–9.

Pietruczuk, L., Rutkowski, L., Jaworski, M. & Duda, P. (2017), 'How to adjust an ensemble size in stream data mining?', *Information Sciences* **381**, 46–54.

Polikar, R., Upda, L., Upda, S. S. & Honavar, V. (2001), 'Learn++: An incremental learning algorithm for supervised neural networks', *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)* **31**(4), 497–508.

Quinlan, J. R. (1986), 'Induction of decision trees', *Machine learning* **1**(1), 81–106.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M. & Herrera, F. (2017), 'A survey on data preprocessing for data stream mining: Current status and future directions', *Neurocomputing* **239**, 39–57.

Reinsel, D., Gantz, J. & Rydning, J. (2017), Data age 2025: The evolution of data to life-critical don't focus on big data; focus on the data that's big, Technical report, IDC.
URL: *https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf*

Ross, Z. E. & Ben-Zion, Y. (2014), 'An earthquake detection algorithm with pseudo-probabilities of multiple indicators', *Geophysical Journal International* **197**(1), 458–463.

Rutkowski, L., Jaworski, M., Pietruczuk, L. & Duda, P. (2015), 'A new method for data stream mining based on the misclassification error', *IEEE transactions on neural networks and learning systems* **26**(5), 1048–1059.

Rutkowski, L., Pietruczuk, L., Duda, P. & Jaworski, M. (2012), 'Decision trees for mining data streams based on the mcdiarmid's bound', *IEEE Transactions on Knowledge and Data Engineering* **25**(6), 1272–1279.

Sakaki, T., Okazaki, M. & Matsuo, Y. (2010), Earthquake shakes twitter users: real-time event detection by social sensors, *in* 'Proceedings of the 19th international conference on World wide web', ACM, pp. 851–860.

Schlimmer, J. C. & Granger, R. H. (1986), 'Incremental learning from noisy data', *Machine learning* **1**(3), 317–354.

Sethi, T. S. & Kantardzic, M. (2017), 'On the reliable detection of concept drift from streaming unlabeled data', *Expert Systems with Applications* **82**, 77–99.

Srimani, P. & Patil, M. (2016), 'Mining data streams with concept drift in massive online analysis frame work', *WSEAS transactions on computers* **15**, 133–139.

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

Tatbul, N., Çetintemel, U., Zdonik, S., Cherniack, M. & Stonebraker, M. (2003), Load shedding in a data stream manager, *in* 'Proceedings of the 29th international conference on Very large data bases-Volume 29', VLDB Endowment, pp. 309–320.

Tran, D.-H., Gaber, M. M. & Sattler, K.-U. (2014), 'Change detection in streaming data in the era of big data: models and issues', *ACM SIGKDD Explorations Newsletter* **16**(1), 30–38.

van Rijn, J. N., Holmes, G., Pfahringer, B. & Vanschoren, J. (2018), 'The online performance estimation framework: heterogeneous ensemble learning for data streams', *Machine Learning* **107**(1), 149–176.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P. & SciPy 1.0 Contributors (2020), 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python', *Nature Methods* **17**, 261–272.

Wald, A. (1973), *Sequential analysis*, Courier Corporation.

Wang, H. & Abraham, Z. (2015), Concept drift detection for streaming data, *in* 'Neural Networks (IJCNN), 2015 International Joint Conference on', IEEE, pp. 1–9.

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

Wang, S., Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P. & Yao, X. (2013), Concept drift detection for online class imbalance learning, *in* 'Neural Networks (IJCNN), The 2013 International Joint Conference on', IEEE, pp. 1–10.

Wares, S., Isaacs, J. & Elyan, E. (2019), 'Data stream mining: methods and challenges for handling concept drift', *SN Applied Sciences* **1**(11), 1412.

Wares, S., Isaacs, J. & Elyan, E. (2021), 'Burst detection-based selective classifier resetting', *Journal of Information & Knowledge Management* **20**(02), 2150027.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L. & Petitjean, F. (2016), 'Characterizing concept drift', *Data Mining and Knowledge Discovery* **30**(4), 964–994.

Widmer, G. & Kubat, M. (1993), Effective learning in dynamic environments by explicit context tracking, *in* 'European Conference on Machine Learning', Springer, pp. 227–243.

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

Wijffels, J. (2014), RMOA: Connect r with moa to perform streaming classifications. R package version 1.0.
URL: *https://github.com/jwijffels/RMOA*

Yan, M. M. W. (2020), 'Accurate detecting concept drift in evolving data streams', *ICT Express* **6**(4), 332–338.

Yang, M., Rashidi, L., Rajasegarar, S., Leckie, C., Rao, A. S. & Palaniswami, M. (2018), Crowd activity change point detection in videos via graph stream mining, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops', pp. 215–223.

Yu, S. & Abraham, Z. (2017), Concept drift detection with hierarchical hypothesis testing, *in* 'Proceedings of the 2017 SIAM International Conference on Data Mining', SIAM, pp. 768–776.

Yu, S., Abraham, Z., Wang, H., Shah, M., Wei, Y. & Príncipe, J. C. (2019), 'Concept drift detection and adaptation with hierarchical hypothesis testing', *Journal of the Franklin Institute* **356**(5), 3187–3215.

Zhang, P., Zhou, C., Wang, P., Gao, B. J., Zhu, X. & Guo, L. (2015), 'E-tree: An efficient indexing structure for ensemble models on data streams', *IEEE Transactions on Knowledge and Data engineering* .

Zhu, Y. & Shasha, D. (2003), Efficient elastic burst detection in data streams, *in* 'Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 336–345.

Zliobaite, I. (2010), 'Learning under concept drift: an overview', *arXiv preprint arXiv:1010.4784* .

Zliobaite, I. (2013), 'How good is the electricity benchmark for evaluating concept drift adaptation', *arXiv preprint arXiv:1301.3524* .

Zliobaite, I., Bifet, A., Read, J., Pfahringer, B. & Holmes, G. (2015), 'Evaluation methods and decision theory for classification of streaming data with temporal dependence', *Machine Learning* **98**(3), 455–482.

Zou, J., Fu, X., Guo, L., Ju, C. & Chen, J. (2021), 'Creating ensemble classifiers with information entropy diversity measure', *Security and Communication Networks* **2021**.

Appendix 1

Configuration values and settings used in experimentation and evaluation of the BD-SCR method proposed in Chapter 4

BD-SCR Settings

- Learner: Drift Detection Method Classifier
  - Base Learner: Naive Bayes
  - Drift Detection Method: DDM/EDDM/CUSUM/PH
  - Burst Threshold: 0 - 3
  - Burst Detection: ON/ENABLED
- Stream: ARFF Filestream containing either Electricity or Forest Covertype
- Evaluator: Basic Classification Performance Evaluator
- Instance Limit: 100 000
- Time Limit: -1 (disable time limit)
- Sample Frequency: 10 000
- Check Frequency: 10 000
- Dump File: Unused - set as appropriate
- Output Prediction File: Unused - set as appropriate
- Width: 1000

Appendix 2

Configuration values and settings used in experimentation and evaluation of the TDI-CDS method proposed in Chapter 6

MOA Settings

- Learner: Drift Detection Method Classifier
  - Base Learner: Naive Bayes
  - Drift Detection Method: DDM/EDDM/CUSUM/PH
  - Burst Threshold: N/A since mode disabled
  - Burst Detection: OFF/DISABLED
- Stream: Concept Drift Stream
  - Stream: AGRAWAL/SEA/STAGGER
  - Drift Stream: AGRAWAL/SEA/STAGGER (see configurations for specific functions)
  - Alpha: 0
  - Position: 0
  - Width: 1000
  - Random Seed: 1
  - Temporal Dependence: ON/ENABLED
  - Temporal Chance: 0 - 1
  - Temporal Range: 563
- Evaluator: Basic Classification Performance Evaluator

- Instance Limit: 100 000
- Time Limit: -1 (disable time limit)
- Sample Frequency: 10 000
- Check Frequency: 10 000
- Dump File: Unused - set as appropriate
- Output Prediction File: Unused - set as appropriate
- Width: 1000

Agrawal Configuration

- Stream: AGRAWAL function 1
- Drift Stream: AGRAWAL function 10

SEA Configuration

- Stream: SEA function 1
- Drift Stream: SEA function 4

STAGGER Configuration

- Stream: STAGGER function 1
- Drift Stream: STAGGER function 3