Anomaly Detection in Univariate Time Series Data in the Presence of Concept Drift

ANOMALY DETECTION IN UNIVARIATE TIME SERIES DATA IN THE PRESENCE OF CONCEPT DRIFT

BY

SOROUSH ZAMANI ALAVIJEH, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2021) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Anomaly Detection in Univariate Time Series Data in the Presence of Concept Drift
AUTHOR:	Soroush Zamani Alavijeh B.Sc. (Computer Engineering), Sharif University of Technology, Tehran, Iran
SUPERVISOR:	Dr. Fei Chiang
NUMBER OF PAGES:	xii, 106

Lay Abstract

Digital programs and hardware record data over time for quality assurance and monitoring purposes. Anomalies are often present in such data for various reasons, including recording or storage errors. Many programs and environments are inherently dynamic and experience changes in their pattern of behavior and subsequently in the data they produce. Such fluctuations are referred to as concept drifts. In this study, we propose and implement a framework that is able to identify anomalies in data recorded over time in dynamic environments.

Abstract

Digital applications and devices record data over time to enable the users and managers to monitor their activity. Errors occur in data, including the time series data, for various reasons including software system failures and human errors. The problem of identifying errors, also referred to as anomaly detection, in time series data is a well studied topic by the data management and systems researchers. Such data are often recorded in dynamic environments where a change in the standard or the recording hardware can result in different and novel patterns arising in the data. Such novel patterns are caused by what is referred to as concept drifts. Concept drift occurs when there is a pattern change in the statistical properties of the data, e.g. the distribution of the data, over time. The problem of identifying anomalies in time series data recorded and stored in dynamic environments has not been extensively studied. In this study, we focus on this problem. We propose and implement a unified framework that is able to identify drifts in univariate time series data and incorporate information gained from the data to train a learning model that is able to detect anomalies in unseen univariate time series data.

To my mom, and my family

who supported me unconditionally

Acknowledgements

I would like to acknowledge my supervisor, Dr. Fei Chiang, for being supportive and encouraging in the path I have come. Her support, patience, and guidance were crucial factors in my accomplishments during my study. It was a great pleasure for me to work with her and I am very grateful for that. Her support of my decisions throughout the path I have come has immensely helped me to gain more confidence in myself and this is an extraordinarily amazing accomplishment. She has always been willing to listen and answer my questions and I would like to express my gratitude for that.

I am profoundly grateful to my mom, my family, and my friends, without whom I would never be able to stand where I am standing now. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Expansion of data driven applications and digital devices that constantly and continuously record data leads to a higher risk of errors being present in the data. Errors might be present in the data due to possible mistakes in reading, storing, analyzing or transferring. For instance, consider a sensor recording the temperature of a conference room every ten minutes and sends it to an embedded software to be stored on a disk locally. At the end of the day, the software also computes the average temperature and sends the data to a remote server that manages and analyzes the data. A dataset on the remote management server could have errors due to following reasons:

- The sensor is defective and records erroneous temperature readings.
- The local storage disk is defective and the numbers are prone to errors upon the saving operation.
- The embedded software has a bug in its program, and experiences an

arithmetic overflow when computing the average temperature at the end of the day.

- The network hardware is defective or the network software requires an update for its checksum algorithm, and it results in wrong data being transmitted to the remote server.

- The remote server's analysis software has a bug in its conversion from Celsius to Kelvin, required for further statistical analysis.

As portrayed in the possible scenarios above, errors are an inseparable part of computer systems that read, analyze, and write data. Time Series data is not an exception. A time series is defined as a sequence of orderly recorded data points in which every point of data has a time step associated with it, i.e. chronically ordered sequence of data. Formally, a time series T of length n is defined as a sequence of data points with a chronological order $\{T(i)|\ i \in [1, n]\}$ where i represents the time step and $T(i)$ represents the value of the time series at time step i. If the data is being recorded indefinitely and in an online fashion, then there is no limit for i to grow.

Errors in a time series data, also referred to as anomalies, are single data point or a consecutive sequence of data points that are known to be abnormalities, i.e. data points that do not have a normal value. Abnormalities could be manually identified by domain experts or automatically identified based on a set of rules.

Some studies refer to and treat anomalies the same way as outliers. In their book on outlier analysis, (Aggarwal, 2016) defines outliers and anomalies as

the same concept: "Outliers are also referred to as abnormalities, discordants, deviants, or anomalies in the data mining and statistics literature."

In order to establish a scientific method to detect such errors in time series data, we need a coherent and single definition. One of the most common definitions of anomalies is given by (Chandola *et al.*, 2009): "Anomalies are patterns in data that do not conform to a well defined notion of normal behavior". Based on this definition, the main goal of this study is to identify such abnormal behavior and in turn, specific points in time series data that are labeled as anomalies.

The first outcome of the above definition is that most of a time series data must be regarded as normal, so that a few points can be regarded as abnormal. The notion of abnormality finds its meaning when the majority of the data is normal. The second outcome is that not all changes in the patterns or the values of data points in a time series are necessarily abnormalities, i.e. a factor of ill-defined or malicious behavior must be present to consider a portion of the time series as anomalous. A change in pattern of the time series that emerges after a period of stability is different from an anomalous pattern. These new patterns, also known as novelties [(Chandola *et al.*, 2009), (Pimentel *et al.*, 2014)], are normal yet unprecedented or rare, but not abnormal. Therefore, we need to distinguish them in our study of anomalies.

Time series data can be modelled via random variables operating as underlying data generation processes, i.e. at each time step the value of the time series is drawn from the random variable. A change in the probability distribution of the underlying data generation process results in novel patterns in

the data (Webb *et al.*, 2016). Novel patterns emerge in time series data due to various reasons. A change in season for temperature data, a change in climate conditions for water level data, or a job promotion for bank account balance data are some examples of possible scenarios leading to a novel pattern in data. Such dynamic and non-stationary environments often experience a drift in their data distribution. Concept drift is a phenomenon that is attributed to a significant change in the distribution of data, in a way that models applied on old data are no longer as accurate as they were upon the emergence of such drifts.

1.1 Our Contributions

The goal of this study is to design an anomaly detector for univariate time series data that explicitly considers the effects of concept drift on anomaly detection. We first provide an extended set of formal definitions to characterize concept drift and then we provide an algorithm that detects concept drift in (univariate) time series data. The extension is done on the definitions and parameters provided by (Webb *et al.*, 2016). The definitions and parameters provided by (Webb *et al.*, 2016) were not sufficient to develop a concept drift detection algorithm. We explain what are the revisited definitions and how and why we extend them in details in the Chapter 3. We then analyze and observe the effect of the detected concept drifts in the process of anomaly detection and show its impact on the performance of the anomaly detector. We use the publicly available Yahoo! dataset (Laptev *et al.*, 2015) that is manually labeled

by domain experts, the publicly available light (Zhu, 2010) and power (Zhu, 2010) datasets with injected anomalies, and a set of synthetically generated datasets with injected anomalies using MOA tool (Bifet *et al.*, 2010). The followings are a list of contributions of this study:

1. An extended set of definitions to formalize and characterize the notion of concept drift, its various types, and its effects on time series data analysis.

2. An algorithm that effectively detects abrupt, gradual, and incremental drifts.

3. A classifier in the form of an anomaly detector that effectively uses detected concept drifts to identify anomalous data points in time series data more accurately compared to a baseline study.

It is noted that we revisit and extend some of the definitions and parameters introduced by (Webb *et al.*, 2016) to provide a more complete and integrated set of definitions according to the first contribution point listed above. We explain such definitions in details in the next chapter.

Figure 1.1 demonstrates the architecture of our proposed framework. Specifically, for each time series, we first identify concept drifts and extract a set of features based on their location, length, and type. We then extract a set of statistical features from the time series. Finally, we feed the extracted features, i.e. concept drift and statistical features, to a trained feed forward neural network to identify anomalies. It is noted that the steps in Figure 1.1 shows the flow for the anomaly detection using a trained neural network. However, there

is a training step that uses the labeled time series data with known anomalies to train the model. The training step has an identical structure as depicted in Figure 1.1.

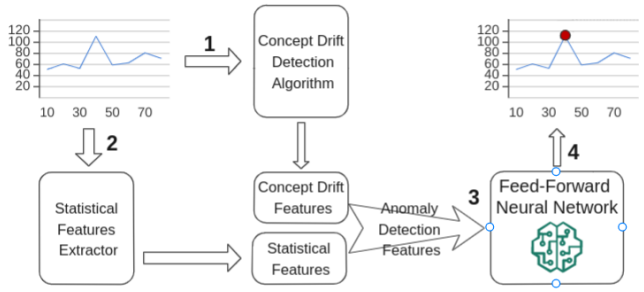


Figure 1.1: Structure and flow of our framework.

In Chapter 2, we provide an extensive review of the existing works and how they studied related problems as well as their limitations. We also explain how our study overcomes such limitations. In Chapter 3, we study the problem of concept drift detection and propose our algorithm to identify concept drifts. Chapter 3 discusses step 1 in the Figure 1.1 . In Chapter 4, we study the problem of anomaly detection and introduce the features (both concept drift based and statistical based), the datasets, and the classifier we use to model and detect anomalies. Chapter 4 discusses steps 2 and 3 in the Figure 1.1. In Chapter 5, we demonstrate the results and discuss the performance of our framework. Chapter 5 discusses step 4 in the Figure 1.1. Finally, in Chapter 6, we conclude this study and provide a groundwork for future studies.

Chapter 2

Related Works

In this chapter we dig deeper in the problem of anomaly detection and concept drift detection by introducing several related and background studies and explaining how they have studied these problems. We then explain what are some of the limitations of these studies and how that motivated us to conduct this study. We finish this chapter by explaining our contributions and how we try to address the existing limitations.

Anomaly detection in time series data is a well studied topic in the literature [(Kumar *et al.*, 2009), Rousseeuw and Leroy (1987), (Aggarwal, 2016), (Box and Jenkins, 1976), (yf *et al.*, 2014), (Yairi *et al.*, 2001), (Çelik *et al.*, 2011), (Oehmcke *et al.*, 2015)]. However, most of the existing studies lack an explicit consideration and analysis of novel patterns and its coexistence with, influence, and impacts on the anomalous data and anomaly detectors.

In the context of time series data, concept drift is often defined as a significant change in the rolling mean of the data. We provide formal definitions in

the next section.

In the next two sections, we explore the existing studies of anomaly detection and concept drift detection problem, respectively.

2.1 Anomaly Detection

In order to categorize our related work review of anomaly detection in a more aligned way with the approach of our framework, we break down the review into two subsections. In the first subsection, we review the related works in anomaly detection based on whether the true labels of data are available or not, i.e. normal and abnormal data points are known. In the next subsection, we review the related works based on the method that is used to study anomaly detection. Specifically, we review the statistical approaches versus the the machine learning based approaches.

2.1.1 Supervised vs. Semi-Supervised vs. Unsupervised

One way of breaking down anomaly detection in time series data is based on whether or not the anomaly detector is trained on labeled data. Depending on the supervision of the domain, i.e. whether anomalies are automatically or manually labeled and identified by domain experts, there are supervised, semi-supervised and unsupervised anomaly detectors. For completely labeled data a supervised model, e.g. K Nearest Neighbors or Neural Network could be used [(Saurav *et al.*, 2018), (Xu *et al.*, 2020)]. For partially labeled data, e.g. not

enough identified anomalous data points exist to be fed to a fully supervised detector, semi-supervised learning or active learning methods are used. Semi-supervised learning methods are used when both labelled and unlabeled data coexist (Zhu and Goldberg, 2009). Active learning is a learning method used in similar environments where the learner chooses which part of unlabeled data should be labelled and then it asks an oracle, e.g. a domain expert, to label that part of data. (Le and Papotti, 2020) recently proposed a distance based anomaly detection algorithm for time series data that uses active learning. In cases where there is no labeled data available, unsupervised methods are used. (Munir *et al.*, 2019) proposed a predictive Deep Convolutional Neural Network (CNN) model named *DeepAnT* that uses its predictions to identify anomalies, i.e. comparing the predicted value for a specific time step with the actual time series value in that time step.

2.1.2 Statistical vs. Machine Learning

(Braei and Wagner, 2020) breaks down the study of anomaly detection in time series data into two main categories: Statistical methods and Machine Learning based methods. Statistical methods are based on the assumption that the time series data points are generated by an underlying data generation process according to a distribution. The goal is to estimate the distribution and model the underlying data generator so that we can predict and analyze the time series data. As opposed to statistical approaches, Machine Learning based approaches consider the data generation process a black box and only focus on and try to learn from the time series data. (Breiman, 2001) takes this

discussion to the next level and describes this categorization as two different cultures and recommends the machine learning based approaches for time series analysis.

As for the statistical approaches, (Kumar *et al.*, 2009) uses Autoregressive Models (AR) to model the underlying data generation process. AR is a model that assumes each data point n_t is computed based on the linear combination of the points before it $n_1, n_2, n_3, \ldots, n_{t-1}$. (Rousseeuw and Leroy, 1987) uses Moving Average Models (MA) to model the underlying data generation process. Similar to AR, MA also tries to predict the value at each data point n_t using the information from the previous data points. But instead of using the actual observed values of previous points, MA uses a linear combination of the errors of previous predictions for data points to predict the value of the current data point. (Box and Jenkins, 1976) uses a combination of both AR and MA methods to predict the value at each data point, i.e. a linear combination of the observed previous points and the prediction errors of the previous points. (Boracchi and Roveri, 2014) proposes an approach that exploits self-similarity of a change-free portion of a time series as a means to identify unexpected changes in the other portion of the time series. They consider a batch of data points to be anomalous if it is not statistically similar enough to at least one batch in the change-free portion.

On the other hand, machine learning approaches with a black box treatment of the data generation process only rely on time series data itself. (Yairi *et al.*, 2001) and (Dunning and Friedman, 2014) use K-Means Clustering method to categorize time series points into normal and abnormal. (Çelik

et al., 2011) uses Density-Based Spatial Clustering of Application with Noise algorithm (DBSCAN) to categorize time series data points into one of the following three classes, core points, border points, and anomaly points. As for the Neural Network based classification, (Haselsteiner and Pfurtscheller, 2000) uses Multi Layer Perceptron (MLP) models which are basically fully connected feed-forward neural networks for the time series classification problem. (Munir *et al.*, 2019) uses a convolutional neural network (CNN) architecture to predict and forecast the future values of a time series and identify anomalies based on how off the time series value is from the predicted one. (Wu *et al.*, 2019) used Gated Recurrent Unit networks (GRU), which belongs to the category of Recurrent Neural Networks (RNN) to detect anomalies in time series.

Statistical-based methods and machine learning based methods mentioned above lack a thorough and explicit integration and consideration of various types of concept drift in their studies. Deep learning models, especially RNNs which are known to consider historical data in the decisions they make, may implicitly learn the behaviors of concept drift in time series data, given their black box and unexplainable nature. However, a lack of concept drift analysis is still present in those studies.

2.2 Concept Drift Detection

Concept drift has been extensively explored in the context of classification problems [(Wang *et al.*, 2017), (Zhang *et al.*, 2008), (Masud *et al.*, 2011)]. In dynamic environments, the distribution of the input data to a classifier

evolves over time. Studies assume that such evolutions are unexpected and unpredictable and could happen at any given time. Though, in some cases a change may be predictable, e.g. the temperature level is expected to fluctuate upon seasonal changes. In the context of classification, concept drift and dynamicity in the distribution of data result in performance deterioration of the models working with that data, if models are not prepared to recognise and adapt the change. (Bifet and Gavaldà, 2007) proposes ADWIN2, an algorithm that utilizes a window based approach to detect concept drift in data using the error rate of a classification model. They use two side-by-side sliding sub-windows, one scanning new data as it slides and one keeping history. They monitor the difference of the mean value of both windows and declare a drift upon significant difference. Recently, (Raab *et al.*, 2020) proposed a window based approach similar to (Bifet and Gavaldà, 2007) that uses two sliding sub-windows to detect concept drift. They use a statistical test named Kolmogorov-Smirnov test to compare the distribution of the data within each of the sub-windows to compute their difference. Upon significant difference, they declare the emergence of a drift period.

In the context of time series data analysis, (Cavalcante *et al.*, 2016) proposes a feature extraction method as well as a classifier that utilizes those extracted statistical features to identify concept drift in time series data. It is important to note that they define a drift period as anywhere in the time series where those extracted features evolve and their classifier of choice recognizes that change. In a more implicit study of concept drift in time series

data, (Guajardo *et al.*, 2010) proposes a strategy to update time series forecasters based on the Support Vector Regressor (SVR) models, to adapt the drift and improve their performance. Their proposed strategy tunes the selection of train, test, and validation sets as well as model and feature selection parameters.

2.2.1 Types of concept drift

There are several types of concept drift studied in the literature, e.g. Abrupt drift, Gradual drift, and Incremental drift. Some researchers have studied and identified different types of concept drifts in the context of the classification problem [(Widmer and Kubat, 1994), (Gama and Rodrigues, 2009), (Žliobaite, 2011), (Hoens *et al.*, 2012), (Moreno-Torres *et al.*, 2012), (Gama *et al.*, 2014)]. Many of the given definitions for various types of concept drifts in the above studies are qualitative or informal. In the context of time series forecasting, (H. F. M. Oliveira *et al.*, 2017) proposed a particle swarm optimization based approach that is able to forecast time series while detecting and considering concept drift. PSO is essentially an optimization method that is used to optimize a model in the training and validation phase.

In their study, (Webb *et al.*, 2016) provided formal definitions for various types of concept drift in an effort to characterize and formalize concept drift. They mention in their study that the provided definitions are probability based and it is not clear as to how they can be used in real world applications (such as time series classification). They also do not provide an algorithm or a method to actually detect concept drift as their study is limited to definitions

and characterizations. For this study, we use some of the basic definitions proposed by (Webb *et al.*, 2016) to construct a complete set of formalizations required to accurately and formally identify concept drifts for the purpose of using them in the context of time series applications. Our work is unique in the sense that it provides formal definitions for concept and various drift types in the context of time series anomaly detection.

2.3 Motivations

The problem of anomaly detection in non stationary environments with concept drift has not been extensively studied. (Zambon *et al.*, 2017) studied the problem of concept drift and anomaly detection in graph streams which is inherently different from time series data. (Saurav *et al.*, 2018) and (Xu *et al.*, 2020) studied the problem of anomaly detection with concept drift adaptation using a temporally enhanced deep neural network called Recurrent Neural Networks (RNN). Deep neural network models are considered as merely black boxes with limited explainability. Concept drift presence is only considered implicitly by the features extracted from the time series data by the deep neural network model. Therefore, the influence of it is not quantifiable with this approach. (Amarasinghe *et al.*, 2018) proposed an explainable RNN model for anomaly detection, but their work lacks discussion and analysis of the role of concept drift and they mostly focus on anomaly detection.

To the best of our knowledge, there are two published studies that explore

the problem of anomaly detection in time series data while explicitly considering concept drift in a more explainable way compared to the deep neural networks. (Ma *et al.*, 2018) proposed *StepWise* which is a framework that helps anomaly detection models (anomaly detectors) to adapt themselves in the event of emergence of a concept drift. It is the first study to explicitly and directly consider concept drift detection for the purpose of anomaly detection in time series data. However, their study is limited in two ways. First, they do not discuss or consider various types of concept drift that have been studied in the literature, e.g. Abrupt, Gradual, and Incremental Concept Drifts. Rather, they consider all types of drift as a general concept drift. As we will study in this research, various types of concept drift have different effects on the anomaly detectors. Second, they do not propose a specific anomaly detection algorithm for time series data. Rather, they propose a means to help with such detectors by providing them with information about concept drift. However, various anomaly detection models work in different ways and given the different standards of input and output, models are not necessarily flexible enough to accept additional concept drift information from the *StepWise* framework to adapt themselves with the novel patterns emerging.

Very recently, (Yu *et al.*, 2020) studied the similar problem and proposed *DDCOL*, an algorithm that identifies abrupt anomalous changes in time series data with the adaptation of concept drifts. Similar to (Ma *et al.*, 2018), their work also lacks a thorough analysis of various types of drifts and their effects on the performance of anomaly detectors.

Previous studies suggest that non-stationary time series data experience

various types of fluctuations over time. A dynamic underlying data generator changes its data generation patterns over time and it results in what is known as concept drift in the literature. Studies also suggest that time series anomaly detectors are affected by concept drifts and their performance deteriorates over time if they are not adaptive towards concept drift. A lack of formal definition and integration of concept drift and anomaly detection in time series data is the main incentive of this work. A few previous studies that also explore and address the effects and impacts of concept drift on anomaly detection in time series data have limitations in at least one of these three categories: 1. Implicitly considering concept drift adaptation through black box deep learning neural network models that cannot quantitatively explain the effects of concept drift on anomaly detection. 2. Not formally defining various types of concept drift and their effects on anomaly detection. 3. No concept drift detection algorithm or an integrated anomaly detector with explicit integration of concept drift is provided. Our study addresses all three issues as we explain in the next section.

Chapter 3

Concept Drift Detection

This chapter contains the following sections:

1. A set of parameters used in the definition of concept drifts, the concept drift detection algorithm, and the preliminary functions.
2. A set of preliminary functions used as helper functions to develop the concept drift detection algorithm.
3. A set of definitions provided to create a concrete foundation to define concept drift.
4. An algorithm to detect various types of concept drifts in univariate time series data.
5. A discussion on how various types of concept drift impact anomaly detection.

Before we continue, it is worth noting that we use "drift" as a shorter term to refer to "concept drift". Also, we will define "concept" as a period of

stability in the time series as opposed to "drift" which represents a period of instability. Between every two consecutive periods of stability, "concepts", lies a period of instability, "drift", which we categorize into three types, i.e. abrupt, gradual, and incremental drifts. We will define the notion of stability and instability with concrete definitions in this chapter. Figure 3.1 shows a period of stability, followed by an abrupt drift that marks a period of instability, followed by the next period of stability.

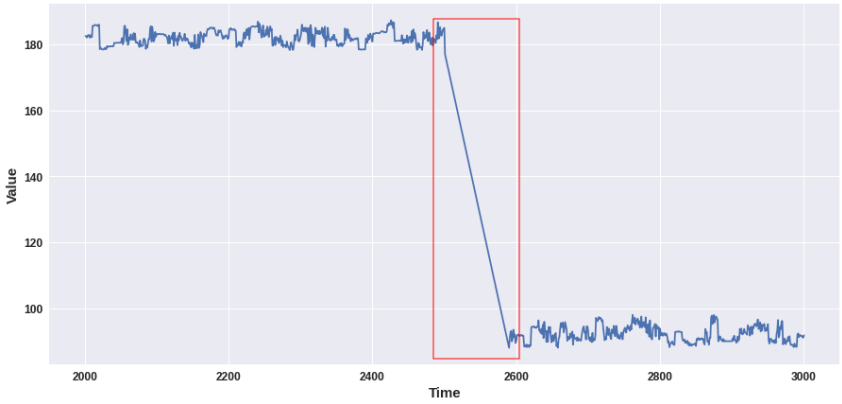


Figure 3.1: An example of concept drift. The marked interval demonstrates a period of instability known as a concept drift and specifically, an abrupt drift.

It is also worth noting that we identify the specific type of a drift after its starting and ending points are identified. We do so by checking the drift characteristics against each of the drift type definitions and matching them.

Finally, it is noted that we revisit some of the parameters in the coming sections of this chapter introduced by (Webb *et al.*, 2016) in their study of characterizing concept drifts. We introduce more parameters to build the ground for our concept drift detection algorithm. Specifically, (Webb *et al.*, 2016) introduced the notion of the following parameters which we revisit and

base some of our definitions on:

- Minimum duration for concept drift. We revisit this parameter as ϕ in the next section.
- Maximum duration for abrupt drift. We revisit this parameter as δ in the next section.
- Gradual drift window period. We revisit this parameter as ν in the next section.
- Maximum allowed fluctuation during the gradual drift period. We revisit this parameter as μ in the next section.

3.1 Symbols and Parameters

In this section we provide a list of the parameters we use for both definitions and the concept drift detection algorithm. It is noted that our framework has the flexibility to compute these parameters based on the statistical properties of the time series if they are not given by the user. We propose our default parameter calculation approach in the Appendix A. We report the impact of changing the value of the novel parameters we introduce (W_b and Δ) on the performance of anomaly detection in Appendix A.

3.1.1 Base Window Size, $W_b \in \mathbb{N}$

In our definition of concept and in our concept drift detection algorithm, we track a certain amount of "history" to account for context and historical data

points leading to the current state of the time series. W_b is the size of the window leading to the current point we want to calculate the concept value for.

In Figure 3.2, W_b is set to 10 and for determining whether time step 25 marks the beginning or end of a stable concept, all of the 10 points leading to it are considered. We explain how this parameter and these "history" points are taken into account in the next section.

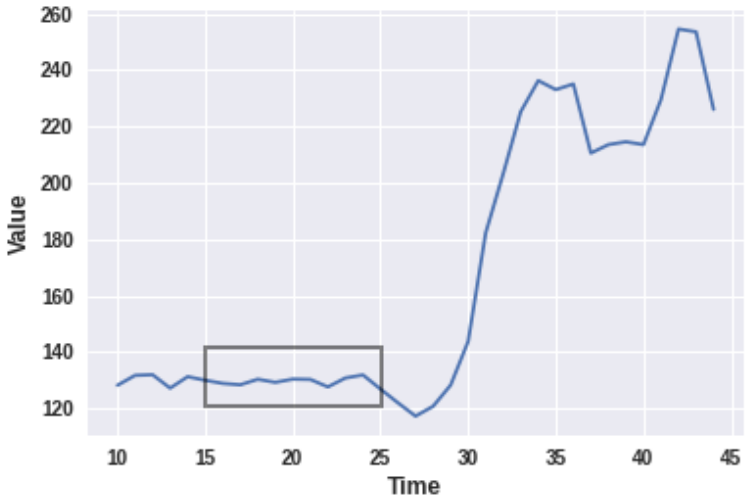


Figure 3.2: W_b is set to 10. Upon examining the point at time step 25, all of the 10 points leading to it are considered.

3.1.2 Minimum Stability Period, $\phi \in \mathbb{N}$

For each continuous portion of time series to be considered as a stable concept, a minimum period of stability of length ϕ points is required. It ensures that each concept has a minimum period of stability required by the drift detection algorithm to function and efficiently detect drifts.

3.1.3 Maximum Tolerable Fluctuation, $\epsilon \in \mathbb{R}^+$

During each concept a tolerable amount of fluctuation in value is allowed. Time series values do not necessarily remain unchanged as time passes by. Room temperature might change by one or a fraction of one Celsius degree and still be considered as stable. In the next subsection, we demonstrate how we calculate the fluctuation with the help of *Stat* function defined in Algorithm 1.

3.1.4 Maximum Abrupt Drift Duration, $\delta \in \mathbb{N}$

Abrupt drift is a specific category of concept drift. A sudden and "abrupt" transition from one stable concept to another stable concept is considered as an abrupt drift. For a drift (i.e. a transition) to be considered abrupt, it must happen in a short period of time, i.e. less than δ time points.

It is important to note that we do not measure time in seconds or minutes, but rather in orderly and chronic points in time. The scale used for time series could be from a fraction of a second to days or even years depending on the basis the time series data are recorded.

3.1.5 Minimum Gradual Drift Period, $\Delta \in \mathbb{N}$

Gradual drift is another type of concept drift. A "gradual" transition from one stable concept to another stable concept is characterized and known as gradual drift. For a drift to be considered as gradual, there must be a minimum transition time, i.e. minimum drift period. Otherwise, a quick gradual drift

could be considered as abrupt drift as well if it is shorter than δ.

The minimum drift period that must be satisfied for a drift to be considered as gradual is defined by parameter Δ. Similar to δ and ϕ, Δ is measured by time points.

3.1.6 Maximum Gradual Drift Stat Fluctuation, $\mu \in \mathbb{R}^+$

A gradual drift, as the name implies, must encompass the idea of graduality. To enforce that, we introduce the maximum gradual drift stat fluctuation, μ, as a cap for fluctuation of stat value (defined in the next subsection) at any point during the drift period for it to be considered as gradual. It is measured in the same metric used for the values of time series, e.g. Celsius, meter, etc.

3.1.7 Gradual Drift Graduality Period, $\nu \in \mathbb{N}$

The maximum gradual drift concept fluctuation, μ, is validated within every continuous ν points of time within the drift period. Concretely, the fluctuation of concept value within every continuous ν points of a gradual drift must be less than or equal to μ. As we will show in the definitions section, this will ensure the "graduality" aspect of gradual drifts.

3.2 Preliminary Functions

We provide definitions for the main and helper functions we use to detect concept drift in a univariate time series. We use the symbol T to represent time series and $T(t)$ returns the value of time series T at point t in time. We

proceed in the next section with the formal definitions for concept and drift.

3.2.1 Stat Function, $Stat : \mathbb{N} \rightarrow \mathbb{R}$

Stat receives a positive integer as its input that represents the time step at which we want to calculate the statistical property (rolling mean) value. It returns a real number that represents the rolling mean at the given point while considering a window of W_b points as history. As discussed before, we consider *mean* as the criterion and statistical property to measure concept stability. The rolling part is added to account for historical changes and values that led the time series to the specific point we are interested in. Concretely, the inner body of *Stat* function is defined in Algorithm 1.

For example, Figure 3.2 shows the W_b history points considered in the computation of the *Stat* function for point $t = 25$ when W_b is set to 10.

Algorithm 1 Stat Function

Require: Time Series T and pre-defined parameters

```
1: function STAT(t)
2:     if t < W_b then
3:         return -1                    ▷ Min W_b points of history required
4:     sum ← 0
5:     for i = t − W_b to t do
6:         sum ← sum + T(i)
7:     res ← sum/W_b
8:     return res
```

The time complexity of the Algorithm 1 is $O(1)$. This is because the number of iterations of the loop at line 5 in the *Stat* function is upper bounded by a constant value, W_b.

3.2.2 Distance function, $Distance : (\mathbb{N}, \mathbb{N}) \rightarrow \mathbb{R}^+$

Distance receives two positive integers as its input that represent the time steps at which we want to calculate the difference in stat (rolling mean) value. It returns a positive real number that represents the size of the difference in stat value at the two given points. *Distance* function is defined in Algorithm 2

Algorithm 2 Distance Function

Require: Time Series T and pre-defined parameters

```
1: function DISTANCE(t_1, t_2)
2:     if t_1 < W_b or t_2 < W_b then
3:         return -1                          ▷ Min W_b points of history required
4:     s_1 ← Stat(t_1)
5:     s_2 ← Stat(t_2)
6:     diff ← abs(s_2 − s_1)
7:     return diff
```

For the similar reason to the Algorithm 1, the time complexity of the Algorithm 2 is also $O(1)$.

3.3 Definitions

In this section, we first provide the definitions of stable concept. We then proceed with introducing the notations we use to define drifts. We then follow up with introducing and defining three categories of drift, i.e. abrupt, gradual, and incremental.

3.3.1 (Stable) Concept

A concept refers to a period of stability in time series during which the time series values do not fluctuate enough to make it "unstable" and start a drifting period. Concretely, given ϕ and ϵ as we defined them, the time interval $[t_1, t_2]$ of a given time series T is considered to be "stable" and constructing a stable concept if the following expression holds true.

$$t2 \geq t1 + \phi \wedge \forall_{i \in [1,\, t_2 - t_1]}\, Distance(t_1, t_1 + i) \leq \epsilon$$

This definition first ensures that the minimum period of stability, ϕ, is respected. It also ensures that the fluctuation of *Stat* value for every point in the interval do not deviate more than ϵ when compared to the beginning of the concept, t_1. It is important to note that we only compare the *Stat* value of each point to the very beginning point of the concept. This is to ensure that a subtle and gradual drift through a long interval does not go unnoticed. Therefore, t_1 remains as the base comparison point.

Figure 3.3 demonstrates how using ϕ as a lower bound for the duration of a stable concept helps our algorithm to avoid identifying short periods of time with relative stability compared to their before and after as stable concepts.

3.3.2 Concept and Drift Notations

The i_{th} identified stable concept, C_i, has a starting and ending point associated with it, $C_i.S$ and $C_i.E$, respectively. What comes after $C_i.E$ and before $C_{i+1}.S$ is a drift. The i_{th} drift, written as D_i takes place between the i_{th} and and

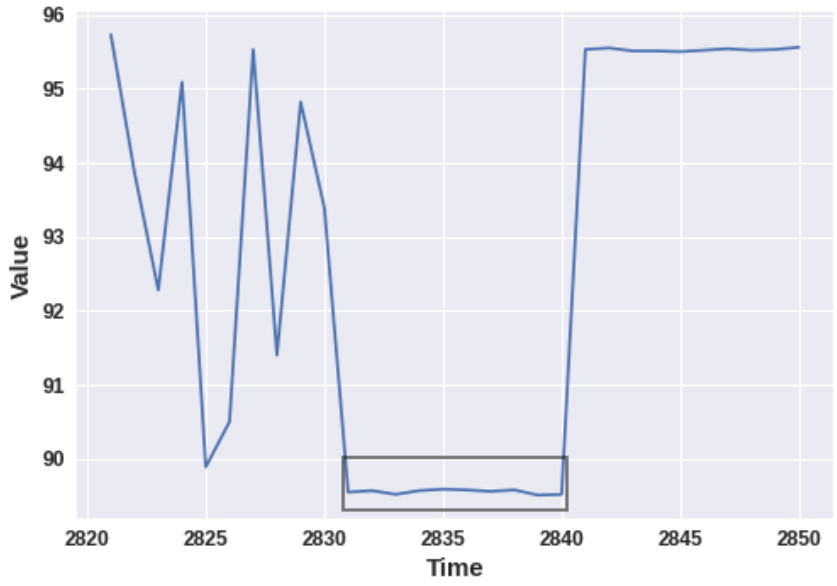


Figure 3.3: If ϕ parameter is set to 10 or a larger number, then the marked period cannot be identified as a stable concept.

$(i+1)_{th}$ concepts. Similarly, a drift has a starting and ending point associated to it. The starting and ending points of the i_{th} drift are noted as $D_i.S$ and $D_i.E$, respectively.

For example, Figure 3.4 shows part of a time series that includes the following periods:

- Third stable concept of the time series starting at $T1 = 2040$ and ending at $T2 = 2480$.
- Third drift period of the time series starting at 2481 and ending at 2602.
- Fourth stable concept of the time series starting at $T3 = 2603$ and ending at $T4 = 2945$.

Table **??** shows how we denote the introduced parameters for this example.

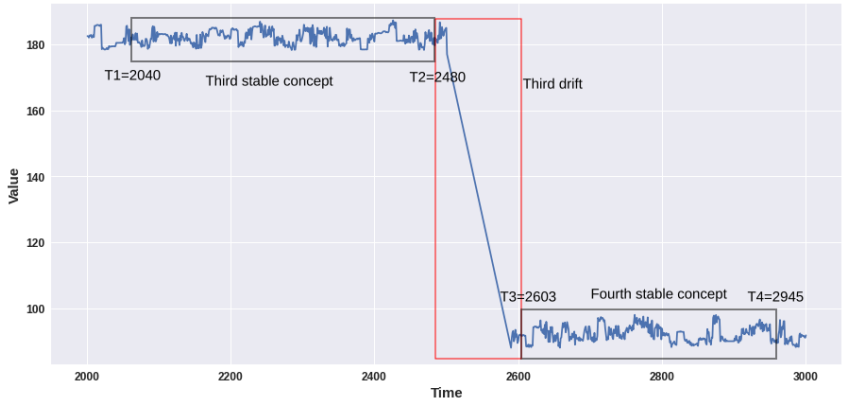


Figure 3.4: A stable concept followed by a drift period followed by another stable concept.

3.3.3 Abrupt Drift

The first type of drift is known and defined here as abrupt drift. It is informally characterized as a sudden change in the statistical properties used as criteria for concept (*Stat* value as defined in our study), i.e. a sudden transition from one stable concept to another.

Given δ as defined, C_i as the trailing concept, i.e. the concept that ends when the abrupt drift starts, and C_{i+1} as the leading concept, i.e. the concept that begins when the abrupt drift ends, the i_{th} drift noted as D_i is considered to be abrupt if the following expression holds true.

$$C_{i+1}.S - C_i.E \leq \delta$$

Figure 3.5 shows an example of abrupt drift. It demonstrate how the drift period must be shorter than or equal to δ for a drift to be considered abrupt.

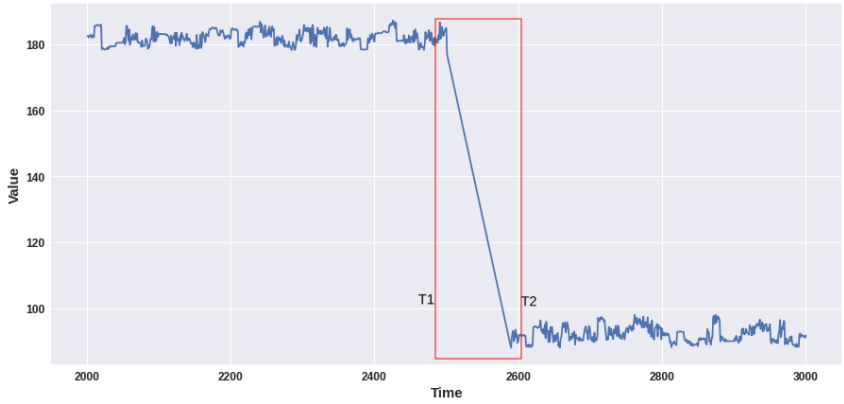


Figure 3.5: A drift that occurs between time steps $T1$ and $T2$. In order to be considered as abrupt, $T2 - T1$ must be less than or equal to δ.

3.3.4 Gradual Drift

The second type of drift is known and defined here as gradual drift. It is informally characterized as a gradual change in the *Stat* value over time. It is noted that this change does not have to be monotonic and steady during the drift period. But the overall trend must express a deviation from one concept to another.

Given Δ, μ, and ν as defined, C_i as the trailing concept, and C_{i+1} as the leading concept, the i_{th} drift noted as D_i is considered to be gradual if the following expression holds true.

$$C_{i+1}.S - C_i.E > \Delta \wedge \forall t \in [C_i.E, C_{i+1}.S - \nu] \; Distance(t, t + \nu) \leq \mu$$

Figure 3.6 shows how the parameters μ and ν are used. Specifically, during the drift period between $T1$ and $T2$, the fluctuation of stat value within every

ν time steps must be lower than μ.

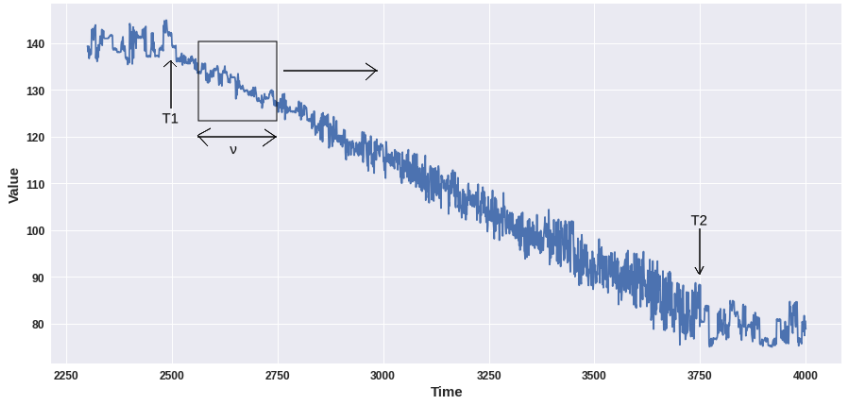


Figure 3.6: An example of gradual drift. Note that during the drift period, time series values can fluctuate as long as the fluctuation within a window of ν points is limited by at most μ.

3.3.5 Incremental Drift

The third type of drift is known and defined here as incremental drift. It is informally characterized as a unidirectional change in the *Stat* value over time. Unlike gradual drift in which only the overall direction of change must be from one concept towards another, in incremental drift the change must be unidirectional during the drift time.

Given C_i as the trailing concept and C_{i+1} as the leading concept, the i_{th} drift noted as D_i is considered to be incremental if the following expression

holds true.

$$\forall t \in [C_i.E, C_{i+1}.S] \forall u \in [t, C_{i+1}.S] Distance(C_i.E, t) \leq Distance(C_i.E, u) \\ \wedge Distance(t, C_{i+1}.S) \geq Distance(u, C_{i+1}.S) \tag{3.3.1}$$

The above definition ensures an incremental property of the drifting period, i.e., as time passes through the drift period, the *Distance* value between each drift point and the end point of the drift must decrease. In Figure 3.7 for each time step t and u such that $t < u < D_i.E$, the expression 3.3.1 must hold true for the drift to be considered as incremental.

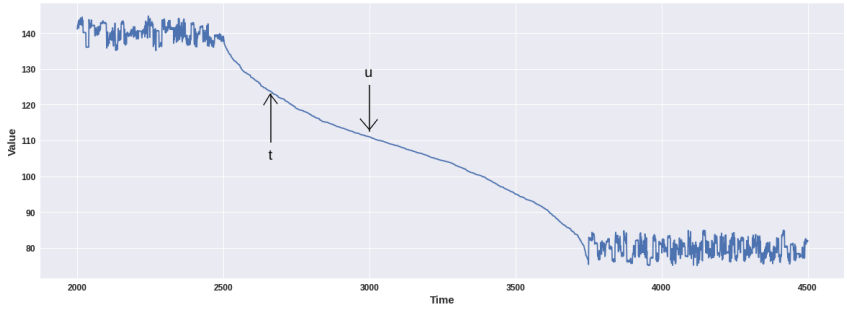


Figure 3.7: An example of incremental drift. Note that throughout the drift period, the *Distance* value between each point and the drift end point keeps decreasing. Also, the *Distance* value between each point and the drift start point keeps increasing.

3.4 Concept Drift Detection Algorithm

In this section we propose our concept detection algorithm that is able to identify all stable concepts in a given time series. In the next section, we

propose the drift identification algorithm that categorizes the identified drifts into one of the introduced drift types.

Given the fact that stability in time series data is subject to relative historical context for each period of time, and that the *Stat* value is derived from W_b historical points leading to the point we are interested in, we mark the ${W_b}^{th}$ point as the beginning of the first stable concept. This is because the very first few points of a time series data does not convey much history, nor does it tell anything about the scale of fluctuation in the time series data. Therefore, as a starting point, we mark the ${W_b}^{th}$ point as $C_0.S$ and the algorithm begins with looking for an end to the first stable concept, i.e. $C_0.E$.

We first introduce some helper functions and then build our algorithm using them.

3.4.1 Concept End Detection Function, $FindConceptEnd : \mathbb{N} \rightarrow \mathbb{N}$

This function is used to find the end of a stable concept by giving it the starting point of the concept. It looks for any fluctuation of *Stat* value that comes after the first ϕ points of the minimum stability period.

Algorithm 3 Concept End Detection

Require: Time Series T and pre-defined parameters

```
1: function FindConceptEnd(Start)
2:     for i = Start + φ + 1 to T.length do
3:         if Distance(Start, i) > ε then
4:             return i                ▷ i is the end of concept started at Start
5:     return -1                       ▷ Reached the end of time series
```

It is noted that in the Algorithm 3, we do not compare the first ϕ points of concept to see if it satisfies the condition for ending the concept. It is because we only mark a period of time as stable concept if it has at least a minimum of ϕ points with stability in the *Stat* value, i.e. it has already been confirmed that the first ϕ points satisfy the stability condition.

The time complexity of the Algorithm 3 is $O(n)$ where n is the length of the time series. This is because the maximum number of time points iterated by the $FindConceptEnd$ function is upper bounded by the number of data points in the time series.

3.4.2 Concept Start Detection Function, $FindConceptStart : \mathbb{N} \rightarrow \mathbb{N}$

This function is used to find the beginning of the next stable concept by giving it the ending point of the previous concept. When the ending point of a concept is reached, via the help of Algorithm 3, a drift period starts. A adrift period ends only when the beginning of the next stable concept is identified. This step is done via the help of Algorithm 4.

A stable concept is identified at a time point if and only if the next ϕ points possess the property of not fluctuating more than ϵ in *Stat* value. Therefore, the Algorithm 4 only marks the starting point of the next stable concept after it made sure about the minimum period of stability.

The time complexity of the Algorithm 4 is also $O(n)$ where n is the length of the time series. This is because the outer loop in the $FindConceptStart$ function at line 3 is upper bounded by the length of the time series and the

Algorithm 4 Concept Start Detection

Require: Time Series T and pre-defined parameters

```
1:  function FindConceptStart(End)
2:      FoundFlag ← True
3:      for i = 1 to T.length − End − ϕ do
4:          for j = 1 to ϕ do
5:              if Distance(End + i, End + i + j) > ϵ then
6:                  FoundFlag ← False
7:                  break
8:          if FoundFlag then
9:              return i
10:     return -1                          ▷ Reached the end of time series
```

inner loop at line 4 is upper bounded by a constant value, ϕ.

3.4.3 Concept Detection Function, *DetectConcept*

Using the two helper functions 3 and 4 and the fact that the first stable concept starts at the time step W_b, i.e. $C_0.S$ is equal to W_b, we present the concept detection in Algorithm 5. It is noted that this function, similar to other functions, has access to the time series T and the pre-defined parameters. It also does not accept any input. As its output, it returns a list of two-tuples that contains the starting point and ending point of each concept, as well as another list of two-tuples that contains the starting point and ending point of each drift.

In the next section, we present our drift identification algorithm that analyzes the drifts detected by Algorithm 5 and categorize them into one of the defined drift groups.

Algorithm 5 has two phases: A stability phase and a drift phase. This duality is tracked by the variable *ConceptPeriod*. If *ConceptPeriod* is True,

Algorithm 5 Concept Detection

Require: Time Series T and pre-defined parameters

```
1: function DetectConcept
2:     ConceptList ← [(W_b, −1)]
3:     DriftList ← [(−1, −1)]
4:     Finished ← False
5:     while not Finished do
6:         ConceptPeriod ← True
                              ▷ -1 as an index returns the last element in a list
7:         if ConceptList[−1][1] == −1 & ConceptPeriod then
8:             ConceptEnd ← FindConceptEnd(ConceptList[−1][0])
9:             if ConceptEnd == −1 then
10:                ConceptList[−1][1] ← T.length − 1
11:                Finished ← True
12:                break
13:            else
14:                ConceptList[−1][1] ← ConceptEnd
15:                DriftList.append((ConceptEnd, −1))
16:                ConceptPeriod ← False
17:        if DriftList[−1][1] == −1 & not ConceptPeriod then
18:            ConceptStart ← FindConceptStart(ConceptList[−1][1])
19:            if ConceptStart == −1 then
20:                DriftList[−1][1] ← T.length − 1
21:                Finished ← True
22:                break
23:            else
24:                DriftList[−1][1] ← ConceptStart
25:                ConceptList.append((ConceptStart, −1))
26:                ConceptPeriod ← True
27:    return ConceptList, DriftList
```

then we are at a stability period and looking for a possible end of the current stable concept, which in turn marks the beginning of the next drift. If $ConceptPeriod$ is False, then we are at a drift period and looking for a possible end of the current instability drift period, which in turn marks the beginning of the next stable concept.

The time complexity of the Algorithm 5 is $O(n)$ where n is the length of the time series. This function iterates over the time series only once and at each step identifies the starting and ending point of a concept until the time series is fully iterated. While the time complexity of the $FindConceptStart$ and $FindConceptEnd$ functions is $O(n)$, every point on the time series is only iterated once. This is because at each step of the $DetectConcept$ function in Algorithm 5, the iteration jumps over the points that are iterated over by either of the $FindConceptStart$ and $FindConceptEnd$. Figure 3.8 shows how all of the time series data points are only processed once.

3.4.4 Drift Identification Function, $IdentifyDrift$

The Algorithm 5 is able to detect all concept and drift periods. As the next step, we propose the Algorithm 6 that identifies all the detected drifts and categorizes them into one of the drift subgroups.

The $IdentifyDrift$ function returns a list of three tuples, $(Start, End, Type)$, that adds a type identifier to the $DriftList$. The $Type$ value is categorical. The value 0 represents abrupt drift, 1 represents gradual drift, and 2 represents incremental drift. If a drift does not satisfy any of the introduced three categories of drift, then we assign a value of 3 to it that represents "unknown"

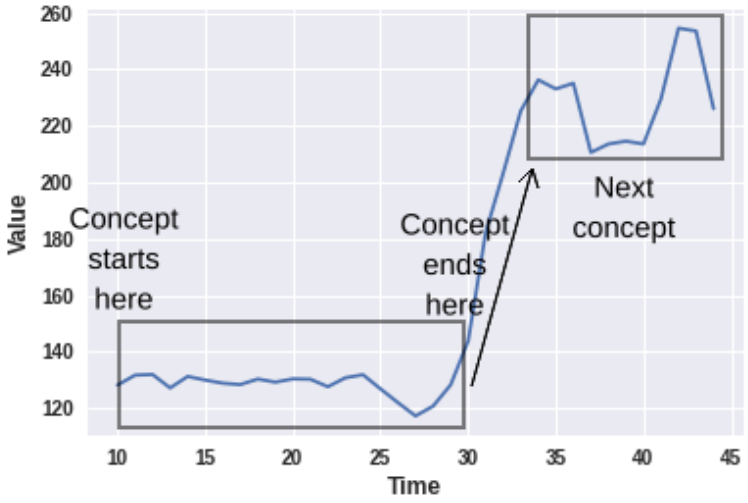


Figure 3.8: When the concept ending point is determined, the concept detection algorithm will skip to the next concept start. It continues until the entire time series is iterated via the $FindConceptStart$ and $FindConceptEnd$ functions.

type of drift.

The time complexity of the Algorithm 6 is $O(n)$ where n is the length of the time series. The outer loop on the line 3 and the inner loops on the lines 11 and 19 in the function $IdentifyDrift$ are together upper bounded by the length of the time series. This is because each drift is only iterated once and even if the entire time series consisted of drifts, then all of the n points are iterated once. Therefore, the number of processed data points is upper bounded by the length of the time series.

Our unified framework starts by first detecting the position of concept drifts using the $DetectConcept$ function in Algorithm 5 and then proceeds to identifying the drift type using the $IdentifyDrift$ function in the Algorithm 6. Each step has a time complexity of $O(n)$. Therefore, our concept

Algorithm 6 Drift Identification

Require: Time Series T and pre-defined parameters

```
1: function IdentifyDrift
                                  ▷ First returned value, ConceptList, is ignored
2:     Ignored, DriftList ← DetectConcept()
3:     for drift in DriftList do
4:         DriftStart ← drift[0]
5:         DriftEnd ← drift[1]
6:         DriftListType ← []
                                                                  ▷ Abrupt Drift
7:         if DriftEnd − DriftStart < δ then
8:             DriftListType.append((DriftStart, DriftEnd, 0))
9:             continue
                                                                  ▷ Gradual Drift
10:        IsGradual ← True
11:        for t = DriftStart to DriftEnd − ν do
12:            if Distance(t, t + ν) > μ then
13:                IsGradual ← False
14:                break
15:        if IsGradual then
16:            DriftListType.append((DriftStart, DriftEnd, 1))
17:            continue
                                                              ▷ Incremental Drift
18:        IsIncremental ← True
19:        for t = DriftStart to DriftEnd do
20:            for u = t to DriftEnd do
21:                ST ← Distance(DriftStart, t)
22:                SU ← Distance(DriftStart, u)
23:                TE ← Distance(t, DriftEnd)
24:                UE ← Distance(u, DriftEnd)
25:                if ST > SU | TE < UE then
26:                    IsIncremental ← False
27:                    break
28:        if IsIncremental then
29:            DriftListType.append((DriftStart, DriftEnd, 2))
30:            continue
                                                                  ▷ Unknown Drift
31:        DriftListType.append((DriftStart, DriftEnd, 3))
32:    return DriftListType
```

drift detection module has an overall time complexity of $O(n)$.

3.5 Concept Drift Impact on Anomaly Detection

Figure 3.9 shows an example of how an abrupt drift can affect the anomaly detection. The time series is experiencing an abrupt drift between time points 25 and 35. The time point 29 has the same value as the time point 32 (both are marked). However, the effects of abrupt drift causes the time point 29 to be considered as anomalous despite its identical value with the normal time point 32. A similar situation occurs in Figure 3.10 but with gradual drift. The two marked points during a gradual drift period have close values but both are considered normal. This is because in gradual drifts, there is enough time for such scenarios to take place, while in abrupt drift the drifting period is limited in length by δ.

This example shows how various types of drift can affect the decision that is made about a point being normal or abnormal. In the next chapter, we show how we take advantage of distinguishing between different types of drift to provide the anomaly classification model with adequate information in the form of concept drift features.

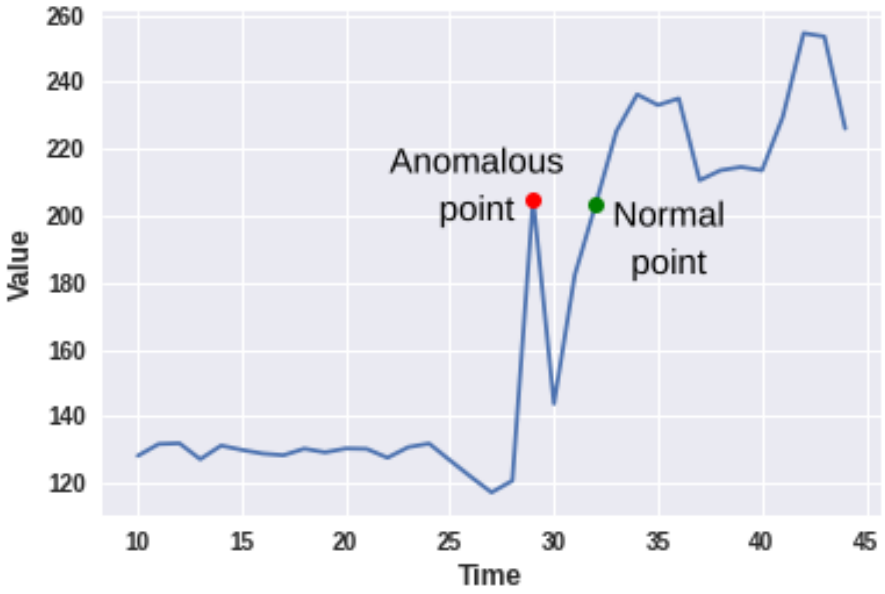


Figure 3.9: Part of a time series with abrupt drift between time 25 and 35. While both marked points have same values, the first one is anomalous and the second one is normal.

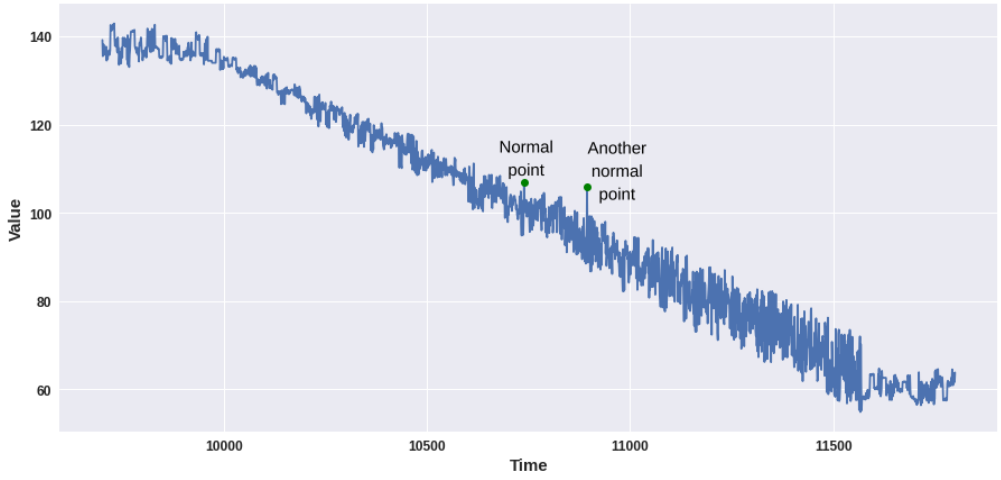


Figure 3.10: Part of a time series with gradual drift between time 10000 and 11500. Both time points are normal.

Chapter 4

Anomaly Detection

In the previous chapter, we presented our concept drift detection algorithm for univariate time series data. We explained how concept drift affects the anomaly detectors' performance and therefore, it is important to consider concept drift for anomaly detection in dynamic settings. We also explained the shortcomings of the similar studies and how our framework fixes them. In this chapter, we present the final step of our framework which is the anomaly detection.

We extract a set of features from the time series data, including several features extracted through the concept drift detection step, and use them to train a classifier on labeled time series data. Some of the datasets we use are already known to have anomalies and are labeled by domain experts. Other datasets are clean of anomalies and we inject anomalies in them to provide labeled data.

In the first section of this chapter, we introduce the features and how we

extract them. In order to study the importance of features and the relationship they have with each other, we present the datasets we use for our experiments with detailed explanation in the second section. We then proceed with Exploratory Data Analysis (EDA) to investigate the datasets. We examine our features using the datasets introduced in the second section. In the third section, we explain the details of how we inject anomalies in the anomaly-free datasets to prepare them for classification. In the fourth and last section, we introduce the classifier we consider to conduct the anomaly detection experiments.

In this chapter and the subsequent chapters, we use the terms error and anomaly interchangeably. Unless explicitly stated, they have the same meaning as anomaly as we defined in the earlier chapters.

4.1 Classification Features

In this section, we introduce the features we extract from the time series data to use them as inputs to a classifier that identifies anomalous data. We break down the features into two categories, statistical features and concept drift features. Statistical features are directly extracted from the time series, but concept drift features are extracted indirectly by using the concept drift algorithms presented in the previous chapter.

4.1.1 Statistical Features

In this section we present seven statistical features that we extract from the time series for the purpose of anomaly detection, i.e. value, first derivative, second derivative, rolling mean, rolling variance, rolling skewness, and rolling kurtosis. For the last four features, we need a window size to compute the "rolling" value over the time. To find the best rolling size, we fixed all parameters and executed the classifier with different rolling sizes ranging from ten to a hundred. Judging by the recall score (given the inherent extreme imbalance in anomaly detection problems) we found that twenty as the window size for rolling features is the best choice. The same approach could be used for new datasets to find the optimal rolling window size.

In Appendix B we provide the definitions as well as examples for each of the following statistical features.

Feature 1: Value

The value of the time series point itself is the first feature we extract. All of the features, either statistical or drift based features, are derived from the actual time series value. We use the value at each time step as a feature that represents that time step.

Feature 2: First Derivative

The first derivative at each point of the time series is the next feature we extract. Our experiments show that the first derivative value has a considerable correlation with the anomaly label, i.e. the higher the first derivative value

the higher the chance that point is anomalous.

Feature 3: Second Derivative

The second derivative is calculated by taking the derivative of the first derivative of the time series. While the first derivative measures the rate of change, the second derivative measures the acceleration of change, i.e. at what rate does the change in the values fluctuate.

Feature 4: Rolling Mean

Mean is a very informative statistical feature widely used in many problems to measure the overall value of the data (Webb *et al.*, 2016). Rolling mean is the mean of values within a rolling window that slides through the time series.

Feature 5: Rolling Variance

Variance measures the extent of volatility in the data. Rolling variance is the variance of values within a rolling window that slides through the time series.

Feature 6: Rolling Skewness

Skewness is a statistical feature that measures the symmetry in the distribution of the data. If the dataset exposes some level of symmetry and there are many anomalous data points on one side of the dataset, then it is likely to expect some anomalous data points on the other side as well. We compute the rolling unbiased skewness obtained by the Fisher-Pearson definition (Fisher, 1930).

Feature 7: Rolling Kurtosis

Kurtosis is a statistical feature that measures to what extent data points are concentrated on the tails of the data distribution when compared to the normal distribution. A high kurtosis means heavily tailed distribution which can be interpreted as tending to have more outliers. This property of the kurtosis value motivated us to use it as a feature for anomaly detection. We compute the rolling unbiased kurtosis obtained by the definition given by Fisher (Fisher, 1930).

4.1.2 Concept Drift Features

In this section we present three concept drift related features that we extract from the results of the proposed concept drift detection algorithm in the previous chapter.

Feature 8: Concept or Drift

The first concept drift related feature is a categorical feature that represents whether the data point belongs to a concept (feature value 0), an abrupt drift (feature value 1), a gradual drift (feature value 2), an incremental drift (feature value 3), or an unknown drift (feature value 4).

Feature 9: Duration of the Period

This feature measures the duration of the stability or instability period that each data point belongs to. For instance, if in a time series data points between time step 231 and 240 belong to an abrupt drift, then the duration feature for

all of the data points equals to 9. Even though this feature is not distinctive among many points that belong to the same period, it is distinctive between the points that belong to different periods. In other words, the lower the value of this feature means the more volatility is present around it.

Feature 10: Percentile of the Point

This feature measures at what percentile of the period the point lies. For instance, if in a time series data points between time step 371 and 492 belong to a stable concept, then the point 490 belongs to the 99th percentile which is interpreted as the tail of a period. In conjunction with the first concept drift based feature, ConceptOrDrift, it can determine that a specific data point belongs to the end of a drift period and this determination could lower the chance for it to be labeled by the classifier as an anomaly compared to when a point is on the same percentile of a stable concept period.

4.2 Datasets and Exploratory Data Analysis

In this section we introduce the datasets we use to evaluate our framework with a detailed Exploratory Data Analysis (EDA) and feature correlation analysis. EDA is a set of statistical and data analytical techniques employed in order to gain an in-depth insight and knowledge prior to using data for the analytical and learning purposes (Tukey *et al.*, 1977). For this study, we use three real world datasets and a set of synthetically generated datasets. We proceed with the introduction of each of the four datasets.

We have the following three objectives for our analyses in this section. We explain our objectives and how they help with the anomaly detection problem below.

- Understanding the distribution of anomalies in the datasets helps us in analyzing the accuracy performance of our model when we perform experiments. For instance, a time series with highly scarce anomalies results in highly imbalanced labels that can impact the performance of the anomaly detector.

- Understanding the length distribution of the datasets helps us to avoid using classification models with higher time complexity. For instance, Support Vector Machine (SVM) classifiers have a training time complexity polynomial to the power of two with respect to the dataset length. Therefore, in order to have a shorter runtime, we need to avoid SVMs if our datasets are too large.

- Calculating features correlation with each other helps us to analyze how much our selected feature set are carrying mutual information with respect to each dataset. For instance, if two individual features highly correlate with each other among all of our datasets, then we should only proceed with one of them.

4.2.1 Yahoo! Time Series Anomaly Detection Datasets

Yahoo! datasets are the first real world dataset we use for our experiments. It consists of a set of time series data gathered from the Yahoo! incoming traffic

to one of their online services. Their online service experienced an abnormal amount of incoming traffic during a seasonal sports event. They employed their domain experts to label the traffic data and identify anomalies. (Laptev *et al.*, 2015)

This dataset has been used as a benchmark for anomaly detection in time series anomaly detection studies [(Le and Papotti, 2020), (Gao *et al.*, 2020)]. We compare our framework's performance against, (Le and Papotti, 2020), the most recent study that reported the highest performance metrics.

EDA

Yahoo! datasets consist of 52 separate time series datasets. Figure 4.1 shows the distribution of the lengths of the Yahoo! datasets. Except for two datasets with the length of 741, the other 50 datasets have a length between 1420 and 1461 which are comparable to each other.

Figure 4.2 shows one of the Yahoo! real world datasets. Anomalies identified by domain experts are marked. Points 400 to 500 are zoomed in to better observe the annotated anomalies.

We also investigate the percentage of anomalous data points within each of the 52 Yahoo! datasets. On average, Yahoo! time series have 1.88% anomalies in them. Figure 4.3 shows the distribution of the percentage of anomalies in Yahoo! time series datasets.

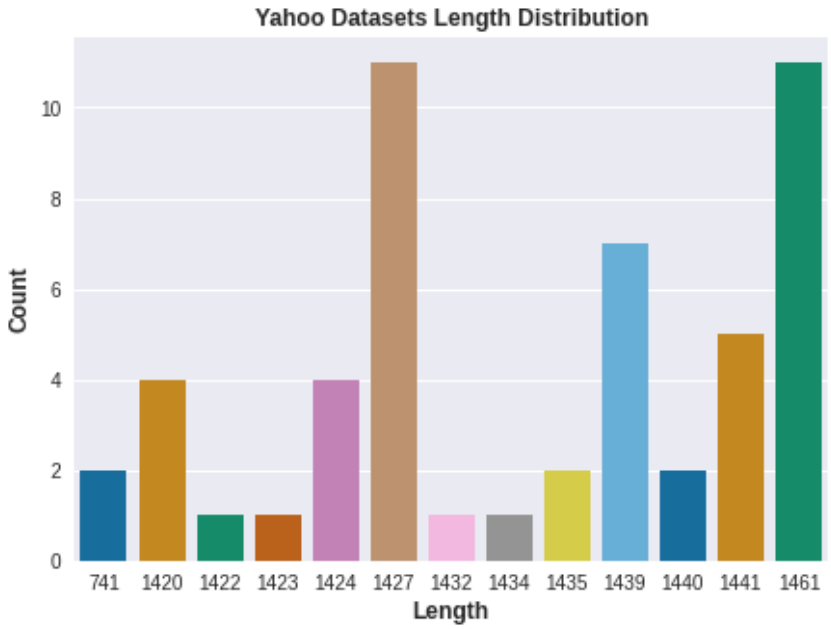
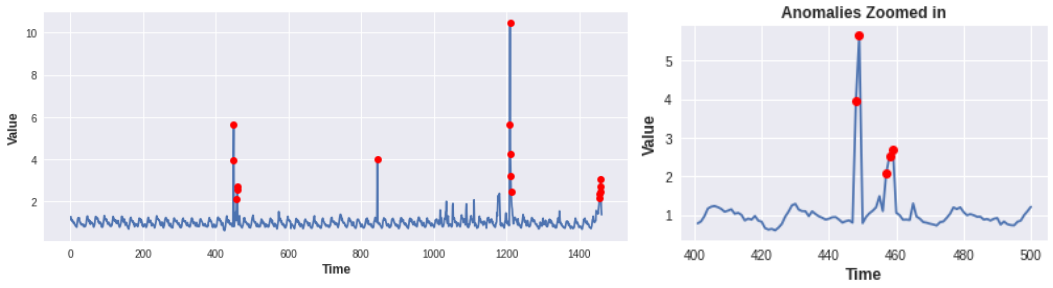


Figure 4.1: Yahoo! real world anomaly datasets length distribution. 50 of 52 datasets have a length between 1420 and 1461.

Feature Correlation

In this subsection, we extract the statistical features introduced in the previous section from the Yahoo! time series datasets to observe how much features are correlated. High feature correlation is a negative indicator for the chosen feature set. Highly correlated features are more linearly dependent and have almost the same effect on the classifier, i.e. classifier can not benefit and gain knowledge from one of them more or less than it could from the other. Figure 4.4 shows statistical features correlation heatmap for two randomly selected Yahoo! datasets. Large negative or large positive correlation is not desired. Figure 4.4 demonstrates that the features are not highly correlated with each other.



(a) Entire dataset with anomalies marked.

(b) Points 400 to 500 zoomed in.

Figure 4.2: One of the Yahoo! real world datasets with anomalous points marked.

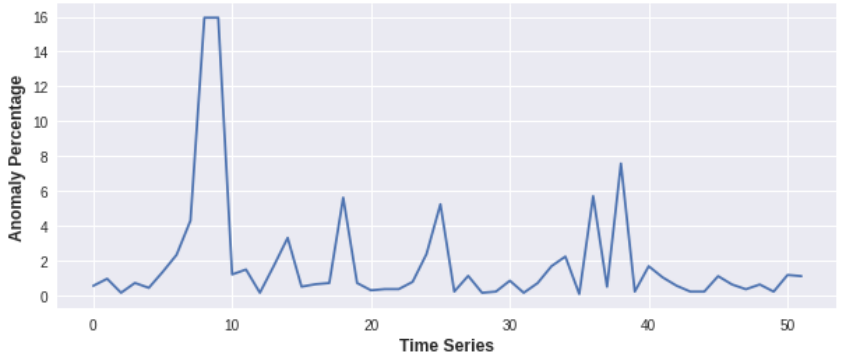
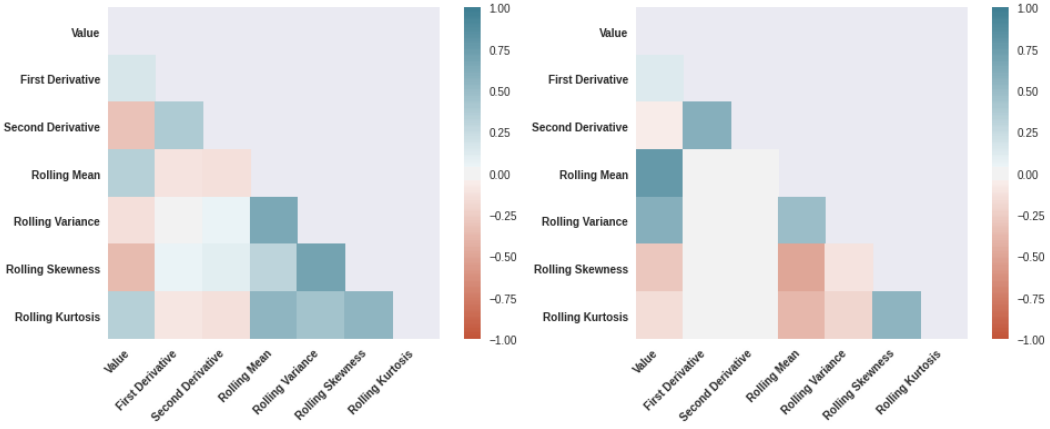


Figure 4.3: Distribution of anomaly percentage within time series for Yahoo! dataset.

4.2.2 Light Sensor Stream Dataset

Light sensor stream dataset is part of a sensor dataset (Zhu, 2010) recorded by 54 sensors deployed at the Intel Berkeley Research Lab. Figure 4.5 shows the map of the building where sensors were recording data (Zhu, 2010). Sensors were deployed at various locations in the building recording the light level over a two month period. Sensors record data every one to three minutes.

Sensor data in this building is affected by many factors, e.g. business hours, holidays, and special events. These factors result in different patterns



(a) Statistical features correlation heatmap for one of the Yahoo! datasets.

(b) Statistical features correlation heatmap for another Yahoo! datasets.

Figure 4.4: Numbers on or above main diagonal are redundant as correlation matrix is symmetric, hence removed from plot.

of change in the recorded data and that can cause concept drift. This dataset is used as a benchmark by concept drift detection studies [(Alves de Souza *et al.*, 2020), (Cabral and Barros, 2020)] and is known to have concept drift. Though, given that this dataset is not labeled for concept drift, the exact location of the drifts are unknown and cannot be agreed upon by various studies defining concept drift in different ways.

It is worth noting that the light sensor dataset does not contain any known anomalies. Therefore, we consider it as anomaly-free. We will inject anomalies into this dataset using the method introduced by (Gierke and Stebner, 2019). We explain the details of the anomaly injection in the subsequent sections.

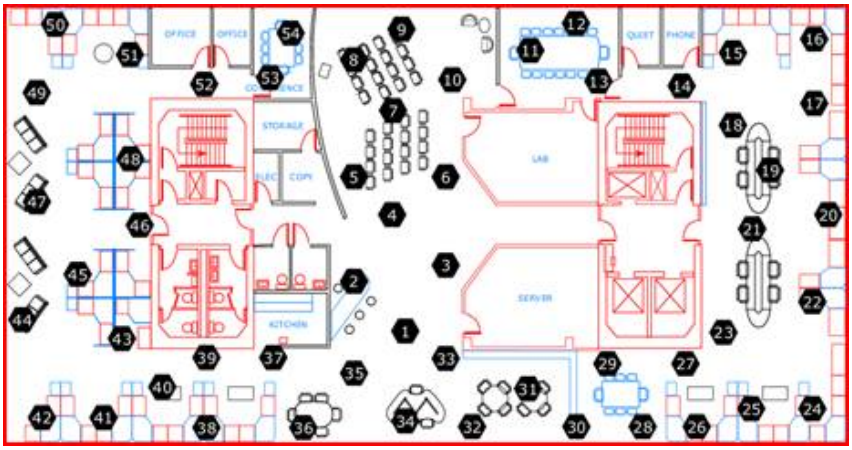


Figure 4.5: Map of the building where sensors were recording data.

EDA

Each of the 54 sensors record data with different rates (1 to 3 minutes) and for different duration. Light data time series lengths range from 2038 time steps to 65689 time steps. Figure 4.6 shows the distribution of the length range of the light time series datasets. Most of the light time series have a range between 30000 and 40000 time steps.

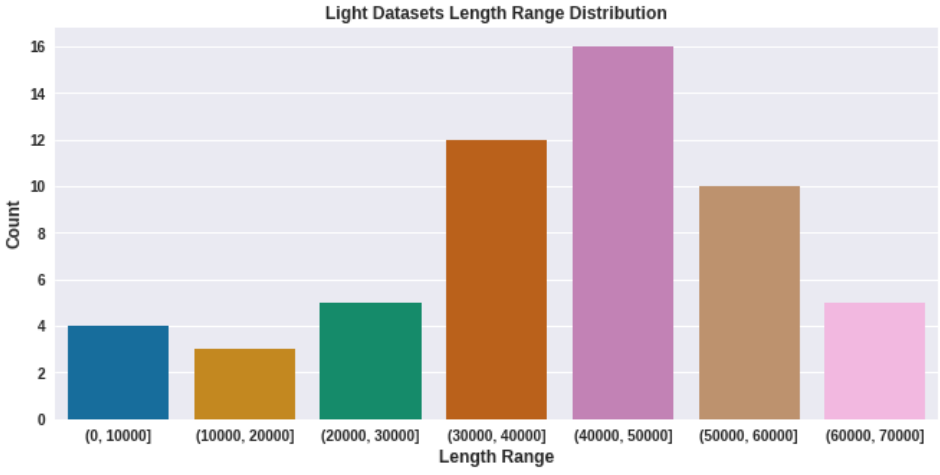


Figure 4.6: Light time series dataset length range distribution.

Figure 4.7 shows a light dataset with 32998 time steps. We observe that the recorded values of the light level change drastically over time. Specifically, at some points in time there is zero amount of light, and suddenly the light value jumps to its maximum level over the entire time series. We attribute that drastic change to the beginning and end of the business hours in the building the data was recorded.

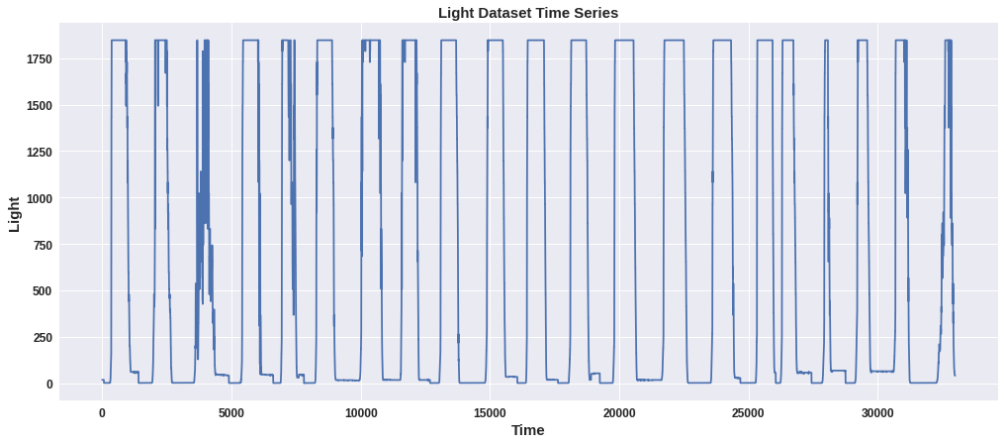
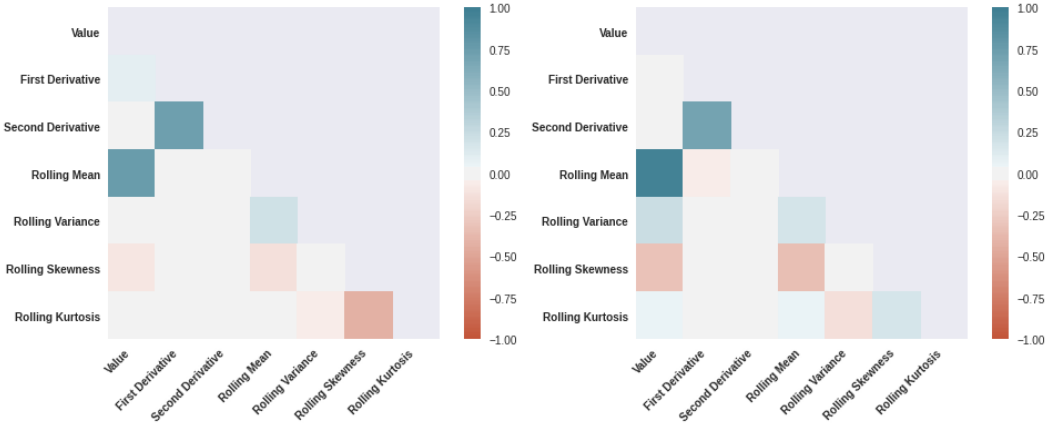


Figure 4.7: A light time series dataset with 32998 time steps of recorded light data.

Feature Correlation

Figure 4.8 shows statistical features correlation heatmap for two randomly selected light datasets. We observe that there is some level of correlation between the feature pair (rolling mean, value) and feature pair (first derivative, second derivative). Both of these cases are expected due to the long periods of stable light level, e.g. zero light after business hours, constant light during meetings or special events. These periods of stability in light level will be identified as stable concepts by our concept drift detection algorithm. The



(a) Statistical features correlation heatmap for one of the light datasets.

(b) Statistical features correlation heatmap for another light dataset.

Figure 4.8: Numbers on or above main diagonal are redundant as correlation matrix is symmetric, hence removed from plot.

anomaly classification model will benefit from concept drift driven features, i.e. concept or drift, duration of period, and percentile of point, the most during these periods. We provide detailed discussions on the performance of the model in such cases in the next chapter.

4.2.3 Power Supply Dataset

Power supply dataset contains the hourly recorded power supply of an electricity company in Italy gathered from 1995 to 1998 [(Zhu, 2010), (Dau *et al.*, 2018)]. This dataset is known to have concept drifts caused by many factors including seasonal change, business hour change, and weather change (Zhu, 2010). It is used as a benchmark dataset for various concept drift detection studies [(Huang *et al.*, 2015), (Alves de Souza *et al.*, 2020)]. Similar to the light dataset, power dataset is not labeled for concept drift, i.e. the exact location

of drift occurrences are unknown. Also, similar to the light dataset, power dataset does not have any known anomalies and we use anomaly injection to prepare it for anomaly detection model training.

EDA

The power dataset consists of 29928 recorded time steps which accounts for 1247 days or 3.4 years. Figure 4.9 shows the power supply time series. This figure gives us a broad overview of the power supply dataset and the various possible drifts that occur in it. The recurrent drops occur approximately every 8500 hours, which is approximately the duration of a year. We attribute that drop to new year holidays when the businesses are closed and the power consumption is lower.

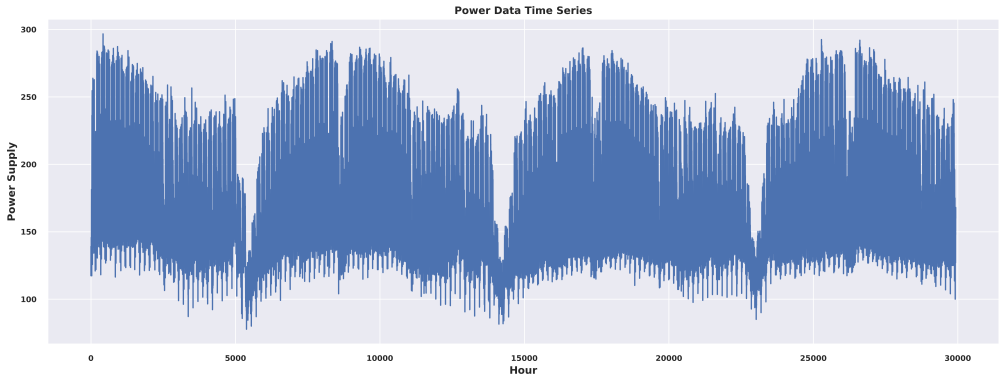


Figure 4.9: Power supply time series dataset recorded hourly over more than three years.

Figure 4.10 shows the power supply time series over a 8760 hours, i.e. one year, period. We observe that as months and seasons change throughout a year, the power consumption and hence the power supply changes.

Figure 4.11 shows the power supply time series over a 744 hours, i.e. one

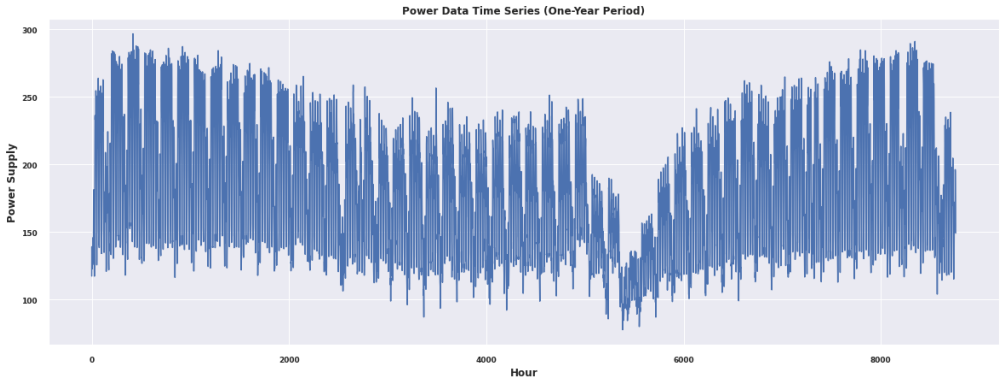


Figure 4.10: Power supply time series dataset over a year period.

month, period. As we further zoom in time, we observe that 5-day work weeks followed by weekends create reoccurring patterns over a short period of time. However, as observed in Figure 4.10, those patterns change over longer periods of time too.

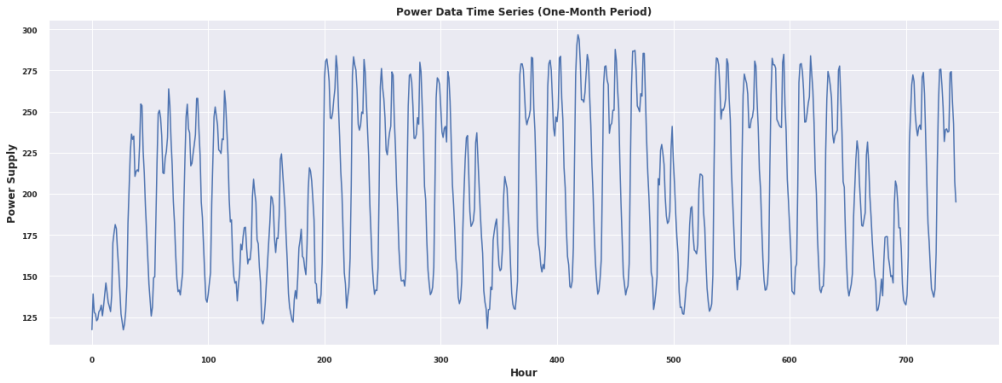


Figure 4.11: Power supply time series dataset over a month period.

Figure 4.12 shows the power supply time series daily average. Daily average is computed by taking average of power supply over a 24 hour period. Daily average supply chart pattern resembles the original hourly supply chart. This shows a reoccurring pattern of change that is consistent over time to an extent

that even the 24-hour average chart has similar change patterns.

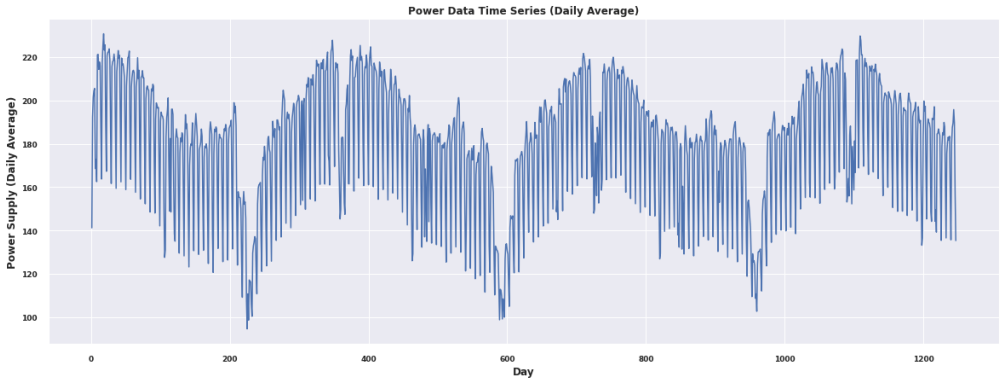


Figure 4.12: Power supply time series dataset daily average.

Feature Correlation

Figure 4.13 shows statistical features correlation heatmap for the power supply time series data. Most of the feature pairs have a low correlation. The pair (rolling mean, rolling variance) show a slightly higher correlation compared to other pairs. Figure 4.11 shows that in the business days as the average power supply increases, the fluctuation (variance) of the supply also increases. This is because despite the higher power supply during the daytime in business days that ranges from 150 to 300 compared to the daytime in weekends that ranges from 125 to 225, the night time supply remains less variable for both business days and weekends in the range (150, 200). Therefore, more average in power supply co-occurs with more variance.

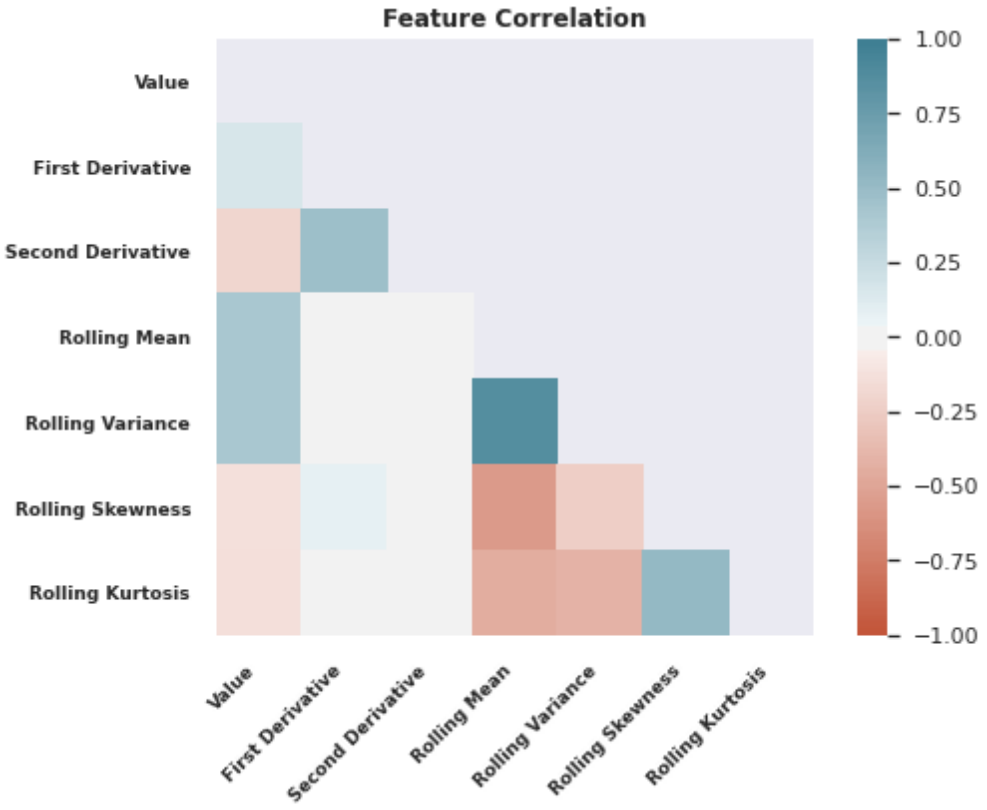


Figure 4.13: Statistical features correlation heatmap for the power supply dataset.

4.2.4 Synthetic Datasets

In order to specifically study each type of concept drift influence on the anomaly detection, we use a tool to synthetically generate such drifts in an isolated settings. We use the Massive Online Analysis (MOA) tool (Bifet *et al.*, 2010) to generate datasets with abrupt, gradual, incremental, and mixed (all three drift types) drifts. Synthetically generated datasets are anomaly-free. Therefore, we will inject anomalies to them to further study how concept drift affect anomaly detection.

EDA

Figure 4.14 shows a synthetic dataset generated with abrupt drifts. It is worth noting that in order to ensure that only the requested drifts are present in the dataset, the MOA tool keeps the non-drifting parts of the dataset with a low variance and smaller fluctuations. Figure 4.15 shows a fraction of the Figure 4.14 with a zoom on the drifting period.

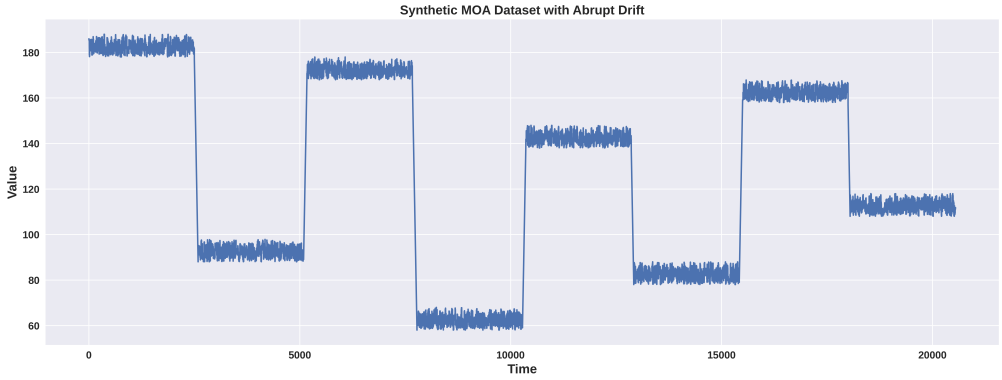


Figure 4.14: Synthetic MOA dataset with multiple abrupt drifts.

Figure 4.16 shows a synthetic dataset generated with gradual drifts. Figure 4.17 shows a fraction of the Figure 4.16 with a zoom on the drifting period.

Figure 4.18 shows a synthetic dataset generated with incremental drifts. Figure 4.19 shows a fraction of the Figure 4.18 with a zoom on the drifting period.

Feature Correlation

Figure 4.20 shows statistical features correlation heatmap for the MOA datasets with abrupt and gradual drifts. Most of the feature pairs have a low correlation. The pairs (rolling mean, value) and (first derivative, second derivative)

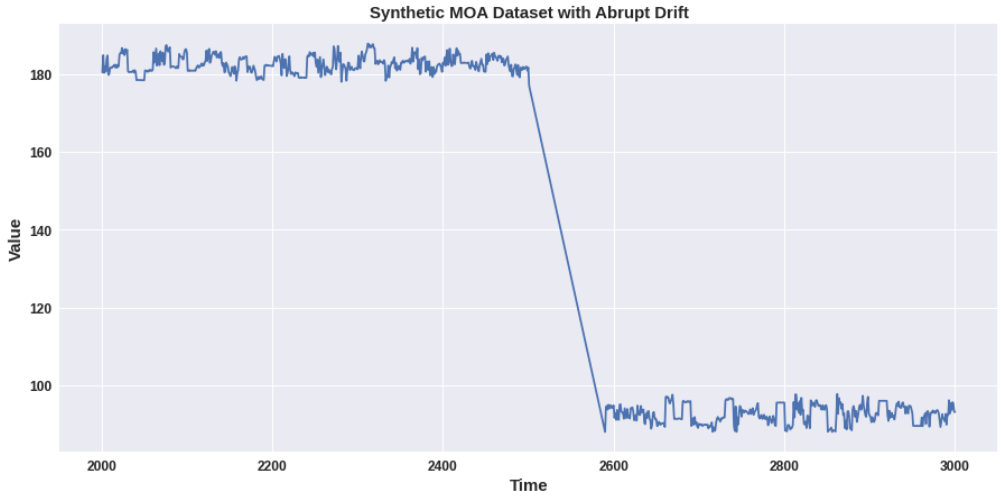


Figure 4.15: The drifting period of the synthetic MOA dataset with abrupt drift.

have a higher correlation. As mentioned, the MOA tool takes extra caution when generating drift datasets to ensure that only the requested drifts are present in the dataset. Therefore, most parts of the dataset experience low variance and maintain high value stability. Therefore, the rolling mean correlates positively with the value feature. The same reasoning is given for the first derivative and the second derivative.

4.3 Anomaly Injection

We introduced four datasets in the previous section, i.e. Yahoo! datasets, light datasets, power supply dataset, and synthetically generated MOA datasets. However, only the Yahoo! datasets are known to have anomalies and labeled by domain experts. For the purpose of this study, we need anomalies with known location, i.e. labeled, to train a supervised classifier. Therefore, we

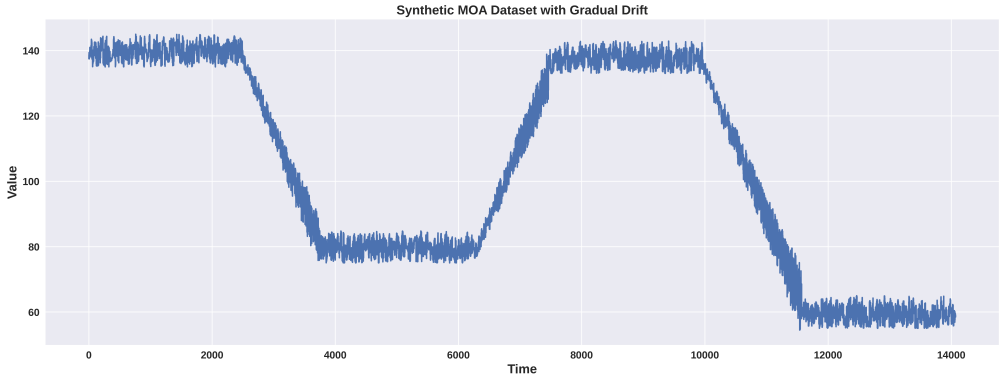


Figure 4.16: Synthetic MOA dataset with multiple gradual drifts.

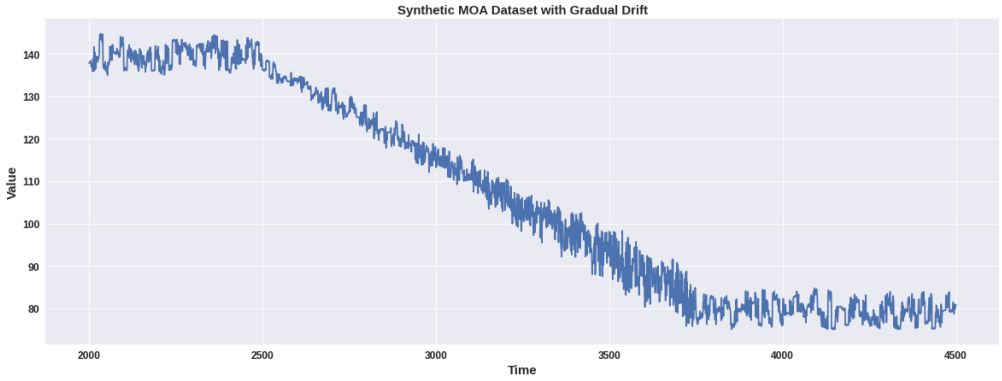


Figure 4.17: The drifting period of the synthetic MOA dataset with gradual drift.

need to manually inject anomalies in those datasets.

In this section, we first introduce the anomaly distributions and the anomaly rates we use to inject anomalies. Then, we proceed with introducing the tool we use to inject anomalies in the time series data.

4.3.1 Anomaly Distribution

In order to establish a fair and robust settings to evaluate our anomaly detection classifier and selection of features, we inject anomalies in the time series

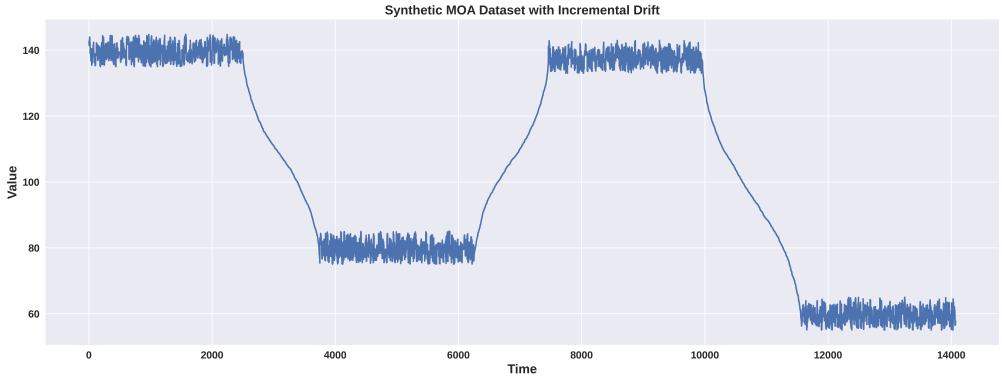


Figure 4.18: Synthetic MOA dataset with multiple incremental drifts.

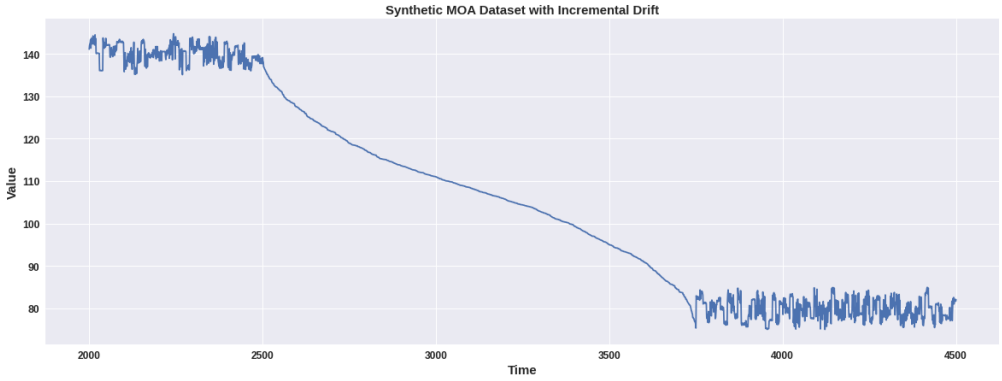
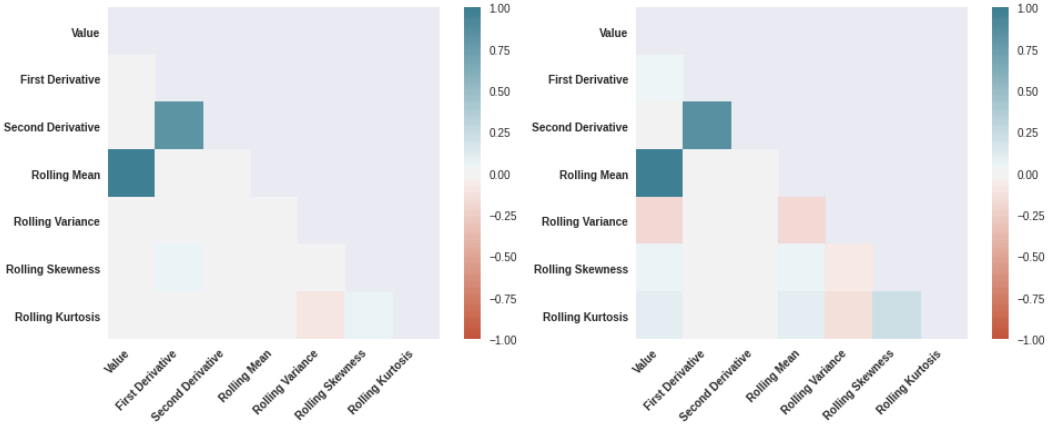


Figure 4.19: The drifting period of the synthetic MOA dataset with incremental drift.

data according to three distributions and ten error rates. Specifically, we consider uniform (random), Gaussian (normal), and Zipfian distributions for injected anomalies. We inject anomalies at the following rates, i.e. percentage of anomalies in the total time series length: 0.5%, 1%, 1.5%, 2%, 2.5%, 3%, 3.5%, 4%, 4.5%, 5%.



(a) Statistical features correlation heatmap for the MOA dataset with abrupt drifts.

(b) Statistical features correlation heatmap for the MOA dataset with gradual drifts.

Figure 4.20: Numbers on or above main diagonal are redundant as correlation matrix is symmetric, hence removed from plot.

4.3.2 Anomaly Injection Method

Once we determine the location of anomalies, we use a technique proposed by (Gierke and Stebner, 2019) to inject them. They also provided an open source implementation that we use as a tool to inject anomalies. Provided with the location or interval at which anomalies must be injected, (Gierke and Stebner, 2019) analyzes the time series at a locality around that area and produce outliers. They calculate outlier values based on the statistical properties of an interval around the designated points where we want to inject anomalies. Then they replace the existing value at each designated point with the anomalous value. Recently, (Preuveneers *et al.*, 2020) used this tool in their study of resource and performance trade-off optimization for anomaly detection in machine learning systems.

4.4 Classifier

We use multi-layer-perceptron as our anomaly detector. Multi-Layer Perceptron (MLP) is a supervised neural network based model. It is a class of Feed-Forward Neural Networks that takes the features as its input, pass them through one or multiple hidden layers, and outputs the final label computed by the network. MLPs are able to learn non-linear relations between the features and the desired output labels. Figure 4.21 shows the architecture of an MLP model with three hidden layers.[1]

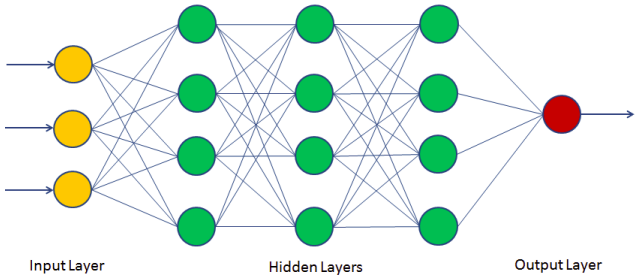


Figure 4.21: Multi-Layer Perceptron models are a class of Feed-Forward Neural Networks

During our initial experiments, MLP performed very promising and we decided to proceed with it. Aside from the MLP, we also tried k nearest neighbors classification, support vector machine classification, and random forest classification to evaluate their performance. Based on their accuracy, we decided to proceed with the MLP model.

We used the randomized hyper-parameter optimization technique (Bergstra

[1]Figure taken from here: https://www.datacamp.com/community/tutorials/neural-network-models-r

and Bengio, 2012) to optimize the hyper-parameters and find the best architecture. Table 4.1 demonstrates the specific details of the MLP architecture used after optimization process.

No. of Hidden Layers	**Batch Size**	**Initial Learning Rate**	**Activation Function**
3	128	0.0005	relu

Table 4.1: MLP architecture details

Chapter 5

Experiments

In the first section of this chapter we present the specs of the hardware we use to run the experiments. We then proceed with presenting the details of the software we programmed to implement the algorithm and analyses. In the third section, we provide the metrics we use to measure the accuracy of our approach. In the fourth section, we discuss the baseline study we use to compare our metric results with. In the fifth section, we provide the metrics results and a comparison with a baseline study. We conclude this chapter by discussing the impact of concept drift on anomaly detection in the last section.

5.1 Hardware

For this study, we use an isolated Ubuntu 20.04.2 LTS operating system running on Linux 5.8.0-63-generic kernel and powered by 8 GB of memory and 8 Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. We flush the memory and kill all of the non-critical background processes every time we run an experiment to

make sure that our runtime environment is fair towards various experiments.

5.2 Software

We implement the program to run the desired experiments using Python programming language version 3.8.10. We use some reputable third party libraries widely used by data science practitioners in both academic and industrial environments. Some of these libraries are listed here.

- Pandas, an open source data analysis tool for Python.[1]
- Numpy, an open source library used for scientific computing with Python.[2]
- Seaborn, an open source data visualization library for Python.[3]
- Scipy, an open source mathematical and engineering computation library for Python.[4]
- Scikit-Learn, an open source classical machine learning library for Python.[5]
- Keras, an open source neural network based machine learning library for Python.[6]

[1] https://pandas.pydata.org/
[2] https://numpy.org/
[3] https://seaborn.pydata.org/
[4] https://www.scipy.org/
[5] https://scikit-learn.org/stable/
[6] https://keras.io/

5.3 Metrics

Many anomaly detection studies use F1-Score as a metric to evaluate their methods [(Munir *et al.*, 2019), Maya *et al.* (2019), (Le and Papotti, 2020)] while others use Area Under Receiver Operating Characteristic Curve (AUC-ROC) [(Sakurada and Yairi, 2014), (Malhotra *et al.*, 2015)]. We decide to proceed with F1-Score for two main reasons. Firstly, the nature of our study and the observations made in the exploratory data analysis (EDA) section in the previous chapter requires a metric that is suitable and robust in highly imbalanced environments. F1-Score is a robust metric often used in such environments. Secondly, the recent study that reported the best performance (Le and Papotti, 2020) used F1-Score as their metric. We compare our results with their reported results. We provide the definition of F1-Score in the Appendix C.

5.4 Baseline Comparison

We use a recent study by (Le and Papotti, 2020) to compare our results with. They employ a symmetric and non-parametric version of k nearest neighbors algorithm called inverse nearest neighbors (INN). INN assigns a clusters of data points into the same group if and only if all of those points are within each others nearest neighbors.

As discussed in the previous chapter and earlier in this chapter, we compare our result metrics with the metrics reported in the recent anomaly detection in time series study by (Le and Papotti, 2020). However, they only reported

the performance of their framework on the Yahoo! datasets and we do not have the performance results of their framework on the other three datasets we use. We implement their approach to have a baseline for comparison for the other three datasets we use to evaluate our approach.

Based on our observations, their average reported f1-score for the Yahoo! datasets is approximately 30% higher than our implementation of their work. We iterated through the algorithm steps introduced in the original paper (Le and Papotti, 2020) several times to ensure its accuracy. We also reached out to the authors for clarifications when needed to ensure an accurate implementation. After carefully inspecting the algorithm steps and exploring the results, we can attribute this gap in f1-score to one specific reason. The last step of the INN algorithm consists of an unsupervised Gaussian Mixture clustering algorithm that identifies anomalous data points. The details of this step were not disclosed to us and despite our efforts to tune and optimize this step to reduce the gap, we could not replicate their results. Therefore, we account for up to 30% gap in our reported results for the INN algorithm and treat them as such.

It is noted that the INN algorithm has an active learning mode where the algorithm asks for human input when it is not confident about its decision. Given that we are unable to perform active learning mode of their algorithm on the light, power, and MOA datasets, and in order to establish a fair comparison, we evaluate and report the baseline results with its default mode.

It is also noted that the (Le and Papotti, 2020) reported that their approach outperforms several existing state-of-the-art anomaly detection methods, including [(Adams and Mackay, 2007), (Vallis *et al.*, 2014), (Solaimani *et al.*, 2014)] without active learning. Therefore, (Le and Papotti, 2020) have reported the highest F1-Score performance and we use their work as the baseline comparison.

5.5 Results and Comparison

In this section, we report the metric results of performing our anomaly detector on the four datasets we introduced in the previous chapter. We also provide a discussion for each of the datasets' metric results.

5.5.1 Yahoo! Datasets

In the previous chapter, we discussed that we proceed with 52 Yahoo! datasets for the purpose of this study. In this subsection, we first provide the average F1-Score and runtime over all of the Yahoo! datasets for our anomaly detection method, our implementation of the baseline INN algorithm, and the official reported INN algorithm (Le and Papotti, 2020). We then proceed with detailed F1-Score and runtime results for each of the Yahoo! datasets individually.

Figure 5.1 demonstrates the average F1-Score and average runtime for the Yahoo! datasets. As mentioned before, our implementation of the INN approach is off by approximately 30% compared to what the authors reported (Le and Papotti, 2020).

(a) Yahoo! datasets average F1-Score comparison.

(b) Yahoo! datasets average runtime comparison.

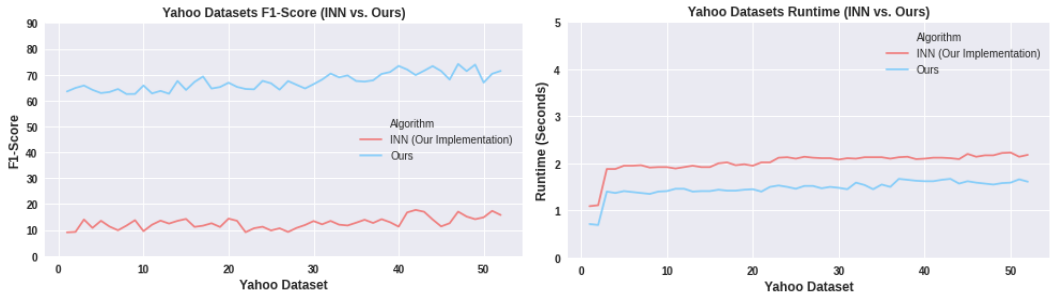
Figure 5.1: Anomaly detection F1-Score and runtime comparison.

We also report the individual results for Yahoo! datasets. Figure 5.2 shows the F1-Score and runtime for all of the 52 Yahoo! datasets examined.

Discussion

Figure 5.1a suggests that our algorithm performs better than the baseline INN algorithm in terms of F1-Score by approximately 20% on average. Also, Figure 5.2a suggests that our algorithm achieves at least 63% of f1-score consistently among all of the Yahoo! datasets. The baseline algorithm authors did not report individual Yahoo! dataset f1-score results. If we account for 30% offset we discussed earlier in this section, out algorithm performs better compared to the baseline algorithm consistently among all Yahoo! datasets.

Given the relatively small size of the Yahoo! datasets, the running time



(a) Yahoo! datasets F1-Score comparison.

(b) Yahoo! datasets runtime (seconds) comparison.

Figure 5.2: Yahoo! datasets F1-Score and runtime comparison for all individual datasets.

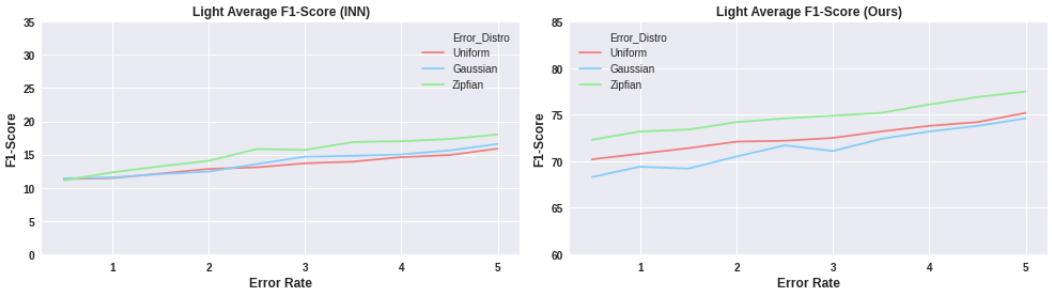
difference is negligible. We will discuss the running time results for the light, power, and MOA datasets which are larger in size compared to the Yahoo! datasets.

5.5.2 Light Datasets

For the purpose of anomaly detection we inject anomalies in the light datasets to study the performance of our concept drift aware anomaly detector. We report the average F1-score and runtime for various error distribution and error rates. We also report the individual light dataset f1-score and runtime with a fixed error distribution and rate to observe how consistent our algorithm performs on each dataset.

Figure 5.3 shows the comparison of average F1-Score for various error distributions and error rates for light datasets.

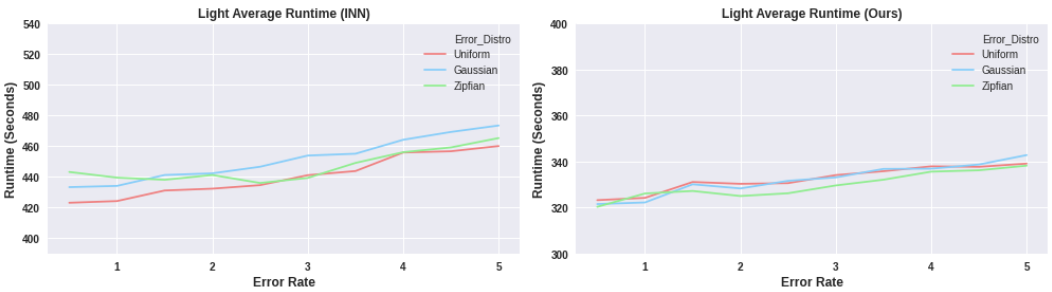
Figure 5.4 shows the comparison of average runtime for various error distributions and error rates for light datasets.



(a) INN algorithm's F1-Score for various anomaly distributions

(b) Our algorithm's F1-Score for various anomaly distributions

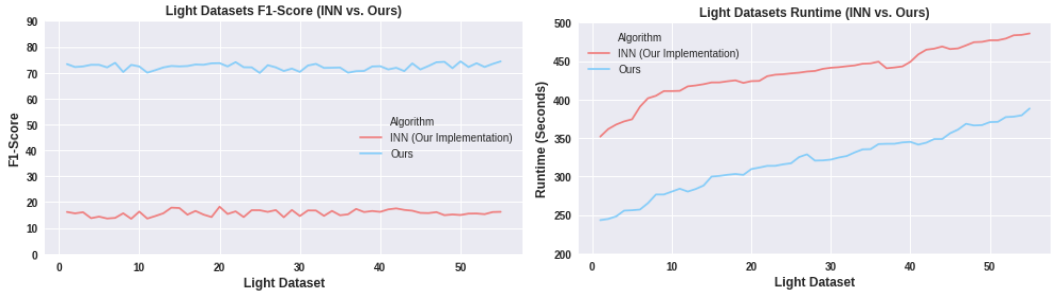
Figure 5.3: Light datasets F1-Score comparison between our algorithm and the baseline INN algorithm for various anomaly distributions.



(a) INN algorithm's runtime for various anomaly distributions

(b) Our algorithm's runtime for various anomaly distributions

Figure 5.4: Light datasets runtime comparison between our algorithm and the baseline INN algorithm for various anomaly distributions.



(a) F1-Score for all light datasets with fixed error rate and distribution.

(b) Runtime for all light datasets with fixed error rate and distribution.

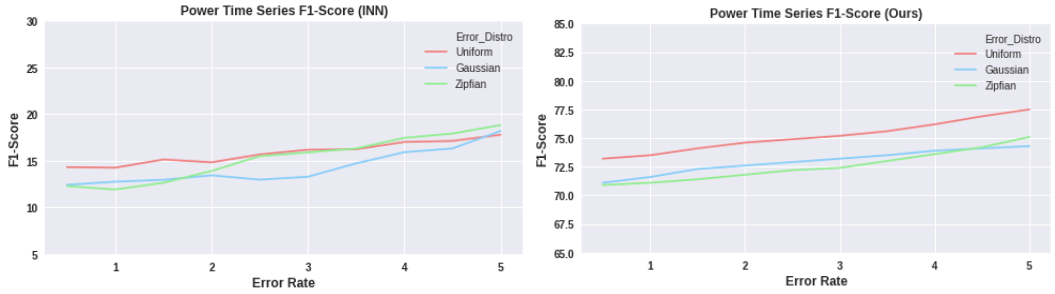
Figure 5.5: Light datasets F1-Score and runtime comparison between our algorithm and the baseline INN algorithm. Uniform error distribution with a 3% rate fixed among all datasets.

In order to compare on a single time series basis, as opposed to the average based comparison where we aggregate the metrics over all time series, we fixed the error distribution and error rate to be uniform and 3% for all of the light time series, respectively. We proceeded with running experiments against all of the datasets to compare our approach with the baseline INN algorithm. Figure 5.5 shows the f1-score and runtime for all of the 55 light datasets with a fixed uniform error distribution at 3% rate.

Discussion

Figure 5.3b suggests that for various error distributions and error rates, our algorithm outperforms the INN results (after adjusting for 30% gap). It is observed that f1-score improves as the error rate increases. This is expected as the MLP model has access to more training data points with anomalous label.

At the individual time series level, Figure 5.5b suggests that our algorithm



(a) INN algorithm's F1-Score for various anomaly distributions

(b) Our algorithm's F1-Score for various anomaly distributions

Figure 5.6: Power supply dataset F1-Score comparison between our algorithm and the baseline INN algorithm for various anomaly distributions.

achieves close to 70% f1-score consistently among all light time series with various lengths. It outperforms the INN baseline, even after adjusting the discussed 30% gap.

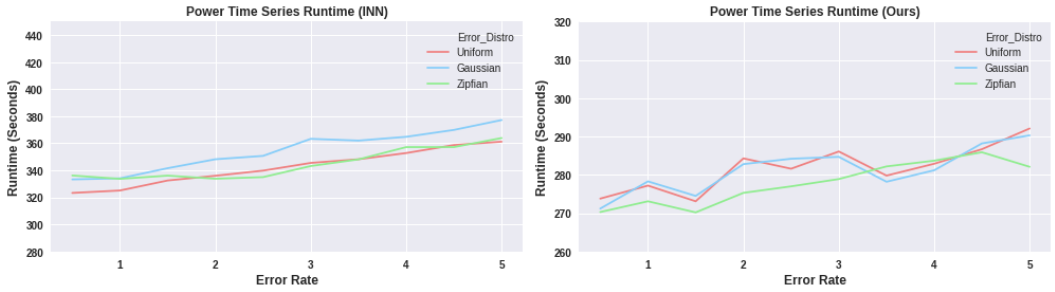
As for the runtime, both Figure 5.4 and 5.5a suggest that our algorithm finishes execution faster than the baseline INN algorithm, both on average and consistently among individual light datasets with various lengths. We discuss this after we present the results for the power dataset.

5.5.3 Power Supply Dataset

Similar to the light datasets, we inject anomalies in the power dataset to study the performance of our anomaly detector. Unlike Yahoo! and light datasets which consist of multiple datasets, power dataset is only one time series.

Figure 5.6 shows the comparison of F1-Score for various error distributions for the power supply dataset.

Figure 5.7 shows the comparison of runtime for various error distributions



(a) INN algorithm's runtime for various anomaly distributions

(b) Our algorithm's runtime for various anomaly distributions

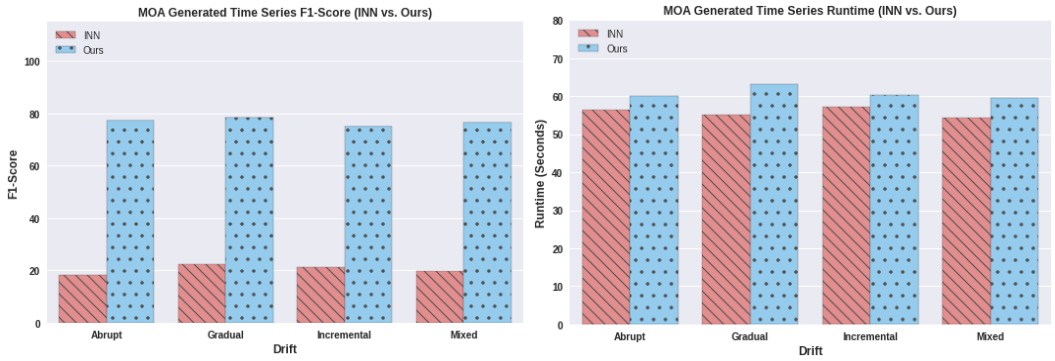
Figure 5.7: Power supply dataset runtime comparison between our algorithm and the baseline INN algorithm for various anomaly distributions.

for the power supply dataset.

Discussion

Figure 5.6 shows that our algorithm outperforms the baseline INN algorithm (after 30% gap adjustment). It also suggests that it performs robustly with different error distributions and error rates with an average of approximately 72% for f1-score.

Figure 5.7 demonstrates that our algorithm outperforms the baseline INN algorithm in terms of the runtime. The INN algorithm uses the Gaussian Mixture Model (GMM) as its anomaly detector (Le and Papotti, 2020). GMMs have a time complexity that is polynomial to the power of 3 with respect to the feature dimension (Pinto and Engel, 2015). However, we use the MLP which has a time complexity polynomial to the power of 1 with respect to the feature dimension. This is why our approach is quicker in execution compared to the baseline INN approach.



(a) F1-Score for various drift types in MOA dataset.

(b) Runtime for various drift types in MOA dataset.

Figure 5.8: MOA generated datasets with various drift types and fixed error distribution and error rate. F1-score and runtime comparison.

5.5.4 Synthetic MOA datasets

We use this datasets to evaluate the robustness of our algorithm against various types of drift. Even though the light and power datasets are known to have various types of drift, the MOA tool incorporates measurements to ensure that only the specified drifts are present in the generated dataset. Therefore, we can evaluate our algorithm's performance against each specific drift type or a mixture of them.

It is noted that we use a fixed uniform error distribution with a rate of 3% to establish a comparison basis to evaluate our algorithm's performance against various types of drift.

Figure 5.8 shows the comparison of f1-score and runtime for various drift types in MOA dataset with a fixed uniform error distribution and 3% error rate.

Discussion

Figure 5.8a suggests that our algorithm outperforms the baseline INN algorithm (considering 30% gap adjustment). It also shows that our algorithm performs robustly in the presence of various types of drifts, individually and mixed. We observe that compared to the previous three datasets, our algorithm performs better on MOA generated datasets. We explored the MOA datasets in the previous chapter and observed that in order to ensure the drift presence, the MOA generated datasets have a low variance in the non-drifting periods. Therefore, anomalies that occur in the non-drifting periods are more distinguished and hence, easier for the classifier to detect.

In terms of the runtime, Figure 5.8b demonstrates that our algorithm and the baseline INN algorithm have a similar execution time for the MOA generated datasets. We attribute this closeness in execution time to the fact that baseline INN algorithm's last step is an unsupervised clustering technique that is able to cluster many of the less fluctuating non-drifting points together. This will save time on the clustering algorithm by grouping many more points together compared to a more fluctuating dataset like the light datasets.

5.6 Concept Drift Impact Discussion

Table 5.1 shows the summary of all of the experiments we conducted to evaluate our framework's performance on various type of datasets. We compare our framework's performance with the recently published INN algorithm that we use as a comparison basis (Le and Papotti, 2020). The average F1-Score

of our approach over the four datasets is 72.7%. The average F1-Score for the INN approach is 45.7%. As for the runtime, the average over the four datasets for our approach is 168.5 seconds. While the runtime average for the INN approach is 212.3 seconds.

Dataset	**Our Approach**		**Baseline INN Approach (F1-Score Adjusted 30%)**	
	Average F1-Score	**Average Runtime (S)**	**Average F1-Score**	**Average Runtime (S)**
Yahoo!	67.8	1.5	43.5 (Reported)	0.4
Light	72.9	331.4	44.1 (Adjusted)	446.1
Power	73.5	280.3	45.1 (Adjusted)	347.0
Synthetic	76.8	60.8	50.4 (Adjusted)	55.7

Table 5.1: Complete metric report for our approach and baseline INN approach. F1-Score for baseline INN approach is adjusted by 30% as discussed earlier to account for implementation vs. reported F1-Score gap.

In order to study the direct impact of concept drift on our anomaly detector's performance, we excluded the concept drift features and executed our experiments with only statistical features. Specifically, Table 5.2 shows the feature vector we used to perform this experiment. The following concept drift specific features are excluded.

- Feature 8: Concept or Drift
- Feature 9: Duration of the Period
- Feature 10: Percentile of the Point

We report the average F1-Score over the four datasets in Table 5.3 when considering all features vs. when excluding concept drift features. It is noted

that the exclusion of concept drift features has a higher impact on the light, power supply, and synthetic datasets compared to the Yahoo! dataset. As we studied in Chapter 4, the Yahoo! datasets are shorter in length compared to the other three datasets. Given the minimum stable concept length parameter, ϕ, there are not enough room in Yahoo! datasets to detect as many stable concepts and drifts as it is possible in the other three datasets. This is why the impact of concept drift on the Yahoo! datasets is lower than the other three datasets. It is also noted in Table 5.3 that the explicit inclusion of concept drift features has improved the performance of the anomaly detector in terms of the F1-Score by 12.5% on average among all datasets.

Table 5.2: Statistical feature vector

Value	First Derivative	Second Derivative	Rolling Mean	Rolling Variance	Rolling Skewness	Rolling Kurtosis

Table 5.3: F1-Score report for our approach when considering all of the features comparison with when excluding concept drift features.

Dataset	**Average F1-Score**	
	With CD features	**Without CD featues**
Yahoo!	67.8	61.2
Light	72.9	59.4
Power	73.5	60.3
Synthetic	76.8	60.2
Average	**72.7**	**60.2**

Chapter 6

Conclusion

In this work, we studied the problem of identifying anomalies in time series data in the presence of concept drift. We studied how previous works neglected in whole or in part the impact of concept drift when studying anomaly detection in time series data. We also studied how concept drift affect time series data in dynamic settings in which data evolve as time passes. We proposed our integrated framework in an attempt to overcome the limitations of the previous studies neglecting the impact of concept drift on anomaly detection in time series data. We proposed a concept drift detection algorithm that is able to identify and locate various types of concept drift, i.e. abrupt, gradual, and incremental drift, in univariate time series data. We then used the information gained from the output of this algorithm in conjunction with a set of statistical features extracted from the time series to construct a feature vector. We used this feature vector to train a feed-forward neural network to learn and model the behavior of anomalies and to be able to identify them. We

performed experiments on various real world datasets used in the literature by the previous studies and on a set of synthetic datasets. We showed that our approach overcomes the recently published baseline study. Specifically, our approach outperforms the baseline INN method in terms of the average anomaly detection F1-Score over four datasets by 27%. As for the runtime, for the same datasets, it is 43.8 seconds faster on average. We also showed that our integration of concept drift detection in the form of three explicit features fed as part of the input to the anomaly detector has improved the average F1-Score over four datasets by 12.5%.

6.1 Future Work

Context aware neural networks such as recurrent neural networks (RNNs) are able to learn the temporal relationships that exist in data (Sherstinsky, 2020). One direction to extend this study is to incorporate RNNs with the result of our concept drift detection algorithm to further study the impact of concept drift on anomaly detection and improve the its performance. It is a challenge to ensure that the extracted concept drift features do not neutralize the temporal relationship based mechanism of RNNs and to ensure both concept drift features and RNN model operate towards the same goal of detecting anomalies with higher accuracy.

One other direction in which this study can be continued and extended is to incorporate context-aware neural network models such as Transformer models (Vaswani *et al.*, 2017) as the anomaly detectors. Transformer models

are able to consider the historical data that goes back as far as the beginning of the train data in the decision making process. It is noted that we consider historical context of the data by incorporating concept drift features in our feature set and transformer models must be incorporated while considering this fact. This method can further investigate the impact of historical data on the anomaly detection problem.

Another direction to extend this study is to incorporate an online anomaly detection method that is able to process the time series data as it becomes available and identify anomalies on the go. Online learning models are used when it is computationally infeasible to process all of the available data at once or when data is in the form of streams that becomes available over the time (Hoi *et al.*, 2014).

Appendix A

Concept Drift Detection Parameters

In this appendix, we present our default values for the concept drift detection parameters. Specifically, we recommend the default values for the following parameters used in the concept drift detection algorithm if they are not provided by the operator:

1. Base Window Size, W_b
2. Minimum Stability Period, ϕ
3. Maximum Tolerable Fluctuation, ϵ
4. Maximum Abrupt Drift Duration, δ
5. Minimum Gradual Drift Period, Δ
6. Maximum Gradual Drift Stat Fluctuation, μ

7. Gradual Drift Graduality Period, ν

We also study the impact of the novel parameters we introduced to extend and complete the concept drift definitions in the context of time series data. Specifically, we report the experimental results on the Yahoo! and the MOA datasets with varying W_b and Δ values while other parameters are fixed. We choose Yahoo! because they are shorter in length and has fewer concept drift and it enables us to study the influence of W_b and Δ when there are fewer drifts. We also choose MOA because they have multiple specific drift types as opposed to the Yahoo! datasets.

It is noted that we only recommend default values for these parameters based on the experiments on the datasets we had. Other users of the algorithm may decide to use customized values based on other datasets instead. We also assume that a time series has a length of at least 500 points in order to contain the notions of stability and drift. We conclude this appendix by providing a discussion on how these parameters are adaptive with new datasets that are not studied by us.

Given time series T demonstrated as $T = \{T[1], T[2], ..., T[n]\}$, we propose the following default values for the given parameters.

Base Window Size, W_b

Our recommended value for this parameter is:

$$min(100, \lfloor 0.01 * n \rfloor)$$

where the $min(a, b)$ returns the minimum value between a and b. Based on our experiments, a maximum of 100 points is optimal to incorporate historical context when identifying concept drift. Increasing this upper bound will reduce the influence of the current examining points in the concept drift detection algorithm introduced in Chapter 3. Decreasing this parameter will amplify the influence of the current examining points and reduce the importance of historical context.

In order to further study the impact of this parameter on anomaly detection, we report the experimental results on the Yahoo! and MOA datasets with W_b values [80, 90, 100, 110, 120] while other parameters are fixed in Table A.1.

W_b	**Yahoo! Average F1-Score**	**MOA Average F1-Score**
80	66.1	74.9
90	67.2	75.4
100	67.8	76.8
110	67.7	76.7
120	65.9	76.1

Table A.1: Average F1-Score for Yahoo! and MOA datasets for different values of W_b. Other parameters are fixed.

Table A.1 shows that the value of 100 for W_b is optimal for both Yahoo! and MOA datasets. It also demonstrates how changing the value of W_b can impact the performance of anomaly detection.

Minimum Stability Period, ϕ

We recommend that this parameter should be at least 2 times larger than the base window size, W_b. This ensures a complete passage of the examining window, used in the concept drift detection algorithm, over the stability period while only considering the non-drifting points as history for *stat* function calculations.

Maximum Tolerable Fluctuation, ϵ

Our recommended value for this parameter is:

$$\frac{\sigma}{5}$$

where σ is the standard deviation of the time series values. Based on our experiments, one fifth of the standard deviation of the time series is a conservative tolerance level for our drift detection algorithm. Increasing this tolerance results in longer stable concept periods and decreasing it results in longer drift periods.

Maximum Abrupt Drift Duration, δ

We do not propose a default value for this parameter as it is highly dependent on the scale of which the time series data are collected. For instance, the notion of an abrupt drift in a time series consisting of temperature data collected every hour in an animal shelter is different than the same notion for radioactivity level data collected every minute in a nuclear power plant. Therefore, the

domain expert is solely responsible to set up a maximum duration for abrupt drift based on their knowledge of the application.

Minimum Gradual Drift Period, Δ

We do not recommend a direct value for this parameter for the similar reason as for the maximum abrupt drift duration. However, we recommend that this parameter should be at least 10 times larger than the maximum abrupt drift duration in order to distinguish between gradual and abrupt drifts in terms of the duration.

In order to further study the impact of this parameter on anomaly detection, we report the experimental results on the Yahoo! and MOA datasets with *Delta* values [$8X\delta$, $9X\delta$, $10X\delta$, $11X\delta$, $12X\delta$] in Table A.2.

Δ	**Yahoo! Average F1-Score**	**MOA Average F1-Score**	**MOA Gradual-Only Dataset F1-Score**
$8X\delta$	67.7	76.1	77.0
$9X\delta$	67.6	76.3	77.6
$10X\delta$	67.8	76.8	78.4
$11X\delta$	67.8	76.8	78.3
$12X\delta$	67.6	76.7	78.4

Table A.2: Average F1-Score for Yahoo! and MOA datasets for different values of Δ. Other parameters are fixed.

Table A.2 shows that the value of $10x\delta$ for Δ is optimal for both Yahoo! and MOA datasets. Specifically, for the gradual-only MOA dataset, we observe that the highest F1-Score happens when the value of Δ is set to $10x\delta$. It also demonstrates how changing the value of Δ can impact the performance of anomaly detection.

Maximum Gradual Drift Stat Fluctuation, μ

Our recommended value for this parameter is:

$$\frac{\sigma}{2}$$

where σ is the standard deviation of the time series values. Based on our experiments, the maximum gradual drift stat fluctuation must be larger than the maximum tolerable fluctuation in order to justify the drift occurrence. In other words, the amount of tolerance for stability must be smaller than the amount of allowed fluctuation during a drift period.

Gradual Drift Graduality Period, ν

Similar to the minimum gradual drift period parameter, we also do not recommend a direct value for this parameter. Instead, we recommend this value as 10% of the length of the minimum gradual drift period parameter, specifically, $\frac{\Delta}{10}$. With this recommended value, the gradual drift property defined in Chapter 3 is enforced and checked upon at every 10% of the drifting period.

New Datasets Adaptation

We discussed earlier that why the ultimate decision for the parameter values is on the domain experts. However, our suggested default values for the parameters are flexible and adaptive with respect to the statistical properties of the dataset. Specifically, the parameters W_b, ϵ, and μ are directly driven from the statistical properties of the dataset and the parameters ϕ, Δ, and ν are

indirectly driven from the statistical properties of the dataset as we explained earlier in this appendix. We also have optimized these parameter values on four group of datasets and they consistently performed well in terms of F1-Score. Lastly, the decision on the parameter δ is solely made the domain experts as we explained why its value highly depends on the nature of the dataset.

Appendix B

Statistical Features

This appendix contains the definitions for the following statistical features we used in Chapter 4 as features for anomaly detection. We also provide an example for each feature.

1. Value
2. First derivative
3. Second derivative
4. Rolling mean
5. Rolling variance
6. Rolling Skewness
7. Rolling Kurtosis

We define each of the statistical features on a time series T represented as $T = \{T[1], T[2], ..., T[n]\}$ which consists of n steps. We also use time series

$TS = \{11, 13, 10, 17, 14\}$ with five time steps as an example to calculate each feature.

Value

The value feature is defined as follows:

$$Values = \{T(i) \mid i \geq 1\}$$

And the example values for TS will be:

$$Values = \{11, 13, 10, 17, 14\}$$

First Derivative

The first derivative feature is defined as follows:

$$FirstDerivatives = \{T[i] - T[i-1] \mid i \geq 2\}$$

And the example values for TS will be:

$$FirstDerivatives = \{NA, 2, -3, 7, -3\}$$

It is noted that given the derivation step, i.e. dt, is basically one time step and is equal to 1, the derivative value is translated into the difference of value between consecutive time steps.

Second Derivative

The second derivative feature is defined as:

$$SecondDerivatives = \{FirstDerivatives[i] - FirstDerivatives[i-1] \mid i \geq 3\}$$

And the example values for TS will be:

$$SecondDerivatives = \{NA, NA, -5, 10, -10\}$$

Rolling Mean

For a rolling window size of W, the rolling mean is defined as:

$$RollingMean = \{\frac{\sum_{i-W}^{i} T[i]}{W} \mid i \geq W\}$$

And the example values for $W = 4$ and given TS will be:

$$RollingMean = \{NA, NA, NA, 12.75, 13.5\}$$

Rolling Variance

For a rolling window size of W, the rolling variance is defined as:

$$RollingVariance = \{\frac{\sum_{i-W}^{i} (T[i] - mean)^2}{W} \mid i \geq W\}$$

where $mean$ is the mean of the points between $i - W$ and i.

And the example values for $W = 4$ and given TS will be:

$$RollingVariance = \{NA, NA, NA, 9.58, 8.33\}$$

Rolling Skewness

Skewness is defined as the third central moment divided by the square root of the cube of variance:

$$RollingSkewness = \{\frac{m_3}{\sigma^3} \mid i \geq W\}$$

where σ is the square root of variance (standard deviation) of the points between $i - W$ and i. m_3 is the third central moment of that range defined as:

$$m_3 = \frac{\sum_{i-W}^{W}(T[i] - mean)^3}{W}$$

where $mean$ is the mean of the points between $i - W$ and i.

And the example values for $W = 4$ and given TS will be:

$$RollingSkewness = \{NA, NA, NA, 1.13, 0\}$$

Rolling Kurtosis

Kurtosis is defined as fourth central moment divided by the square of variance subtracted by three.

$$RollingKurtosis = \{\frac{m_4}{\sigma^4} - 3 \mid i \geq W\}$$

where σ is the square root of variance (standard deviation) of the points between $i - W$ and i. m_4 is the fourth central moment of that range defined as:

$$m_4 = \frac{\sum_{i-W}^{W}(T[i] - mean)^4}{W}$$

where $mean$ is the mean of the points between $i - W$ and i.

And the example values for $W = 4$ and given TS will be:

$$RollingKurtosis = \{NA, NA, NA, 0.75, 0.91\}$$

Appendix C

Metrics

In this appendix, we provide the definition of the F1-Score, a metric we used to evaluate the performance of our model. F1-Score is computed as the harmonic mean of precision and recall. In this section, we first introduce precision and recall, and then we proceed with the introduction of F1-Score.

Confusion matrix helps better understand and introduce various metrics used in classification, including precision, recall, and F1-Score. In order to define F1-Score, both precision and recall must be defined first. Table C.1 shows how a confusion matrix is constructed. As an example for our use case, if a data point is labeled as anomaly by the classifier (positive label) but in reality it was not anomalous, it is classified as a false positive.

		Predicted Label	
		Positive (1)	Negative (0)
Actual Label	Positive (1)	True Positive (TP)	False Negative (FN)
	Negative (0)	False Positive (FP)	True Negative (TN)

Table C.1: Confusion Matrix

Precision

Precision is defined as the ratio of correctly identified positive labels to the total number of identified positive labels. In our settings, precision is defined as the ratio of correctly identified anomalies to the total number of identified anomalies by the classifier.

$$Pr = \frac{TP}{TP + FP}$$

Recall

Recall, also referred to as sensitivity, is defined as the ratio of correctly identified positive labels to the total number of positive labels regardless of how they are labeled by the classifier. In our settings, recall is defined as the ratio of correctly identified anomalies to the total number of anomalies.

$$Re = \frac{TP}{TP + FN}$$

F1-Score

F1-Score is defined as the harmonic average of precision and recall. It takes both false positives and false negatives into consideration. Therefore, if the model is biased towards one label, one of the precision or recall metrics will be high but not both. For instance, if the classifier favors the anomaly class and mistakenly classifies many data points as anomalies, then the recall score will be high, but the precision, and consequently the f1-score, will be low. F1-Score is widely used to evaluate classification models, especially when the dataset in

highly imbalanced.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Bibliography

Adams, R. P. and Mackay, D. J. C. (2007). Bayesian online changepoint detection. *arXiv: Machine Learning*.

Aggarwal, C. C. (2016). *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition.

Alves de Souza, V., Reis, D., Maletzke, A., and Batista, G. (2020). Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, **34**, 1–54.

Amarasinghe, K., Kenney, K., and Manic, M. (2018). Toward explainable deep neural network based anomaly detection. In *2018 11th International Conference on Human System Interaction (HSI)*, pages 311–317.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, **13**, 281–305.

Bifet, A. and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 7th SIAM International Conference on Data Mining*, **7**.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). MOA: massive online analysis. *J. Mach. Learn. Res.*, **11**, 1601–1604.

Boracchi, G. and Roveri, M. (2014). Exploiting self-similarity for change detection. *Proceedings of the International Joint Conference on Neural Networks*, pages 3339–3346.

Box, G. and Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Braei, M. and Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *ArXiv*, **abs/2004.00433**.

Breiman, L. (2001). Statistical modeling: The two cultures. *Quality Engineering*, **48**, 81–82.

Cabral, D. and Barros, R. (2020). Emzd: Equal means z-test concept drift detector. pages 1037–1044.

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

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, **41**(3).

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen,

A., Batista, G., and Hexagon-ML (2018). The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Dunning, T. and Friedman, E. (2014). *Practical Machine Learning: A New Look at Anomaly Detection*. O'Reilly Media.

Fisher, R. (1930). The moments of the distribution for normal samples of measures of departure from normality. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, **130**, 16–28.

Gama, J. and Rodrigues, P. P. (2009). *An Overview on Mining Data Streams*, pages 29–45. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gama, J. a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, **46**(4).

Gao, J., Song, X., Wen, Q., Wang, P., Sun, L., and Xu, H. (2020). Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks. *ArXiv*, **abs/2002.09545**.

Gierke, W. and Stebner, A. (2019). Kdd-opensource/agots: Anomaly generator on time series. https://github.com/KDD-OpenSource/agots.

Guajardo, J. A., Weber, R., and Miranda, J. (2010). A model updating strategy for predicting time series with seasonal patterns. *Applied Soft Computing*, **10**(1), 276–283.

H. F. M. Oliveira, G., C. Cavalcante, R., G. Cabral, G., L. Minku, L., and L. I. Oliveira, A. (2017). Time series forecasting in the presence of concept drift: A pso-based approach. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 239–246.

Haselsteiner, E. and Pfurtscheller, G. (2000). Using time-dependent neural networks for eeg classification. *IEEE Transactions on Rehabilitation Engineering*, **8**(4), 457–463.

Hoens, T., Polikar, R., and Chawla, N. (2012). Learning from streaming data with concept drift and imbalance: An overview. *Progress in Artificial Intelligence*, **1**.

Hoi, S. C., Wang, J., and Zhao, P. (2014). Libol: A library for online learning algorithms. *Journal of Machine Learning Research*, **15**(1), 495.

Huang, D. T. J., Koh, Y. S., Dobbie, G., and Bifet, A. (2015). Drift detection using stream volatility. In A. Appice, P. P. Rodrigues, V. Santos Costa, C. Soares, J. Gama, and A. Jorge, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 417–432, Cham. Springer International Publishing.

Kumar, V., Banerjee, A., and Chandola, V. (2009). Anomaly detection for symbolic sequences and time series data.

Laptev, N., Amizadeh, S., and Billawala, Y. (2015). Webscope — yahoo labs. `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`.

Le, K.-H. and Papotti, P. (2020). User-driven error detection for time series with events. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 745–757.

Ma, M., Zhang, S., Pei, D., Huang, X., and Dai, H. (2018). Robust and rapid adaption for concept drift in software system anomaly detection. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 13–24.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, volume 89, pages 89–94.

Masud, M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. M. (2011). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, **23**(6), 859–874.

Maya, S., Ueno, K., and Nishikawa, T. (2019). dlstm: a new approach for anomaly detection using deep learning with delayed prediction. *International Journal of Data Science and Analytics*, **8**.

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., and Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, **45**(1), 521–530.

Munir, M., Siddiqui, S. A., Dengel, A., and Ahmed, S. (2019). Deepant: A

deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, **7**, 1991–2005.

Oehmcke, S., Zielinski, O., and Kramer, O. (2015). Event detection in marine time series data. In S. Hölldobler, R. Peñaloza, and S. Rudolph, editors, *KI 2015: Advances in Artificial Intelligence*, pages 279–286, Cham. Springer International Publishing.

Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, **99**, 215–249.

Pinto, R. and Engel, P. (2015). A fast incremental gaussian mixture model. *PLoS ONE*, **10**.

Preuveneers, D., Tsingenopoulos, I., and Joosen, W. (2020). Resource usage and performance trade-offs for machine learning models in smart environments. *Sensors (Basel, Switzerland)*, **20**.

Raab, C., Heusinger, M., and Schleif, F.-M. (2020). Reactive soft prototype computing for concept drift streams. *Neurocomputing*, **416**.

Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. John Wiley amp; Sons, Inc., USA.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, page 4–11, New York, NY, USA. Association for Computing Machinery.

Saurav, S., Malhotra, P., TV, V., Gugulothu, N., Vig, L., Agarwal, P., and Shroff, G. (2018). Online anomaly detection with concept drift adaptation using recurrent neural networks. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CoDS-COMAD '18, page 78–87, New York, NY, USA. Association for Computing Machinery.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, **404**, 132306.

Solaimani, M., Iftekhar, M., Khan, L., and Thuraisingham, B. M. (2014). Statistical technique for online anomaly detection using spark over heterogeneous data from multi-source vmware performance data. *2014 IEEE International Conference on Big Data (Big Data)*, pages 1086–1094.

Tukey, J. W. *et al.* (1977). *Exploratory data analysis*, volume 2. Reading, Mass.

Vallis, O., Hochenbaum, J., and Kejariwal, A. (2014). A novel technique for long-term anomaly detection in the cloud. In *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'14, page 15, USA. USENIX Association.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing*

Systems, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Žliobaite, I. (2011). Controlled permutations for testing adaptive classifiers. In *Proceedings of the 14th International Conference on Discovery Science*, DS'11, page 365–379, Berlin, Heidelberg. Springer-Verlag.

Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585.

Webb, G., Hyde, R., Cao, H., Nguyen, H.-L., and Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, **30**.

Widmer, G. and Kubat, M. (1994). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, **23**.

Wu, W., He, L., and Lin, W. (2019). Local trend inconsistency: A prediction-driven approach to unsupervised anomaly detection in multi-seasonal time series. *ArXiv*, **abs/1908.01146**.

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

Yairi, T., Kato, Y., and Hori, K. (2001). Fault detection by mining association rules from housekeeping data. *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS)*.

yf, y., Zhu, Y., Li, S., and Wan, D. (2014). Time series outlier detection based on sliding window prediction. *Mathematical Problems in Engineering*, **2014**.

Yu, G., Cai, Z., Wang, S., Chen, H., Liu, F., and Liu, A. (2020). Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes. *IEEE Transactions on Network and Service Management*, **17**(3), 1294–1308.

Zambon, D., Alippi, C., and Livi, L. (2017). Concept drift and anomaly detection in graph streams. *IEEE Transactions on Neural Networks and Learning Systems*, **PP**.

Zhang, P., Zhu, X., and Shi, Y. (2008). Categorizing and mining concept drifting data streams. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 812–820, New York, NY, USA. Association for Computing Machinery.

Zhu, X. (2010). Stream data mining repository. `https://www.cse.fau.edu/~xqzhu/stream.html`.

Zhu, X. and Goldberg, A. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **3**(1), 1–130.

Çelik, M., Dadaşer-Çelik, F., and Dokuz, A. (2011). Anomaly detection in temperature data using dbscan algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95.